Universitat Oberta de Catalunya
uoc.edu

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

Meritxell Bosch i Garcia
Màster Universitari en Ciència de Dades
Àrea Medicina

Tutors: Dr. Elisenda Bonet Carne i Dr. Xavier P. Burgos Artizzu
BCNatal Fetal Medicine Research Center (Hospital Clínic i Sant Joan de Déu de Barcelona)

Professor Responsable: Dr. Ferran Prados Carrasco

Juny 2021

FITXA DEL TREBALL FINAL

Títol del treball:	Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals
Nom de l'autor:	Meritxell Bosch i Garcia
Nom del consultor/a:	Dr. Elisenda Bonet Carne i Dr. Xavier P. Burgos Artizzu
Nom del PRA:	Ferran Prados Carrasco
Data de lliurament (mm/aaaa):	06/2021
Titulació o programa:	Màster Universitari en Ciència de Dades
Àrea del Treball Final:	Àrea Medicina
Idioma del treball:	Català
Paraules clau	Ventriculomegalia, aïllada, aprenentatge automàtic, mineria de dades, detecció, classificació, caracterització, dades reduïdes, medicina fetal

Resum del Treball (màxim 250 paraules): Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball

La ventriculomegalia (VMG) és una malaltia rara que afecta entre 0,3 i 22 de cada 1000 embarassos. D'aquests, entre el 25 i el 60% són casos aïllats, és a dir, que no tenen una altra patologia associada. L'evolució de l'afectació i l'efecte sobre el neurodesenvolupament no és previsible.
La detecció de la VMG s'efectua a través de la mesura de dos únics paràmetres: la mida de l'Atrium Esquerre i de l'Atrium Dret.
L'objectiu és usar una base de dades de 81 participants (41 d'ells afectats i 40 no afectats) per a millorar la caracterització de l'afectació i de la prognosi i per a fer una detecció precoç per poder predir els efectes post-natals en base a les dades pre-natals disponibles.
La reduïda mida de la base de dades ha suposat un repte que ha impedit trobar noves caracteritzacions de l'afectació tot i que s'ha detectat certa correlació entre la VMG i altres dos paràmetres: la Banya Anterior Esquerra i la Banya Anterior Dreta.
S'ha aconseguit entrenar un model de classificació binari amb una valor de F1 al voltant del 90%. S'ha aconseguit amb xarxes neuronals o amb una cascada de xarxes neuronals amb els models d'aprenentatge supervisat més potents.
Es conclou que un nombre reduït de dades inclús desbalancejat permet oferir models amb suficients prestacions però que un major coneixement del context podria millorar els resultats.

Abstract (in English, 250 words or less):

Ventriculomegaly (VMG) is a rare disease that affects between 0.3 and 22 out of every 1000 pregnancies. 25-60% of these are isolated cases, which means that it has no other associated pathology. The evolution of the disease and the effect on neurodevelopment is not predictable.

The detection of VMG is done through the measurement of only two parameters: the size of the Left Atrium and the Right Atrium .
The aim is to use a database of 81 participants (41 affected and 40 unaffected) to improve characterisation of the condition and the prognosis and to provide an early detection to predict postnatal effects based on the available prenatal data.
The small size of the database has been a challenge to find new characterisations of the disease, although some correlation has been detected between VMG and two other parameters: the Left Anterior Horn and the Right Anterior Horn.
A binary classification model has been trained with a value of F1 of about 90%. This has been achieved applying neural networks or by cascading neural networks with the most powerful supervised learning models.
The conclusion is that a small number of even unbalanced data allows to provide models with sufficient performance but more knowledge on the context might improve the results.

Als meus pares, Maria Dolors i Josep Lluís, amb enyorança

Als meus germans, Elisenda Montserrat i Jordi, amb devoció

I al Martí i la Laia, als que estimo com d'aquí a la lluna infinit vegades... i tornar!

ÍNDEX

LLISTA DE FIGURES

1. INTRODUCCIÓ

1.1. CONTEXT I JUSTIFICACIÓ DEL TREBALL

La ventriculomegalia (VMG) es defineix com la dilatació d'un o dels dos ventricles laterals del cervell i és present entre 0,3 i 1,5 per cada 1000 embarassos, tot i que alguns autors i estudis ho eleven fins a 22 casos cada 1000 embarassos[1][4][6]. Es tracta, per tant, d'una malaltia rara.

La VMG apareix com a conseqüència d'una obstrucció a la circulació del líquid que es troba dins dels ventricles cerebrals, fent que s'acumuli. També pot estar associada a un problema en el desenvolupament del cervell, ja sigui a causa d'un problema aparegut al llarg de l'embaràs o bé present des del seu inici. El risc més important és el de la coexistència d'anomalies, que habitualment consisteixen en infeccions, malformacions dins i fora del cervell, anomalies cromosòmiques i síndromes genètiques[1].

Quan la VMG no es pot associar a una causa coneguda o a una altra patologia serà considerada VMG aïllada[1]. I en aquests casos la determinació del risc no és tant evident ja que no es pot relacionar amb les patologies o causes associades.

Actualment no existeix un tractament a aplicar durant l'embaràs per a millorar els casos de VMG aïllada, per això una bona caracterització i detecció precoç és important per ajudar a:

- Determinar els riscos als que pot desembocar.
- Millorar la informació a proporcionar a les families.
- Generar una finestra d'oportunitat per a trobar nous tractaments.

Una detecció precoç i una caracterització de la VMG també permet determinar quin és el seguiment mèdic més apropiat i establir el pronòstic, ja que existeixen casos amb pronòstics bons, moderats i dolents.

1.2. OBJECTIUS DEL TREBALL

La doctora Elisenda Eixarch junt amb el grup de recerca BCNatal Fetal Medicine Research Center (amb la participació de l'Hospital Clínic i de Sant Joan de Deu de Barcelona) disposa de diverses fonts d'informació amb dades pre i post-natals de fetus/nadons que han estat diagnosticats amb VMG aïllada.

En concret aquestes dades configuren un conjunt de fitxers excel en els que es recullen dades de 81 mostres (81 nadons): 41 casos positius d'afectació i 40 casos sense afectació que s'usen com a control. Els paràmetres són tant pre-natals com post-natals.

Els paràmetres pre-natals inclouen ecografies i ressonàncies magnètiques del cervell fetal en diversos moments de l'embaràs. Aquestes tècniques d'imatge ja han estat tractades i se n'han obtingut les mesures dels paràmetres més habituals.

Les dades post-natals es basen en els resultats de 3 tests:

1. Brazelton: La Brazelton Neonatal Assessment Scale (BNAS) va ser desenvolupada per T. Berry Brazelton per analitzar, en una prova de 30 minuts de durada, el comportament d'un nounat (fins als 2 mesos).[24] En el cas de la base de dades de la que es disposa, el temps promig en el que s'ha realitzat el test és als 35 dies des del naixement.
2. Escala Bayley: Escales Bayley de desenvolupament infantil (Les Bayley Scales of Infant Development, BSID) mesuren el desenvolupament mental i el comportament dels infants entre 1 i 42 mesos d'edat.[24] En el cas de la base de dades de la que es disposa el test ha estat efectuat als 19,5 mesos de promig.
3. Vineland: Les Vineland Adaptive Behavior Scales (VABS) estan designades per analitzar les capacitats autosuficients dels individus des del naixement fins a prop de l'edat adulta. Permet diagnosticar i avaluar discapacitats mentals. [24] En el grup d'estudi que conforma la base de dades es va efectuar entre els 18 i els 42 mesos de vida dels infants, amb un promig de poc més de 28 mesos.

Amb aquestes eines, el present treball es planteja 2 objectius principals:

- Trobar relacions i associacions entre els diversos paràmetres pre-natals per avaluar si algun d'ells es pot relacionar directament amb la VMG i millorar la seva caracterització.
- Caracterizar del pronòstic de la VMG en base als paràmetres provinents de les ressonàncies magnètiques fetals disponibles i tenint en compte els resultats dels tests post-natals (any i mig de vida aproximadament), per tal de predir-lo de forma precoç.

Per a complir amb els objectius d'anàlisi de les dades disponibles es preveu fer ús de mineria de dades i aprenentatge automàtic.

En concret, donat que es disposa de dades etiquetades, s'aplicaran tècniques d'aprenentatge automàtic supervisat. S'analitzaran quins algoritmes de regressió i classificació encaixen millor en les dades disponibles.

A banda de disposar de les fonts de dades que permeten efectuar el treball no es considera necessari disposar de recursos software addicionals als disponibles (Anaconda, Jupyther notebook/lab, google colab o Kaggle, RStudio, ... i llibreries gratuïtes) ni hardware addicional al disponible (MacBook Pro 2020, 2,3 GHz Intel Core i7 de 4 nuclis 32 GB 3733 MHz LPDDR4X, GPU Intel Iris Plus Graphics 1536 MB).

2. ESTAT DE L'ART

S'ha analitzat l'estat de l'art des de dos punts de vista:

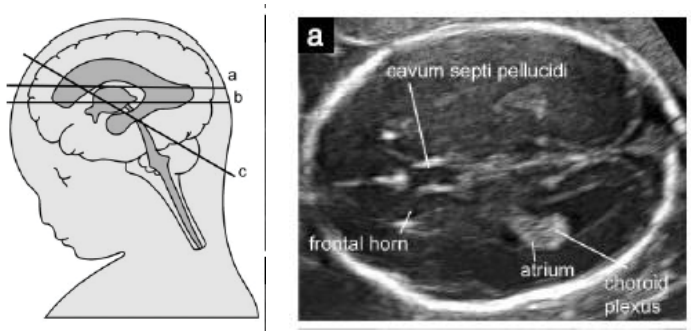
- Estat de l'art de l'estudi clínic de la VMG.
- Estat de l'art de les tècniques de caracterització i classificació. I, dins d'aquest àmbit, amb especial atenció a les tècniques emprades en casos clínics, i encara més específicament en casos de bases de dades de poques mostres. No es consideren les publicacions relacionades amb l'automatització del procés de mesura dels ventricles donat que els valors de tots els paràmetres han estat proporcionats.

2.1. ESTUDI CLÍNIC DE LA VENTRICULOMEGALIA

2.1.1. DEFINICIÓ I TIPOLOGIA

Existeix un acord en considerar que l'ample de l'atrium dels ventricles laterals del cervell dels fetus, visualitzat amb tècniques d'imatge en un pla axial ha de mesurar menys de 10 mm cadascun per poder descartar la VMG[1][2]. La Figura 1 il·lustra el pla axial i la ubicació de l'atrium:

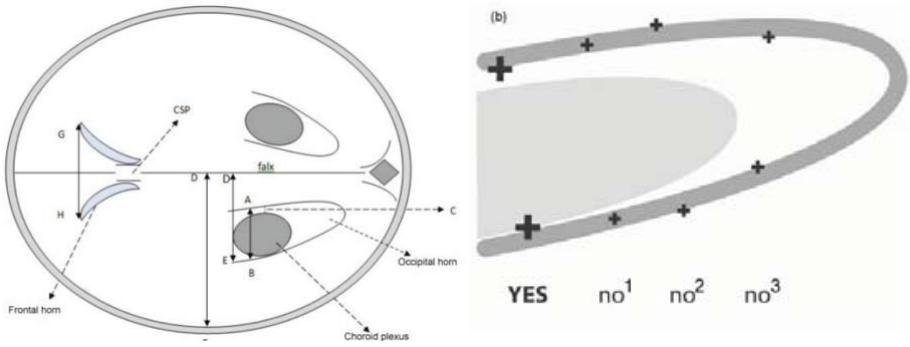
Figura 1 Pla axial (línia 'a' de la imatge de l'esquerra) i localització de l'atrium en un ultrasò (a la dreta).



Font: Nabila, M i altres[6]

Si els ventricles superen aquesta mida, es considera que existeix una VMG, que podrà ser unilateral o bilateral segons si és present en un o en els dos ventricles. La Figura 2 permet identificar de forma esquemàtica la forma correcta de mesurar les dimensions que han de servir per al diagnòstic de la VMG.

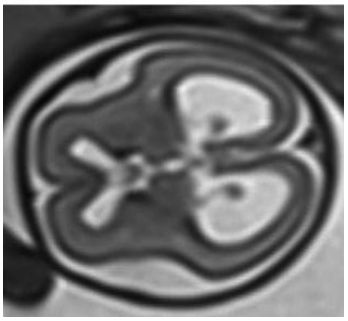
Figura 2 Localització de l'atrium en el pla axial, ubicació de l'ample (mesura AB de la imatge de l'esquerra) i forma correcta de mesura (a la imatge de la dreta).



Font: Nabila, M i altres[6]

Encara que no hi ha un autèntic consens i hi ha estudis que consideren VMG lleu superar els 10 mm i severa a partir de 15mm[11], la majoria dels estudis recomanen dividir els casos de VMG en tres nivells: consideren que la VMG es considera lleu si no supera els 12mm (10 a 12mm) i és moderada entre 12 i 15 mm (12,1 a 15 mm) [4][9]. A partir de 15 mm es considera que es tracta d'una ventriculomegalia severa (com la que es mostra en la Figura 3) i es parla d'hidrocefàlia.

Figura 3 Exemple de ventriculomegalia severa en una ressonància magnètica (MRI).



Font: Cardoen, L i altres[5]

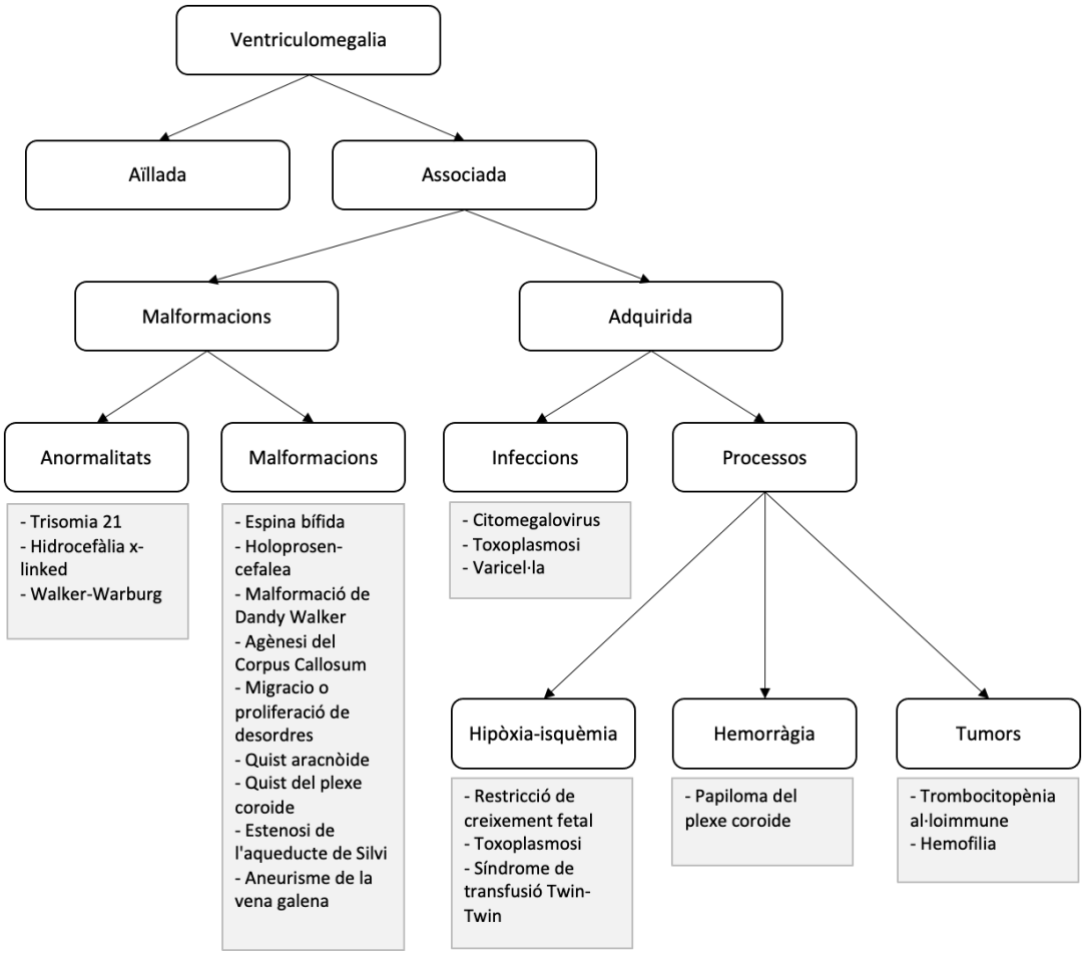
2.1.2. ETIOLOGIA

L'etiologia és la ciència centrada en l'estudi de la causalitat o els orígens d'una patologia o malaltia.

Habitualment la VMG està associada a altres patologies o bé és adquirida a causa d'algun incident durant la gestació (una infecció, una hemorràgia, un tumor o una hipoxico-isquèmia, per exemple). La Figura 4 ens ofereix una classificació esquemàtica de la VMG segons la seva etiologia en la que es pot observar que, a més de les associades, existeix, un percentatge de casos, en els que la VMG es considera aïllada perquè no està associada a altres patologies o a cap incident[1][3].

La variabilitat en la incidència de la VMG no està clarament determinada, ja que la seva prevalència està valorada en un rang d'entre 0,3 i 22 per 1000 de tots els embarassos segons els autors i els estudis[4][6] la qual cosa pot explicar la variabilitat del dimensionament de les VMG aïllades.

Figura 4 Classificació de la ventriculomegalia segons la seva etiologia.



Font: Adaptació de Hahner, Nadine[3]

Els percentatges d'incidència de les diferents etiologies també tenen molta variabilitat: no hi ha consens en el percentatge de VMG que tenen patologies prèvies o adquirides versus el percentatge de casos de VMG aïllada.

Els casos aïllats són entre un 25% i un 60%[7] segons els autors i els estudis.

2.1.3. PROGNOSI

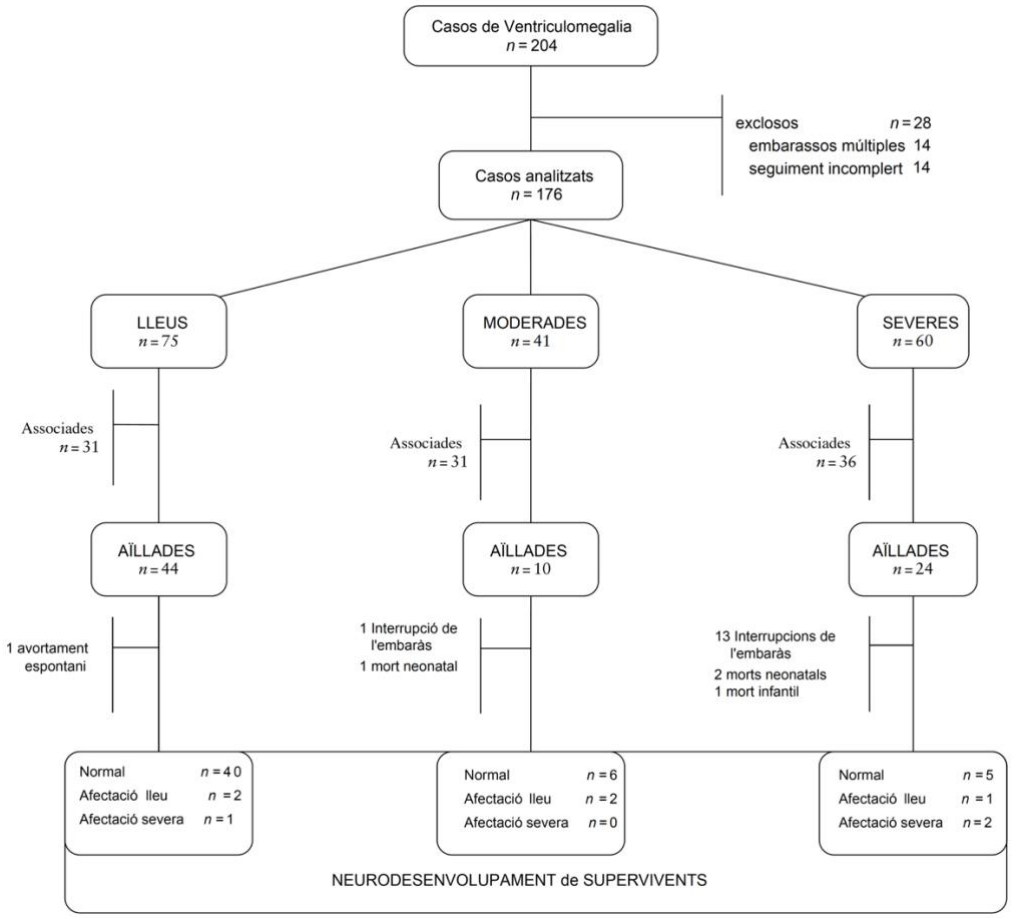
La prognosi és un terme d'origen grec que significa preveure. Per això s'usa aquest terme per referir-se en medicina a la previsió del progrés d'un pacient o l'evolució futura de la seva patologia.

La majoria d'estudis consultats consideren provat que quan una VMG apareix amb una patologia prèvia o associada, la seva prognosi està més relacionada amb aquesta patologia que amb la VMG en sí.[4][7][8][10] Per contra, en els casos de VMG aïllada la prognosi és més difícil de determinar, per això s'han localitzat molts d'estudis se centren en analitzar la prognosi específicament per a casos aïllats.[3][10][11]

Els estudis realitzats es concentren en analitzar la prognosi a efectes estadístics. Per exemple, l'estudi més ampli trobat, efectuat amb 176 casos [8] permet fer un resum força consensuat amb la resta de literatura consultada sobre la prognosi de la VMG.

En aquest estudi s'hi diferencien 3 grups segons l'ample dels ventricles: lleu (10 a 12mm en 75 casos), moderada (12,1 a 14,9 mm, en 41 casos) i severa (major o igual a 15 mm en 60 casos).

Figura 5 Resum de resultats d'un estudi amb 176 casos d'afectats de VMG.



Font: Adaptació de Gaglioti P i altres [8]

La VMG aïllada és més present en els casos lleus. En aquest cas hi ha un 58,7% de casos aïllats. En el grup de VMG moderada la presència de casos aïllats és d'un 24,4% i en el grup sever d'un 40%.

La supervivència més enllà de 24 mesos està directament associada a l'ample dels ventricles. És d'un 97,7% en els casos lleus, un 80% en els moderats i un 33,3% en els severs.

Pel que fa al neurodesenvolupament, no presenta anormalitat en el 93% dels casos lleus, en el 75% dels moderats i en el 62,5% dels severs.

Aquest 7% de casos lleus que sí que presenten neurodesenvolupaments anormals, són molt similars als resultats obtinguts en altres estudis[9][11] on s'han obtingut ràtios del 10-11%. Aquesta ràtio no difereix substancialment de la població general.

Es poden observar aquests resultats en l'esquema de la Figura 5 que es presenta de forma adjunta.

2.1.4. ESTAT DE L'ART CLÍNIC

Atenent a l'anàlisi efectuat en els apartats previs es pot concloure que existeixen múltiples publicacions que analitzen l'etiologia i la prognosi de la VMG.

Fins i tot hi ha bibliografia nombrosa relacionada amb estudis de l'afectació amb el focus en alguna de les seves particularitats: el seu nivell (lleu, moderada o greu) o el seu origen (aïllada o no) o fins i tot la seva tipologia (unilateral o bilateral).

Pel que fa a les VMG aïllades hi ha consens en establir que, les que són lleus i estables no presenten una prognosi diferent que en la població general.

Així mateix, existeixen múltiples publicacions que analitzen com han evolucionat i quines problemàtiques han presentat els nadons segons el seu tipus de VMG i la seva causa.

El sexe i la unilateralitat/bilateralitat no són paràmetres concloents.[11] Hi ha consens aparent en considerar que la VMG és més present en sexe masculí, però no en el pronòstic, ja que alguns estudis conclouen que és pitjor en el sexe femení[12] mentre que altres no han trobat diferències significatives [11].

No s'ha trobat cap treball relacionat amb la caracterització de les VMG aïllades, per a que puguin ser detectada de forma precoç la seva evolució. Ni amb analitzar altres paràmetres que permetin determinar a priori quins casos tindran una evolució favorable o no, més enllà de la mida de l'ample dels ventricles, la seva estabilitat i el fet de ser o no aïlades.

És a dir, els estudis trobats s'han limitat a analitzar quina és l'etiologia o la prognosi de la VMG. En alguns estudis es concentren en casos aïllats o en casos greus. Però sempre analitzant els resultats, en percentatges de tipologies diferents. Però sense analitzar l'origen o la causa de la diferent evolució.

2.2. TÈCNIQUES DE CARACTERITZACIÓ I CLASSIFICACIÓ

A l'hora d'analitzar l'estat de l'art en l'àmbit tecnològic, partim de l'objectiu que tenim previst aconseguir: identificar paràmetres que permetin a priori classificar els diagnòstics de VMG aïllada entre aquells que tindran o no un neurodesenvolupament normal.

Per tant, ens trobem davant de la necessitat de disposar d'eines de caracterització i regressió (si es pot predir el valor continu dels resultats dels tests) o de classificació (si es vol predir el resultat del test per categories), de la que existeix abundant bibliografia. En el cas que ens ocupa, resulta d'especial interès aquella relacionada amb aprenentatge automàtic (machine learning en anglès) aplicat a casos clínics o a jocs de dades petits.

No s'analitza en aquest treball la bibliografía (també força àmplia) relacionada amb l'automatització de les ressonàncies magnètiques (MRI) que són les que ens permeten disposar dels paràmetres que s'usaran en l'estudi. És molta i variada la bibliografia relacionada amb l'automatització per a l'obtenció dels valors o, fins i tot, aquella que permet diagnosticar la VMG directament amb l'anàlisi de les MRI o altres tècniques

d'imatge.

2.2.1. APRENENTATGE AUTOMÀTIC

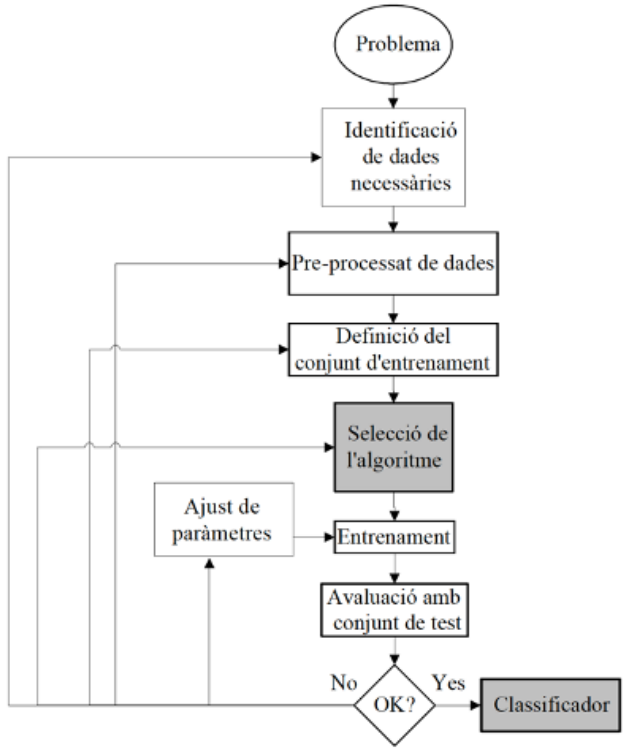
Ens trobem en un cas d'aprenentatge automàtic inductiu, és a dir, un procés en el que volem detectar les regles que operen entre els diversos paràmetres dels que disposem per crear un classificador[13].

El problema radica en localitzar un algoritme que permeti extreure regles amb poques mostres i molts atributs com tenim en el nostre joc de dades.

Podem resumir les diferents tècniques disponibles en la llista següent: Classificadors lineals, regressió logística, Classificador Naïve Bayes, Perceptró, Maquina de Vector de Suport (o SVM, de l'anglès Support Vector Machine), Anàlisi de Components Principals (PCA de Principal Components Analysis), Classificadors quadràtics, K veïns més propers (o K-NN, de K-Nearest Neighbourgs), Agrupaments (en anglès Clustering), Potenciació (Boosting), Arbres de decisió (Decision Trees), Boscos aleatoris (o Random Forest), etc. A més de xarxes neuronals o xarxes bayesianes, per exemple.

El procés habitual a seguir seria el que mostra la Figura 6.

Figura 6 Procés de l'aprenentatge automàtic supervisat emprat.



Font: Adaptació de S. B. Kotsiantis[13]

La dificultat en seleccionar el millor algoritme radica en el fet que no es pot deduir com de bo serà un algoritme pels resultats obtinguts en aplicar-lo en un altre joc de dades[14]. És a dir, la precisió o exactitud que ofereix un algoritme en un joc de dades no té perquè mantenir-se en un altre tipus de joc de dades.

Podem observar, per a cadascun dels principals algoritmes, quines prestacions ofereix i quines limitacions té. De forma molt simplificada, es poden resumir les següents, basant-nos en diversos dels estudis i literatura existents que han analitzat les seves avantatges i inconvenients[13][14][15][16]:

2.2.1.1. XARXES BAYESIANES

Usar xarxes bayesianes no és una bona alternativa quan no es disposa d'un coneixement profuns dels paràmetres disponibles[14] (ja que dóna la possibilitat de tenir en compte coneixements previs per a definir l'estructura de relació entre paràmetres) i incrementa notablement la seva complexitat quan hi ha una gran dimensionalitat.[15]

Les xarxes Bayesianes Naive, encara que siguin més simples i resolguin el problema de la complexitat, no ofereixen millor rendiment que altres més habituals com els arbres de decisió.[14] A més, necessiten que les etiquetes disponibles de classificació siguin molt independents.[15]

2.2.1.2. ARBRES DE DECISIÓ

Aquest sistema de classificació és molt simple i ràpid d'entrenar. A més, no requereix cap mena de coneixement previ sobre els paràmetres que s'analitzen i no té problema amb jocs de dades d'alta dimensionalitat.[16] Ofereixen un resultat molt fàcil d'interpretar i no tenen problema amb els valors fora de rang (outliers en anglès).[15]

La facilitat que ofereix per interpretar el mecanisme de classificació fa que sigui molt usat per entitats bancàries o asseguradores per prendre decisions empresarials, però també en casos relacionats amb la salut.[27]

En contrapartida, poden oferir models poc acurats si no es disposa d'un nombre de mostres suficientment representatiu (en relació amb el nombre de paràmetres). [15]

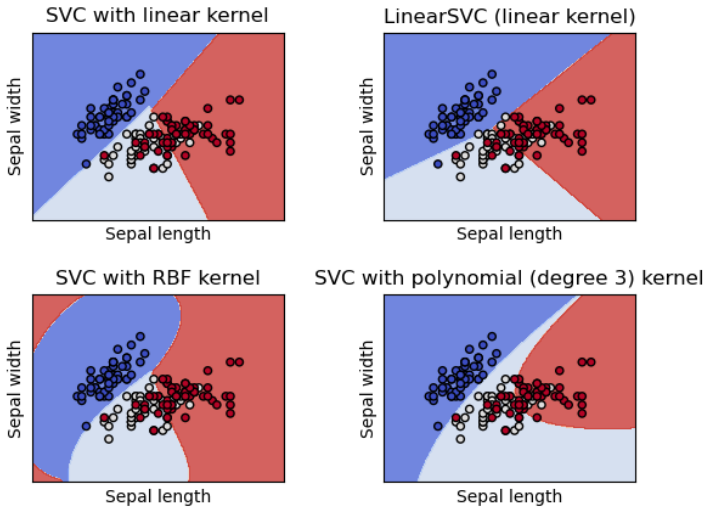
2.2.1.3. MÀQUINES DE VECTOR DE SUPORT (SVM)

Aquests tipus d'algoritmes ofereixen unes de les millors prestacions en quant a exactitud tot i que tenen una taxa d'aprenentatge menor i una major dificultat per interpretar les dades[13]. Junt amb les xarxes neuronals, són les que millor tracten jocs de dades amb una alta dimensionalitat de paràmetres. [14]

És un dels models més potents per a reconèixer patrons i per això és extensament usat en els casos més complexos, com els reconeixements de textos o les series temporals o en camps com la sequenciació de proteïnes i el diagnòstic de diversos tipus de càncer. [17][27][28] Aquest tipus d'algoritme pot ser tant lineal com no lineal en funció del tipus de kernel que s'utilitzi. En la imatge de la Figura 7 adjunta podem observar-hi la diferència en el resultat del model de classificació aplicat en el joc de dades 'iris', segons el kernel usat.

Davant d'aquesta diferència de resultats, s'opta per considerar aquests tipus d'algoritmes com a models diferents i no considerar el tipus de kernel com un paràmetre més quan es configuri i entreni.

Figura 7 Exemples de diferents kernels usats en SVM aplicats al dataset 'iris'.



Font: Pedregosa F i altres[25]

2.2.1.4. K VEÏNS MÉS PROPERS (K-NN)

Aquest tipus d'algoritmes són molt sensibles quan es tenen en consideració paràmetres irrelevants. [14] I són molt poc tolerants davant de valors inexistents o soroll [13], sobretot en casos on hi ha un nombre reduït de mostres[16]. Com a contrapartida poden treballar de forma eficient (en termes de computació) si es tenen poques dades sobretot en casos on la 'k' està predefinida[15].

2.2.1.5. REGRESSIÓ LOGÍSTICA

Alguns algoritmes de regressió poden ser usats per a resoldre problemes de classificació i viceversa[26]. La regressió logística és un model de regressió que té com a resultat un valor discret o una categoria (o la probabilitat de pertànyer a una o altra categoria), per tant, és perfectament aplicable com a model classificador. De fet, aquesta forma de classificar usant una probabilitat la converteix en una de les eines més comuns en l'analisi de dades discretes.[14]

La regressió logística té en comú amb les xarxes neuronals l'ús del descens del gradient com a mecanisme d'optimització.

2.2.1.6. COMBINACIÓ DE MODELS

L'estat de l'art en la mineria de dades i l'aprenentatge automàtic treballa en dues vies paral·leles. Per una banda, es treballa per a desenvolupar nous algoritmes i, per l'altra, intenta la millora del rendiment dels algoritmes ja existents.[27]

En aquest darrer àmbit, una de les noves tècniques per a optimitzar els resultats dels algoritmes preexistents, consisteix en l'agrupació o combinació de classificadors. Algunes combinacions són seqüencials i altres són paral·leles. Es considera l'ús de les tècniques de ensacat (bagging en anglès), impuls (boosting en anglès) , apilament (stacking en anglès) i cascada (cascading en anglès).

L'ensacat (bagging) consisteix en la generació de múltiples conjunts de dades similars per a poder aplicar-hi un classificador a cada conjunt. Finalment la decisió de la classificació es pren per majoria combinant els resultats de totes les classificacions parcials. Una aplicació habitual d'aquesta tècnica és la que representa la configuració de boscos aleatoris (random forests en anglès).[27]

L'impuls (boosting) és similar, però enlloc de crear de forma paral·lela els classificadors per cada conjunt de dades, el que es fa és generar un primer classificador i s'utilitza el resultat del primer entrenament i els errors comesos per a definir un nou classificador que posi ènfasi en els errors de classificació del primer. I així de forma successiva. [27]

L'apilament (stacking) consisteix en entrenar el joc de dades amb diversos classificadors, de manera que es construeix un nou classificador amb les prediccions obtingudes enlloc de amb les dades originals.[27][29]

La cascada (cascading) és similar a l'apilament, però enlloc d'usar només les prediccions dels models individuals per a entrenar el nou model, es fan servir també les dades originals.[27][29]

2.2.2. SELECCIÓ DE TÈCNIQUES

A la vista de la revisió de l'estat de l'art i de les tècniques disponibles es considera adequat testejar totes les tècniques de classificació que s'han analitzat i comentat, excepte les xarxes bayesianes que no s'han considerat adequades segons l'estat de l'art i la bibliografia consultada.

S'hi inclourà també el bosc aleatori com a model de combinació tipus bagging i s'aplicaran també els altres mecanismes de combinació sobre els millors models individuals per trobar models més ajustats.

S'analitzarà també quin rendiment poden aportar les xarxes neuronals. En casos on el nombre de mostres és reduït l'ús de les xarxes neuronals té molta aplicació quan es poden aprofitar models preentrenats. No existeixen jocs de dades pre-entrenats que puguem usar, però tot i així s'analitzarà la seva possible aplicació en aquest cas.

Per valorar els diferents models o combinacions de models s'aplicarà una metodologia que consisteix en usar matriu de confusió, també coneguda com taula de contingència o matriu d'errors.[27] D'aquesta manera es podrà comparar de forma gràfica la seva capacitat predictiva.

2.2.3. JOCS DE DADES REDUÏDES

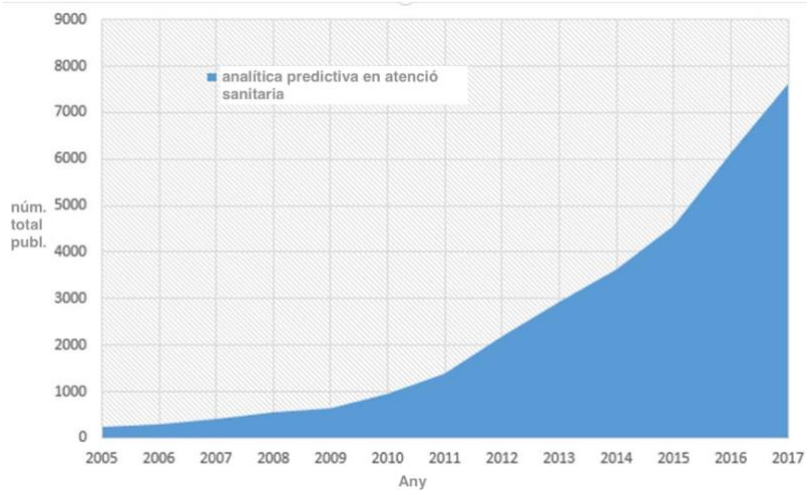
En el cas que ens ocupa, no només cal tenir en consideració el millor algoritme segons el nombre de paràmetres sinó tenint en compte que es tracta d'una base de dades mèdica d'una malaltia rara amb un baix número de casos, però molt ben caracteritzats clínicament.

S'ha analitzat, per tant, bibliografia relacionada amb tècniques d'aprenentatge

automàtic aplicades a bases de dades petites o a aplicacions mèdiques.

Com es pot observar en la Figura 8 adjunta, les tècniques d'aprenentatge automàtic per a fer anàlisi predictiu en termes de salut han estat cada vegada més presents al llarg del temps[17]. Encara que la gràfica finalitza el 2017 la tendència és clara.

Figura 8 Evolució del nombre de publicacions relacionades amb Analítica Predictiva en atenció sanitaria.



Font: Adaptació de M. A. Sarwar i altres[17]

D'alguns d'aquests estudis se n'ha extret les conclusions següents:

- L'exactitud (accuracy en anglès) com a mètrica.

L'eina principal per a comparar i analitzar sistemes de classificació és l'exactitud (accuracy) o la corva ROC (acrònim en anglès de Característica Operativa del Receptor) amb la seva mètrica AUC-ROC (Area Sota la Corvar de l'anglès Area Under the Curve ROC).[17][19][21]

- Representativitat de la mostra

En jocs de dades petits (< 2000 mostres) és difícil obtenir resultats superiors al 75-80% [17] [19] ja que cal partir de la hipòtesis de que tots els valors possibles estan balancejats en la mostra i això no sempre és cert.[18]

- Grandària de la mostra

El principal problema és el nombre de mostres i l'existència de valors perduts.[17] Per això alguns estudis apliquen mètodes de generació virtual de mostres o VSG (de l'anglès Virtual Sample Generation) per minimitzar aquests efectes[18].

S'han arribat a obtenir valors d'exactitud molt bons, amb menys de 50 mostres però en casos on s'ha pogut aplicar xarxes neuronals i s'han pogut congelar capes i aplicar la tècnica de tranferència d'aprenentatge (transfer learning en anglès): aprofitant el coneixement obtingut amb xarxes pre-entrenades amb altres jocs de dades amb moltes més mostres[19][22]. Aquest procés no precisa que s'hagi entrenat una xarxa amb casos idèntics, ni tan sols molt similars, però sí relacionats[20].

- El preprocessat i selecció de paràmetres és clau

Es refereix al procés de selecció de paràmetres, de detecció de dades perdudes o errònies, d'identificació de valors fora de rang, etc. És a dir, el processat previ de les dades és clau per millorar els resultats. [20][21]

- Algoritmes adeqüats

Els millors resultats, quan no es disposa de xarxes neuronals preentrenades s'obtenen amb mecanismes com el K-NN i el SVM[17] tot i recordar que, amb dades diferents s'obtenen resultats diferents[14].

3. SELECCIÓ DE DADES I PARÀMETRES

3.1. DADES DISPONIBLES

El projecte parteix de recepció de 7 fitxers en format excel. El nom dels fitxers i el seu contingut es descriu a continuació:

- Postpartal data base.xlsx: dades incomplertes de 81 nadons. Amb 168 paràmetres, principalment relacionats amb dades del nadó (identificador, sexe, pes al néixer, etc.) i els resultats dels tests Brazelton i Bayley. Tots els valors parcials de cadascuna de les valoracions, tot i que només s'ha efectuat el test Brazelton a 62 dels nadons i el test Bayley a 42 nadons. Tots els valors són numèrics o categòrics, excepte els camps destinats a les dates (data de naixement, data prevista de naixement, data de realització dels tests). Hi ha 90 paràmetres que detallen els diferents aspectes avaluats en el test Brazelton, mentre que hi ha 67 paràmetres que detallen els diversos aspectes avaluats en el test de Bayley.
- Bayley-Vineland.xls: només 3 columnes de dades. Les dues primeres indiquen, amb un 1 o un 0 si la mostra ha obtingut resultat anormal en el test Bayley o el test Vineland. La tercera indica si un dels dos ha donat resultat anormal. Destaca que és la única dada del resultat del test Vineland de la que es disposa i que només es té el resultat corresponent a 18 subjectes de l'estudi.
- Data_Eco_Brazelton_ASQ.xls: dades dels nadons, dividits per grups segons el moment en què s'han incorporat al projecte. Dades de les ecografies a la setmana 26 i 30. No tots els subjectes tenen dades en les dues ecografies. Alguns només tenen dades de la setmana 26, altres només de la setmana 30, altres de cap i en alguns casos hi ha registres de les dues proves. També inclou dades resumides del test Brazelton. En total 155 paràmetres diferents.
- INSVM_All_noNDdata.xls: la taula més complerta, amb 250 paràmetres. El nom del fitxer prové de Insulated Non Severe VentriculoMegalia (ventriculomegalia aïllada no severa). És la informació que es va fer servir per dur a terme una Tesi per la vessant clínica[3] al respecte. Inclou dades dels tests Brazelton i Bayley i de les ecografies (setmanes 26 i 30).
- Data Base Eco.xlsx: dades de les ecografies a les 26 i 30 setmanes. Alguns paràmetres tenen un valor nominal i altres paràmetres només tenen una graduació (grading en

anglès) que és una categorització en 5 nivells (del 0 al 4).

- Data base fetal MRI.xlsx: aquest fitxer inclou el diagnostic de VMG (indicant si existeix i si és lateral dret, esquerre o bilateral), el sexe del nadó i les dades d'una ressonància magnètica o MRI (Magnetic Resonance Imaging). La ecografia és la prova estàndard de seguiment mèdic dels embarassos i és on es detecta i es fa el seguiment de la VMG. Però en aquest cas disposem també de les dades de ressonàncies com a part de la Tesi[3] mencionada anteriorment. En el marc d'aquell estudi es va efectuar una ressonància magnètica a gairebé totes les participants. Per aquesta raó es considera important aquest fitxer i la informació que aporta ja que ens permet disposar de registres de dades pre-natals de tots els subjectes d'estudi, excepte un cas.
- Maternal data base.xlsx: informació relacionada amb les gestants. Identificadors, dades mèdiques i socioeconòmiques.

3.2. SELECCIÓ DE PARÀMETRES

De totes les dades recollides i tenint en compte que tenim un nombre molt reduït de mostres i els objectius que es pretenen assolir, s'opta per escollir els paràmetres de treball en base als següents criteris:

- utilitzar paràmetres que tinguin pocs valors perduts: donat que no es disposa de coneixements mèdics que permetrien aplicar mètodes de recuperació d'aquests valors perduts (usant una mitjana o mediana absoluta o dependent d'altres paràmetres). Per això es descarten les dades ecogràfiques i em concrentro en valors de la MRI.
- totes les dades post-natals: malgrat que s'ha indicat que només es disposa de 18 resultats de Vineland, de 42 resultats Bayley i de 62 resultats Brazelton s'utilitzen i, quan correspongui, es considerarà que no tenir dades representa un resultat de normalitat.
- utilitzar valors numèrics i no categòrics: sobretot per als resultats d'ecografies i ressonàncies, es descarten les graduacions (gradings) i s'usen els valors mesurats en mil·límetres.

Per tant, els paràmetres finalment escollits es configuren en un fitxer excel, anomenat 'bbdd_TFM.xlsx'. Els paràmetres es recullen en l'annex ANNEX 1: LLISTAT DE PARÀMETRES SELECCIONATS I DESCRIPCIÓ.

Considero que cal mencionar de forma especial els paràmetres LA (Left Atrium) i RA (Right Atrium). Donat que aquests paràmetres són els que determinen la presència o no de VMG, decideixo recopilar totes les mesures efectuades d'aquests dos paràmetres al llarg de les diferents proves practicades i escollir els valors majors.

Per tant, finalment, parteixo d'un joc de dades de 81 mostres amb 38 paràmetres.

3.3. PREPARACIÓ DE DADES

En l'annex ANNEX 2: CODI PYTHON: PREPARACIÓ DE DADES s'hi recull el procés de preparació de les dades fins obtenir una trama de dades (dataframe en anglès), anomenat data, amb el que es treballarà.

Els canvis efectuats han estat molt bàsics:

- S'ha recodificat l'identificador com un objecte.
- S'ha creat una columna nova de Sexe codificant en '1' i '0': el valor '0' correspon al sexe masculí mentre que el valor '1' correspon al sexe femení.
- S'ha creat un camp addicional per a un valor categòric del diagnòstic. S'ha assignat un valor de 0 a 3 per cadascun dels 4 diagnòstics possibles:
    - '0' per a 'No VMG'.
    - '1' per a 'VMG lateral esquerra'.
    - '2' per a 'VMG lateral dreta'.
    - '3' per a 'VMG bilateral'.

Així s'obté una trama de dades definitiva de 40 paràmetres.

Un cop definit i determinat el joc de dades es poden efectuar les primeres visualitzacions i anàlisi exploratòria de dades. Estan recollides en l'annex mencionat però no se n'ha després cap informació no evident.

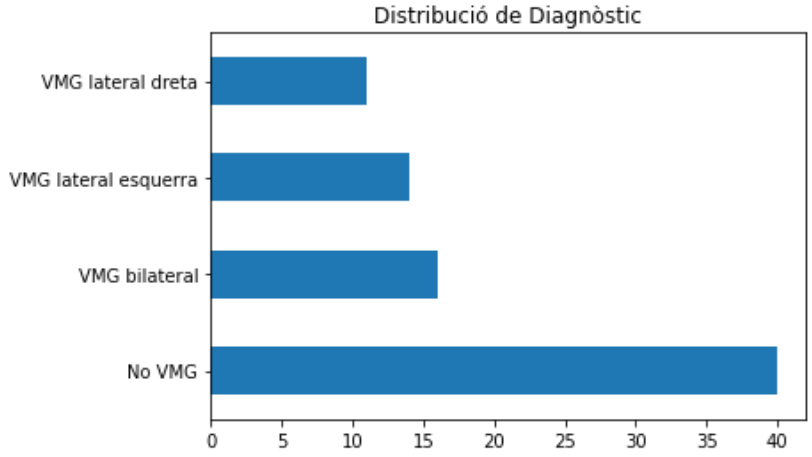
S'observa, en la Figura 9, que tenim moltes més mostres de nens que de nenes i que hi ha major taxa de VMG bilateral en nenes que en nens.

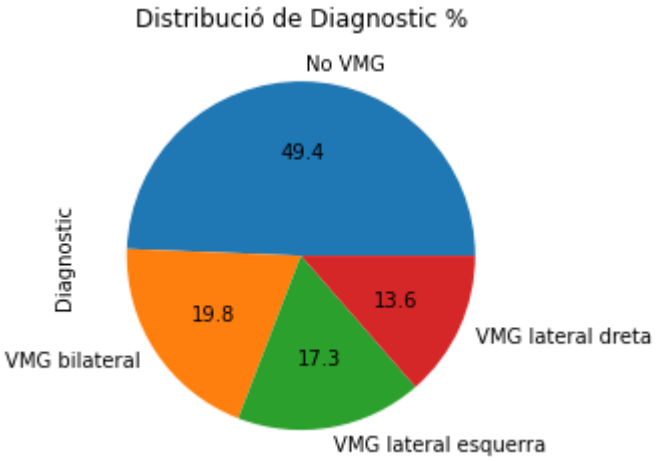
Malgrat això, per la dimensió de la mostra i pel fet de ser casos seleccionats per un estudi, no ens permet confirmar que sigui una patologia més present en nens que en nenes ni que els efectes en nenes siguin més greus com afirmen alguns estudis[11][12].

Podem confirmar, però, que la mostra està equilibrada pel que fa als casos d'afectació (41 casos) i casos de no afectació (40 casos).

Figura 9 Distribució de les mostres segons diagnòstic i distribució del diagnòstic per sexe.

Diagnostic	No VMG	VMG bilateral	VMG lateral dreta	VMG lateral esquerra	All
Sexe					
Fem	14	5	1	1	21
Masc	26	11	10	13	60
All	40	16	11	14	81





Font: elaboració pròpia

4. CARACTERITZACIÓ DE LA VENTRICULOMEGALIA I DE LA SEVA PROGNOSI

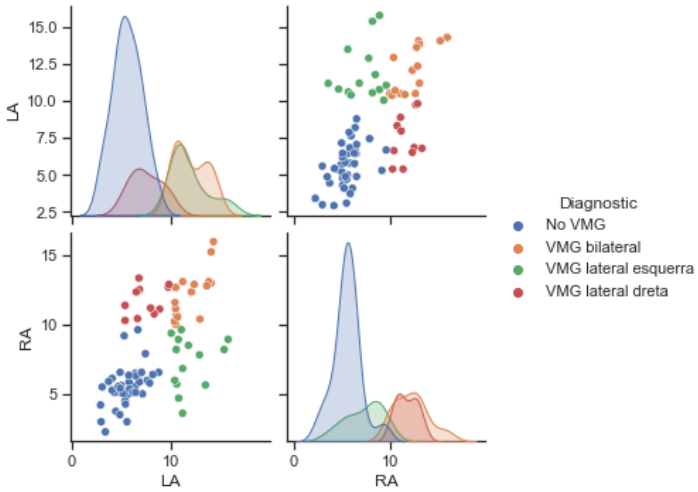
El primer dels objectius del treball era la caracterització de la VMG i dels seus efectes. En l'annex ANNEX 3: CODI PYTHON: ANÀLISI EXPLORATÒRIA DE DADES s'hi recull el contingut del notebook amb el codi en python utilitzat en el procés d'exploració qualitativa i quantitativa de les dades i el procés seguit per localitzar paràmetres o conjunts de paràmetres que permetin caracteritzar tant la patologia (la VMG) com els seus efectes (la prognosi).

És a dir, intentar caracteritzar amb les dades pre-natals el diagnòstic i els resultats de les proves i tests que es faran post-natals.

4.1. CARACTERITZACIÓ DE LA VENTRICULOMEGALIA

La VMG és present quan l'amplada d'un o dels dos atriums en el pla axial supera els 10mm. Aquests paràmetres, per tant, és acceptat per la comunitat mèdica que determinen la VMG[1][2].

Figura 10 Pairplot dels valors de LA i RA diferenciats per diagnòstic.



Font: elaboració pròpia

I així es comprova amb les dades que tenim. Podem veure en la Figura 10 com la combinació dels paràmetres LA i RA generen 4 conjunts disjunts de dades per diagnòstic.

Per tal d'intentar localitzar altres paràmetres que puguin servir també per a

caracteritzar la VMG i, per tant, per a facilitar-ne un millor coneixement procedim amb una sèrie d'anàlisis, principalment gràfiques consistents en:

- relació de paràmetres del joc de dades i resum de valors: per tal de detectar valors perduts i confirmar el nombre de registres i de paràmetres.
- histogrames de cadascun dels paràmetres: per identificar valors fora de rang (outliers en anglès) i observar si existeixen patrons.
- correlacions entre paràmetres per identificar els que estan relacionats.

De les dues primeres proves no se n'obté cap resultat a destacar, més enllà de confirmar que tenim 81 registres i 40 columnes de dades. Que hi ha 7 paràmetres que no tenen valors buits (l'identificador, les dues columnes que recullen el valor del sexe, les dues columnes del diagnòstic i els valors de LA i de RA).

Tots els paràmetres relacionats amb mesures internes del cervell tenen només 1 valor perdut, excepte dos paràmetres que tenen dos valors perduts.

Els resultats dels tests, com ja s'ha indicat anteriorment tenen molts valors perduts. De fet, es confirma que tenim 25 resultats del test Vineland, 42 resultats del test Bayley i 64 resultats del test Brazelton.

Pel que fa als histogrames, no s'han identificat patrons i no s'han identificat valors fora de rang.

4.1.1. ANÀLISI DE LA CORRELACIÓ

El resultat de l'anàlisi de la correlació sí que ha donat algún resultat interessant.

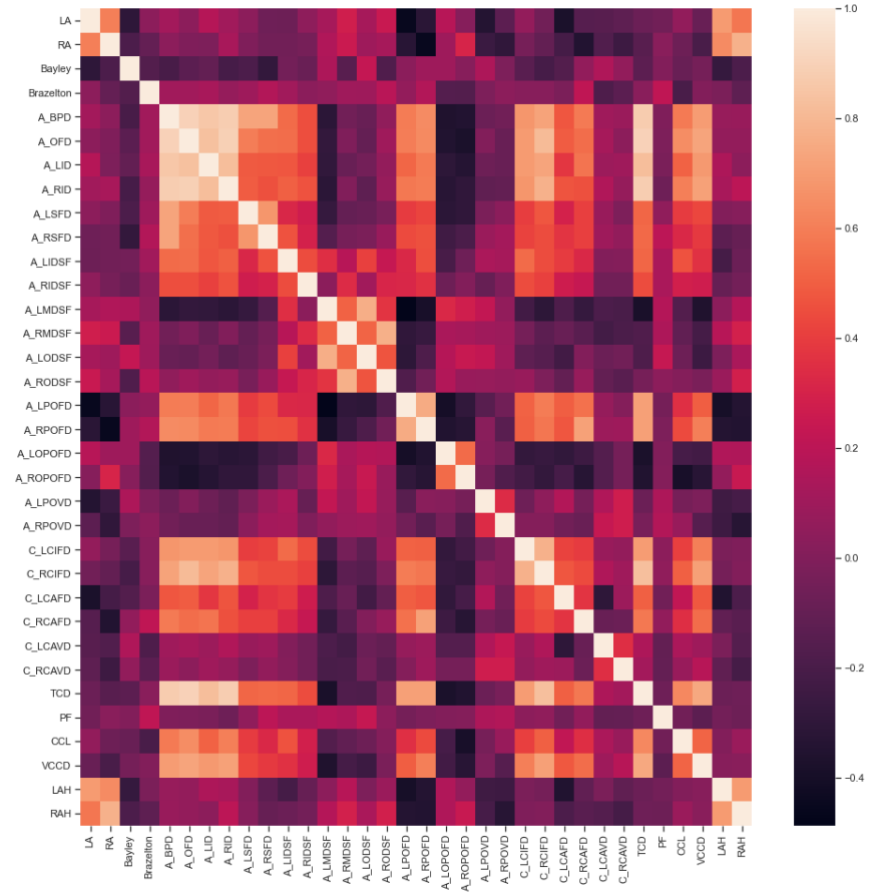
S'ha aplicat la correlació a tot el joc de dades però naturalment només ha donat resultat per als atributs que són numèrics. Per tant, obtenim correlacions creuades de 34 dels 40 paràmetres.

S'ha optat per usar un mapa de calor perquè l'anàlisi numèrica hauria estat més complexa. El resultat es pot observar en la Figura 11. Tal com mostra la llegenda, el valor de correlació 1 és el color més clar.

L'observació del mapa de calor permet concloure que:

- Els paràmetres que serveixen per diagnosticar la VMG, els paràmetres Atrium Esquerre i Atrium Dret (LA i RA respectivament) estan molt poc correlats amb els resultats dels tests Bayley i Brazelton.
- En canvi LA i RA estan molt correlats amb els paràmetres identificats com LAH i RAH que es corresponen a Banya Anterior Esquerra (Left Anterior Horn) i Banya Anterior Dreta (Right Anterior Horn).
- LA i RA no estan correlats amb cap altre paràmetre.
- Els resultats dels tests no estan correlats amb cap paràmetre.

Figura 11 Mapa de calor de la correlació creuada entre els paràmetres numèrics del conjunt de dades.



Font: elaboració pròpia

D'aquestes observacions se'n desprèn que la caracterització del diagnòstic no està directament relacionada amb la caracterització de la prognosi.

Cal analitzar la relació de LAH i RAH amb la caracterització de la VMG, ja que sembla que podria ser un indicador que ha de permetre diagnosticar-la.

Per seguir intentant trobar relacions que permetin caracteritzar la VMG s'efectuen anàlisis pairplot dels paràmetres identificant el diagnòstic.

Aquests pairplots es fan de forma separada per grups de paràmetres. Es fan tres grups:

- Paràmetres del pla axial: tots els paràmetres que es corresponen a mesures efectuades en la MRI en un pla axial.
- Paràmetres del pla coronal: tots els paràmetres que es corresponen a mesures efectuades en la MRI en un pla coronal.
- Paràmetres generals: els que no es mesuren específicament en el pla axial o coronal i no són diagnòstics.

L'anàlisi d'aquests pairplots es fa visualment. Els resultats d'aquests pairplots per paràmetres, disponibles a l'annex ANNEX 3: CODI PYTHON: ANÀLISI EXPLORATÒRIA DE DADES, no ofereixen resultats a destacar.

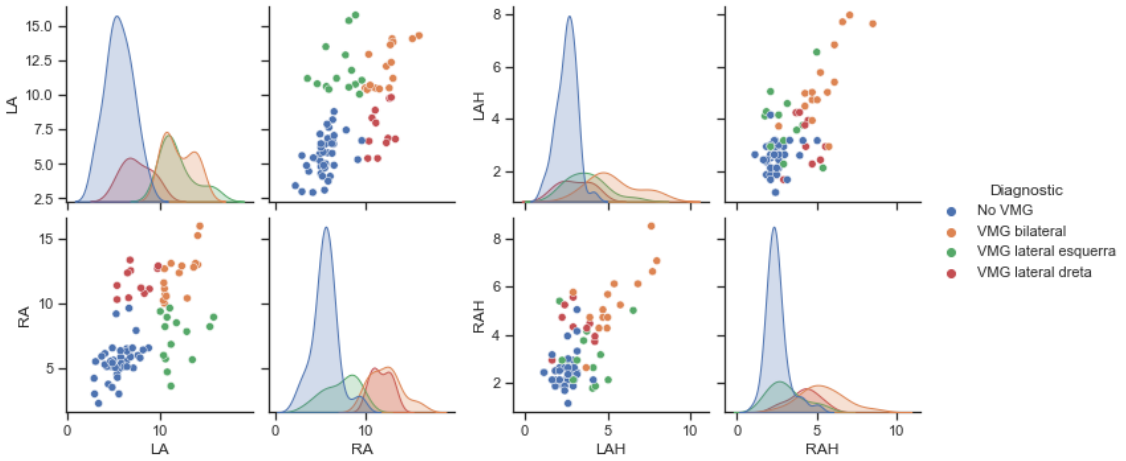
Els valors dels paràmetres que formen part de cada grup són molt similars i, per tant, l'ordre de magnitud dels valors no hauria de tenir efecte, però de tota manera es comprova repetint els gràfics amb els valors normalitzats, sense que apareguin canvis. Per a la normalització s'utilizen llibreries de prepocessat de sklearn (scikit-learn)[25].

4.1.2. ANÀLISI DE LES MESURES DE LA BANYA ANTERIOR

Donat que no s'ha observat cap altre element significatiu, s'estudia la possible relació dels paràmetres LAH i RAH amb la VMG.

S'ha observat que aquests dos paràmetres tenen una relació amb el diagnòstic molt similar a la que tenen el LA i el RA.

Figura 12 Comparació dels valors dels paràmetres LA+RA i LAH+RAH en relació amb el diagnòstic.



Font: elaboració pròpia

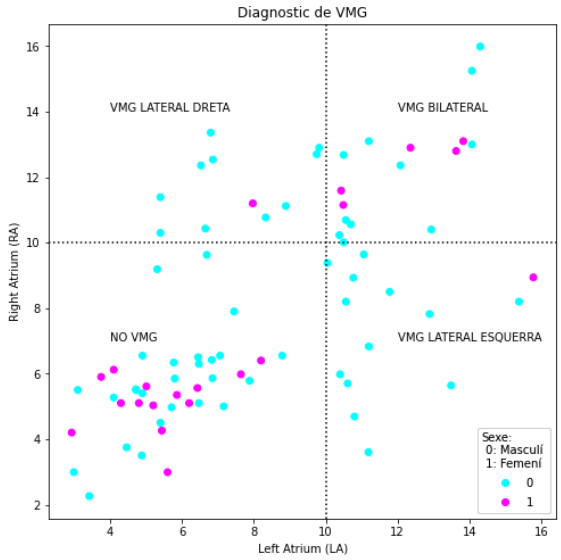
Per comparar-ho es presenten els dos pairplots junts en la Figura 12: el que relaciona LA i RA entre si diferenciant per diagnòstic (a l'esquerra) i el que fa el mateix però amb LAH i RAH (a la dreta). S'observa que amb LAH i RAH no s'obté una caracterització que ofereixi una separació entre diganòstics tant clara com que la ofereix amb LA i RA.

4.1.3. ANÀLISI DE LA RELACIÓ AMB EL SEXE

Donat que en la bibliografia consultada hi ha cert debat sobre la relació del sexe amb la VMG es considera adeqüat fer una anàlisi específica de l'afectació en relació amb el sexe. S'observa que, efectivament, la presència del sexe masculí en els casos amb la patologia són molt més nombroses, també és cert que dins del grup femení hi ha major rati de VMG bilateral.

Aquestes observacions s'extreuen de la gràfica de la Figura 13 en la que, per cada mostra, es pot determinar el diagnòstic, el valor de LA i RA i el sexe. No se n'extreu més informació d'interès.

Figura 13 Caracterització del diagnòstic en relació amb els valors dels paràmetres LA i RA i del sexe.



Font: elaboració pròpia

4.2. CARACTERITZACIÓ DE LA PROGNOSI

La prognosi és a la previsió del progrés d'un pacient o l'evolució futura de la seva patologia. En el cas que ens ocupa, els pacients que pateixen aquesta afectació poden presentar problemes de neurodesenvolupament. Cal recordar que aquest treball es centra només en casos de VMG aïllada, és a dir, no associada a altres patologies[1].

Donades les característiques de la base de dades disponible, la prognosi queda definida fonamentalment en els test de Bayley i Brazelton. No podem fer cap anàlisi amb les dades del test Vineland perquè tenim molt pocs resultats i els que tenim són binaris ('1' per indicar valor anormal i '0' per indicar valor normal).

4.2.1. RELACIÓ DE LA PROGNOSI AMB EL DIAGNÒSTIC

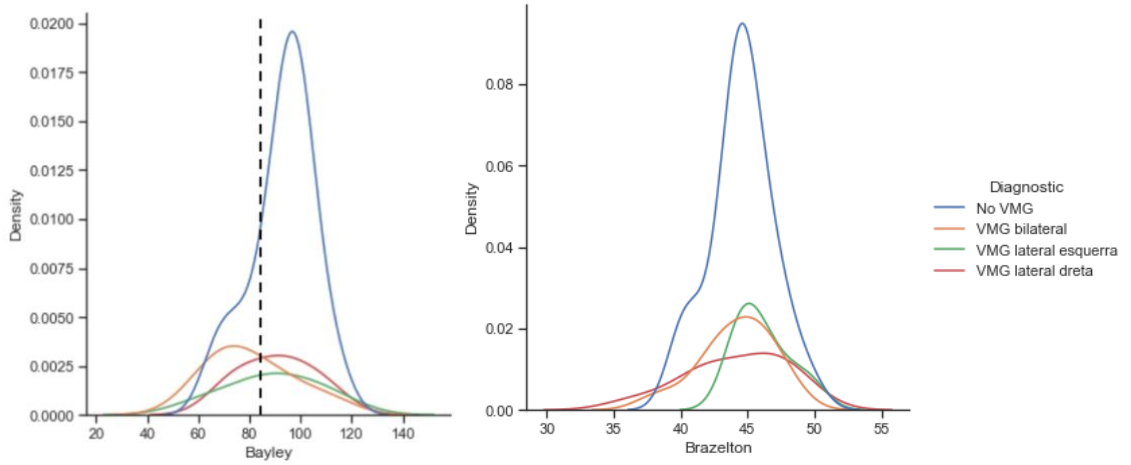
Tant el test de Bayley com el Brazelton ens ofereixen valors numèrics i no per a totes les mostres, però si en un nombre suficient per poder, si més no, intentar una anàlisi explotatòria. En l'annex ANNEX 3: CODI PYTHON: ANÀLISI EXPLORATÒRIA DE DADES hi ha la relació de representacions i anàlisis que s'han efectuat en aquest àmbit:

- Histograma dels resultats de Bayley, tant de forma global com individualitzada per diagnòstic.
- Histograma dels resultats de Brazelton, tant de forma global com individualitzada per diagnòstic.

La Figura 14 ofereix una versió resumida i visual dels histogrames elaborats.

L'escala de Bayley per sota de 85, marcat amb una línia en la imatge de l'esquerra és el que s'estableix com a límit[11]. Un valor menor indica problemes de neurodesenvolupament. No es disposa d'un valor mínim de l'escala de Brazelton per a determinar el valor límit que indiqui la presència de problemes.

Figura 14 Relació del diagnòstic (pre-natal) amb els dels tests de Bayley i Brazelton (post-natals).



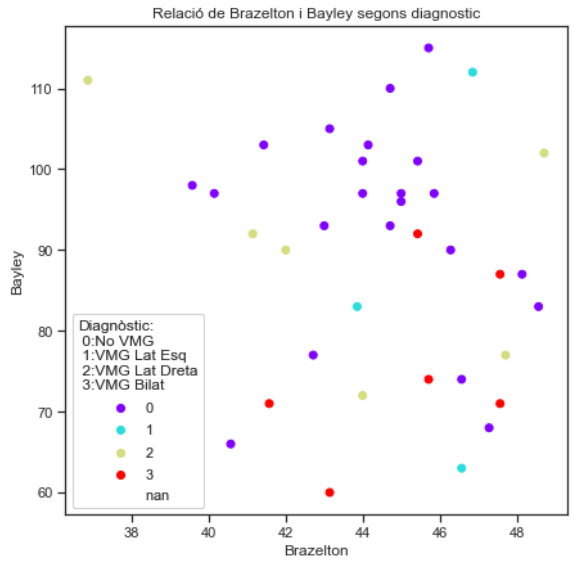
Font: elaboració pròpia

En qualsevol cas es conclou que no hi ha relació entre els resultats del test de Bayley i el test de Brazelton amb el diagnòstic. El diagnòstic no determina la prognosi, ja que els rangs dels resultats dels tests no es diferencien de forma destacable en funció del diagnòstic.

S'analitza la possibilitat que la combinació dels resultats dels dos tests permeti observar una relació amb el diagnòstic.

La Figura 15 presenta aquest resultat en forma de diagrama de dispersió de la que tampoc se'n desprèn cap conclusió.

Figura 15 Relació creuada entre resultats del test Bayley i Brazelton segons diagnòstic.



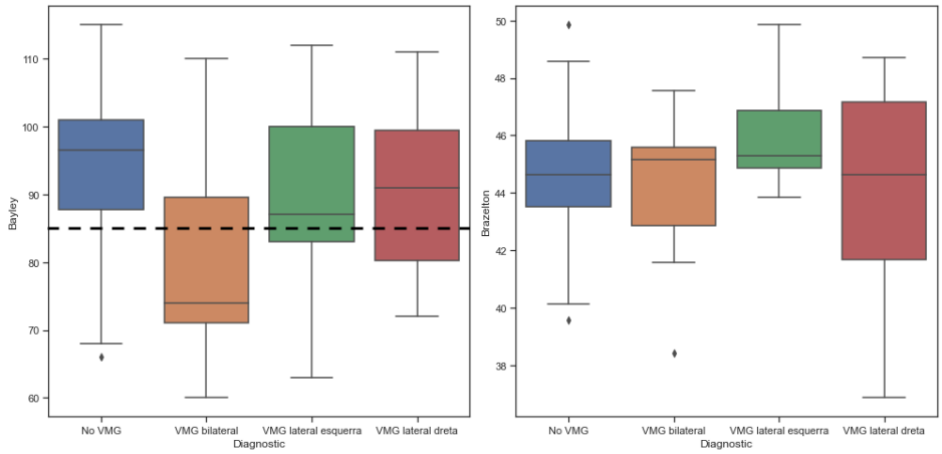
Font: elaboració pròpia

Com a darrera ilustració que permet concloure que no es pot deduir la prognosi directament del diagnòstic, es presenten dos diagrames de caixes (boxplots en anglès), un per cada test en relació amb el diagnòstic.

En els boxplots de la Figura 16 també es pot observar que el valor mig del test de Bayley

per cada diagnòstic és més dispers que el valor mig del resultat del test Brazelton.

Figura 16 Diagrames de caixa dels test de Bayley i Brazelton, en relació amb el diagnòstic.



Font: elaboració pròpia

4.2.2. RELACIÓ DE LA PROGNOSI AMB LA RESTA DE PARÀMETRES

En l'apartat 4.1.1 ANÀLISI DE LA CORRELACIÓ ja s'ha pogut observar que no hi ha cap correlació entre els paràmetres i els tests que determinen la prognosi.

Malgat això, tal com s'ha efectuat amb la caracterització del diagnòstic, també en aquest cas s'efectuen anàlisis de relacions creuades del resultat del test de Bayley amb la resta de paràmetres de forma individualitzada (usant pairplots), perquè ens ofereix la possibilitat de poder observar la relació separant els diagnòstics, per si aportés valor.

Malgrat tenir més dades del test de Brazelton que del test de Bayley el procediment per a caracteritzar la prognosi es basa principalment en el resultat de test de Bayley perquè és el que es considera més validat.

Es fa una revisió manual de tots els diagrames pairplots obtinguts sense que se n'obtingui cap conclusió. Es repeteixen tots els pairplots però usant el test de Brazelton com test a relacionar, sense que s'obtinguin resultats destacables.

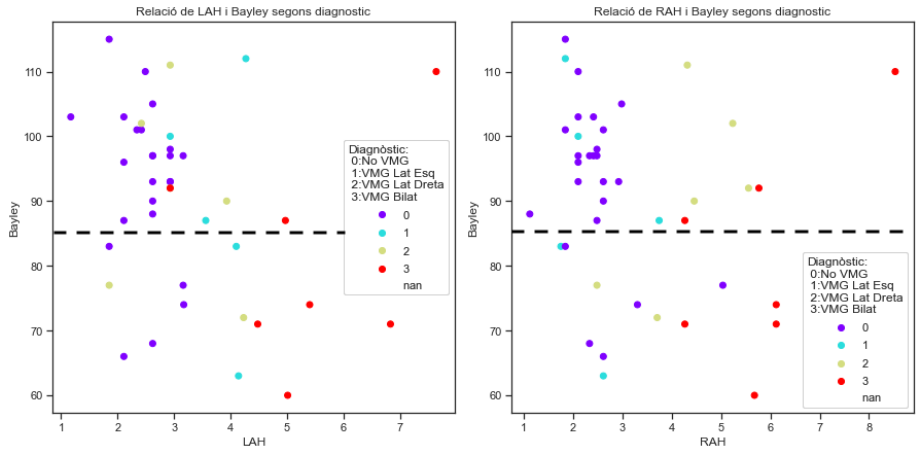
Finalment s'efectuen pairplots comparant cada paràmetre de forma simultània amb els resultats dels dos tests, però tampoc es poden observar en la seva revisió manual cap patró o element que destaqui.

Tots els gràfics es poden consultar a l'annex l'annex ANNEX 3: CODI PYTHON: ANÀLISI EXPLORATÒRIA DE DADES.

Finalment i donat que els valors LAH i RAH han destacat en la caracterització de la VMG, s'analitzen aquests dos valors de forma especial.

La Figura 17 recull dos diagrames de dispersió, un per al resultat del test Bayley en relació amb el paràmetre LAH i l'altre idèntic però amb el paràmetre RAH. A més, es discrimina per color el diagòstic i es marca amb una línia el nivell 85 del test Bayley que marca el límit del resultat anormal.

Figura 17 Relació creuada entre resultats del test Bayley i els paràmetres LAH i RAH segons diagnòstic.



Font: elaboració pròpia

Com es pot observar aquests paràmetres poden permetre caracteritzar la VMG però no la prognosi, ja que el rang de valors de LAH i de RAH per sobre o per sota de la linia que marca el resultat 85 del test no és molt diferent.

5. DETECCIÓ PRECOÇ DE LA PROGNOSI

No es pot caracteritzar la prognosi de la VMG aillada amb cap dels paràmetres dels que es disposa. Haver trobat algun element caracteritzador hauria permès centrar la tasca de la detecció precoç o predicció de la prognosi de forma més clara.

Del treball ja efectuat i detallat en l'apartat 4.2 CARACTERITZACIÓ DE LA PROGNOSI se'n conclou que qualsevol parametrització o predicció serà una combinació de diversos paràmetres no detectables amb l'anàlisis exploratòria dels atributs ja efectuats.

Predir de forma precoç la prognosi es tradueix en tractar de modelar els resultats dels tests que es fan de forma post-natal. En concret l'objectiu és modelar el resultat del test de Bayley, que com ja s'ha mencionat és el que es considera més validat.

Donat que és un valor continu, s'opta per dissenyar un model de regressió.

5.1. PREDICCIÓ PER REGRESSIÓ DEL RESULTAT DEL TEST BAYLEY

En l'annex ANNEX 4: CODI PYTHON: MODEL DE REGRESSIÓ BAYLEY s'hi recull el contingut del notebook amb el codi en python utilitzat en el procés de disseny del model de regressió més adient per a aquest projecte.

El primer pas és seleccionar les dades. Donat que volem predir el valor del test Bayley, eliminarem el resultat obtingut de la base de dades i convertirem aquest valor en la variable objetiu a predir.

Però eliminarem també de la base de dades el resultat de tots els tests pots-natals. No tindria sentit utilitzar-los com a paràmetre ja que són dades post-natals i l'objectiu, precisament és efectuar la predicció tan sols amb dades pre-natals per poder fer un pronòstic precoç. El resultat d'aquesta selecció és un nou conjunt de dades anomenat 'data_regression_Ba'.

5.1.1. REGRESSIÓ LINEAL

S'intentarà, en primer lloc entrenar un model de regressió lineal. Per a configurar i entrenar el model de regressió lineal s'utilitzen les llibreries sklearn (scikit-learn)[25], en concret la "linear_model.LinearRegression". S'aplica un model de regressió lineal a la totalitat de les dades disponibles per analitzar en primer lloc el seu poder de predicció.

Per a avaluar els models s'utilitzen dos mètriques de la mateixa llibreria específicament

destinades a models de regressió: l'error quadràtic mig (o mse, de l'anglès mean squared error) i el coeficient de determinació (identificat en les mètriques amb l'identificador r2_score).

El primer valor és el que mesura l'error de forma que ha de ser el més petit possible, amb un valor òptim de 0[25].

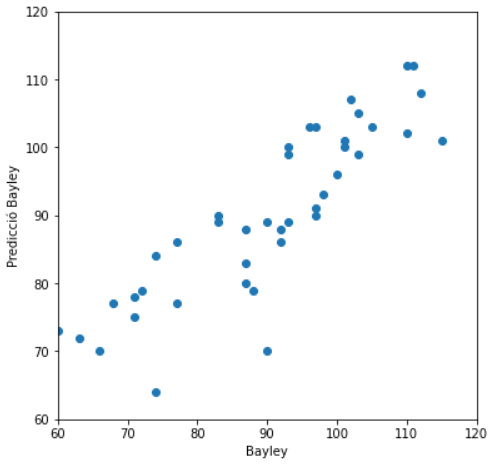
El coeficient de determinació és el que avalua quina part de la variança del valor que es vol predir és explicada amb les dades que formen part del model. Es considera un indici de la bondat del model i de com de bé es preveu que les noves dades s'ajustin al model. El seu valor òptim és 1[25].

Una primera prova de disseny del model ens ofereix un resultat no òptim però aparentment bo (mse= 50 i R2= 0,75).

Figura 18 Prediccions del resultat del test Bayley efectuades amb un model de regressió lineal.

	y	pred
0	74.0	64.0
1	77.0	77.0
2	110.0	112.0
3	87.0	80.0
4	93.0	99.0
5	83.0	90.0
6	103.0	105.0
7	93.0	100.0
8	115.0	101.0
9	97.0	90.0
10	83.0	89.0
11	111.0	112.0
12	88.0	79.0
13	98.0	93.0
14	97.0	103.0
15	93.0	89.0
16	103.0	99.0
17	66.0	70.0
18	101.0	100.0
19	71.0	78.0
20	60.0	73.0
21	112.0	108.0
22	90.0	89.0
23	105.0	103.0
24	101.0	101.0
25	90.0	70.0
26	97.0	91.0
27	96.0	103.0
28	72.0	79.0
29	77.0	86.0
30	100.0	96.0
31	110.0	102.0
32	71.0	75.0
33	92.0	88.0
34	102.0	107.0
35	87.0	88.0
36	74.0	84.0
37	92.0	86.0
38	68.0	77.0
39	63.0	72.0
40	87.0	83.0

Mean squared error: 50.4878
R2 (coeficient de determinacio): 0.7534



Font: elaboració pròpia

Hem utilitzat totes les dades disponibles per a entrenar el model. De manera que no tenim mostres per a validar el resultat que no hagin participat en l'entrenament.

S'ha procedit d'aquesta manera per comprovar la viabilitat d'utilitzar aquest tipus de regressió però és imprescindible disposar d'un conjunt de test sobre el que aplicar el model entrenat. Per a poder efectuar una mínima avaluació s'aplica una validació creuada, usant eines de la mateixa llibreria sklearn[25].

Encara que existeixen altres tècniques que poden ser inicialment més recomanades per al nostre cas, com el mètode LeaveOneOut[27], també hi ha autors (fins i tot els que faciliten llibreries), que indiquen que aquesta tècnica pot generar una alta variança perquè les diferents succesions de jocs de dades creats són subtancialment iguals.

Per això es recomana, fins i tot en conjunts de dades petits, fer servir una validació creuada (anomenada validació de k plecs o k-fold validation, en anglès)[25].

Tenint en compte el reduït nombre de mostres, s'utilitzen diversos valors per a determinar l'estratègia de divisió en plecs. En tots els casos els resultats són molt negatius.

Figura 19 Avaluació del model per al test Bayley de regressió lineal per validació creuada segons nombre de plecs.

nombre de plecs	error quadràtic mig	coeficient de determinació
2	-1204,46	-4,80
4	-4518,84	-24,91
5	-12089,45	-78,32
7	-929452,56	-3248,93
10	-7969,80	-202,34

Font: elaboració pròpia

Tot i que, de fet, la validació creuada ja suposa avaluar el model separant sempre una part de les dades per a la validació, aquests resultats porten a una revisió de la política aplicada en el disseny del model i es repetirà el procés amb una separació de les mostres en un conjunt de d'entrenament i un conjunt de test.

5.1.1.1. GENERACIÓ D'UN CONJUNT D'ENTRENAMENT I DE TEST

Prèvia a la generació del conjunt de test, la decisió a prendre és la mida d'aquesta divisió. El reduït nombre de mostres ens posa davant d'un difícil dilema: si s'usen moltes mostres per a la validació es confirmarà si el model és bo o és dolent de forma més efectiva, però haurem reduït la mida de la mostra que es farà servir per a dissenyar el model, de manera que el model s'entrenarà amb molt poques mostres. Si es destinen moltes mostres a l'entrenament és probable que el model resulti més ajustat però no ho podrem validar perquè tindrem molt poques mostres de test.

Davant aquest dilema, s'opta per crear una rutina que paulatinament incrementi la mida del conjunt de test, intentant, d'aquesta manera trobar un equilibri. Per a que aquesta prova resulti vàlida s'ha seleccionat una llavor d'aleatorietat. Això ens permet assegurar dos aspectes importants:

- que les mostres usades com a test són representatives.
- que el conjunt test creix però mantenint les mostres ja destinades en el conjunt anterior. És a dir, s'afegeixen noves mostres al conjunt test, no es canvien o renoven completament.

Figura 20 Resultats de la rutina elaborada per a determinar la grandària del conjunt test.

mida conjunt test	error quadràtic mig (entrenament)	coeficient de determinació (entrenament)	error quadràtic mig (test)	coeficient de determinació (test)
0,05	24,82	0,8741	22322	-73,35
0,1	24,63	0,8754	12476,94	-49,88
0,15	0	1	22703,56	-99,33
0,2	0	1	5957,99	-30,35
0,25	0	1	3822,19	-22,83
0,3	0	1	3433,10	-22,89
0,35	0	1	1237,86	-6,46
0,4	0	1	1136,28	-5,30
0,45	0	1	1032,25	-4,69

Font: elaboració pròpia

Dels resultats que es mostren a la taula de la Figura 20 ens permeten evidenciar que si dediquem al test més del 10% de les mostres, el model sobreentrena. El coeficient de determinació més proper a 1 es dóna quan el conjunt d'entrenament és el més petit possible. Però cap dels resultats és bo.

A mode d'exemple, es presenten en la els resultats de predicció per a dos valors extrems (10% i 50% de mostres per a test).

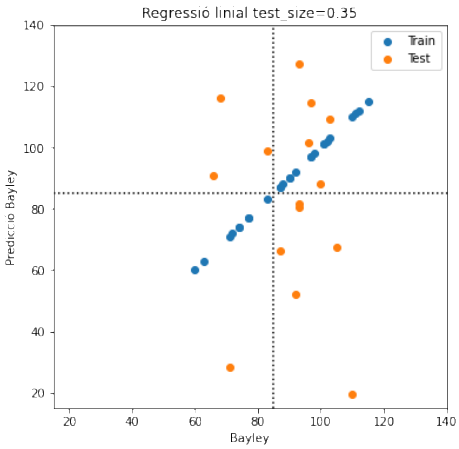
Figura 21 Prediccions de Bayley obtingudes en un model de regressió lineal per a dos valors de mida de test extrems.

Font: elaboració pròpia

Els valors més habituals utilitzats estan en un rang que va del 20% al 40% de les mostres destinades al test. Tot i haver determinat ja que el model de regressió lineal no és una solució viable, analitzem els resultats per a un valor intermig (35%).

Cal recordar que el test de Bayley té un valor considerat límit que discerneix entre un neurodesenvolupament normal o anormal. Un valor inferior a 85 és indicador de problemes en el neurodesenvolupament.

Figura 22 Predicció del valor del test Bayley amb un model de regressió lineal usant el 35% de mostres per a test.



Font: elaboració pròpia

La Figura 22 permet observar si els errors de predicció provoquen que mostres amb

valors per sobre de 85 es prediuen per sota de 85 i viceversa.

5.1.1.2. NORMALITZACIÓ

En l'anàlisi exploratori s'ha observat que la normalització no té un efecte perceptible en els resultats, ja que hi ha poca dispersió en els rangs dels valors del paràmetres.

Malgrat això, donat que les bones pràctiques així ho recomanen, es fa una normalització de les dades, de forma similar i amb les mateixes eines utilitzades en les anàlisis exploratòries.

La diferència és gairebé imperceptible. No podem comparar de forma directa el valor quadràtic mig perquè els valors estan normalitzats, però el coeficient de determinació gairebé no canvia. S'obté un 0,7527 enlloc del 0,7534 obtingut sense haver normalitzat.

La normalització s'efectua amb la funció StandardScaler que és adequada per a la majoria de casos. Es proven altres sistemes de normalització, per exemple el RobustScaler, de la mateixa llibreria que l'anterior[25]. Amb aquesta modificació, el canvi és nul, tal com es pot comprovar en el codi presentat en l'annex ANNEX 4: CODI PYTHON: MODEL DE REGRESSIÓ BAYLEY.

5.1.2. REGRESSIÓ POLINOMIAL

Davant la impossibilitat de generar un model amb resultats vàlids amb un model de regressió lineal, es configura i entrena un model polinomial. No es preveu que els resultats millorin perquè el problema és principalment de sobreentrenament i no sembla probable que usar un model polinomial solucioni aquest problema, però es fa la prova per a confirmar-ho.

Tenim una base de dades amb 34 paràmetres. És habitual considerar que la ràtio 15:1 permet determinar el nombre de graus necessaris per al model. En el nostre cas, per tant, seria un grau 2.

S'ha provat, però, un model amb grau 2, 3 i 4. Tal com s'ha efectuat en el cas del model de regressió lineal el primer que es fa és usar totes les dades per a observar si una solució és viable. Fins i tot amb totes les dades dedicades a entrenament, en tots els casos el model sobreentrena.

Seguint el mateix procediment que en el cas anterior, apliquem validació creuada i recopilem en la Figura 23 els resultats obtinguts segons el grau del model polinomial i el nombre de plecs aplicat.

Podem observar que, efectivament tots els casos són inaplicables però que el menys dolent és el que usa un model de grau 2 i només 2 plecs, que es correspon a dedicar la meitat de les mostres a test (com ja ens ha resultat en el model lineal).

Una vegada més, encara que sabem que el model està sobreentrenat, fem la predicció amb un conjunt d'entrenament i de test (50% de mida de test) i amb un model de regresió no-lineal de grau 2 per analitzar l'error suportat.

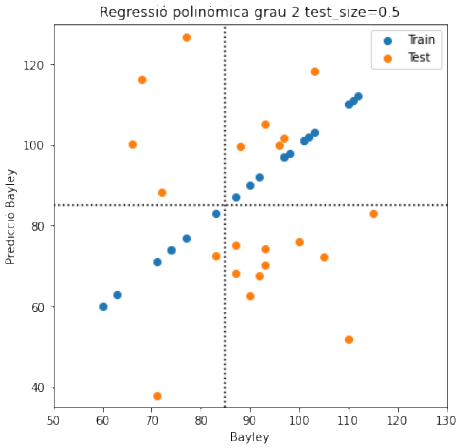
Figura 23 Resultats de l'avaluació del model de regressió polinomial per validació creuada segons nombre de plecs i el grau del model, per al valor del test Bayley.

grau	nombre de plecs	error quadràtic mig	coeficient de determinació
2	2	-1124,93	-4,42
2	5	-2003,93	-10,31
2	10	-2856,52	-41,81
3	2	-1525,11	6,32
3	5	-1994,97	-9,57
3	10	-2818,49	-32,16
4	2	-1866,01	-7,95
4	5	-2261,28	-10,41
4	10	-3247,33	-30,23

Font: elaboració pròpia

La Figura 24 permet observar si els errors de predicció provoquen que mostres amb valors per sobre de 85 es prediguin per sota de 85 i viceversa.

Figura 24 Predicció del valor del test Bayley amb un model de regressió polinomial de grau 2 usant el 50% de mostres per a test.



Font: elaboració pròpia

Tots els resultats obtinguts i que es poden comprovar de forma extensa i detallada en el fitxer de codi recollit en l'annex ANNEX 4: CODI PYTHON: MODEL DE REGRESSIÓ BAYLEY, ens porten a considerar que qualsevol model de regressió que es pugui desenvolupar amb les dades del test Bayley tindran poca precisió a causa de l'escassetat de mostres disponibles que porten de forma inevitable a un sobreentrenament.

5.2. PREDICCIÓ PER REGRESSIÓ DEL RESULTAT DEL TEST BRAZELTON

Encara que el Test de Brazelton no dóna la mateixa informació que el test Bayley ni està tant validat, es considera una bona alternativa dissenyar un model de regressió per a predir aquest atribut. Cal recordar que tenim més valors de test Brazelton que de test Bayley, per això podria ser que un major nombre de mostres minimitzés l'efecte de sobreentrenament que hem patit amb el model aplicat al Bayley.

L'aproximació i estratègia de resolució és la mateixa que hem aplicat en el cas anterior. El codi del desenvolupament dut a terme es pot consultar a l'annex ANNEX 5: CODI

PYTHON: MODEL DE REGRESSIÓ BRAZELTON.

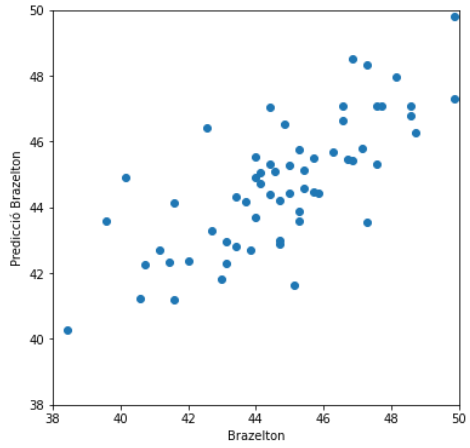
5.2.1. DISSENY AMB DADES SENSE NORMALITZAR

Per a configurar i entrenar el model de regressió lineal s'utilitzen les llibreries sklearn (scikit-learn)[25], en concret la "linear_model.LinearRegression".

S'aplica també en aquest cas, doncs, el mateix model de regressió lineal a la totalitat de les dades disponibles. I s'avalua amb les mateixes mètriques: l'error quadràtic mig i el coeficient de determinació.

Una primera prova de disseny del model ens ofereix un resultat molt pobre, amb un error quadràtic mig de 2,7996 i un coeficient de determinació de 0,6047. El gràfic que permet valorar les predicions es pot consultar en la Figura 25.

Figura 25 Prediccions efectuades amb el model de regressió lineal per al resultat del test Brazelton.



Font: elaboració pròpia

Procedim, però, amb el mateix protocol aplicat en el cas anterior i per a validar-lo s'aplica una validació creuada, usant eines de la mateixa llibreria sklearn[25].

Figura 26 Avaluació del model per al test Brazelton de regressió lineal per validació creuada segons nombre de plecs.

nombre de plecs	error quadràtic mig	coeficient de determinació
2	-106,53	-17,84
4	-22,70	-2,91
5	-24,32	-5,28
6	-22,33	-3,20
7	-20,77	-5,19
10	-21,43	-4,94

Font: elaboració pròpia

Com en el cas anterior, el resultats obtinguts són insuficients i es canvia l'estratègia aplicant una separació de dades en un conjunt d'entrenament i un de test.

Encara que l'entrenament no disposarà de tantes mostres, el resultat de la validació i per tant l'eina per avaluar la bondat del model serà millor.

5.2.1.1. GENERACIÓ D'UN CONJUNT D'ENTRENAMENT I DE TEST

Ens trobem amb el mateix dilema que en el cas anterior. La diferència, però, és que en aquest cas tenim 62 mostres etiquetades (que tenen valor assignat) que segueixen sent molt poques, però són un 50% més de les que teniem per al model de Bayley.

S'utilitza la mateixa rutina que paulatinament incementi la mida del conjunt de validació, intentant, d'aquesta manera, trobar un equilibri.

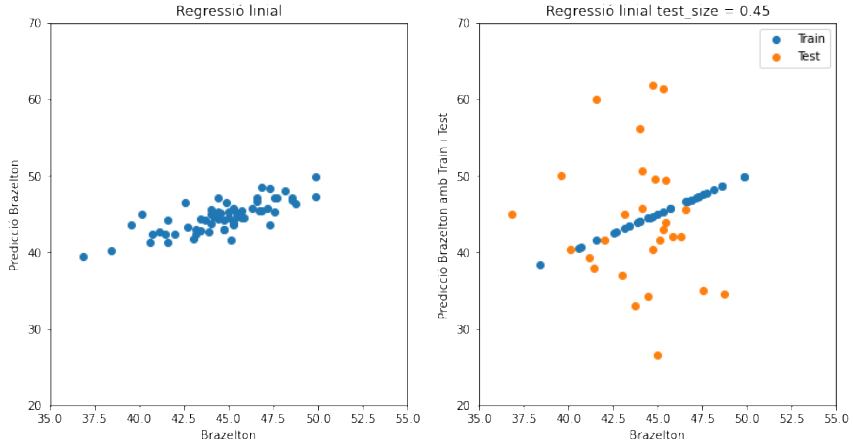
Figura 27 Resultats de la rutina elaborada per a determinar la grandària del conjunt test.

mida conjunt test	error quadràtic mig (entrenament)	coeficient de determinació (entrenament)	error quadràtic mig (test)	coeficient de determinació (test)
0,05	2,06	0,6947	31,74	-1,77
0,1	1,87	0,7306	23,91	-2,04
0,15	1,87	0,7365	19,72	-1,94
0,2	1,69	0,7746	15,93	-1,98
0,25	1,46	0,8158	24,98	-4,55
0,3	1,42	0,7956	17,77	-1,64
0,35	1,40	0,8060	20,32	-2,39
0,4	0,78	0,8845	80,75	-11,84
0,45	0	1	83,61	-12,46

Font: elaboració pròpia

Les dades de la taula presentades en la Figura 27 ens permeten observar que si es dedica més del 40% de les mostres a conjunt de test es provoca un sobreentrenament.

Figura 28 Prediccions de Brazelton obtingudes en un model de regressió lineal per a dos valors de mida de test extrems.

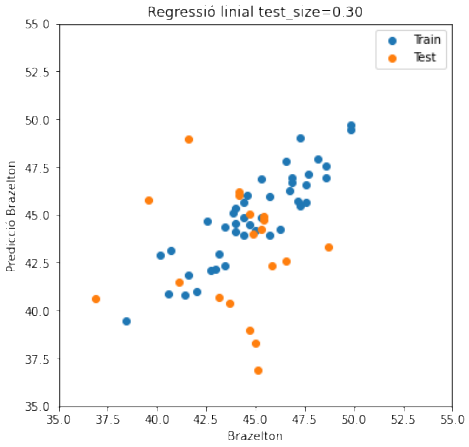


Font: elaboració pròpia

La Figura 28 presenta les prediccions comparades dels dos extrems: no dedicar cap dada a test i dedicar el 45% de les mostres a test provocant un sobreentrenament.

Així mateix la Figura 29 presenta la predicció que ofereix el model amb la grandària del conjunt de test que ofereix un millor resultat, segons el que s'ha observat en la taula anterior.

Figura 29 Predicció del valor del test Brazelton amb un model de regressió lineal usant el 30% de mostres per a test.



Font: elaboració pròpia

5.2.1.2. NORMALITZACIÓ

De nou en aquest cas, s'aplica normalització per a descartar que existeixi alguna afectació sobre la predicció. La diferència és inexistent i s'obté exactament el mateix resultat en la mètrica R2 i exactament les mateixes prediccions. Es comprova fent un diagrama de dispersió que permet observar la identitat dels dos conjunts de prediccions (amb i sense normalització).

Tampoc és detectat cap canvi si es genera un conjunt de test sobre els valors normalitzats tal com es pot comprovar en el codi presentat en l'annex ANNEX 5: CODI PYTHON: MODEL DE REGRESSIÓ BRAZELTON.

5.2.2. REGRESSIÓ POLINOMIAL

El model lineal de regressió obtingut és millor que el que s'ha aconseguit amb el test Bayley, però també s'analitza l'aplicació de models no lineals.

Ja s'ha comentat en el capítol anterior que el grau que previsiblement ha de donar un millor resultat és 2. Però també aquí es testejarà el model amb grau 2, 3 i 4.

Figura 30 Resultats de l'avaluació del model de regressió polinomial per validació creuada segons nombre de plecs i el grau del model, per al valor del test Brazelton.

grau	nombre de plecs	error quadràtic mig	coeficient de determinació
2	2	-38,31	-5,04
2	5	-31,16	-5,95
2	10	-35,72	-8,35
3	2	-32,67	-4,01
3	5	-32,79	-6,16
3	10	-35,81	-8,05
4	2	-33,48	-4,02
4	5	-35,58	-6,64
4	10	-37,36	-8,22

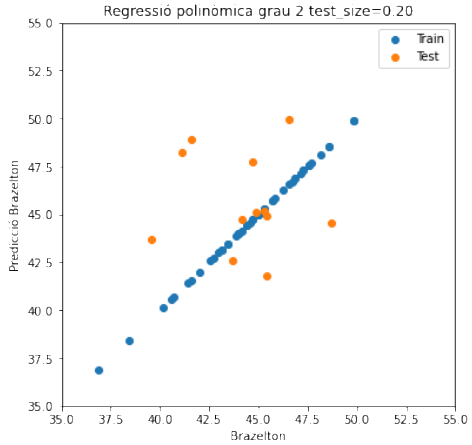
Font: elaboració pròpia

Seguint el mateix procediment que en el cas anterior, apliquem validació creuada i recopilem en la Figura 30 els resultats obtinguts segons el grau del model polinomial i el nombre de plecs aplicat.

Podem observar que, efectivament tots els casos són allunyats d'un model bo però que el menys dolent és el que usa un model de grau 2 i 5 plecs (que correspon a dedicar un 20% de les mostres a test.

Una vegada més, encara que sabem que el model està sobreentrenat, fem la predicció amb un conjunt d'entrenament i de test (20% de mida de test) i amb un model de regresió no-lineal de grau 2 per analitzar l'error suportat.

Figura 31 Predicció del valor del test Brazelton amb un model de regressió polinomial de grau 2 usant el 20% de mostres per a test.



Font: elaboració pròpia

5.3. PREDICCIÓ PER CLASSIFICACIÓ

S'han obtingut models de regressió per a determinar els valors dels tests Bayley i Brazelton. Els millors resultats s'han obtingut amb models lineals. El model Brazelton ha donat resultats prou bons, però el test de Brazelton no està tant validat com el test Bayley. I no hem aconseguit un model regressiu prou acurat per aquest test a causa del nombre reduït de mostres amb resultat del test Bayley (només 42 dels 81 registres).

En una revisió d'objectius amb la doctora Eixarch es considera que una classificació de l'afectació en el neurodesenvolupament també seria interessant. És a dir, disposar d'un classificador que predigui si hi haurà problemes de neurodesenvolupament post-natals.

En aquest sentit, en un dels documents usats com a font de dades, el Bayley-Vineland.xls ja contenia 3 columnes.

La primera indicant amb un 1 o un 0 la anormalitat detectada en el test Bayley (que es considera anormal per valors inferior a 85). Una segona columna amb indicació, també binaria sobre la anormalitat del resultat del test Vineland. En aquest segon cas es desconeixen les dades en brut que han donar orígen a aquesta valoració. I finalment una tercera columna en la que es considera, també de forma binaria si un dels dos (o els dos)

tests han ofert resultats anormals.

En aquest cas es decideix no utilitzar el valor del test de Brazelton donat que la doctora considera més vàlids els resultats dels altres dos tests per a aquesta classificació.

El desenvolupament en codi d'aquestes tasques està recollit al ANNEX 6: CODI PYTHON: MODEL DE CLASSIFICACIÓ DE LA PROGNOSI.

5.3.1. PREPARACIÓ DE LES DADES I ESTRATÈGIA

Es decideix, per tant, dissenyar i configurar un model de classificació per a predir quan un dels dos tests Bayley o Vineland sortirà anormal.

Per tal de maximitzar les mostres disponibles (recordem que només tenim 42 resultats del test Bayley i 25 resultats del test Vineland), es considera que, a falta de dades, el resultat és normal. Encara que no es disposi del resultat numèric que ha portat al diagnòstic, el fet que no hi hagi diagnòstic s'interpreta com que no ha reportat mostres de problemes en el seu neurodesenvolupament.

Com a primer pas, es genera un nou atribut que codifica en binari la anormalitat del test Bayley (atribut 'Bayley_cod'). I en segon lloc es genera un nou atribut que conté un '1' quan el valor de 'Bayley_cod' és '1' o el valor de l'atribut 'Vineland' és 'SI'.

D'aquesta manera disposem d'un nou atribut ('B_o_V_cod') que serà utilitzat com a nova etiqueta de supervisió per al model de classificació.

Es crea un nou joc de dades al que s'anomena 'data_logic' en el que s'eliminen les dades de diagnòstic post-natal ('Brazelton', 'Vineland', 'Bayley', 'Bayley_cod'). També s'elimina 'ID' per ser innecessària i 'Sexe' perquè ja existeix un atribut amb el sexe codificat en binari. Per tant, només es conserven els atributs de la MRI, a més del 'Sexe_cod' i el diagnòstic pre-natal de la VMG ('Diagnòstic_cod').

5.3.1.1. ANÀLISI EXPLORATÒRIA

Els paràmetres ja han estat analitzats. La única variable que no hem analitzat anteriorment és l'etiqueta 'B_o_V_cod'. De l'anàlisi se'n desprèn que no tenim un joc de dades balancejat.

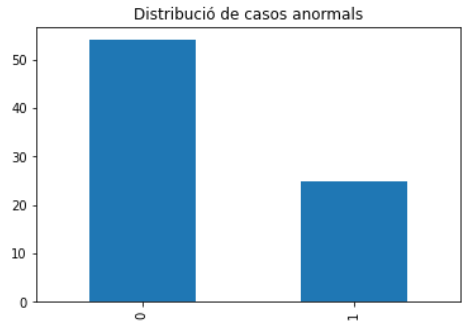
No està balancejat en quant a la prognosi, però sí que està força balancejat en quant a les classes per cadascun dels diagnòstics que no són casos de control.

La Figura 32 adjunta facilita l'observació d'aquesta situació. I també ens permet apreciar que ara treballarem amb un total de 79 mostres totalment operatives, 39 de casos de control i 40 de casos d'afectació i amb 25 casos amb prognosi anormal i 54 amb prognosi normal.

Aquest desbalanceig serà important i caldrà tenir en consideració altres mètriques a banda de l'exactitud per a seleccionar un bon model o comparar els models entrenats. S'opta per usar el F1_score, que és la mètrica que habitualment s'usa perquè ens dóna un valor que és el compromís entre la precisió i el recall[26][29].

Per a que aquest valor, a més, tingui en compte el desbalanceig de la presència de mostres de cada classe, es pot indicar a través de l'argument "average = weighted".[26]

Figura 32 Balanceig de les mostres per etiqueta de classe i per cada diagnòstic pre-natal.



Llegenda (B_o_V_cod)

Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

Diagnostic_cod	0	1	2	3	All
B_o_V_cod					
0	33	7	6	8	54
1	6	7	4	8	25
All	39	14	10	16	79

Llegenda (Diganostic_cod)

Classe '0'	No VMG
Classe '1'	VMG lateral esquerra
Classe '2'	VMG lateral dreta
Classe '3'	VMG bilateral

Font: elaboració pròpia

5.3.1.2. CONJUNT D'ENTRENAMENT I TEST

Tot i que 79 mostres segueixen sent un joc de dades molt petit, per a la classificació s'opta per treballar en tot moment amb un conjunt d'entrenament i un conjunt de test. De fet, caldria disposar de 3 conjunts: entrenament, validació i test, però el nombre de mostres suggereix treballar només amb entrenament i test.

Per mantenir el balanceig existent en els dos jocs de dades, s'usa el paràmetre random_state per garantir que sempre aconseguim una conjunt d'entrenament i test idèntic per comparar tots els classificadors amb les mateixes condicions. S'escull un valor de llavor d'aleatorietat que ofereixi uns jocs el més equilibrats possibles en relació amb la prognosi que és el que volem classificar.

5.3.1.3. ESTRATÈGIA DE DISSENY

En aquest cas s'usa un procés habitual de validació creuada per a entrenar de forma òptima el model. La Figura 33 mostra dos imatges que il·lustren aquest procés.

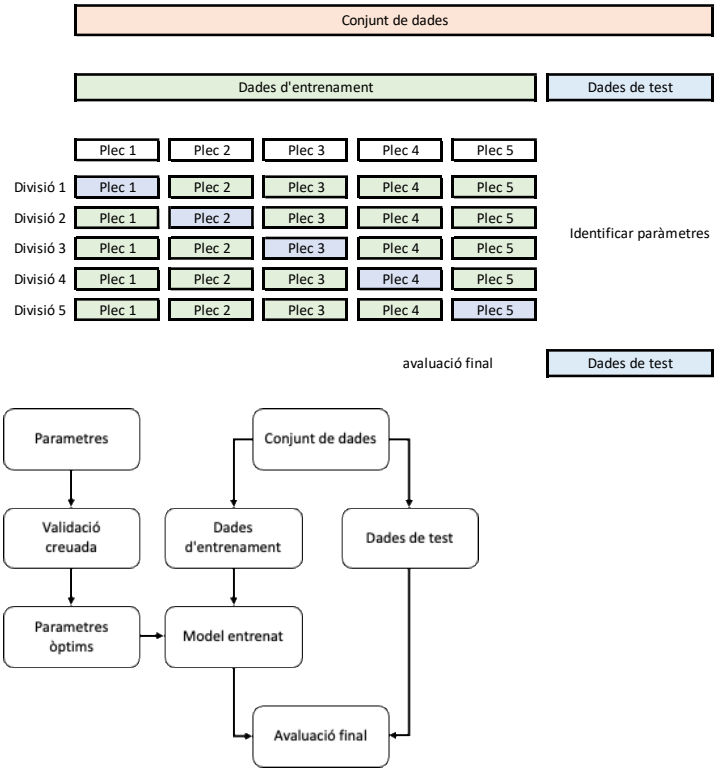
Bàsicament el que permet aquesta metodologia és evitar el sobreentrenament. S'usen els jocs de dades per separat. S'usa una validació creuada sobre el conjunt d'entrenament per determinar els millors paràmetres per obtenir els millors resultats i finalment, un cop seleccionats els paràmetres adients, aquests són els que s'usen per a entrenar el model amb el conjunt d'entrenament.

El conjunt de test que no ha participat ni en la selecció de paràmetres ni en l'entrenament, permet una avaluació objectiva de la bondat del model.

Aplico la validació creuada amb una busca per graella (grid-search en anglès) que permet fer una graella de paràmetres de forma molt més ràpida i intuïtiva que amb una

sèrie successiva de bucles. A més permet guardar com a variable el millor valor per a cada paràmetre i així usar-lo en l'entrenament del model sense haver-lo d'introduir manualment.

Figura 33 Metodologia d'entrenament de models usant validació creuada.



Font: Pedregosa F i altres[25]

Abans d'iniciar el procés de configuració i entrenament del model, cal tenir en compte les dades que tenim. En un model amb dos classes balancejades, qualsevol model amb una exactitud superior al 50% ja és millor que un model aleatori. No és el nostre cas, al no tenir un conjunt de dades balancejat.

Disposem de 54 de 79 mostres de classe '0' a l'entrenament i 10 de 16 mostres al conjunt de test. És a dir, tenim un 68,35% de les mostres d'entrenament i un 62,5% de les mostres de test que tenen una classe '0'. Qualsevol model que ens ofereixi una predicció sempre '0', tindrà una exactitud del 62,5%.

Per això tal com s'ha avançat anteriorment usarem F1 com a mètrica de comparació. Amb jocs de dades tant desbalancejats, si un model prediu sempre la categoria més habitual tindrà una exactitud proporcional al desbalanceig.

5.3.2. K VEÏNS MÉS PROPERS

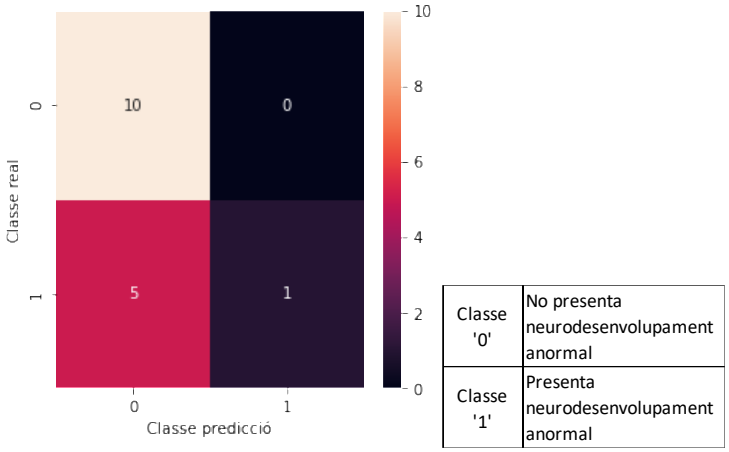
El primer model que s'aplica és el k veïns més propers (K-NN) perquè el joc de dades típic d'aquest tipus d'algoritmes està format per diversos atributs descriptius i un sol atribut objectiu anomenat classe o etiqueta[27].

A més a més, aquest és un dels models que en l'estat de l'art ja s'ha indicat que és un

dels més recomanats quan no es disposa de xarxes neuronals preentrenades.

Entre els paràmetres a entrenar en la busca per graella no hi consignarem el nombre de classes, perquè no estem en un problema d'agrupament sinó de classificació i ja sabem que el nombre de classes és de 2. Sí que farem una graella per determinar el paràmetre de pesos. Obtenim el mateix resultat tant si usem valors uniformes o per distància.

Figura 34 Matriu de confusió de la predicció del model K-NN.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

Font: elaboració pròpia

La predicció després d'entrenar el model dóna un resultat molt pobre: una exactitud del 68,75% i una F1 de 60,71%.

Per a poder comparar els models es presenta per a cada cas la seva matriu de confusió, com la de la Figura 34. En aquest cas, el model prediu correctament totes les mostres etiquetades de classe '0' però només és capaç de predir correctament una de les sis mostres de test de classe '1'.

5.3.3. ARBRE DE DECISIÓ

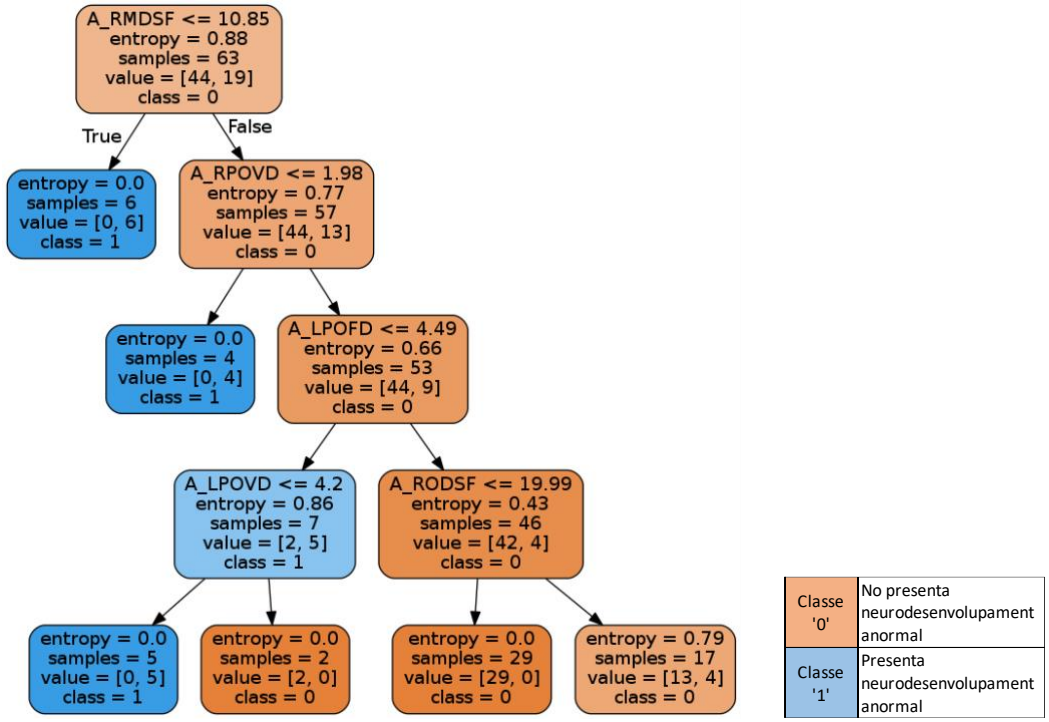
El següent model a configurar i entrenar és l'arbre de decisió. No perquè esperi obtenir un resultat millor (que no sol ser el cas perquè hi ha models molt més complexos) però aporta l'enorme avantatge de la seva capacitat autoexplicativa.

En aquest cas, la graella prèvia de busca de paràmetres ens permet determinar que els millors resultats s'obtindran amb:

- criteri de decisió: per entropia.
- màxima profunditat: 4.
- nombre mínim de mostres per fulla: 2.
- nombre mínim de divisió de mostres: 2.

Amb aquests paràmetres s'entrena el model i s'obté un resultat exportable a un fitxer d'imatge. S'utilitza una web gratuïta (https://onlineconvertfree.com/complete/dot-png/) per a convertir el fitxer en una imatge i poder-la presentar en la Figura 35.

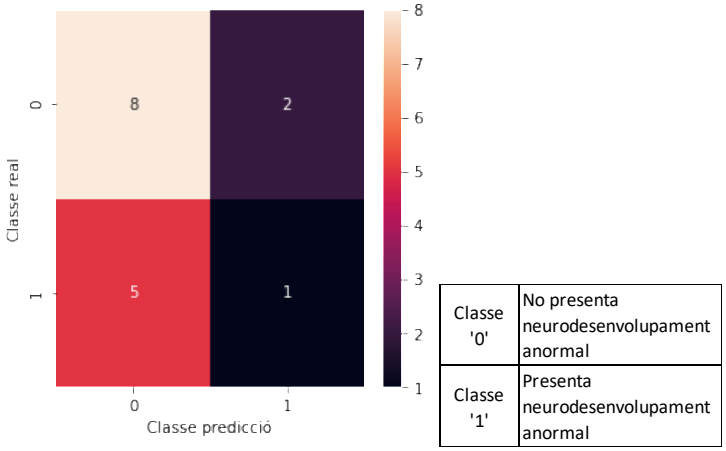
Figura 35 Model d'arbre de decisió.



Font: elaboració pròpia

Lamentablement no ofereix sempre un model igual. Cada simulació pot oferir un arbre diferent. Aquest és el cas en el que ens trobem. Tot i que les variacions no són molt substancials, sí que existeixen.

Figura 36 Matriu de confusió del model d'arbre de decisió sobre el conjunt de test.



Font: elaboració pròpia

A més de la informació en format text que apareix en cada node de l'arbre, es pot observar que el color de les caselles serveix per indicar quina de les dues classes es classifica (en aquest cas '0' per al taronja i el '1' per al blau). I el to del color indica el nivell d'entropia. A major to de color, menor entropia i menor indefinició en la classificació.

Aquest model no dóna tampoc bons resultats. No supera el 56,25% d'exactitud i el 51,81% de F1. És a dir, no és millor que si haguéssim decidit classificar-ho tot com a '0'.

5.3.4. BOSC ALEATORI

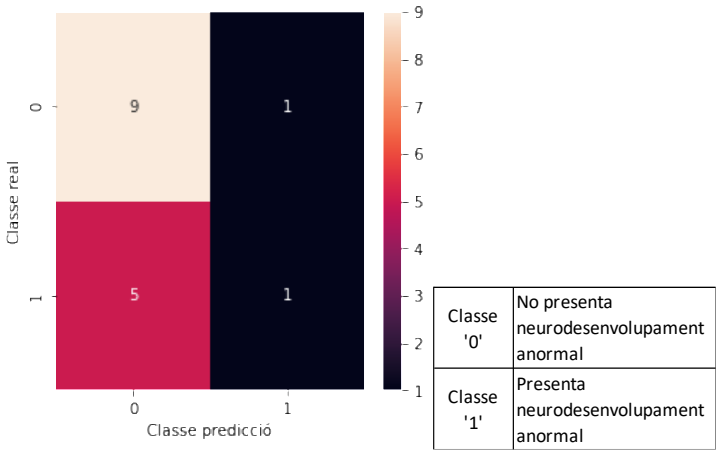
El bosc aleatori (Random Forest en anglès) com el seu nom porta a intuir és un conjunt d'arbres contruits de forma aleatòria. Utilitza, per tant, un arbre de decisió com a classificador de base i aplica com a tècnica d'agrupació el Bagging. Cada arbre es construeix a partir d'una mostra del conjunt de dades original.[28]

El disseny del bosc aleatori té dos paràmetres principals: la profunditat i el nombre d'arbres. S'aplica una busca per graella (grid search en anglès) per a determinar els millors valors per aquests paràmetres i es determina que són 3 i 27 respectivament.

S'entrena el model obtenint valors molt similars que amb l'arbre de decisió. Cal recordar que és un model aleatori per tant, cada execució pot aportar valors diferents.

En concret, en la majoria d'aplicacions s'obtenen valors idèntics que amb l'arbre de decisió i altres valors una mica millors (no en exactitud, que es manté en 62,50% però sí en F1 que puja fins a 56,25%). La matriu de confusió es presenta en la Figura 37.

Figura 37 Matriu de confusió del model Random Forest entrenat.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

Font: elaboració pròpia

5.3.5. MÀQUINES DE VECTOR DE SUPORT

Les màquines de vector de suport (SVM, de Support Vector Machines) són un tipus d'algoritme, lineal o no lineal, supervisat, capaç de resoldre problemes de classificació, però també de regressió. Es considerarà que cada kernel disponible és un tipus de model diferent.

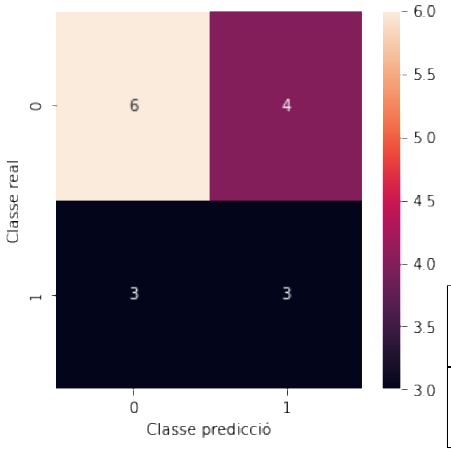
5.3.5.1. ÚS DE KERNEL LINEAL

Era d'esperar que l'ús del kernel lineal no oferiria bons resultats, ja que en cas de tenir un fort component lineal és probable que els resultats obtingut en els models de regressió lineals haguessin estat millors.

Com en els casos anteriors, la determinació dels paràmetres es fa aplicant una graella encara que només hi ha un paràmetre. "C" és el paràmetre de regularització i el valor òptim aplicat és de 0,1.

La matriu de confusió de la Figura 38 acredita que no és un model ni precís ni exacte. L'exactitud és de 56,25% i el F1 score és de 56,78%.

Figura 38 Matriu de confusió del model basat en una SVM lineal.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

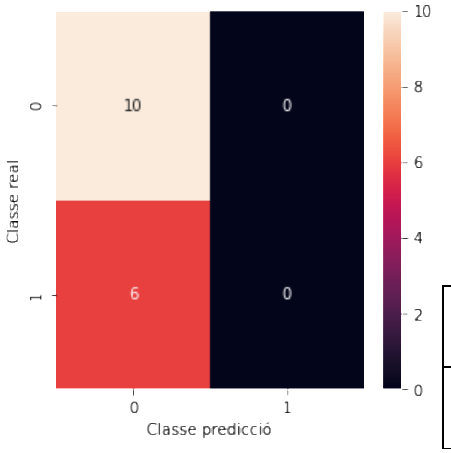
Font: elaboració pròpia

La diferència amb els casos anteriors és que millora la classificació de la classe '1' en perjudici de la classe '0'.

5.3.5.2. ÚS DE KERNEL RADIAL

En cas d'aplicar un kernel radial ('rbf') es pot aplicar un paràmetre addicional: la gamma del kernel. La busca per graella ens indica que els millors paràmetres són: paràmetre de regulació: 0,05; gamma: 0,001.

Figura 39 Matriu de confusió del model basat en una SVM radial.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

Font: elaboració pròpia

S'obté una classificació òptima per a la classe '0' però nul·la per a la classe '1'. Això fa que l'exactitud sigui 62,5% però el F1 sigui 48,08%.

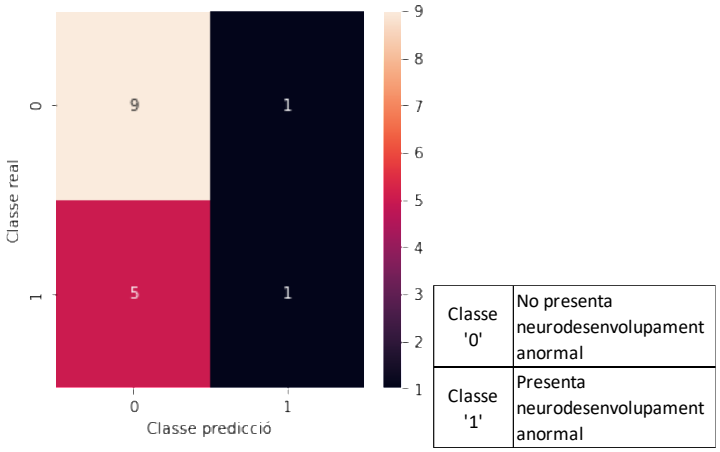
5.3.5.3. ÚS DE KERNEL POLINOMIAL

Aquest model permet afegir un paràmetre addicional que precisament és el grau del polinomi.

La busca per graella ens indica que els paràmetres òptims són: grau 3, paràmetre de regularització 0,01 i gamma 0,001.

Ens permet obtenir un model una mica millor que l'obtingut fins el moment amb els altres kernels amb una exactitud de 62,5% i el F1 arriba a 56,25%.

Figura 40 Matriu de confusió del model basat en una SVM polinomial.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

Font: elaboració pròpia

5.3.5.4. ÚS DE KERNEL SIGMOIDE

S'ha configurat un model de support vector machine amb kernel tipus sigmoide.

No s'obtenen resultats millors usant aquest kernel, malgrat que en les xarxes neuronals de classificació binaria s'acostuma a usar com a capa de sortida una activació de tipus sigmoide[26][27][29][29].

Els paràmetres determinats per la graella han estat 0,05 pel paràmetre de regularització i 0,001 per la gamma. Els resultats tampoc resulten favorables. Es repeteixen els mateixos valors resultants de les mètriques i la matriu de confusió del model amb kernel radial.

5.3.5.5. VERSIÓ LINEAL

Existeix una segona tipologia de SVM anomenada linearSVM que es diferencia de la que hem configurat uns apartats més amunt en que intenta minimitzar la pèrdua de frontissa (hinge loss) al quadrat enlloc de la pèrdua de frontissa com el cas anterior, de manera que no és realment un kernel lineal.

A priori, l'ús de la SVM lineal que hem usat anteriorment és més adequat per a separar les classes una-vs-una classe, mentre que la Linear SVM que ara intentarem és millor per a separar una-vs-resta de classes. En un problema de classificació binari com el que tenim no hauria de resultar molt diferent[25].

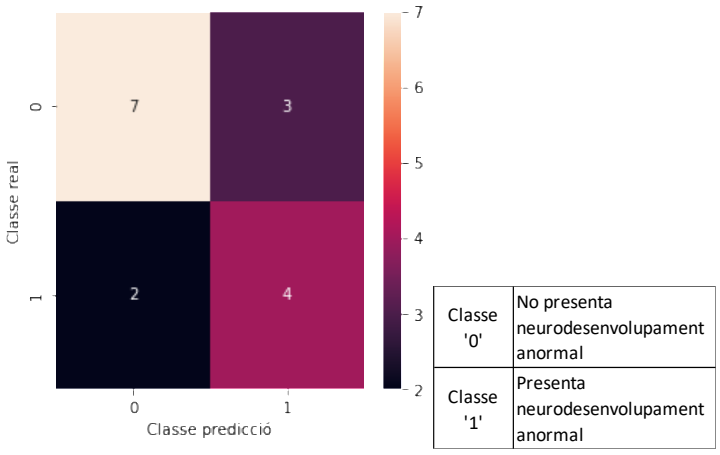
El primer problema que ens trobem a l'entrenar aquest model és en la seva dificultat de convergir en una solució. S'ha d'incrementar de forma notable el límit d'iteracions. Per a observar la magnitud d'aquest problema cal tenir en compte que el nombre d'iteracions per defecte és de 1000 i s'ha hagut d'ajustar a 500000 (500 vegades superior) per obtenir

alguns resultats. El paràmetre de regularització resultat de la busca per graella és de 0,05.

S'aconsegueix una millora sobre els altres tipus de models, amb un 68,75% de l'exactitud, però també millora la F1 amb un 69,13%, la qual cosa ens indica que classifica de forma més equilibrada les dues classes.

Si observem la matriu de confusió de la Figura 41 veiem com, en aquest cas, classifica pitjor mostres de classe '0' que amb altres models es classificaven perfectament, mentre que hem millorat en les mostres de classe '1'. Cal tenir en compte que no sempre un fals negatiu és igual de bo o de dolent que un fals positiu.

Figura 41 Matriu de confusió del model basat en una LinearSVM.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

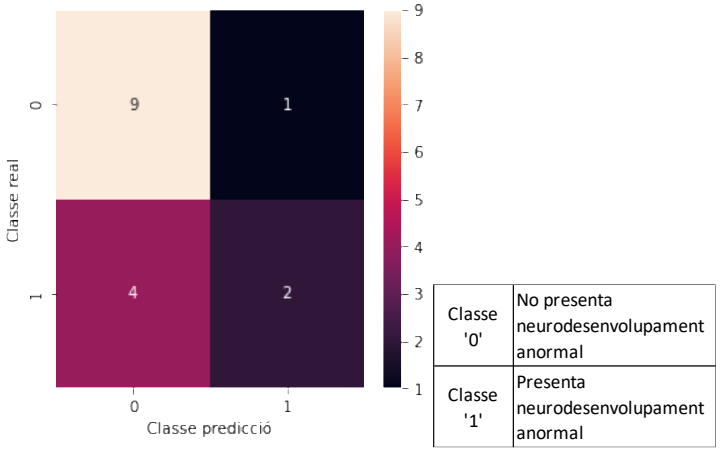
Font: elaboració pròpia

5.3.6. REGRESSIÓ LOGÍSTICA

Malgrat s'anomena regressió logística, és vàlida per a classificar.

En aquesta ocasió cal determinar el valor de os paràmetres. Per una banda el paràmetre de regularització i per altra amb el tipus d'algoritme usat per a l'optimització.

Figura 42 Matriu de confusió del model de regressió logística.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

Font: elaboració pròpia

Encara que per a jocs de dades petits 'liblinear' és el recomanat[25], per a valors de regularització baixos s'obtenen millors resultats amb l'algoritme 'newton-cg'.

La graella ens indica que cal usar el newton-cg i un parametre de regularització de 0,01.

També en aquest model cal incrementar el límit d'iteracions per a que convergeixi.

Els resultats obtinguts són similars als obtinguts amb el darrer model SVM entrenat: una exactitud del 68,75% però una F1 de 65,58%. Hem tornat a empitjorar la classificació dels casos afectats (classe '1').

La Figura 42 adjunta mostra la matriu de confusió obtinguda amb aquest model.

5.3.7. XARXA NEURONAL

Encara que la base de dades de la que disposem és molt petita, els resultats obtinguts ens porten a pensar que o bé no tenim dades suficients per interpretar els patrons o bé els models utilitzats fins el moment no són prou potents o complexos.

Per comprovar quina és realment la raó que ha impedit fins al moment aconseguir resultats que ofereixin millors prestacions es decideix configurar i entrenar una petita xarxa neuronal. No es pot aplicar cap tècnica de transferència d'aprenentatge perquè no existeixen xarxes entrenades amb aquestes dades, de manera que caldrà entrenar la xarxa des de zero. El risc en aquests casos és el sobreentrenament, de manera que haurà de ser realment molt senzilla.

Previ al disseny de la xarxa s'efectua una anàlisi per determinar les neurones de la capa inicial (les neurones de la capa de sortida en un model de classificació binari ha de ser 2). Aquestes proves i anàlisis prèvies estan recollides al ANNEX 7: CODI PYTHON: ANÀLISI PRÈVIA A LA XARXA NEURONAL.

5.3.7.1. PREPARACIÓ DE DADES

El primer que cal fer, com amb els models anteriors és disposar d'un conjunt de mostres de test.

La mida del conjunt de test serà el 20% de les mostres. És un valor habitual i s'ha testejat empíricament altres valors i no s'han aconseguit resultats millors amb altres valors. Es comprova que el nombre de mostres test que són de classe '0' i de classe '1' són representatives del conjunt.

A més, les dades es normalitzen i s'aplica una codificació del tipus 'One Hot Encoding' sobre el valor que volem predir.

5.3.7.2. DETERMINACIÓ DE PARÀMETRES

La funció d'activació de la capa de la sortida ha de ser Softmax perquè la literatura recent recomana aquest tipus de funció quan les classes són mútuament excloents com en casos binaris.[26][29][29]

En la capa de sortida el nombre de neurones ha de ser 2, una per cadascuna de les classes possibles. De la mateixa manera, l'entrada és tan gran com el nombre d'atributs que té la base de dades que fem servir (en aquest cas, 34 atributs). També cal determinar

altres paràmetres de disseny: mida del lot (batch), taxa d'aprenentatge (learning rate), nombre de capes, neurones per capa i optimitzador.

D'aquests decisions n'hi ha que son relativament senzilles de prendre, com per exemple, el nombre de capes. Si fem una xarxa molt complexa no podrem evitar sobreentrenar, de manera que es treballarà només amb una capa d'entrada i una de sortida.

El nombre de neurones d'aquesta capa tampoc pot ser molt gran. S'estima, a priori, que hauran de ser més de 2 i menys de 8.

La taxa d'aprenentatge és el que ha de permetre afinar l'entrenament. Si és molt petita o si és massa gran pot fer que sobreentreni o no convergeixi. Però els valors més habituals oscil·len entre 0,00001 i 0,001.

Pel que fa a l'Optimitzador, podríem utilitzar Adam sense més anàlisi ja que és l'optimitzador que es considera el més adequat per a la majoria de casos[25][26][29]. Però es provaran altres optimitzadors habituals, com el SGD, el Adagrad o el Adadelta per comprovar si el problema de classificació al que ens enfrontem és una de les excepcions a l'ús d'Adam.

Per determinar quins valors han de prendre la resta de paràmetres s'usa el mateix procediment seguit al llarg de tot el treball: definir una busca per graella per provar totes les combinacions possibles de neurones a la capa d'entrada, la taxa d'aprenentatge i l'optimitzador.

La mida del lot (batch) es fixa en 1 perquè després de fer diverses proves, resulta el millor valor, probablement per causa del nombre de mostres que, en cas d'agrupar-les en lots encara quedarien més reduïdes. S'ha testejat amb altres valors i no es pot evitar sobreentrenar el model o no convergir. La mida del batch té molta relació amb la taxa d'aprenentatge, per això es decideix fixar el batch a 1 per a la selecció dels millors paràmetres i determinar el valor final en l'entrenament a l'afinar el model junt amb la taxa d'aprenentatge.

Cal tenir en compte que en aquesta anàlisi prèvia s'està treballant amb un entrenament prefixat de 50 èpoques, mentre que en el disseny de la xarxa es programarà un mecanisme de parada (anomenat EarlyStopping) per a que la xarxa aturi el seu entrenament quan la pèrdua del conjunt de test deixi de millorar.

S'arriba a la conclusió que l'únic optimitzador que es pot descartar és Adadelta, perquè els altres tres prenen valors força similars, segons els resultats de la graella presentats en la Figura 43. També es considera que les neurones de la capa d'entrada han de ser 8.

La graella no permet concretar més, de manera que s'opta per configurar i entrenar 3 xarxes iguals, només canviant l'optimitzador per veure quina respon millor ajustant els valors de taxa d'entrenament i de lot.

Figura 43 Resultat de la graella que s'ha programat per a la selecció dels paràmetres de construcció i entrenament d'una xarxa neuronal.

Neurones	Optimitzador	Velocitat aprenentatge	Accuracy	F1	val_Accuracy	val_F1
4	Adam	0.0001	0.619048	0.580931	0.6875	0.686275
4	Adam	0.0005	0.888889	0.856305	0.8125	0.768116
4	Adam	0.0010	0.984127	0.980869	0.6250	0.600000
4	Adam	0.0050	1.000000	1.000000	0.8125	0.768116
4	SGD	0.0001	0.603175	0.558700	0.7500	0.733333
4	SGD	0.0005	0.777778	0.678571	0.9375	0.930736
4	SGD	0.0010	0.809524	0.724490	0.5625	0.458937
4	SGD	0.0050	0.984127	0.980869	0.6875	0.653680
4	Adadelta	0.0001	0.444444	0.382180	0.3750	0.365079
4	Adadelta	0.0005	0.523810	0.508836	0.6875	0.686275
4	Adadelta	0.0010	0.571429	0.498674	0.6875	0.653680
4	Adadelta	0.0050	0.428571	0.424949	0.5625	0.560784
4	Adagrad	0.0001	0.619048	0.497340	0.5625	0.360000
4	Adagrad	0.0005	0.714286	0.465094	0.6250	0.384615
4	Adagrad	0.0010	0.634921	0.545055	0.6250	0.500000
4	Adagrad	0.0050	0.746032	0.664894	0.7500	0.709091
8	Adam	0.0001	0.746032	0.664894	0.6875	0.653680
8	Adam	0.0005	0.936508	0.919437	0.8125	0.811765
8	Adam	0.0010	0.968254	0.961111	0.8750	0.854545
8	Adam	0.0050	1.000000	1.000000	0.6250	0.546559
8	SGD	0.0001	0.634921	0.572944	0.8750	0.866667
8	SGD	0.0005	0.730159	0.636333	0.5625	0.515152
8	SGD	0.0010	0.809524	0.709231	0.5625	0.360000
8	SGD	0.0050	0.984127	0.980869	0.6875	0.653680
8	Adadelta	0.0001	0.380952	0.338271	0.3750	0.333333
8	Adadelta	0.0005	0.539683	0.532138	0.2500	0.250000
8	Adadelta	0.0010	0.412698	0.412698	0.4375	0.417004
8	Adadelta	0.0050	0.539683	0.507945	0.7500	0.733333
8	Adagrad	0.0001	0.619048	0.418462	0.7500	0.666667
8	Adagrad	0.0005	0.666667	0.529684	0.3750	0.272727
8	Adagrad	0.0010	0.571429	0.541880	0.6875	0.686275
8	Adagrad	0.0050	0.825397	0.774194	0.6875	0.676113

Font: elaboració pròpia

5.3.7.3. DISSENY I ENTRENAMENT

La xarxa neuronal, per tant, té 2 capes una de 8 neurones d'entrada i una de sortida de 2 neurones que ofereix un total de 298 paràmetres entrenables.

Figura 44 Resum de la xarxa neuronal bàsica implementada com a model de classificació.

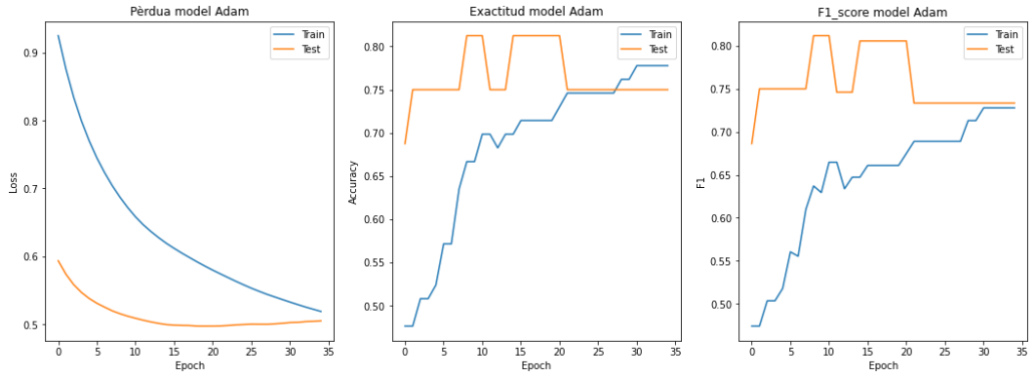
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 8)                 280
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 18
=================================================================
Total params: 298
Trainable params: 298
Non-trainable params: 0
_________________________________________________________________
```

Font: elaboració pròpia

En l'annex ANNEX 8: CODI PYTHON: XARXA NEURONAL s'hi recull el procés de d'entrenament d'aquesta xarxa amb els 3 models i l'ajust fi. Es fan diverses proves amb la taxa d'aprenentatge i mida de lot fins que s'arriba a un valor que sembla que respon de forma òptim per a tots els optimitzadors.

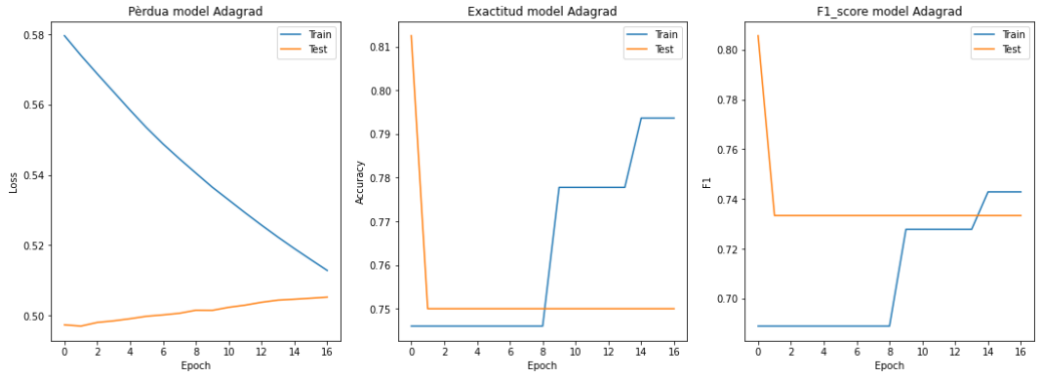
No es limiten les èpoques d'entrenament, s'ha configurat un nombre molt elevat d'èpoques i s'ha definit un procés d'aturada (early stopping en anglès) per a deixar d'entrenar en el moment que la pèrdua de validació deixi de millorar. Es configura amb una paciència de 15 èpoques però amb recuperació dels millors pesos.

Figura 45 Història de la xarxa amb un optimitzador Adam. Evolució de la pèrdua l'exactitud i el F1.



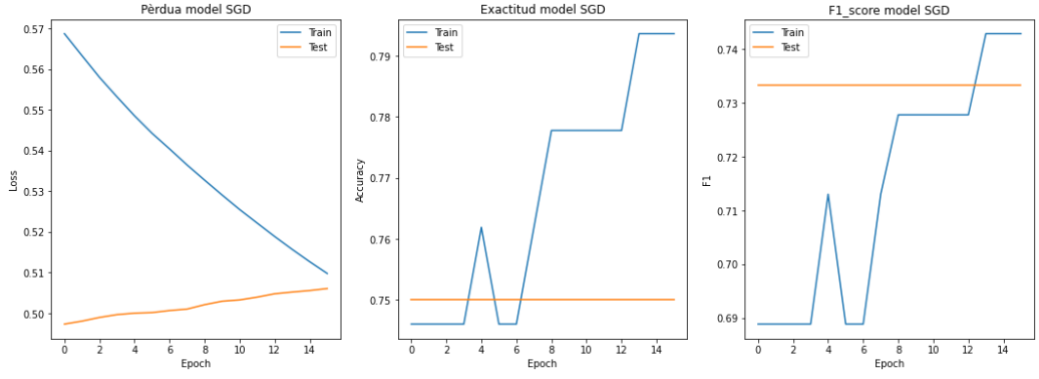
Font: elaboració pròpia

Figura 46 Història de la xarxa amb un optimitzador Adagrad. Evolució de la pèrdua l'exactitud i el F1.



Font: elaboració pròpia

Figura 47 Història de la xarxa amb un optimitzador SGD. Evolució de la pèrdua l'exactitud i el F1.



Font: elaboració pròpia

La Figura 45, la Figura 46 i la Figura 47 mostren la 'historia' de l'entrenament de cadascuna de les tres xarxes neuronals construïdes. Podem observar resultats finals molt similars però després d'un nombre d'èpoques diferent per a cada optimitzador i després d'una convergència diferent cap al valor final.

5.3.7.4. PREDICCIONS

Pel tipus de capa de sortida utilitzada (una softmax), a la sortida de la xarxa neuronal no hi tenim una predicció de classe sinó un array de probabilitats de ser cadascuna de les dues classes, per tant, no obtenim una predicció sinó un array de 2 valors per cada mostra, el primer valor és la probabilitat de ser classe '0' i el segon valor és la probabilitat de ser classe '1'.

Per determinar definitivament quina és la predicció de classe, s'escull la que té major probabilitat. Això només és fa als efectes de poder elaborar les matrius de confusió.

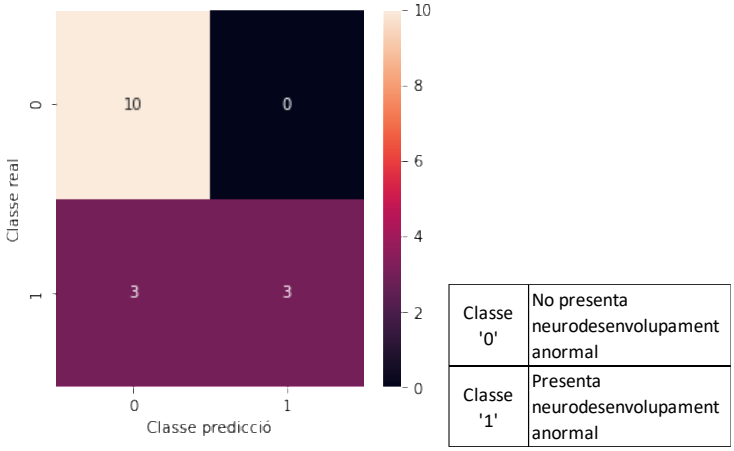
Cal recordar que una mateixa xarxa neuronal, entrenada de la mateixa manera vegades, no té perquè donar sempre els mateixos resultats.

En la majoria de les proves efectuades l'Optimitzador Adam ha donat resultats millors que la resta, però en altres els resultats han estat idèntics. S'ha guardat el model amb optimitzador Adam que ha donat millor resultat com a model definitiu.

Es recull en la Figura 48 la matriu de confusió obtinguda.

En l'annex ANNEX 8: CODI PYTHON: XARXA NEURONAL a més d'aquesta matriu de confusió també s'ha elaborat la matriu de confusió de tot el conjunt de dades només als efectes de comprovar si el model estava sobreentrenat.

Figura 48 Matriu de confusió del millor model de xarxa neuronal.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

Font: elaboració pròpia

Els millors resultats d'exactitud i F1 han estat 81,25% i 79,35% respectivament.

5.3.8. COMBINACIÓ DE MODELS

Malgrat que la xarxa neuronal ha permès disposar d'un model amb una ratio predictiva millor que la que s'ha aconseguit amb models anteriors, no s'han esgotat tots els recursos

que tenim a la nostra disposició per a millorar resultats. I un d'aquests recursos és la combinació de models.

Es descriuen els resultats obtinguts en les versions seqüencials de la combinació de models, encara que a ANNEX 9: CODI PYTHON: COMBINACIÓ DE MODELS també s'ha analitzat com a opció l'augment de gradient (Gradient Boosting) com a alternativa paral·lela, pero els resultats no han millorat els preexistents.

A nivell de metodologia, cal mencionar encara que resulti repetitiu, que abans d'aplicar cada model s'efectua una graella per a seleccionar els millors paràmetres.

Abans de procedir a dissenyar els models de apilament (stacking en anglès) i de cascada (cascading en anglès), és necessari fer un resum dels resultats dels models ja entrenats.

Observem en la Figura 49 adjunta que els dos millors models configurats i entrenats són la regressió logística i el LinearSVM. En tercer lloc hi havia en k veïns més propers, a bastant distància.

La bondat dels models no es mesura amb la mètrica d'exactitud sinó pel valor de F1 que és el que ens interessa.

Per tant, quan es fa un apilament o una cascada amb 2 models es farà amb la regressió logística i el LinearSVM, i si és fa amb tres, s'hi afegirà el k veïns més propers.

Figura 49 Resultats dels models de classificació utilitzats.

Model	Accuracy (%)	F1 score (%)	TN	FP	FN	TP
k veïns més propers	68,75	60,71	10	0	5	1
Arbre de decisió	56,25	51,81	8	2	5	1
Bosc aleatori	62,5	56,25	9	1	5	1
SVM Kernel linial	56,25	56,78	6	4	3	3
SVM Kernel radial	62,5	48,08	10	0	6	0
SVM Kernel polimonial	62,5	56,25	9	1	5	1
SVM Kernel sigmoide	62,5	48,08	10	0	6	0
Linear SVM	68,75	69,13	7	3	2	4
Regressió Logística	68,75	65,58	9	1	4	2
Xarxa Neuronal (Adam)	81,25	79,35	10	0	3	3
Xarxa Neuronal (Adagrad)	75	75	8	2	2	4
Xarxa Neuronal (SGD)	75	75	8	2	2	4

TN = True Negative (verdaders negatius) **FP = False Positive (falsos positius)**
FN = False Negative (falsos negatius) **TP = True Positives (verdaders positius)**

Font: elaboració pròpia

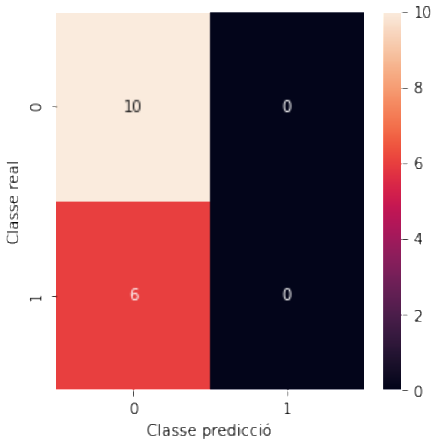
5.3.8.1. APILAMENT

En primera instància es dissenya un model que combina els dos millors classificadors. Davant dels resultats obtinguts, s'opta per intentar combinar els tres millors classificadors, sense que els resultats canviin en absolut.

Els resultats obtinguts estan il·lustrats en la Figura 50, però numèricament, un apilament tant de 2 com de 3 models ofereix una exactitud de 62,5% i una F1 de 48,08%.

És a dir, aquesta combinació no millora els resultats sino que els empitjora.

Figura 50 Matriu de confusió obtingut aplicant apliament, tant de 2 models com de 3 models.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

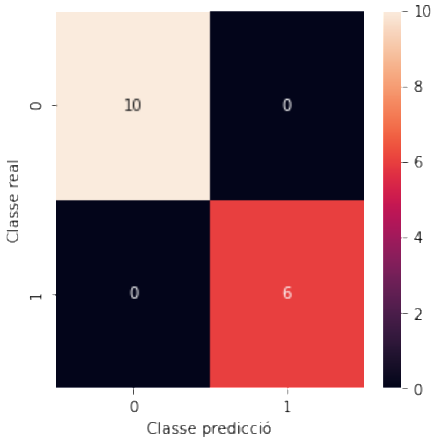
Font: elaboració pròpia

5.3.8.2. CASCADA

En aquest model també s'obtenen els mateixos resultats idèntics tant quan es fa una cascada (cascading) amb els dos millors models com si es fa amb tres models.

La diferència és que en aquest cas s'obtenen prediccions sobre el conjunt de test perfectes: 100% d'exactitud i de F1. Tal com mostra la matriu de confusió.

Figura 51 Matriu de confusió obtingut aplicant cascading, tant de 2 models com de 3 models.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

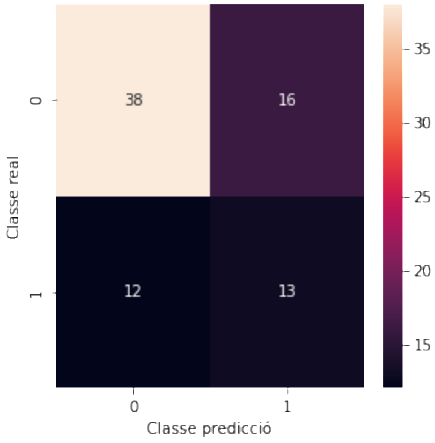
Font: elaboració pròpia

5.3.8.3. VALIDACIÓ

Els models d'apilament i cascada utilitzen les prediccions dels models base usant dades de test del model base, és a dir, amb dades que no han participat en l'entrenament del model.

Per tant aquests models de combinació tenen, en aquest cas, només 16 mostres per a entrenar-se. I no es disposa de dades de test addicionals per poder contrastar els resultats obtinguts amb dades que no hagin estat usades ni en els models base ni en els models combinats.

Figura 52 Matriu de confusió obtingut sobre el global de totes les mostres disponibles.



Classe '0'	No presenta neurodesenvolupament anormal
Classe '1'	Presenta neurodesenvolupament anormal

Font: elaboració pròpia

Per tant, no hi ha possibilitat de validar aquests models de forma convencional. Alternativament, es recorre al global de les dades, és a dir, s'aplica el model no només a les prediccions del conjunt de test sinó a les prediccions de tot el conjunt de dades inicials.

Un cop aplicat el model de cascading al global de les mostres amb els tres models seleccionats, els resultats obtinguts ens oferèixen les mètriques següents: 68,35% d'exactitud i 68,52% de F1.

5.3.9. COMBINACIONS AMB XARXES NEURONALS

Es defineix una nova cascada en què s'incorporen les prediccions de la xarxa neuronal. No la predicció de la classe (recordem que es fa escollint el valor amb major proporció) sino els valors de probabilitat a la sortida. Aporta més informació el percentatge de probabilitat d'una classe que el valor de classe de la predicció.

S'ha testejat la possibilitat d'incorporar les prediccions de les tres xarxes neuronals entrenades, una amb cada optimitzador, però finalment s'opta per utilitzar només les dades de la xarxa entrenada amb l'optimitzador Adam perquè no s'ha detectat cap millora en els resultats si s'usen les tres xarxes.

Aquesta cascada, es fa afegint aquesta nova predicció a les dades del model de cascada de 3 models anterior. És a dir, s'efectua un model de cascada de 4 models: Xarxa neuronal amb optimitzador Adam, k veïns més propers, LinearSVM i regressió logística.

Es pot comprovar el codi emprat al ANNEX 10: CODI PYTHON: COMBINACIÓ DE MODELS AMB XARXA NEURONAL.

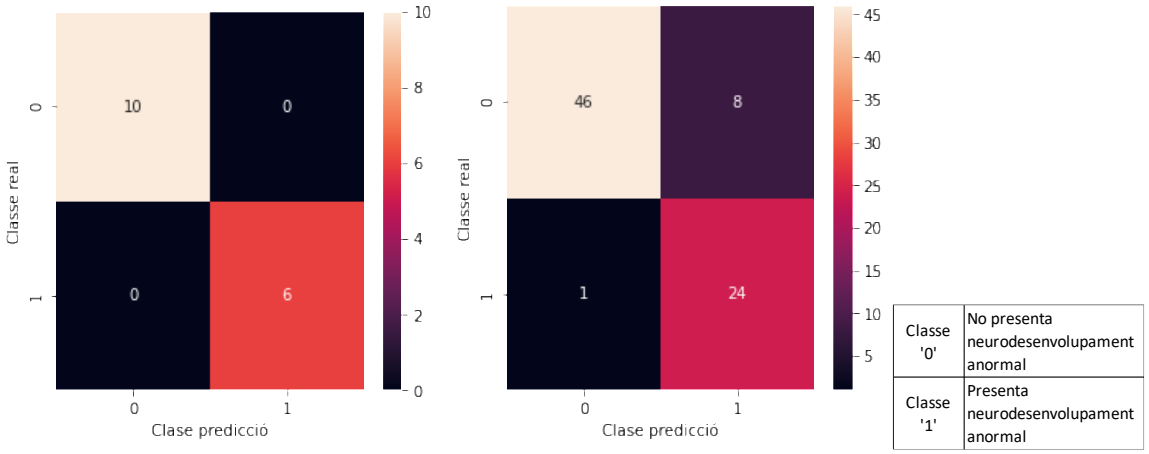
S'aconsegueix un model que aconsegueix, com ja passava amb les cascades sense xarxa neuronal, un 100% d'exactitud i de F1 contra el conjunt de test. Però la millora s'aprecia quan s'aplica sobre el total de les mostres, ja que l'exactitud arriba al 88,61% i a un 88,91% de F1. Les dues matrius de confusió, amb les dades test i amb el global de les dades es presenta en la Figura 53.

A més, tal com mostra la matriu de confusió, ens trobem que hi ha una predicció

encertada de 70 de les 79 mostres. I els errors de predicció consisteixen en 1 fals negatiu i 8 falsos positius. Resulta desitjable tenir falsos positius per davant de tenir falsos negatius en el marc de la sanitat.

Malgrat saber que aquesta verificació no és convencional sí és indicativa de la bondat del model.

Figura 53 Matriu de confusió obtingut aplicant una cascada de 4 models sobre les dades de test (esquerra) i sobre el global de les dades (dreta).



Font: elaboració pròpia

5.3.10. RESUM DE RESULTATS DE CLASSIFICACIÓ

Donat que s'han aplicat diferents models i algoritmes i combinacions, es fa necessari efectuar un resum de resultats.

Figura 54 Resum de resultats dels diferents models de classificació incloent combinacions.

Model	Accuracy (%)	F1 score (%)	TN	FP	FN	TP
k veïns més propers	68,75	60,71	10	0	5	1
Arbre de decisió	56,25	51,81	8	2	5	1
Bosc aleatori	62,5	56,25	9	1	5	1
SVM Kernel linial	56,25	56,78	6	4	3	3
SVM Kernel radial	62,5	48,08	10	0	6	0
SVM Kernel polimonial	62,5	56,25	9	1	5	1
SVM Kernel sigmoide	62,5	48,08	10	0	6	0
Linear SVM	68,75	69,13	7	3	2	4
Regressió Logística	68,75	65,58	9	1	4	2
Xarxa Neuronal (Adam)	81,25	79,35	10	0	3	3
Xarxa Neuronal (Adagrad)	75	75	8	2	2	4
Xarxa Neuronal (SGD)	75	75	8	2	2	4
Gradient Boosting	50	47,27	7	3	5	1
Stacking (Linear SVM, Regressió Logística)	62,5	48,08	10	0	6	0
Stacking (Linear SVM, Regressió Logística, K-NN)	62,5	48,08	10	0	6	0
Cascading (Linear SVM, Regressió Logística)	100	100	10	0	0	6
Cascading (Linear SVM, Regressió Logística, K-NN)	100	100	10 (38)	0 (16)	0 (12)	6 (13)
Cascading (Xarxa Neuronal Adam, Linear SVM, Regressió Logística, K-NN)	100	100	10 (46)	0 (8)	0 (1)	6 (24)

TN = True Negative (verdaders negatius) **FP = False Positive (falsos positius)**
FN = False Negative (falsos negatius) **TP = True Positives (verdaders positius)**

Font: elaboració pròpia

6. CONCLUSIONS

La ciència de dades ens ha de permetre, a partir d'un conjunt de dades, decidir quins mètodes, tècniques o processos són els més adients per a extreure el coneixement necessari per a les preguntes que ens plantegem i també ens ha de permetre aplicar-los per a obtenir la resposta que busquem.

Encara que la ciència de dades es fa especialment necessària quan el volum de dades és tant ingent que no existeixen alternatives per a trobar aquestes respostes, hauria de ser fidel als seus orígens i ser capaç de facilitar-nos eines i mecanismes per extreure coneixement no evident de qualsevol dada.

Aquest treball final de master ha posat a prova aquesta realitat.

S'ha treballat amb un conjunt de dades extremadament petit però que en l'àmbit de la medicina es considerava perfectament caracteritzat i molt gran.

La VMG és una la dilatació d'un o ambdós ventricles laterals del cervell i s'identifica entre el 0,3 i el 22 per 1000 dels embarassos. Aquest treball se centra en VMG aïllades i que poden ser entre un 25% i un 60% dels casos, segons la literatura consultada. Això ens porta a una casuïstica amb baixissima incidència, però que no per això mereix menor esforç.

Efectivament disposar de poques dades ha estat un enorme handicap. No tenir coneixements mèdics per posar les dades en context i poder detectar relacions o correlacions o dependències en alguns casos tampoc ha facilitat la tasca.

Actualment hi ha consens en basar-se en els paràmestre LA i RA per a determinar, únicament pel valor numèric que se n'obté, si ens trobem o no davant d'un cas de VMG. S'ha detectat una forta correlació amb dos paràmetres addicionals com són el LAH i el RAH (banya anterior esquerra i dreta), de la que no s'havia localitzat literatura que s'hi fes referència. Tot i això, no s'ha pogut oferir cap millora substancial en el procés de caracterització de la VMG.

Pel que fa a la caracterització de la prognosi (l'evolució futura de la patologia), tampoc s'ha pogut aportar informació a aquest àmbit. Cap dels paràmestre ni cap de les combinacions de paràmetres disponibles permetia preveure quins casos desenvocarien en problemes greus o lleus de neurodesenvolupament.

Per això, la determinació de la caracterització de la VMG o de la seva prognosi no és

possible amb un nombre de mostres tant reduït.

La següent conclusió a la que s'arriba és que els models de regressió amb predicció continua requereixen moltes més dades que una regressió a valors discontius (és a dir, a una classificació). I més si es tracta d'una classificació binaria com en el nostre cas.

A continuació es contraposen les conclusions a les que s'ha arribat amb el que s'havia conclòs en l'estat de l'art:

- Que l'element principal per comparar sistemes de classificació és l'exactitud

En la literatura consultada sempre es partia de jocs de dades petits però balancejats. En el nostre cas, en canvi, s'ha optat pel seu equivalent per a casos desbalancejats, el F1, per no haver de treballar amb l'exactitud, la precisió i el recall per separat. Malgrat tot, s'ha mantingut sempre un seguiment de l'exactitud.

- Que en jocs de dades amb menys de 2000 mostres és molt difícil obtenir resultats superiors al 75-80%.

En la major part del treball això es compleix ja que no s'ha aconseguit superar el límit del 70% fins que hem introduït les xarxes neuronals i les combinacions de models. De manera que, hem de concloure que, malgrat ser eines molt potents per a treballar amb grans volums de dades, poden arribar a quotes de precisió i exactitud que no s'aconsegueixen amb cap model individualment ni amb la combinació dels millors d'ells.

- Que el preprocessat i la selecció de paràmetres és clau.

Aquest extrem no s'ha pogut confirmar ja que no s'ha pogut fer una selecció conscient i ponderada dels paràmetres utilitzats. Malgrat disposar de molts atributs no tots eren complerts, de manera que s'ha hagut de treballar amb tots els possibles.

La mostra era de 81 casos i es disposava d'entre 34 i 37 atributs (segons el cas). No s'ha tingut ni la ocasió ni els coneixements per determinar els que eren reiteratius. Tot i que s'ha fet una anàlisi de correlacions per procurar tenir paràmetres el més independents possibles. Però no s'ha pogut experimentar amb la selecció d'uns o altres valors.

El que sí que s'ha pogut tractar és el preprocessat. Malgrat seguir pensant que és una bona praxi, en el nostre cas, on tot eren mesures internes de regions cerebrals dels nadons, la dispersió de dades era minsa i els resultats amb i sense normalització no aportava cap canvi.

Hi ha models de regressió molt potents, però classificar ha resultat més senzill.

Poques dades poden ser importants posades en context, no disposar del context és un enorme problema: a l'hora de seleccionar, a l'hora de recuperar dades perdudes i a l'hora d'interpretar resultats.

- Elecció acurada dels algoritmes

Efectivament, el treball presentat ha treballat amb els algoritmes d'aprenentatge supervisat més importants segons la literatura consultada: K-NN, arbres de decisió i

random forests, SVMs, regressions lineals i logístiques i xarxes neuronals. I tal com s'esperava els millors resultats s'obtenen amb el KNN i el SVM. En concret amb el Linear SVM. Però també s'obté resultats molt bons la regressió logística. Tal com es preveia, però, cap dels models arriba al llindar del 70% de F1.

El que no recollia la literatura i que sí conclou el meu treball és que, resulta molt útil aplicar tècniques de combinació de models. Especialment de models secuencials. I especialment la cascada, perquè l'apilament només permetia utilitzar com a valors les prediccions. En un model tan petit i amb gran desbalanceig no donava resultats favorables. En canvi, amb la cascada s'aconseguia una lleugera millora.

Finalment, la darrera conclusió a la que arribo és que les xarxes neuronals combinades amb altres models més senzills però potents poden oferir solucions a jocs de dades realment reduïts.

Així mateix, considero que cal afegir un element addicional a la llista d'aspectes a tenir en compte quan es treballa amb jocs de dades tan petits: la determinació del conjunt d'entrenament i de test. És realment complexe prendre la decisió sobre el nombre de mostres que es sacrifiquen per a poder entrenar un millor model. I no poder disposar d'un conjunt de validació, complementari al de test, dificulta seguir un procediment d'optimització convencional.

Sense cap mena de dubte aquest ha estat el punt del procés que genera major indeterminació i, de fet, és el que fa que els resultats d'aquest treball hagin de ser testejats a posteriori amb noves dades per assegurar que no s'ha aconseguit un model sobreentrenat.

7. LÍNIES DE TREBALL FUTURES

Les línies de treball futures que es proposen són les que es relacionen a continuació:

- Cal intentar disposar de dades noves per poder comprovar la validesa del model generat.

Les dades emprades per al model no han estat molt nombroses. I 34-40 paràmetres no és una quantitat molt gran d'atributs.

Per tant, el següent pas en la línia de treball hauria de ser intentar obtenir noves dades per validar el model. I en cas que no es validi, el que és segur és que es podrà reentrenar el model per ajustar-lo.

- Validar els resultats amb dades provinents d'ecografies.

En aquest treball s'han utilitzat dades extretes de ressonàncies magnètiques perquè la base de dades havia estat elaborada per a un estudi clínic que incloia aquesta prova. Però resulta més fàcil i habitual disposar de dades ecogràfiques (ultrasò).

Caldria, per tant, crear un model amb dades d'ultrasò, però assegurant que les dades obtingudes siguin tan robustes com les que s'obtenen amb una ressonància magnètica.

- Consensuar criteris per a recuperar més dades perdudes.

Moltes de les dades disponibles no han pogut ser utilitzades perquè hi havia molts elements perduts. El coneixement mèdic podria ajudar a determinar dependències o paràmetres associats que permetessin recuperar aquestes mostres amb la mitjana o la mediana dels casos similars, per exemple. També es podrien recuperar de la història clínica si hi apareixen.

- Recodificar els valors discrets dels tests de Bayley i Vineland per intentar classificar amb més classes.

Fins a cert punt, treballar amb classe binària simplifica molt el problema, però en cas d'error, aquest és màxim. Si es poguessin recodificar els resultats del test de Bayley i de Vineland amb valors 0, 1 i 2, es podrien sumar els resultats i tindriem diagnòstics en 5 nivells, del 0 al 4. De manera que amb un model ben entrenat l'error creuat de classificació seria menor, segur, entre les categories 0 i 4, que són els extrems.

- Revisar la incidència dels paràmetres RAH i LAH amb la VMG.

En l'anàlisi efectuat s'ha pogut intuir que la RAH i la LAH poden tenir certa relació amb

la presència de VMG d'una forma similar a com ho fan tradicionalment la LA i la RA.

No han resultat útils per a caracteritzar la prognosi, però potser permeten una detecció precoç del diagnòstic.

- Dissenyar una nova xarxa neuronal.

El disseny d'una xarxa neuronal no estava dins del planificat. Realment estava previst trobar un model millor basat en els algoritmes més potents o en la seva combinació. Davant la impossibilitat d'arribar al 75-80% previst inicialment pel llegit a la literatura, s'ha optat per una xarxa neuronal.

Per tant, no s'ha pogut dedicar temps suficient a afinar correctament la xarxa. Encara que s'ha fet un enorme esforç per a seleccionar els millors paràmetres, no s'ha pogut optimitzar. A priori, pel nombre de mostres disponibles s'ha pensat, per pura intuició, que havia de ser una xarxa el més senzilla possible per no sobrealimentar, però potser caldria aplicar més capes més senzilles o capes més sofisticades o eines més pròpies de l'aprenentatge profund que de la mineria de dades.

- Disposar d'informació per fer una analisi de paràmetres evolutius.

La base de dades disposava de dades d'ecografies de 26 i 30 setmanes. Però s'ha treballat amb les dades de la resonància magnètica perquè era la que oferia més dades disponibles. La majoria dels registres només disposava de les dades d'una de les dues ecografies i fins i tot en alguns casos no hi havia dades de cap de les ecografies. La literatura científica apunta a que la prognosi està molt relacionada amb l'evolució de les dimensions de LA i RA. Aquesta anàlisi no s'ha pogut fer.

8. ANNEXOS

A continuació es presenta informació addicional que considero que no és imprescindible per a entendre aquest Treball Final però sí que és important per a disposar d'una imatge complerta de l'abast del treball que s'ha dut a terme.

No s'hi inclouen les fonts de dades que se'm van facilitar per a dur a terme el treball, perquè són dades mèdiques.

Tampoc incloc el joc de dades que finalment he usat per al projecte perquè està suficientment referenciat en el codi en python que he elaborat i que sí que incloc.

ANNEX 1: LLISTAT DE PARÀMETRES SELECCIONATS I DESCRIPCIÓ

Formen part del fitxer 'bbdd_TFM.xlsx', que és la base de dades de la que parteix aquest projecte, els atributs o paràmetres següents:

- ID: Identificador únic dels subjectes d'estudi
- Sexe: Sexe del nadó. Valors possibles: Masc ; Fem
- LA: Amplada de l'Atrium del Ventricle Dret (Right Atrium)
- RA: Amplada de l'Atrium del Ventricle Esquerre (Left Atrium)
- Diagnostic: Diagnòstic de la Ventriculomegàfia. Valors possibles: No VMG; VMG lateral esquerra; VMG lateral dreta; VMG bilateral
- Bayley: Resultat del test Baley. El paràmetre reculll el "Bayley test composite Score"
- Brazelton: Resultat numèric final del test Brazelton
- A_BPD: Biparietal diameter (pla axial)
- A_OFD: Occipital frontal diameter (pla axial)
- A_LID: Left insular depth (pla axial)
- A_RID: Right insular depth (pla axial)
- A_LSFD: Left sylvian fissure depth (pla axial)
- A_RSFD: Right sylvian fissure depth (pla axial)
- A_LIDSF: Left inner distance of sylvian fissure (pla axial)
- A_RIDSF: Right inner distance of sylvian fissure (pla axial)
- A_LMDSF: Left median distance of sylvian fissure (pla axial)
- A_RMDSF: Right median distance of sylvian fissure (pla axial)
- A_LODSF: Left outer distance of sylvian fissure (pla axial)
- A_RODSF: Right outer distance of sylvian fissure (pla axial)
- A_LPOFD: Left parieto-occipital fissure depth (pla axial)
- A_RPOFD: Right parieto-occipital fissure depth (pla axial)
- A_LOPOFD: Left outer parieto-occipital fissure distance (pla axial)
- A_ROPOFD: Right outer parieto-occipital fissure distance (pla axial)
- A_LPOVD: Left parieto-ocipital-ventricular distance (pla axial)
- A_RPOVD: Right parieto-ocipital-ventricular distance (pla axial)
- C_LCIFD: Left cingulate fissure depth (pla coronal)
- C_RCIFD: Right cingulate fissure depth (pla coronal)

- C_LCAFD: Left calcarine fissure depth (pla coronal)
- C_RCAFD: Right calcarine fissure depth (pla coronal)
- C_LCAVD: Left calcarine ventricular distance (pla coronal)
- C_RCAVD: Right calcarine ventricular distance (pla coronal)
- TCD: Transverse cerebellar diameter
- PF: Posterior fossa
- CCL: Corpus callosum length
- VCCD: Vermis cranio-caudal diameter
- LAH: Left anterior horn
- RAH: Right anterior horn

ANNEX 2: CODI PYTHON: PREPARACIÓ DE DADES

TREBALL FINAL DE MASTER
MERITXELL BOSCH I GARCIA
2020/21-2 · Màster universitari en ciència de dades (*Data science*)
Estudis d'informàtica, multimèdia i telecomunicacions

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import numpy as np
```

In [2]:
```
#Importació del dataset

#data=pd.read_csv("../input/data-tfm/bbdd_TFM.csv", sep = ";", decimal="," ) # a Kaggle
data=pd.read_csv("bbdd_TFM.csv", sep = ";", decimal="," )  # a Anaconda + JupyterLab
```

1. PREPARACIÓ DADES

In [3]:
```
# primera visió de dades
data.head(10)
```

Out[3]:

	ID	Sexe	LA	RA	Diagnostic	Bayley	Brazelton	Vineland	A_BPD	A_OFD	...	C_LCAFD	C_RCAFD	C_LCAVD	C_RCAVD	TCD	PF	CCL	VCCD	LAH	R
0	1	Masc	7.16	5.00	No VMG	NaN	43.714286	NaN	89.75	103.94	...	14.31	14.81	2.25	4.41	43.29	8.20	38.40	21.78	2.59	3
1	2	Masc	14.07	13.00	VMG bilateral	74.0	45.714286	SI	85.55	107.81	...	9.50	11.70	1.76	1.76	39.86	10.55	37.27	18.25	5.40	6
2	3	Masc	10.80	4.69	VMG lateral esquerra	NaN	49.857143	SI	72.11	89.16	...	7.03	18.00	1.76	1.76	31.65	6.45	37.45	21.74	2.11	5
3	4	Masc	8.89	11.12	VMG lateral dreta	77.0	47.714286	SI	89.13	105.79	...	15.20	11.86	2.11	1.13	40.45	5.04	29.09	18.75	1.85	2
4	5	Masc	11.20	6.83	VMG lateral esquerra	NaN	NaN	NaN	66.69	81.43	...	4.97	8.71	2.93	1.66	27.95	6.23	29.56	13.28	2.62	2
5	6	Masc	4.46	3.75	No VMG	110.0	44.714286	NaN	79.57	106.64	...	13.01	14.75	1.76	1.76	41.62	7.50	37.57	24.87	2.49	1
6	7	Masc	10.56	10.69	VMG bilateral	87.0	47.571429	SI	73.64	90.12	...	5.31	6.55	2.99	1.85	32.27	5.02	34.33	15.18	4.97	4
7	8	Fem	7.64	5.98	No VMG	93.0	NaN	NaN	74.05	90.69	...	6.83	8.14	2.62	2.11	32.98	11.95	33.81	15.94	2.93	2
8	9	Masc	5.40	11.39	VMG lateral dreta	NaN	41.571429	NaN	64.36	77.39	...	7.97	6.12	1.17	1.85	26.16	5.04	30.76	13.73	2.26	4
9	10	Fem	15.78	8.94	VMG lateral esquerra	NaN	NaN	NaN	96.21	107.19	...	10.77	14.87	1.85	1.85	50.75	6.31	45.21	19.69	3.75	4

10 rows × 38 columns

In [4]:
```
#visió de tipus de dades
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 38 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           81 non-null     int64
 1   Sexe         81 non-null     object
 2   LA           81 non-null     float64
 3   RA           81 non-null     float64
 4   Diagnostic   81 non-null     object
 5   Bayley       42 non-null     float64
 6   Brazelton    64 non-null     float64
 7   Vineland     25 non-null     object
 8   A_BPD        80 non-null     float64
 9   A_OFD        80 non-null     float64
 10  A_LID        80 non-null     float64
 11  A_RID        80 non-null     float64
 12  A_LSFD       80 non-null     float64
 13  A_RSFD       80 non-null     float64
 14  A_LIDSF      80 non-null     float64
 15  A_RIDSF      80 non-null     float64
 16  A_LMDSF      80 non-null     float64
 17  A_RMDSF      80 non-null     float64
 18  A_LODSF      80 non-null     float64
 19  A_RODSF      80 non-null     float64
 20  A_LPOFD      80 non-null     float64
 21  A_RPOFD      80 non-null     float64
 22  A_LOPOFD     80 non-null     float64
 23  A_ROPOFD     80 non-null     float64
 24  A_LPOVD      80 non-null     float64
 25  A_RPOVD      80 non-null     float64
 26  C_LCIFD      80 non-null     float64
 27  C_RCIFD      80 non-null     float64
 28  C_LCAFD      80 non-null     float64
 29  C_RCAFD      80 non-null     float64
 30  C_LCAVD      80 non-null     float64
 31  C_RCAVD      80 non-null     float64
 32  TCD          79 non-null     float64
```

```
 33  PF          79 non-null     float64
 34  CCL         80 non-null     float64
 35  VCCD        80 non-null     float64
 36  LAH         80 non-null     float64
 37  RAH         80 non-null     float64
dtypes: float64(34), int64(1), object(3)
memory usage: 24.2+ KB
```

In [5]:
```
# canvi de paràmetres a tipus qualitatiu
data = data.astype({'ID': 'object'})
```

In [6]:
```
# mètriques estadístiques bàsiques
data.describe()
```

Out[6]:

	LA	RA	Bayley	Brazelton	A_BPD	A_OFD	A_LID	A_RID	A_LSFD	A_RSFD	...	C_LCAFD	C_RCAFD	C_LCAVD	C_R
count	81.000000	81.000000	42.000000	64.000000	80.000000	80.000000	80.000000	80.000000	80.000000	80.000000	...	80.000000	80.000000	80.00000	80.0(
mean	8.199259	8.003827	90.142857	44.651786	75.701125	92.044000	22.410125	22.583500	12.892125	12.368625	...	9.277250	10.066000	2.49700	2.6(
std	3.323337	3.306623	14.350272	2.641631	7.790142	8.316367	2.498251	2.560044	2.264685	1.833851	...	2.763682	3.520595	1.00972	1.1(
min	2.930000	2.260000	60.000000	36.857143	61.200000	77.390000	17.390000	18.250000	8.480000	8.140000	...	2.110000	4.140000	1.13000	0.8(
25%	5.400000	5.500000	78.500000	43.357143	70.457500	86.025000	20.345000	21.062500	11.430000	11.180000	...	7.320000	7.527500	1.76000	1.7(
50%	7.160000	6.550000	92.500000	44.785714	73.870000	90.485000	21.935000	22.085000	12.740000	12.320000	...	9.170000	9.325000	2.29500	2.3(
75%	10.610000	10.690000	100.750000	46.571429	79.382500	95.527500	24.605000	24.072500	14.110000	13.565000	...	10.630000	11.807500	3.19750	3.2(
max	15.780000	15.990000	115.000000	49.857143	96.210000	114.550000	28.610000	28.850000	18.760000	16.910000	...	16.280000	22.850000	4.97000	6.8(

8 rows × 34 columns

In [7]:
```
# preparar la llista de tots els paràmetres a banda dels diagnostics
parametres = data.columns.values[2:4]
parametres = np.append(parametres, data.columns.values[8:])
```

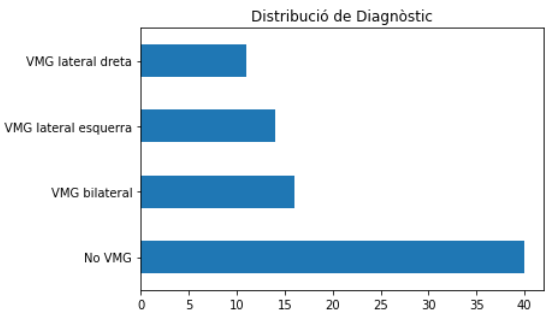
In [8]:
```
# llistat de paràmetres
parametres
```

Out[8]:
```
array(['LA', 'RA', 'A_BPD', 'A_OFD', 'A_LID', 'A_RID', 'A_LSFD', 'A_RSFD',
       'A_LIDSF', 'A_RIDSF', 'A_LMDSF', 'A_RMDSF', 'A_LODSF', 'A_RODSF',
       'A_LPOFD', 'A_RPOFD', 'A_LOPOFD', 'A_ROPOFD', 'A_LPOVD', 'A_RPOVD',
       'C_LCIFD', 'C_RCIFD', 'C_LCAFD', 'C_RCAFD', 'C_LCAVD', 'C_RCAVD',
       'TCD', 'PF', 'CCL', 'VCCD', 'LAH', 'RAH'], dtype=object)
```

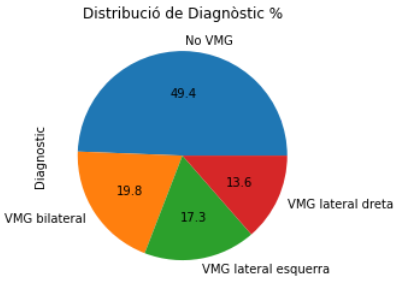
In [9]:
```
# recompte de participants per diagnòstic
data.value_counts(data['Diagnostic'])
```

Out[9]:
```
Diagnostic
No VMG                  40
VMG bilateral           16
VMG lateral esquerra    14
VMG lateral dreta       11
dtype: int64
```

In [10]:
```
plot = data['Diagnostic'].value_counts().plot(kind='barh',
                                              title='Distribució de Diagnòstic')
```



In [11]:
```
plot = data['Diagnostic'].value_counts().plot(kind='pie', autopct='%.1f',
                                              title='Distribució de Diagnòstic %')
```



In [12]:
```
pd.crosstab(index=data['Sexe'],
            columns=data['Diagnostic'], margins=True)
```

Out[12]:

Diagnostic	No VMG	VMG bilateral	VMG lateral dreta	VMG lateral esquerra	All

Diagnostic Sexe	No VMG	VMG bilateral	VMG lateral dreta	VMG lateral esquerra	All
Sexe					
Fem	14	5	1	1	21
Masc	26	11	10	13	60
All	40	16	11	14	81

In [13]:
```
# creo columnes de classe per al sexe i el diagnostic
data['Sexe_cod']=np.where(data['Sexe']=="Masc",0,1)

data = data.astype({'Sexe_cod': 'object'})

data['Diagnostic_cod']=data['Diagnostic']
target_map = {'No VMG':0,
              'VMG lateral esquerra':1,
              'VMG lateral dreta':2,
              'VMG bilateral':3}

data['Diagnostic_cod'] = data['Diagnostic_cod'].apply(lambda x: target_map[x])


data = data.astype({'Diagnostic_cod': 'object'})
```

In [14]:
```
# revisió final de valors
data.head(10)
```

Out[14]:

	ID	Sexe	LA	RA	Diagnostic	Bayley	Brazelton	Vineland	A_BPD	A_OFD	...	C_LCAVD	C_RCAVD	TCD	PF	CCL	VCCD	LAH	RAH	Sexe_cod	Diagn
0	1	Masc	7.16	5.00	No VMG	NaN	43.714286	NaN	89.75	103.94	...	2.25	4.41	43.29	8.20	38.40	21.78	2.59	3.94	0	
1	2	Masc	14.07	13.00	VMG bilateral	74.0	45.714286	SI	85.55	107.81	...	1.76	1.76	39.86	10.55	37.27	18.25	5.40	6.12	0	
2	3	Masc	10.80	4.69	VMG lateral esquerra	NaN	49.857143	SI	72.11	89.16	...	1.76	1.76	31.65	6.45	37.45	21.74	2.11	5.40	0	
3	4	Masc	8.89	11.12	VMG lateral dreta	77.0	47.714286	SI	89.13	105.79	...	2.11	1.13	40.45	5.04	29.09	18.75	1.85	2.49	0	
4	5	Masc	11.20	6.83	VMG lateral esquerra	NaN	NaN	NaN	66.69	81.43	...	2.93	1.66	27.95	6.23	29.56	13.28	2.62	2.93	0	
5	6	Masc	4.46	3.75	No VMG	110.0	44.714286	NaN	79.57	106.64	...	1.76	1.76	41.62	7.50	37.57	24.87	2.49	2.11	0	
6	7	Masc	10.56	10.69	VMG bilateral	87.0	47.571429	SI	73.64	90.12	...	2.99	1.85	32.27	5.02	34.33	15.18	4.97	4.27	0	
7	8	Fem	7.64	5.98	No VMG	93.0	NaN	NaN	74.05	90.69	...	2.62	2.11	32.98	11.95	33.81	15.94	2.93	2.62	1	
8	9	Masc	5.40	11.39	VMG lateral dreta	NaN	41.571429	NaN	64.36	77.39	...	1.17	1.85	26.16	5.04	30.76	13.73	2.26	4.72	0	
9	10	Fem	15.78	8.94	VMG lateral esquerra	NaN	NaN	NaN	96.21	107.19	...	1.85	1.85	50.75	6.31	45.21	19.69	3.75	4.14	1	

10 rows × 40 columns

In [15]:
```
# guardar dataset de treball per a ús en el TFM
data.to_pickle("./data.pkl")
```

ANNEX 3: CODI PYTHON: ANÀLISI EXPLORATÒRIA DE DADES

TREBALL FINAL DE MASTER
MERITXELL BOSCH I GARCIA
2020/21-2 · Màster universitari en ciència de dades (*Data science*)
Estudis d'informàtica, multimèdia i telecomunicacions

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler

from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
# recuperar data set de treball
data = pd.read_pickle("./data.pkl")
```

2. ANÀLISI EXPLORATÒRIA DE DADES

In [3]:
```
# recopilació i resum
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 40 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              81 non-null     object
 1   Sexe            81 non-null     object
 2   LA              81 non-null     float64
 3   RA              81 non-null     float64
 4   Diagnostic      81 non-null     object
 5   Bayley          42 non-null     float64
 6   Brazelton       64 non-null     float64
 7   Vineland        25 non-null     object
 8   A_BPD           80 non-null     float64
 9   A_OFD           80 non-null     float64
 10  A_LID           80 non-null     float64
 11  A_RID           80 non-null     float64
 12  A_LSFD          80 non-null     float64
 13  A_RSFD          80 non-null     float64
 14  A_LIDSF         80 non-null     float64
 15  A_RIDSF         80 non-null     float64
 16  A_LMDSF         80 non-null     float64
 17  A_RMDSF         80 non-null     float64
 18  A_LODSF         80 non-null     float64
 19  A_RODSF         80 non-null     float64
 20  A_LPOFD         80 non-null     float64
 21  A_RPOFD         80 non-null     float64
 22  A_LOPOFD        80 non-null     float64
 23  A_ROPOFD        80 non-null     float64
 24  A_LPOVD         80 non-null     float64
 25  A_RPOVD         80 non-null     float64
 26  C_LCIFD         80 non-null     float64
 27  C_RCIFD         80 non-null     float64
 28  C_LCAFD         80 non-null     float64
 29  C_RCAFD         80 non-null     float64
 30  C_LCAVD         80 non-null     float64
 31  C_RCAVD         80 non-null     float64
 32  TCD             79 non-null     float64
 33  PF              79 non-null     float64
 34  CCL             80 non-null     float64
 35  VCCD            80 non-null     float64
 36  LAH             80 non-null     float64
 37  RAH             80 non-null     float64
 38  Sexe_cod        81 non-null     object
 39  Diagnostic_cod  81 non-null     object
dtypes: float64(34), object(6)
memory usage: 25.4+ KB
```

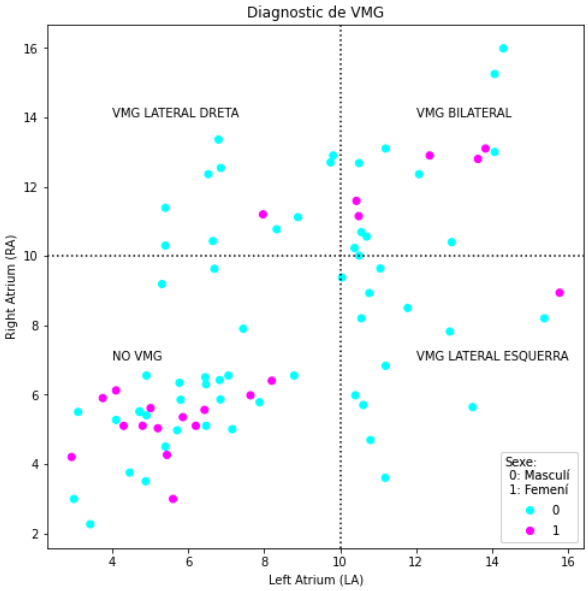
In [4]:
```
# llistar paràmetres
parametres = data.columns.values[2:4] # llista de tots els paràmetres a banda dels diagnostics
parametres = np.append(parametres, data.columns.values[8:38])
parametres
```

Out[4]:
```
array(['LA', 'RA', 'A_BPD', 'A_OFD', 'A_LID', 'A_RID', 'A_LSFD', 'A_RSFD',
       'A_LIDSF', 'A_RIDSF', 'A_LMDSF', 'A_RMDSF', 'A_LODSF', 'A_RODSF',
       'A_LPOFD', 'A_RPOFD', 'A_LOPOFD', 'A_ROPOFD', 'A_LPOVD', 'A_RPOVD',
       'C_LCIFD', 'C_RCIFD', 'C_LCAFD', 'C_RCAFD', 'C_LCAVD', 'C_RCAVD',
       'TCD', 'PF', 'CCL', 'VCCD', 'LAH', 'RAH'], dtype=object)
```

2.1 ANALISI GRÀFICA DEL DIAGNÒSTIC

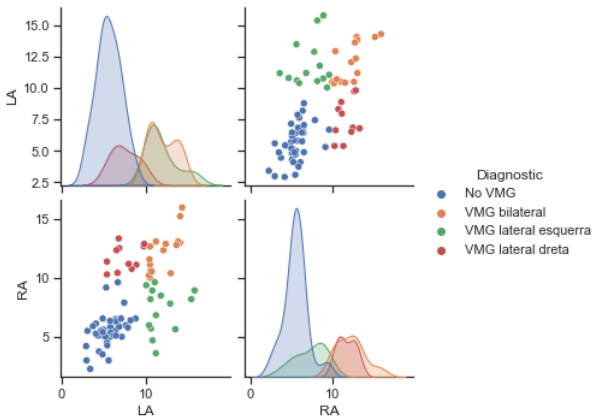
In [5]:
```
# anàlisi de dades per sexe
fig, ax = plt.subplots(figsize=(8, 8))
scatter= ax.scatter(data['LA'], data['RA'], c = data['Sexe_cod'], cmap = "cool")
plt.axhline(y=10, color="black", linestyle=":")
plt.axvline(x=10, color="black", linestyle=":")
plt.xlabel('Left Atrium (LA)')
plt.ylabel('Right Atrium (RA)')
plt.title('Diagnostic de VMG')
plt.text(4, 7, "NO VMG")
```

```
plt.text(4, 14, "VMG LATERAL DRETA")
plt.text(12, 7, "VMG LATERAL ESQUERRA")
plt.text(12, 14, "VMG BILATERAL")
legend1 = ax.legend(*scatter.legend_elements(), loc = 'lower right', title="Sexe:\n 0: Masculí \n 1: Femení")
ax.add_artist(legend1)
plt.show()
```



In [6]:

```
# Diagnòstic de la VMG es basa en la mida de LA i RA
sns.set_theme(style="ticks")
sns.pairplot(data[['LA', 'RA', 'Diagnostic']] , hue="Diagnostic")
plt.show()
```



2.2 ANÀLISI DELS PARÀMETRES

In [7]:

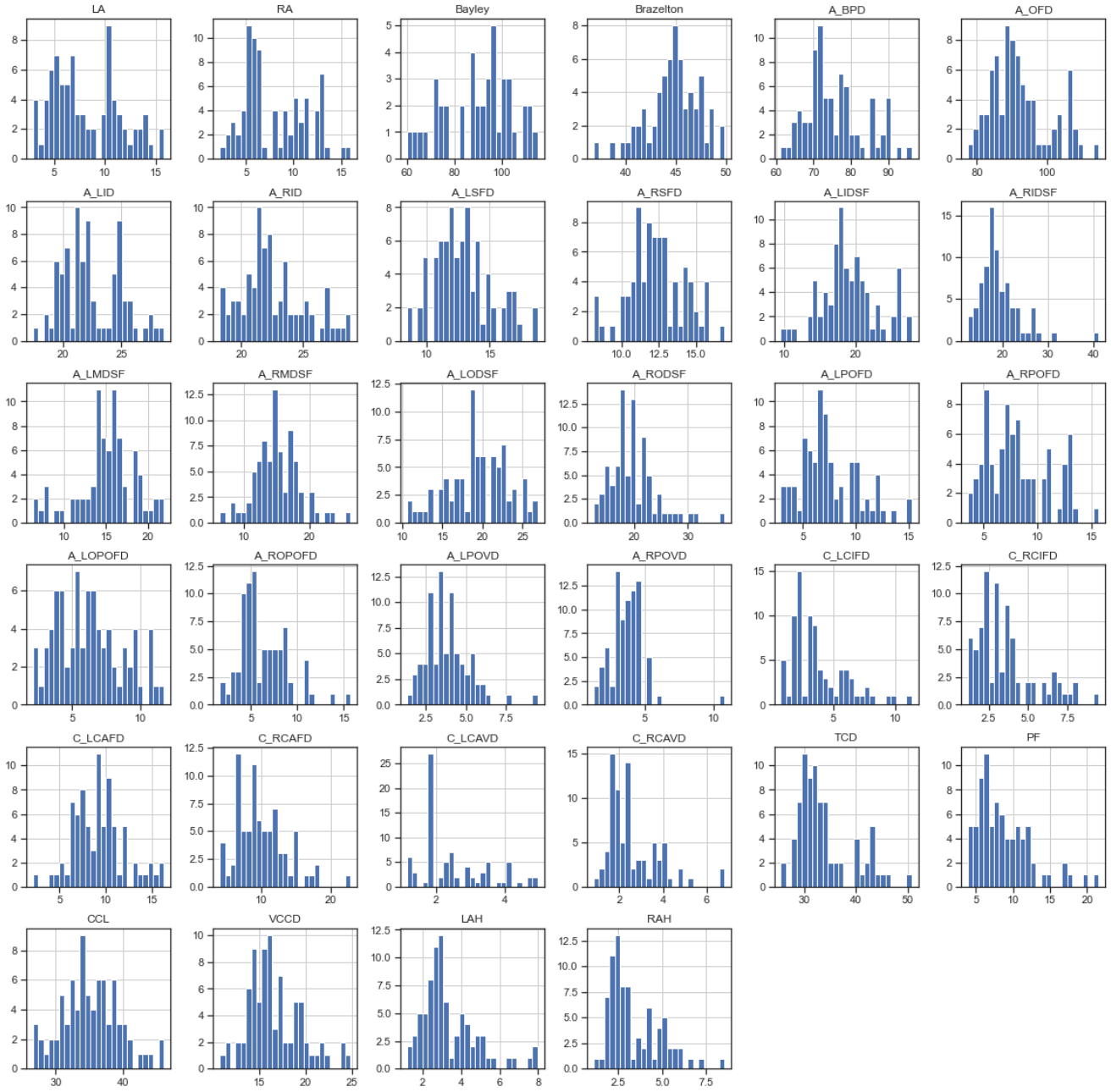
```
# resum del dataframe
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 40 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              81 non-null     object
 1   Sexe            81 non-null     object
 2   LA              81 non-null     float64
 3   RA              81 non-null     float64
 4   Diagnostic      81 non-null     object
 5   Bayley          42 non-null     float64
 6   Brazelton       64 non-null     float64
 7   Vineland        25 non-null     object
 8   A_BPD           80 non-null     float64
 9   A_OFD           80 non-null     float64
 10  A_LID           80 non-null     float64
 11  A_RID           80 non-null     float64
 12  A_LSFD          80 non-null     float64
 13  A_RSFD          80 non-null     float64
 14  A_LIDSF         80 non-null     float64
 15  A_RIDSF         80 non-null     float64
 16  A_LMDSF         80 non-null     float64
 17  A_RMDSF         80 non-null     float64
 18  A_LODSF         80 non-null     float64
 19  A_RODSF         80 non-null     float64
 20  A_LPOFD         80 non-null     float64
 21  A_RPOFD         80 non-null     float64
 22  A_LOPOFD        80 non-null     float64
 23  A_ROPOFD        80 non-null     float64
 24  A_LPOVD         80 non-null     float64
 25  A_RPOVD         80 non-null     float64
 26  C_LCIFD         80 non-null     float64
 27  C_RCIFD         80 non-null     float64
 28  C_LCAFD         80 non-null     float64
 29  C_RCAFD         80 non-null     float64
 30  C_LCAVD         80 non-null     float64
```

```
 31  C_RCAVD          80 non-null     float64
 32  TCD              79 non-null     float64
 33  PF               79 non-null     float64
 34  CCL              80 non-null     float64
 35  VCCD             80 non-null     float64
 36  LAH              80 non-null     float64
 37  RAH              80 non-null     float64
 38  Sexe_cod         81 non-null     object
 39  Diagnostic_cod   81 non-null     object
dtypes: float64(34), object(6)
memory usage: 25.4+ KB
```

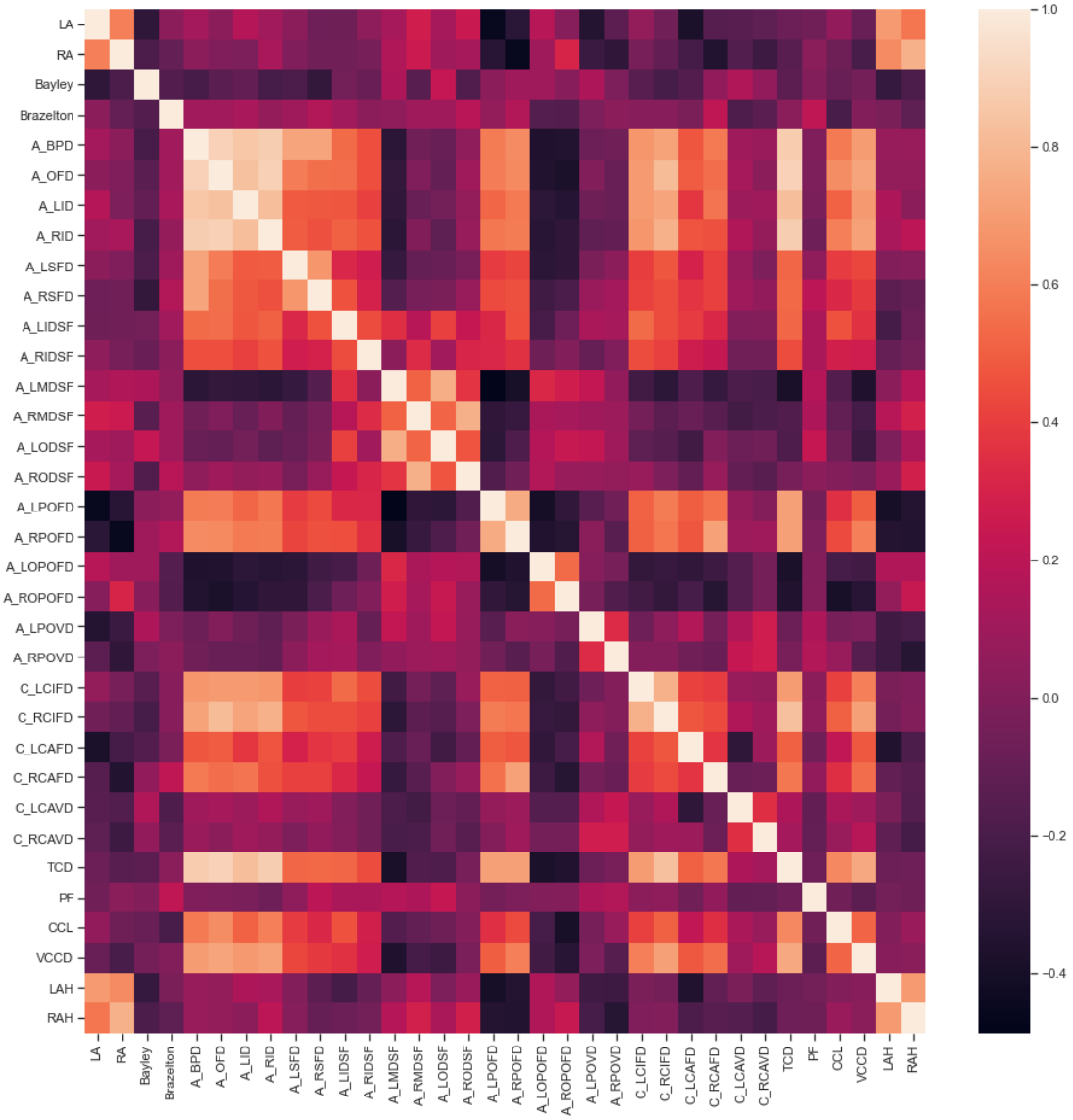
In [8]:

```
# histogrames per cadascun dels paràmetres
data.hist(bins = 25, figsize=(20,20))
plt.show()
```



In [9]:

```
#heatmap de la correlació entre paràmetres
plt.figure(figsize=(16, 16))
sns.heatmap(data.corr())
plt.show()
```



2.3 CARACTERITZACIÓ DE LA VMG

In [10]:
```
# separació de dades per categoria
data_axial = data.iloc[:, 8:26]
data_coronal = data.iloc[:, 26:32]
data_general = data.iloc[:, 32:38]
data_postnatal = data.iloc[:, 2:8]

# Afegir el diagnòstic a cada categoria
data_axial["Diagnostic"]= data["Diagnostic"]
data_coronal["Diagnostic"]= data["Diagnostic"]
data_general["Diagnostic"]= data["Diagnostic"]
```

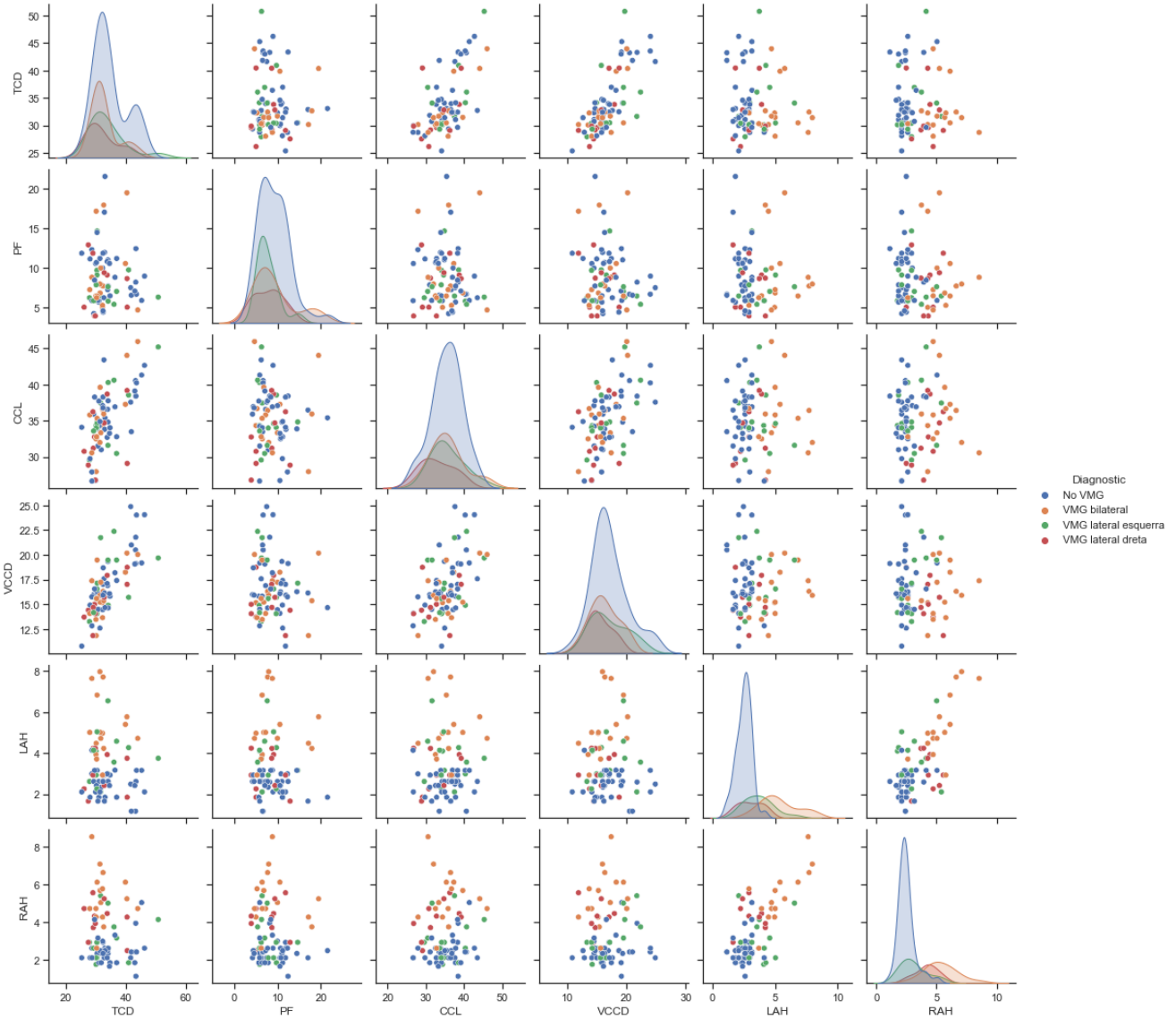
PARAMETRES GENERALS

In [11]:
```
data_general.head(5)
```

Out[11]:

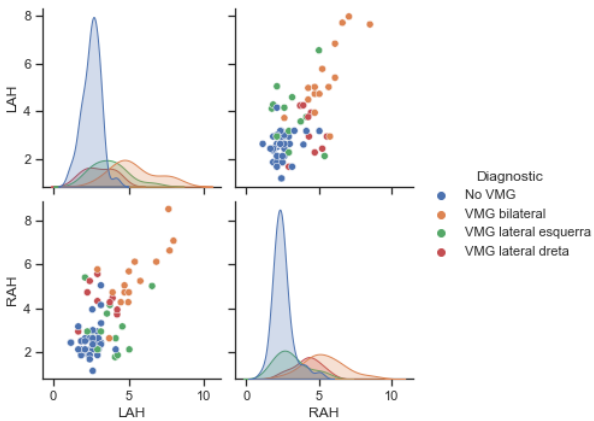
	TCD	PF	CCL	VCCD	LAH	RAH	Diagnostic
0	43.29	8.20	38.40	21.78	2.59	3.94	No VMG
1	39.86	10.55	37.27	18.25	5.40	6.12	VMG bilateral
2	31.65	6.45	37.45	21.74	2.11	5.40	VMG lateral esquerra
3	40.45	5.04	29.09	18.75	1.85	2.49	VMG lateral dreta
4	27.95	6.23	29.56	13.28	2.62	2.93	VMG lateral esquerra

In [12]:
```
# pairplot de tot el grup de dades
sns.set_theme(style="ticks")
sns.pairplot(data_general, hue="Diagnostic")
plt.show()
```



In [13]:

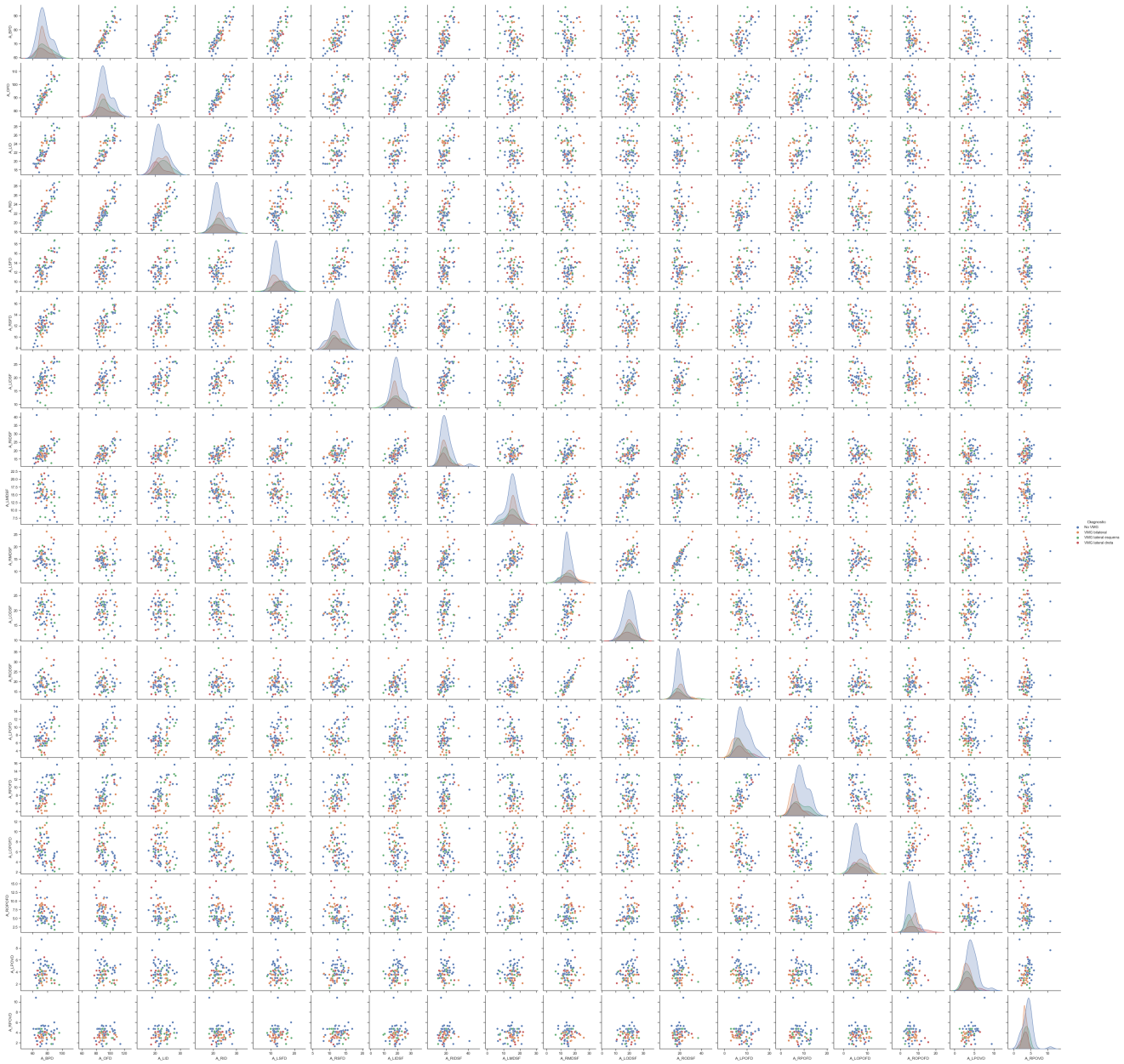
```
# pairplot especial per als dos paràmetres identificadors del diagnòstic
sns.set_theme(style="ticks")
sns.pairplot(data[['LAH', 'RAH', 'Diagnostic']] , hue="Diagnostic")
plt.show()
```



PARAMETRES PLA AXIAL

In [14]:

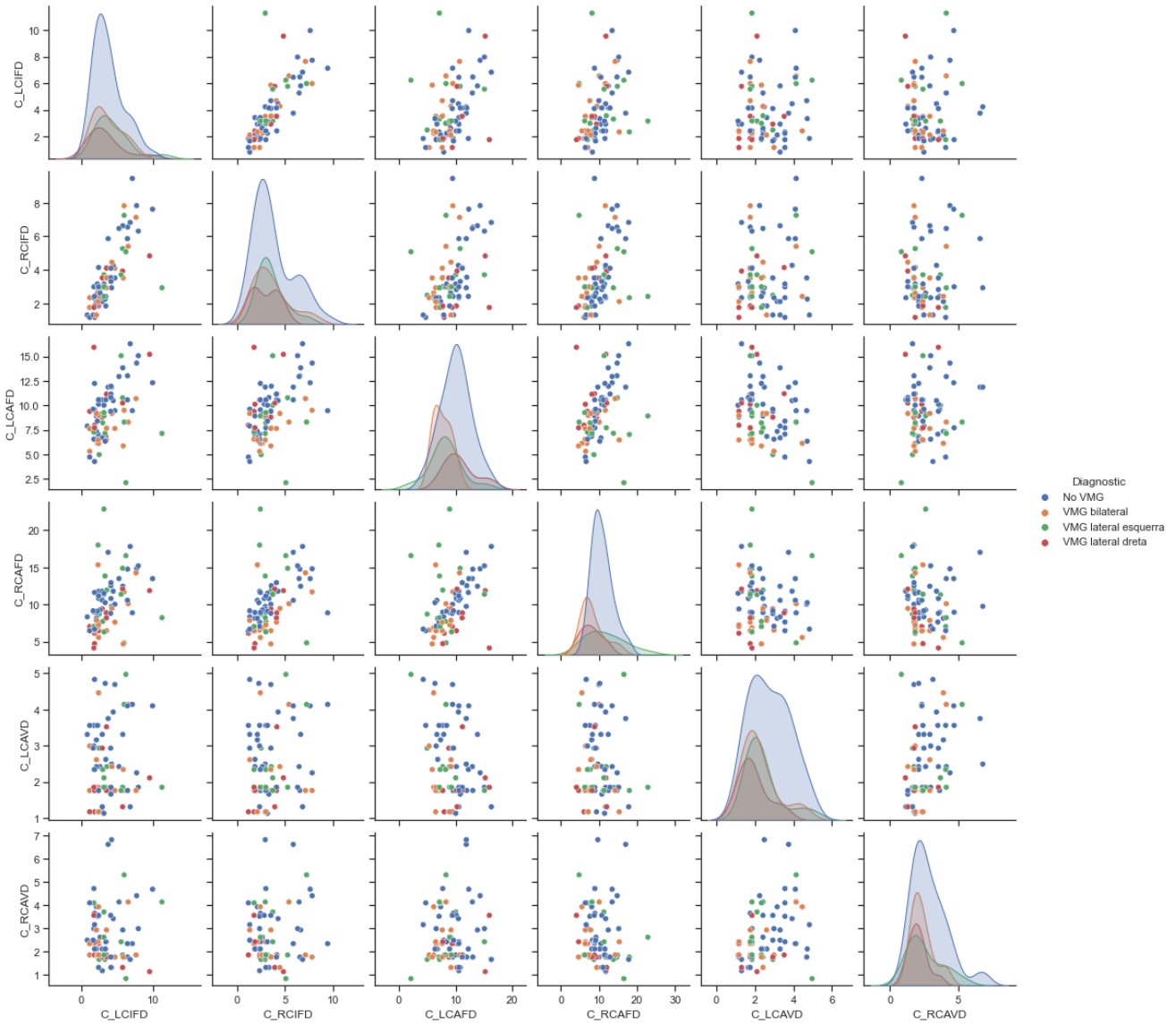
```
sns.set_theme(style="ticks")
sns.pairplot(data_axial, hue="Diagnostic")
plt.show()
```



PARAMETRES PLA CORONAL

In [15]:

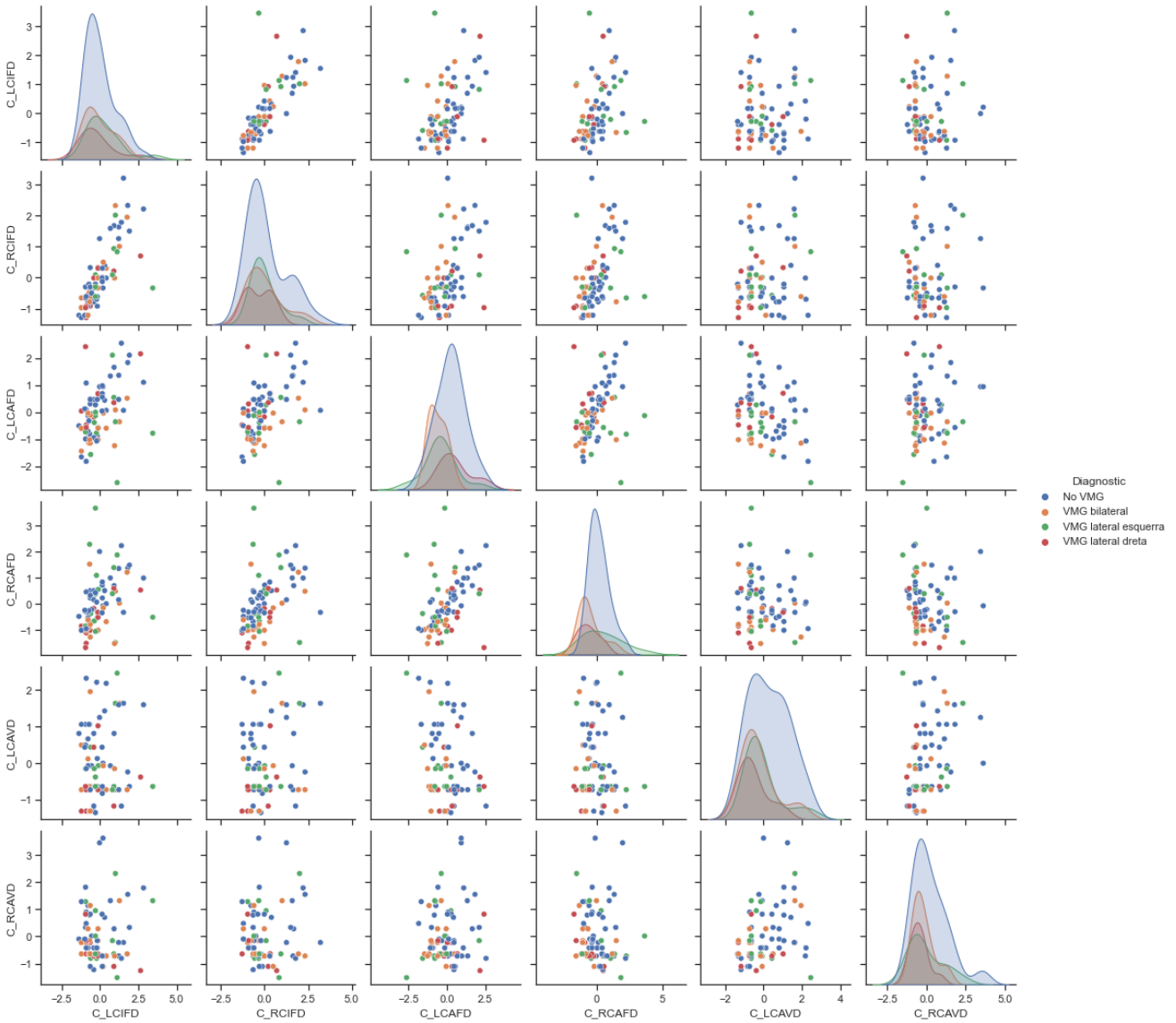
```
sns.set_theme(style="ticks")
sns.pairplot(data_coronal, hue="Diagnostic")
plt.show()
```



2.4 CARACTERITZACIO DE LA VMG AMB NORMALITZACIO

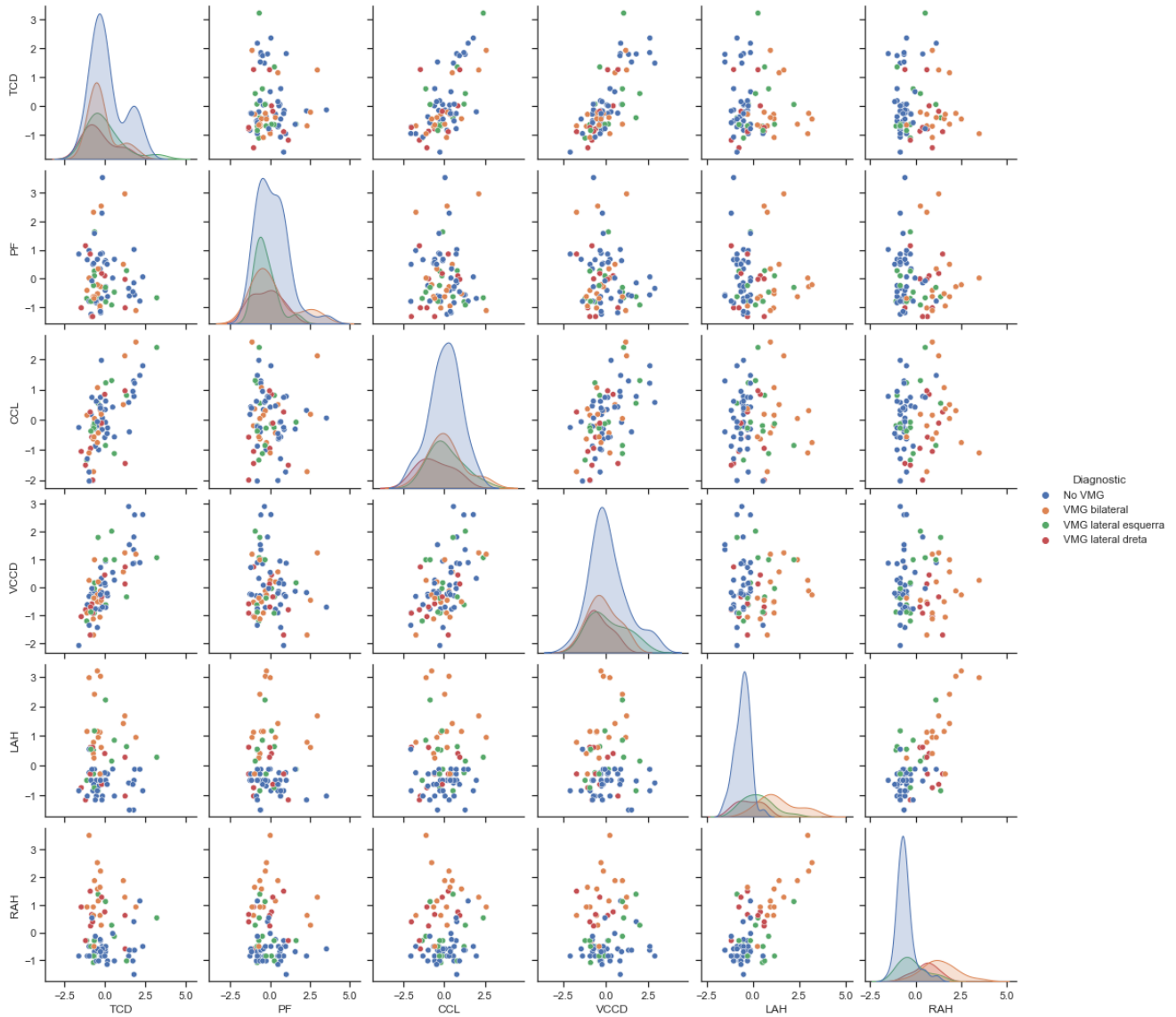
In [16]:
```
# Normalització i repetició de les proves

scaler =  StandardScaler()
scaled_values = scaler.fit_transform(data_axial.iloc[:,0:-1])
data_axial.iloc[:,0:-1] = scaled_values

scaled_values = scaler.fit_transform(data_coronal.iloc[:,0:-1])
data_coronal.iloc[:,0:-1] = scaled_values

scaled_values = scaler.fit_transform(data_general.iloc[:,0:-1])
data_general.iloc[:,0:-1] = scaled_values
```

In [17]:
```
sns.set_theme(style="ticks")
sns.pairplot(data_coronal, hue="Diagnostic")
plt.show()
```



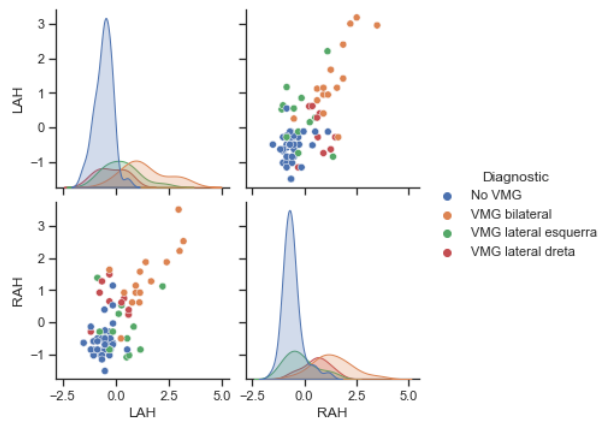
In [18]:

```
sns.set_theme(style="ticks")
sns.pairplot(data_general, hue="Diagnostic")
plt.show()
```



In [19]:

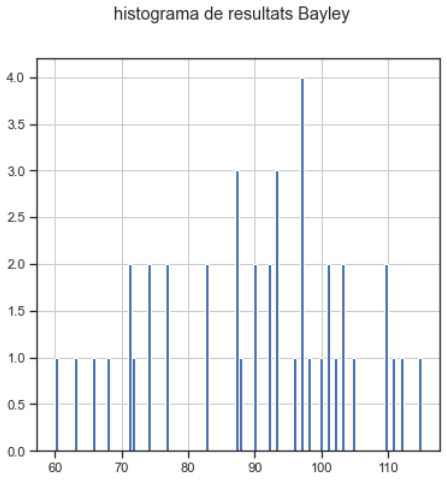
```
sns.set_theme(style="ticks")
sns.pairplot(data_general[['LAH', 'RAH', 'Diagnostic']] , hue="Diagnostic")
plt.show()
```



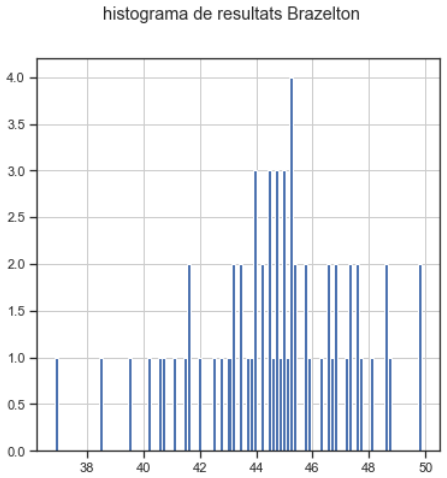
2.5 CARACTERITZACIÓ DE LA PROGNOSI

In [20]:

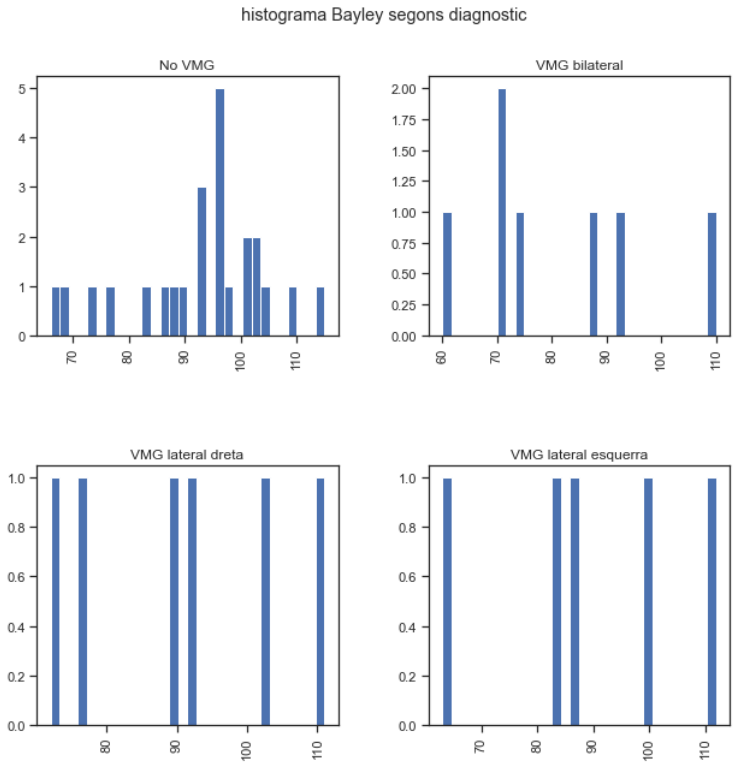
```
# Histograma del test Bayley
data['Bayley'].hist(bins = 100, figsize=(6,6))
plt.suptitle("histograma de resultats Bayley")
plt.show()
```



In [21]:
```
# Histograma del test Brazelton
data['Brazelton'].hist(bins = 100, figsize=(6,6))
plt.suptitle("histograma de resultats Brazelton")
plt.show()
```

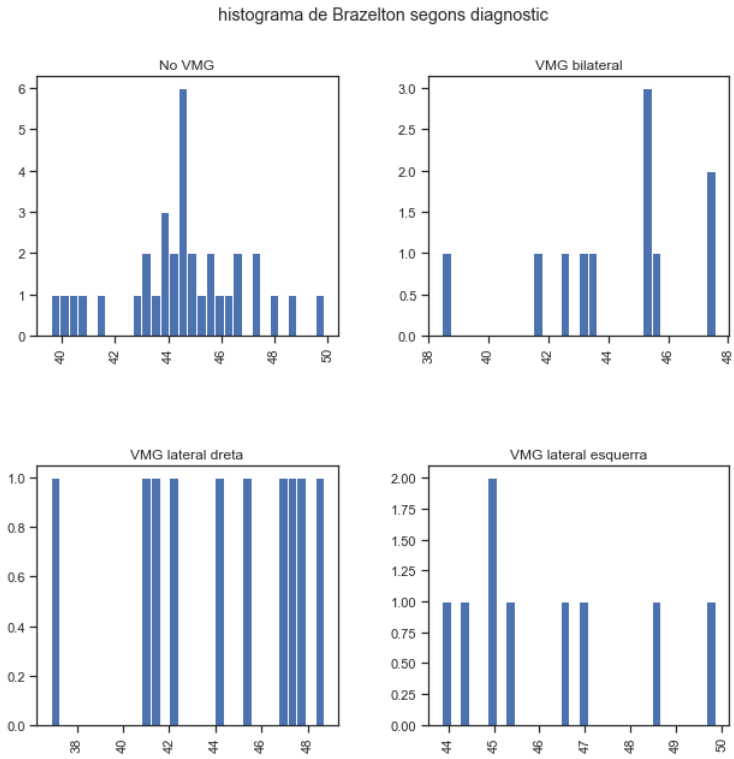


In [22]:
```
# histogrames de test Bayley per diagnòstic
data['Bayley'].hist(bins = 30, figsize=(10,10), by = data['Diagnostic'])
plt.suptitle("histograma Bayley segons diagnostic")
plt.show()
```



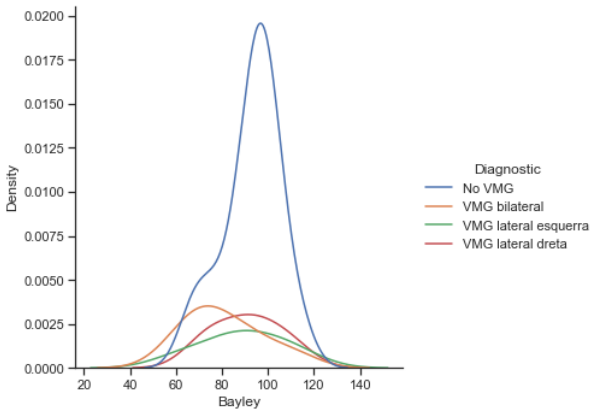
In [23]:
```
# histogrames de test Brazelton per diagnòstic
data['Brazelton'].hist(bins = 30, figsize=(10,10), by = data['Diagnostic'])
```

```
plt.suptitle("histograma de Brazelton segons diagnostic")
plt.show()
```



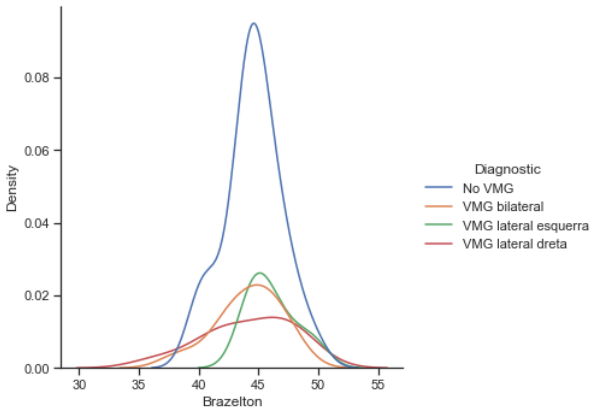
In [24]:

```
# Diagrama de densitat del resultat del test Bayley per diagnòstic
sns.displot(data=data, x="Bayley", hue="Diagnostic", kind="kde")
plt.show()
```



In [25]:

```
# Diagrama de densitat del resultat del test Brazelton per diagnòstic
sns.displot(data=data, x="Brazelton", hue="Diagnostic", kind="kde")
plt.show()
```

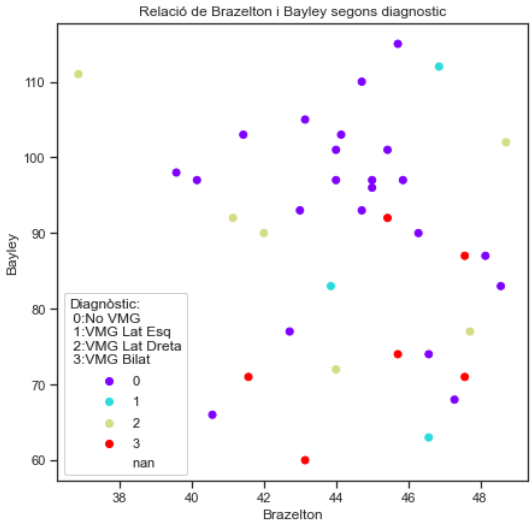


In [26]:

```
# Relació entre el resultat del test Bayley i Brazelton per diagnòstic
fig, ax = plt.subplots(figsize=(7, 7))
scatter= ax.scatter(data['Brazelton'], data['Bayley'], c = data['Diagnostic_cod'], cmap = "rainbow")
plt.xlabel('Brazelton')
plt.ylabel('Bayley')
plt.title('Relació de Brazelton i Bayley segons diagnostic')
legend1 = ax.legend(*scatter.legend_elements(), loc = 'lower left',
                    title="Diagnòstic:\n 0:No VMG\n 1:VMG Lat Esq\n 2:VMG Lat Dreta\n 3:VMG Bilat")
```

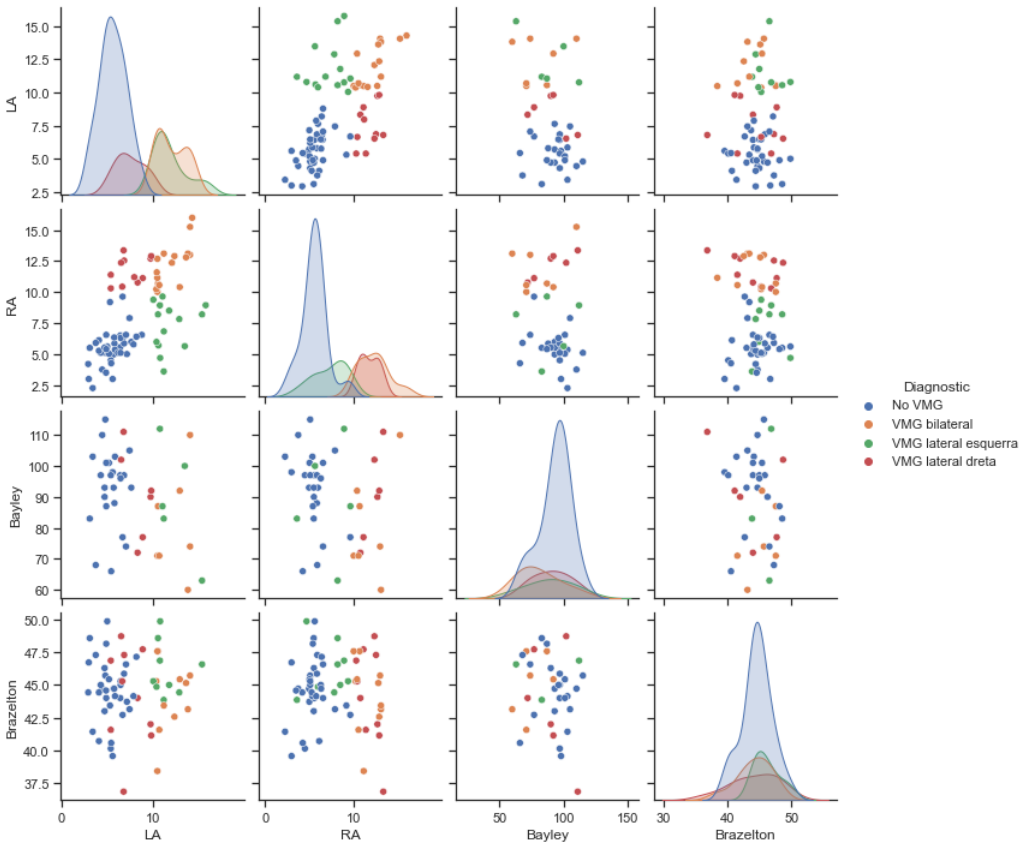
```
ax.add_artist(legend1)
plt.show()
```

```
<__array_function__ internals>:5: UserWarning: Warning: converting a masked element to nan.
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/numpy/core/_asarray.py:83: UserWarning: Warning: converting a masked element to nan.
  return array(a, dtype, copy=False, order=order)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/matplotlib/ticker.py:633: UserWarning: Warning: converting a masked element to nan.
  if self._useLocale else fmt % arg)
```



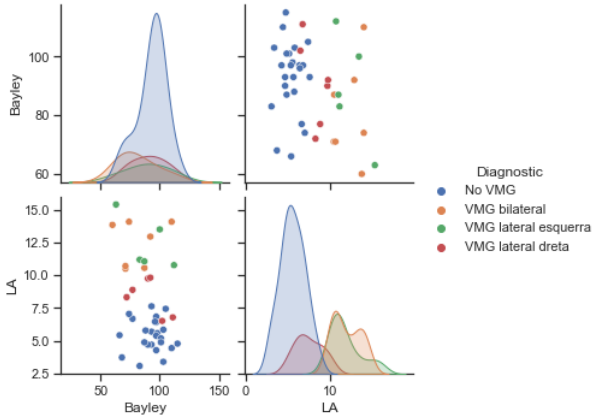
REVISIÓ PARÀMETRES POST-NATALS

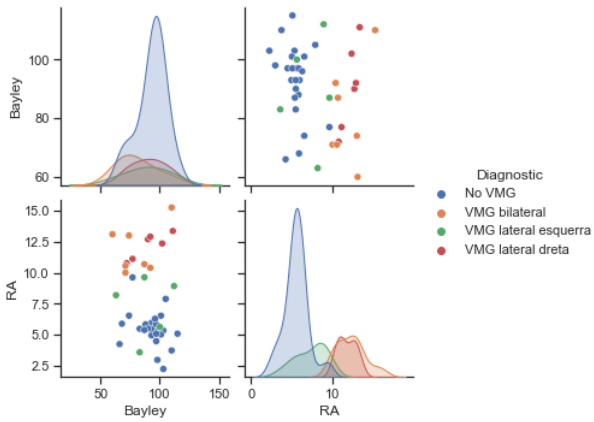
In [27]:
```
# comprovar la relació dels paràmetres que determinen el diagnòstic amb els que determinen la prognosi
sns.set_theme(style="ticks")
sns.pairplot(data_postnatal, hue="Diagnostic")
plt.show()
```

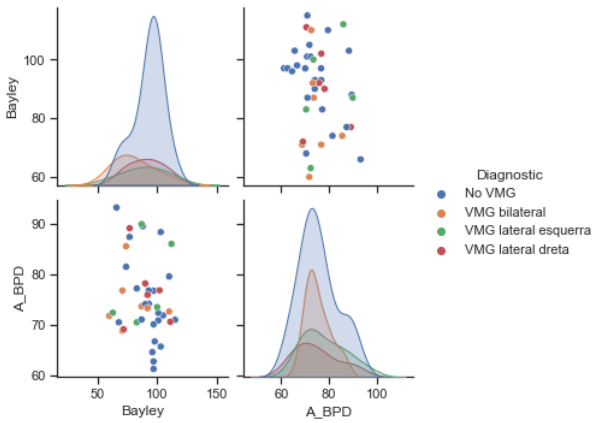


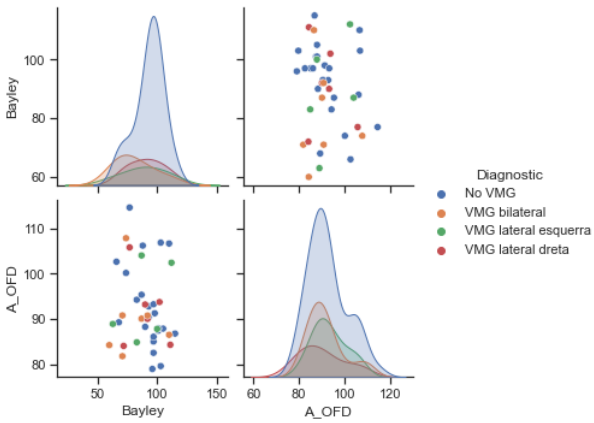
ANÀLISI DEL TEST DE BAYLEY

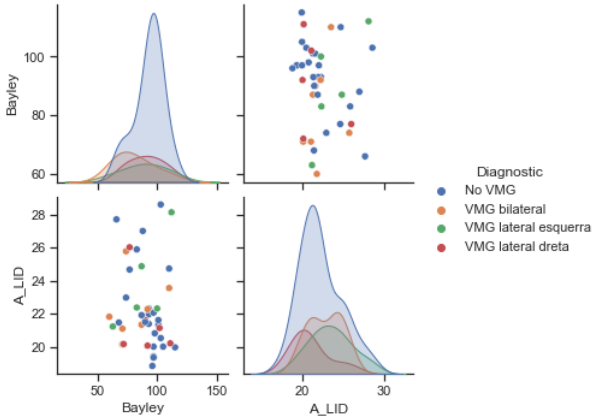
In [28]:
```
# comprovar la relació dels paràmetres amb els tests que determinen la prognosi
for parametre in parametres:
    sns.set_theme(style="ticks")
    sns.pairplot(data[["Diagnostic","Bayley", parametre]], hue="Diagnostic")
    plt.show()
```

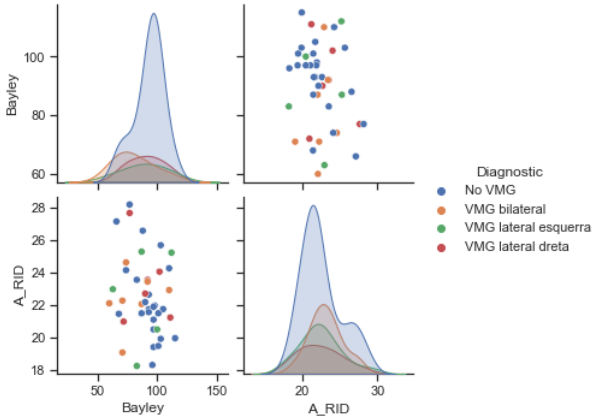

Bayley
LA
Bayley
LA
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

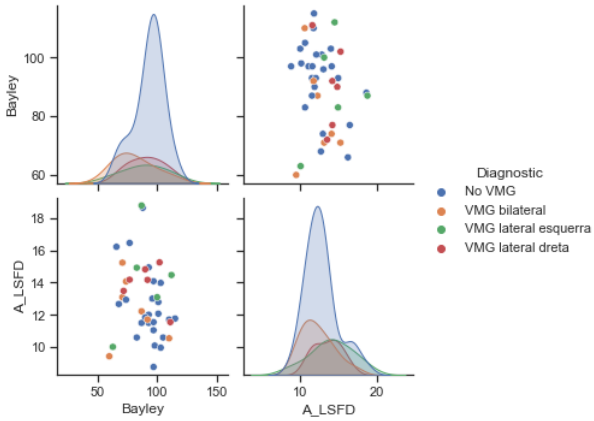

Bayley
RA
Bayley
RA
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

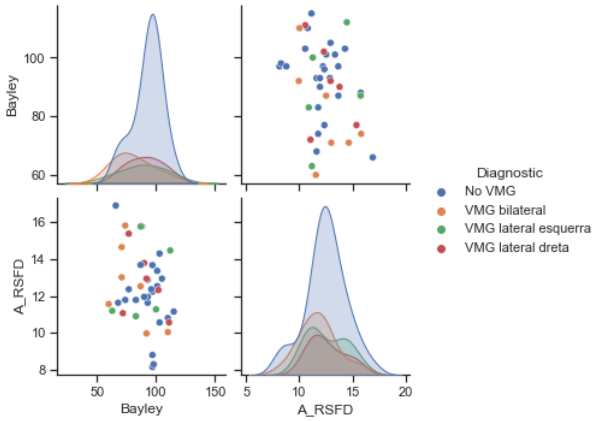

Bayley
A_BPD
Bayley
A_BPD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

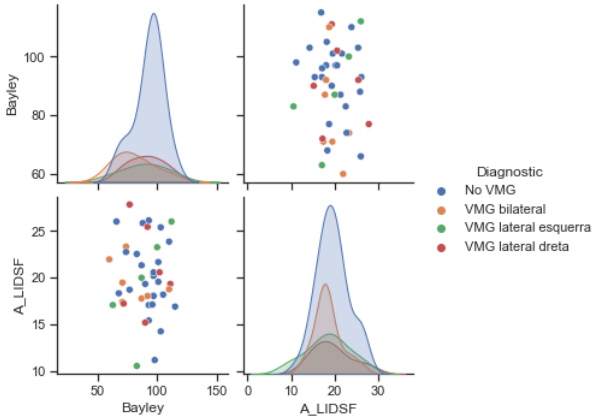

Bayley
A_OFD
Bayley
A_OFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

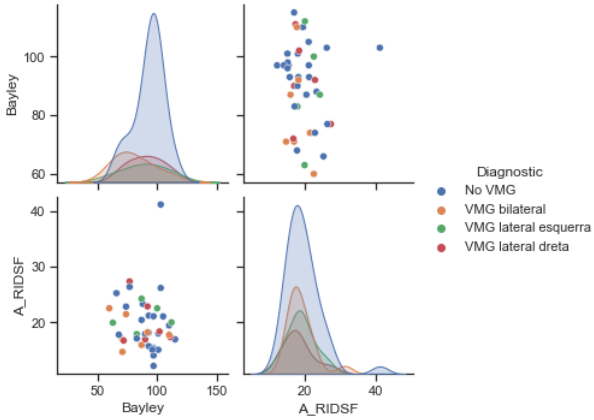

Bayley
A_LID
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

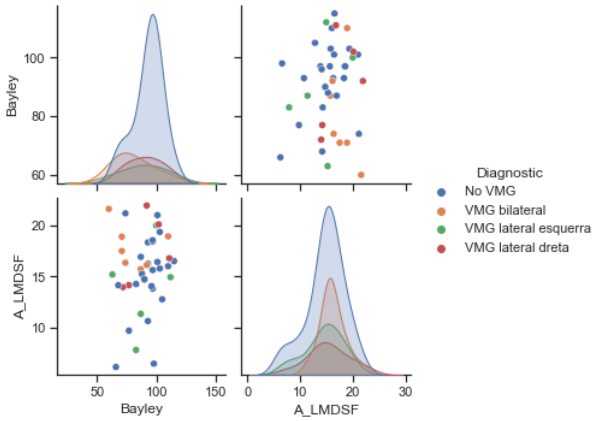

Bayley
A_RID
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

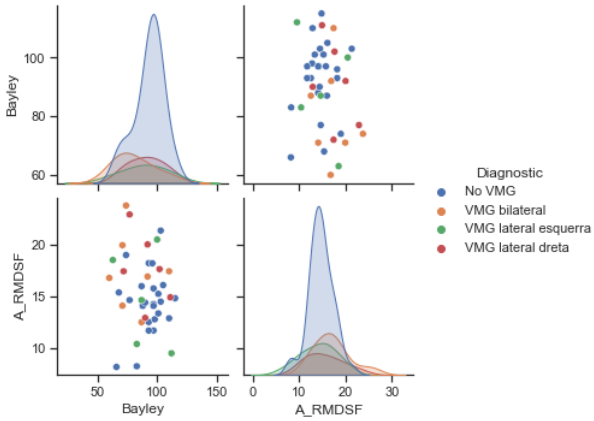

Bayley
A_LSFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

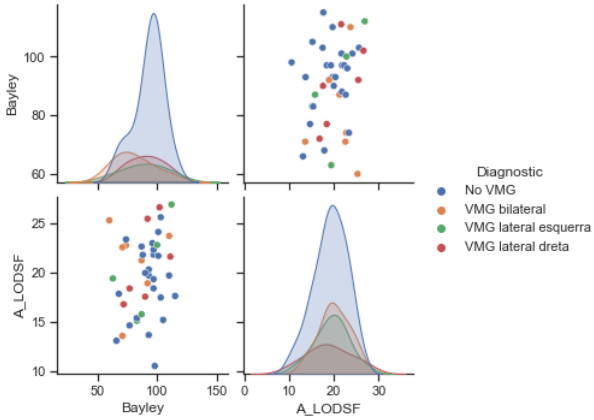

Bayley
A_RSFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

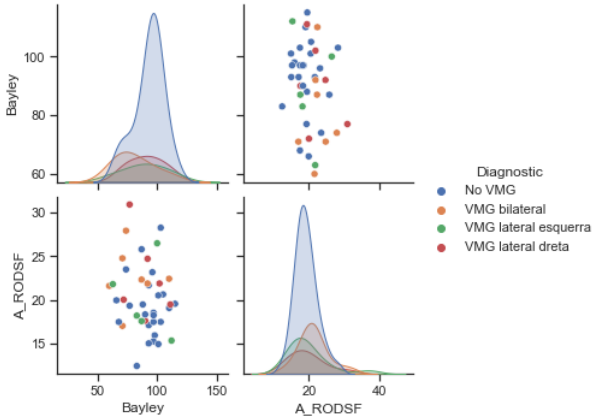

Bayley
A_LIDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

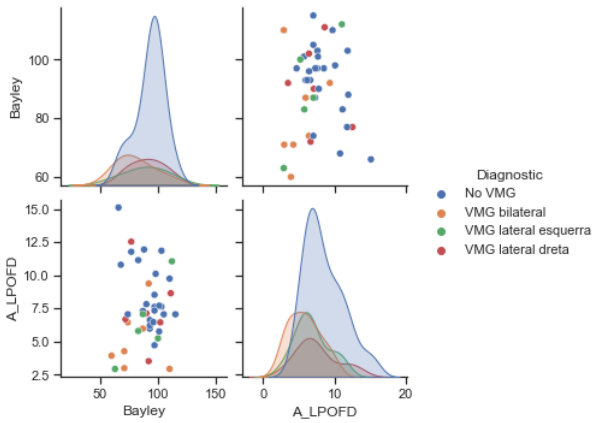

Bayley
A_RIDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

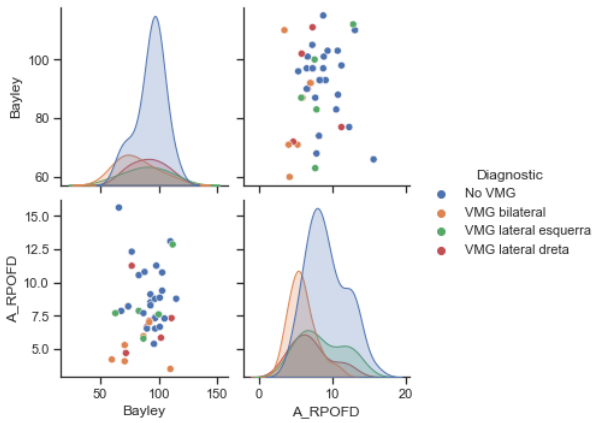

Bayley
A_LMDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

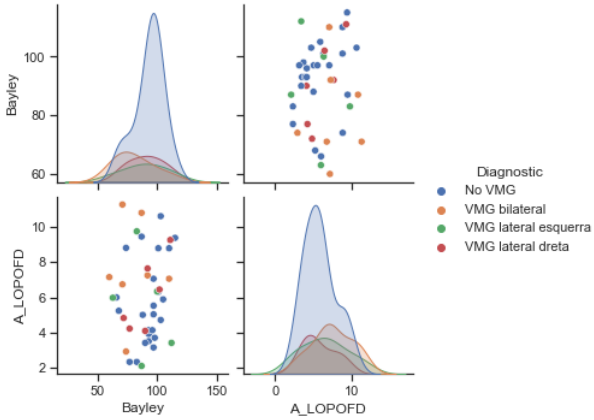

Bayley
A_RMDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

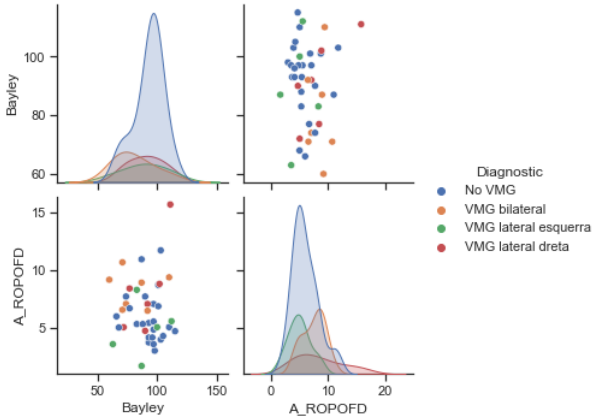

Bayley
A_LODSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

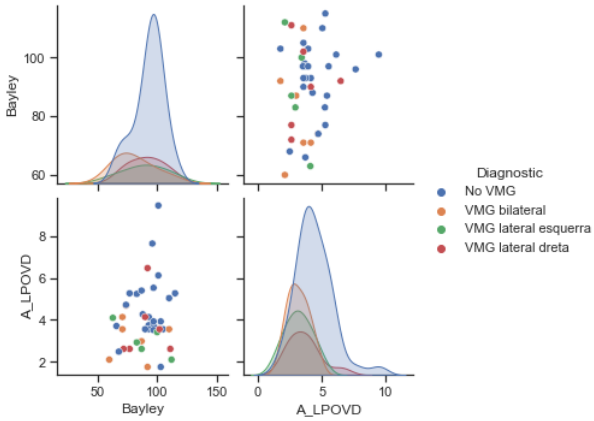

Bayley
A_RODSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

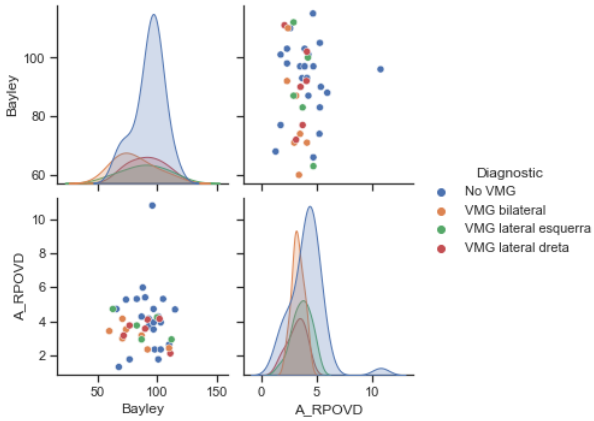

Bayley
A_LPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

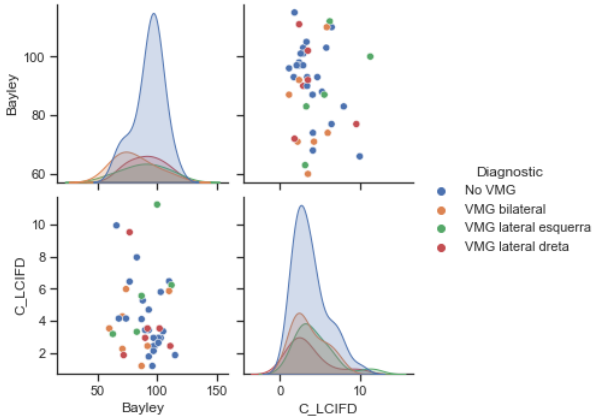

Bayley
A_RPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

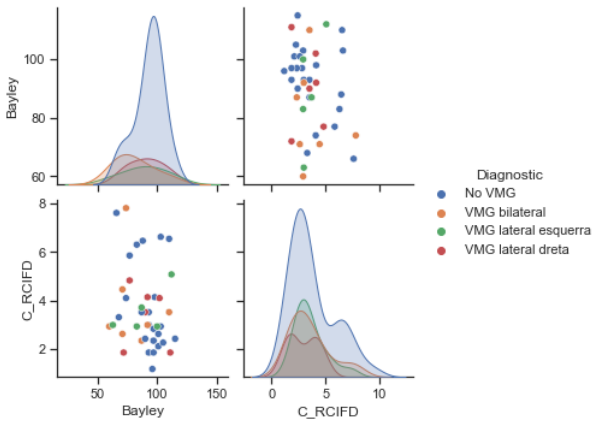

Bayley
A_LOPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

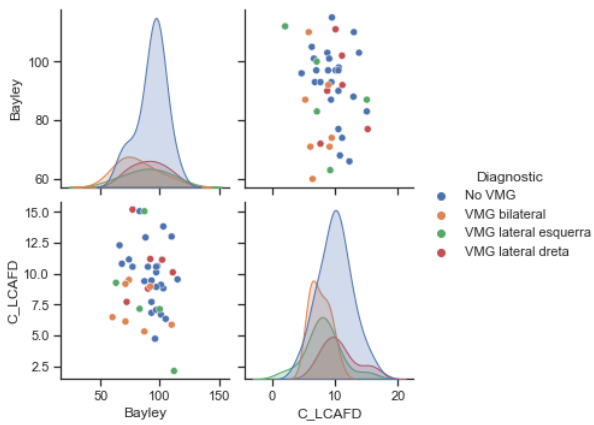

Bayley
A_ROPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

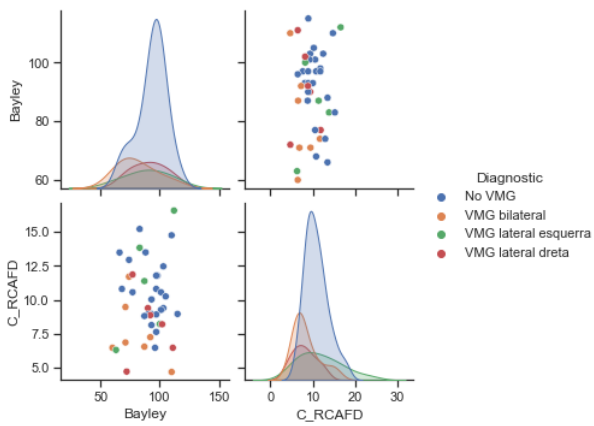

Bayley
A_LPOVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

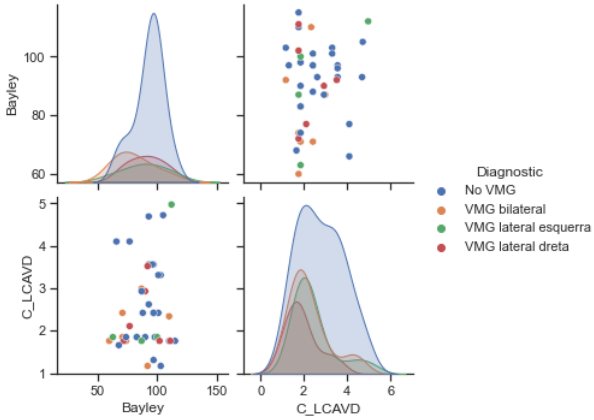

Bayley
A_RPOVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

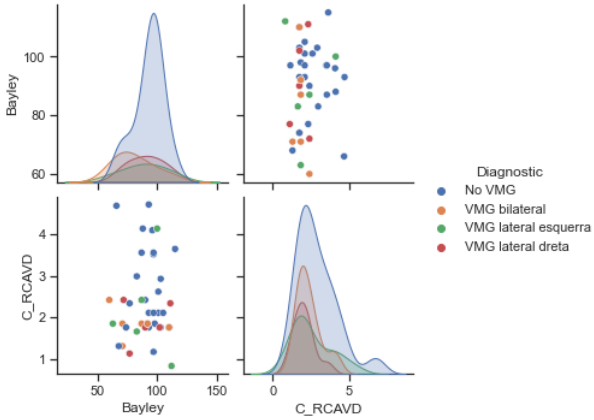

Bayley
C_LCIFD
Bayley
C_LCIFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

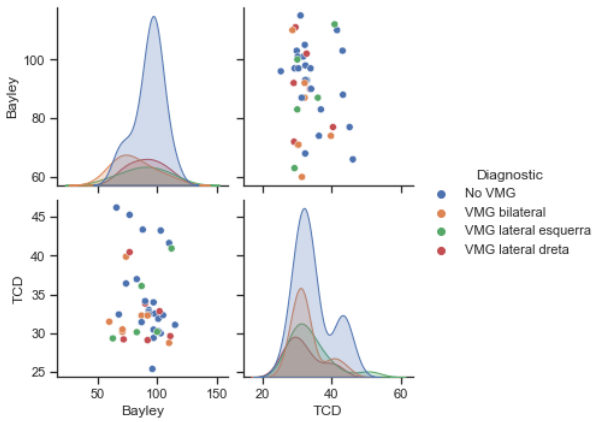

Bayley
C_RCIFD
Bayley
C_RCIFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

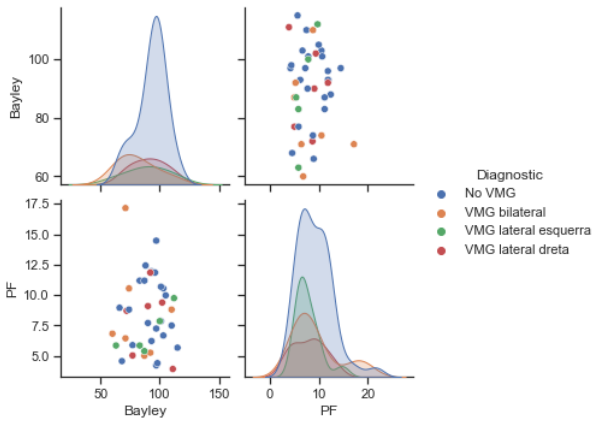

Bayley
C_LCAFD
Bayley
C_LCAFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

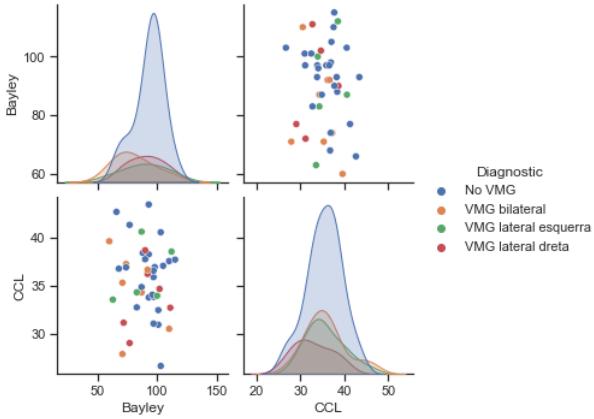

Bayley
C_RCAFD
Bayley
C_RCAFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

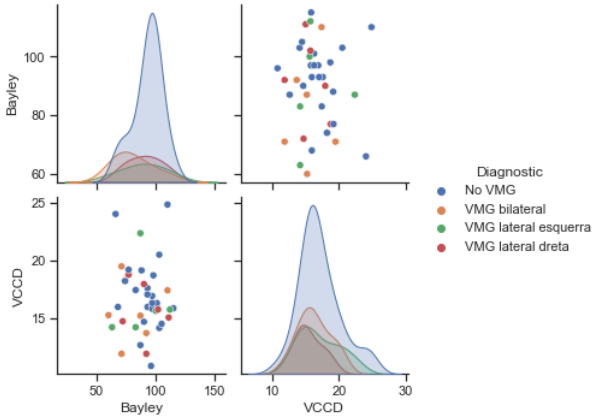

Bayley
C_LCAVD
Bayley
C_LCAVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

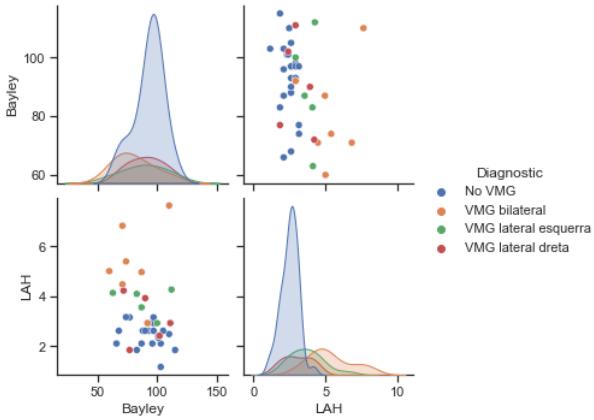

Bayley
C_RCAVD
Bayley
C_RCAVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

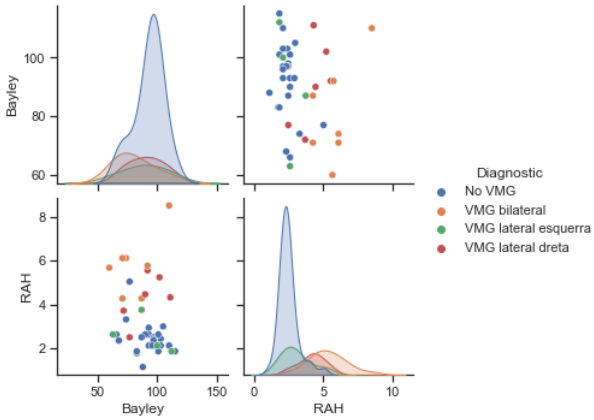

Bayley
TCD
Bayley
TCD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta


Bayley
PF
Bayley
PF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta





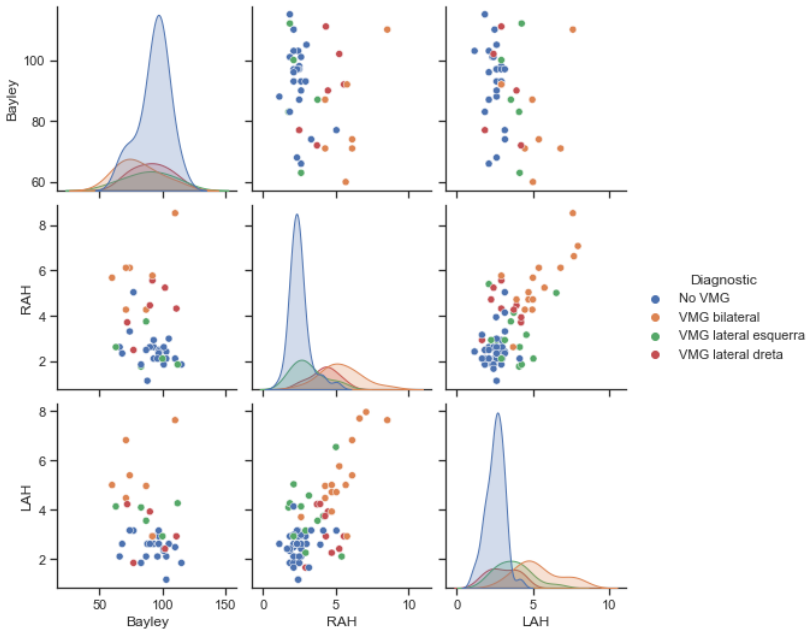




ANÀLISI CONJUNTA AMB RAH I LAH

In [29]:

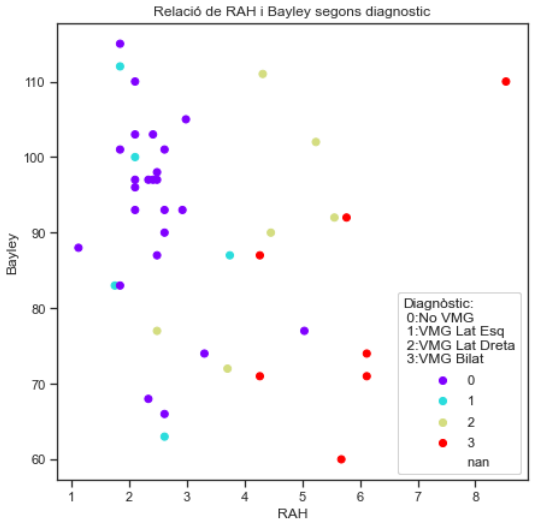
```
sns.set_theme(style="ticks")
sns.pairplot(data[["Bayley",'Diagnostic', 'RAH', 'LAH']], hue="Diagnostic")
plt.show()
```



In [30]:

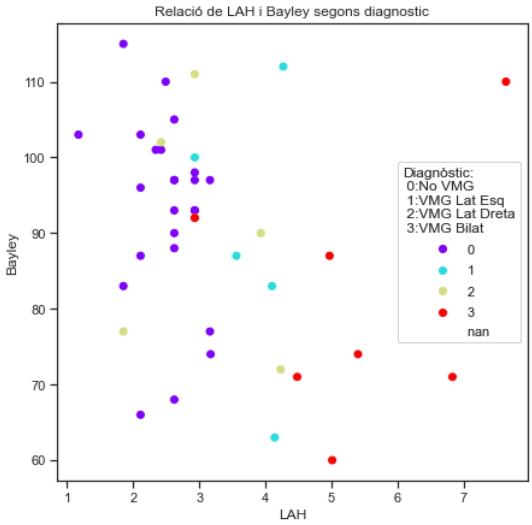
```
# relacio de Bayley amb RAH
fig, ax = plt.subplots(figsize=(7, 7))
scatter= ax.scatter(data['RAH'], data['Bayley'], c = data['Diagnostic_cod'], cmap = "rainbow")
plt.xlabel('RAH')
plt.ylabel('Bayley')
plt.title('Relació de RAH i Bayley segons diagnostic')
legend1 = ax.legend(*scatter.legend_elements(), loc = 'lower right',
                    title="Diagnòstic:\n 0:No VMG\n 1:VMG Lat Esq\n 2:VMG Lat Dreta\n 3:VMG Bilat")
add_artist(legend1)
plt.show()
```

```
<__array_function__ internals>:5: UserWarning: Warning: converting a masked element to nan.
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/numpy/core/_asarray.py:83: UserWarning: Warning: converting a
masked element to nan.
  return array(a, dtype, copy=False, order=order)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/matplotlib/ticker.py:633: UserWarning: Warning: converting a
masked element to nan.
  if self._useLocale else fmt % arg)
```

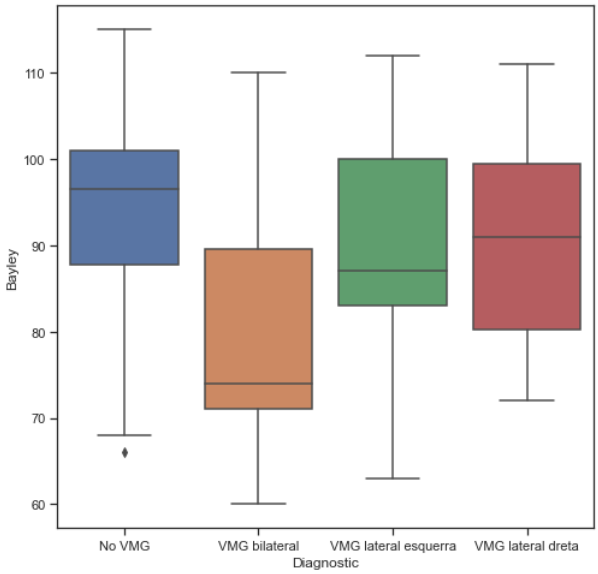


In [31]:

```
# relacio de Bayley amb LAH
fig, ax = plt.subplots(figsize=(7, 7))
scatter= ax.scatter(data['LAH'], data['Bayley'], c = data['Diagnostic_cod'], cmap = "rainbow")
plt.xlabel('LAH')
plt.ylabel('Bayley')
plt.title('Relació de LAH i Bayley segons diagnostic')
legend1 = ax.legend(*scatter.legend_elements(), loc = 'center right',
                    title="Diagnòstic:\n 0:No VMG\n 1:VMG Lat Esq\n 2:VMG Lat Dreta\n 3:VMG Bilat")
ax.add_artist(legend1)
plt.show()
```

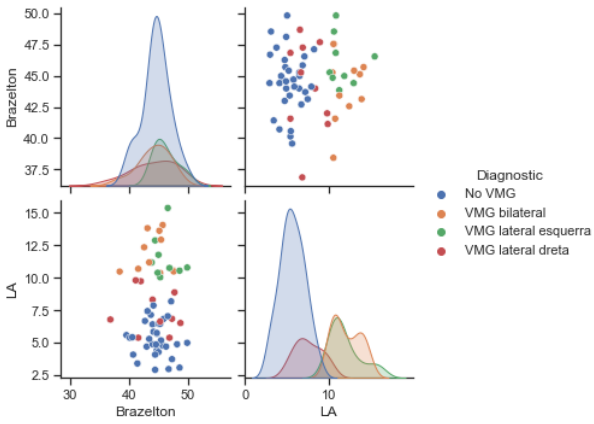


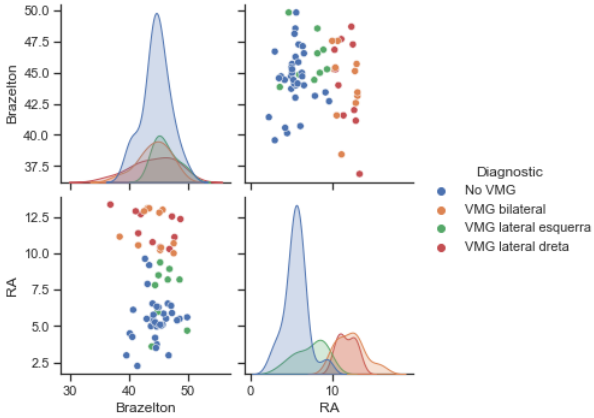
In [32]:
```
# boxplot del test de bayley per diagnostic
fig, ax = plt.subplots(figsize=(8,8))
sns.boxplot(y='Bayley', x='Diagnostic', data=data)
plt.show()
```

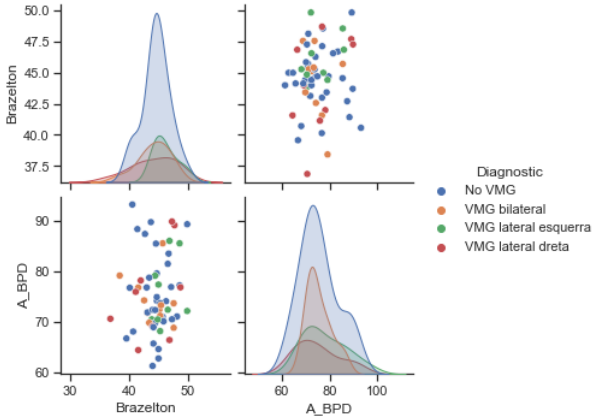


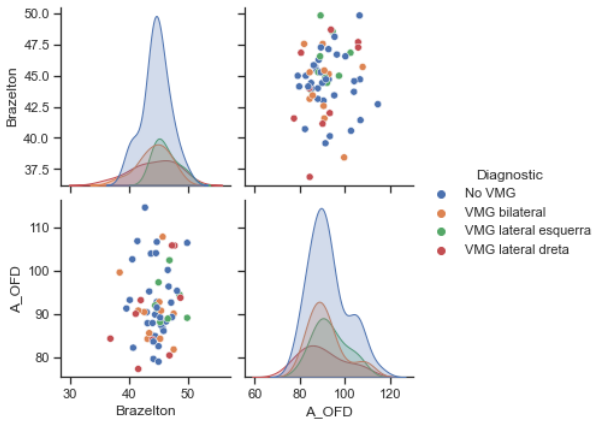
ANÀLISI DEL TEST DE BRAZELTON

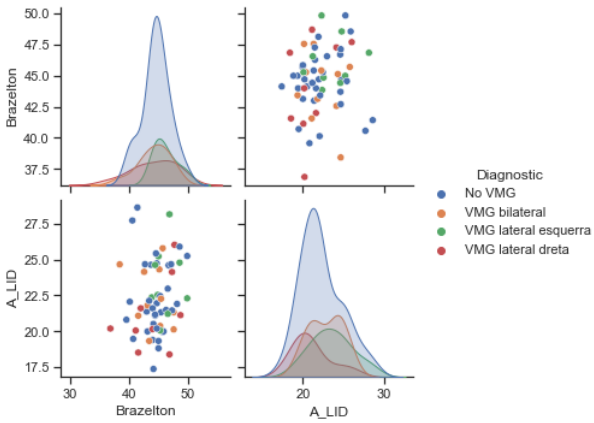
In [33]:
```
# relació de tots els paràmetres amb els test de Brazelton
for parametre in parametres:
    sns.set_theme(style="ticks")
    sns.pairplot(data[["Diagnostic","Brazelton", parametre]], hue="Diagnostic")
    plt.show()
```

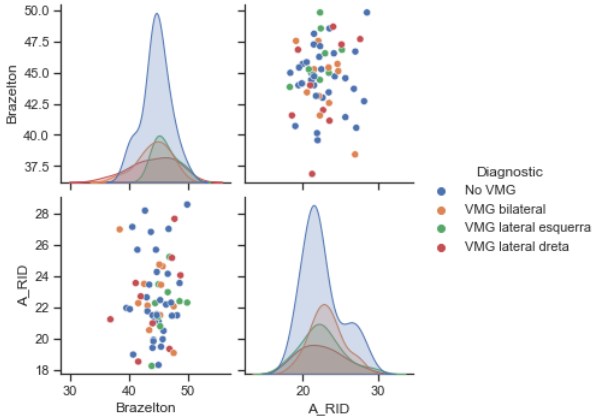


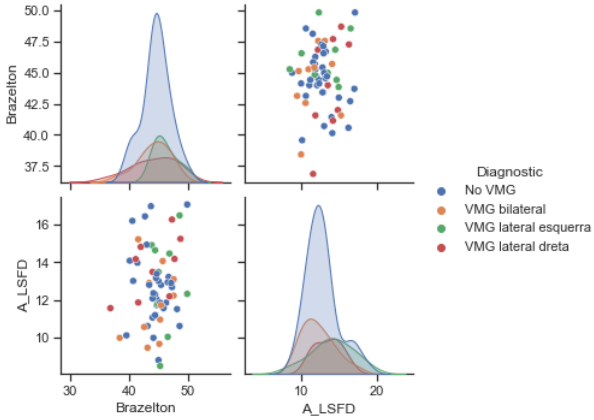

Brazelton
RA
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

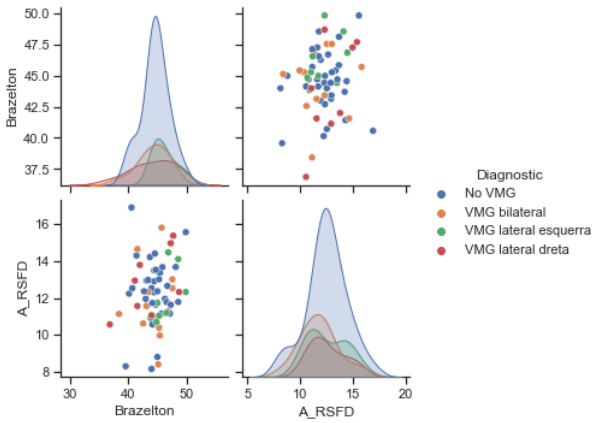

Brazelton
A_BPD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

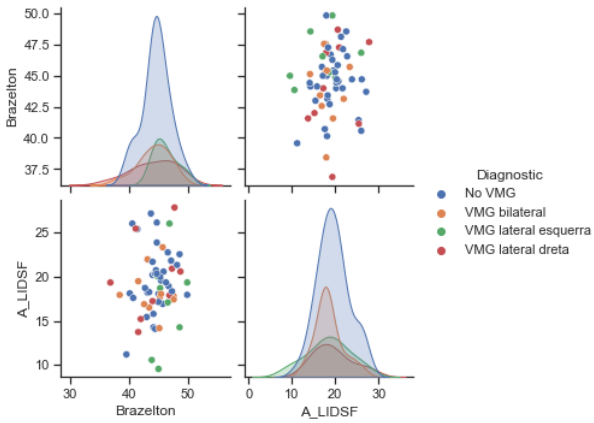

Brazelton
A_OFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

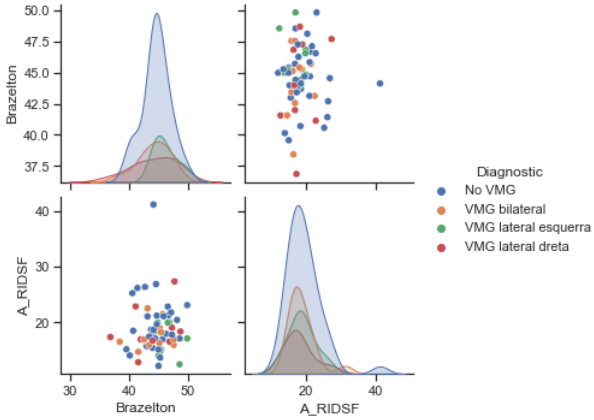

Brazelton
A_LID
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

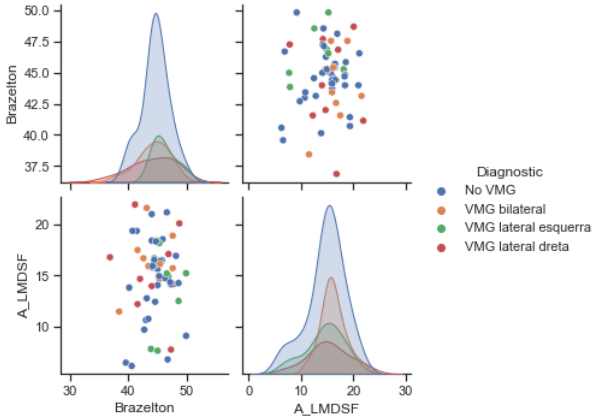

Brazelton
A_RID
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

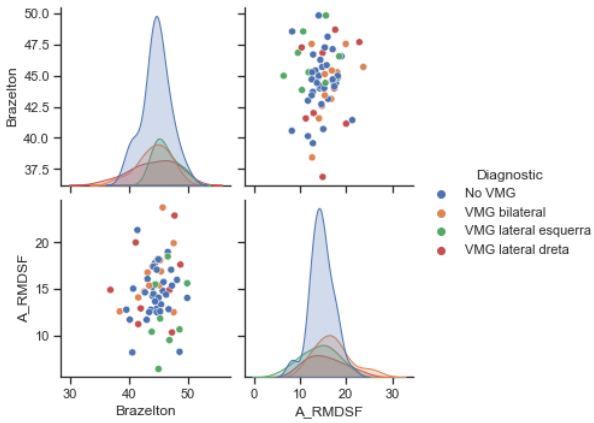

Brazelton
A_LSFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

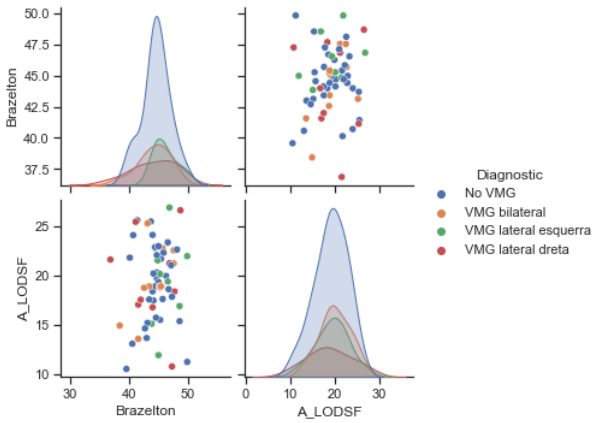

Brazelton
A_RSFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

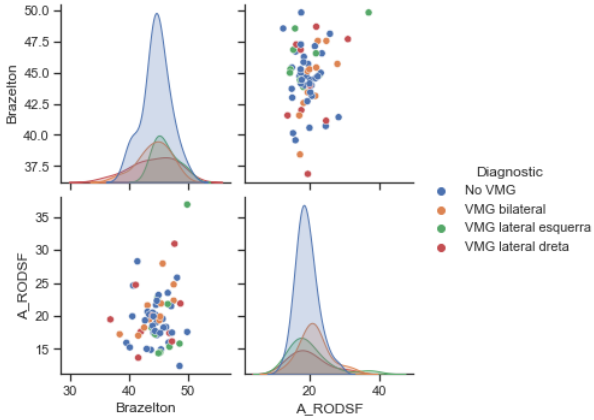

Brazelton
A_LIDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

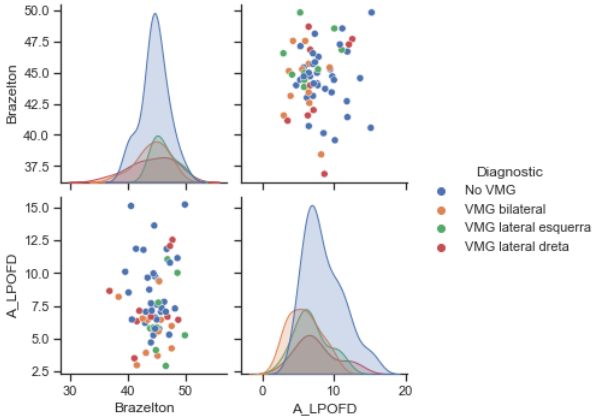

Brazelton
A_RIDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

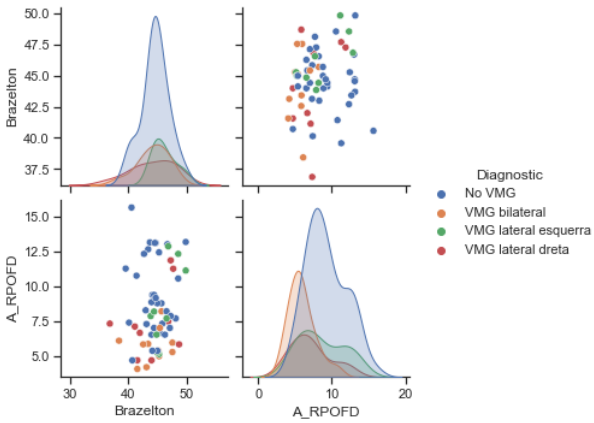

Brazelton
A_LMDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

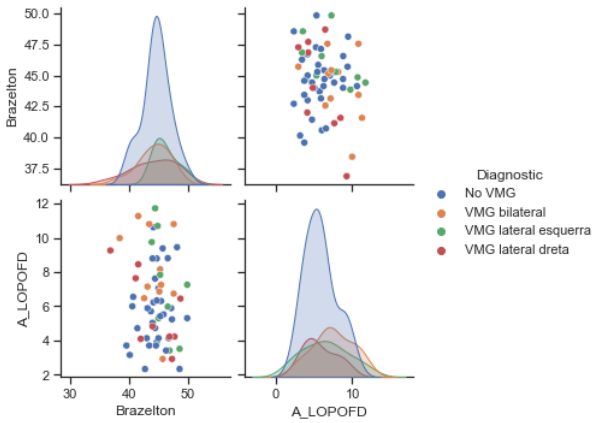

Brazelton
A_RMDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

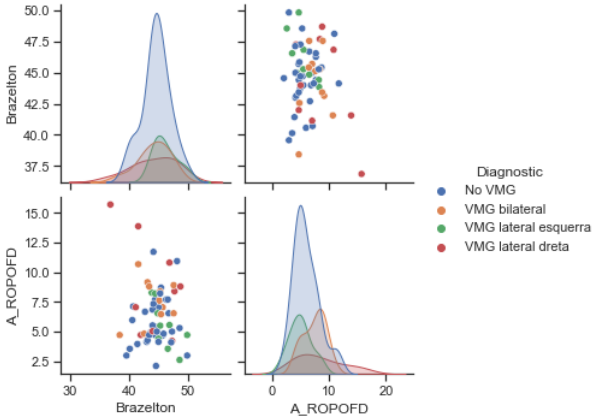

Brazelton
A_LODSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

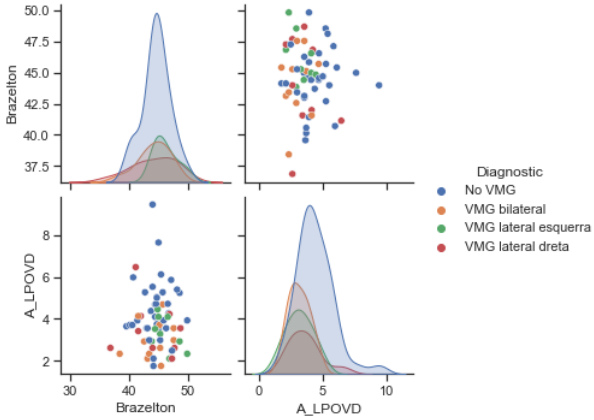

Brazelton
A_RODSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

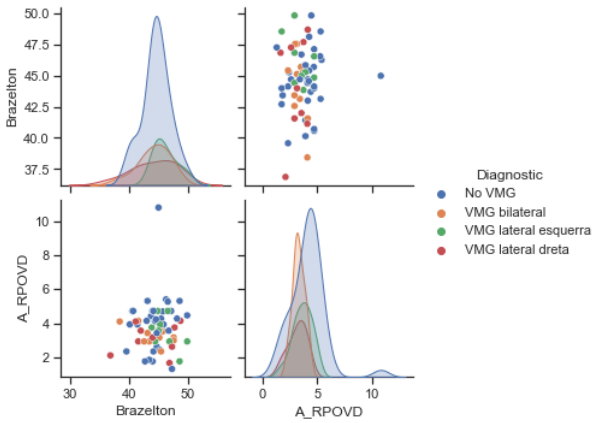

Brazelton
A_LPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

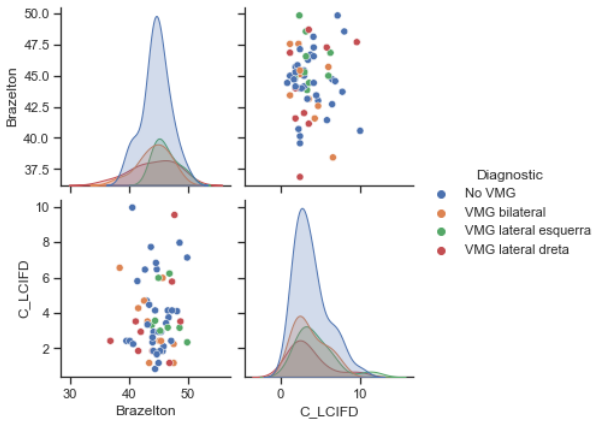

Brazelton
A_RPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

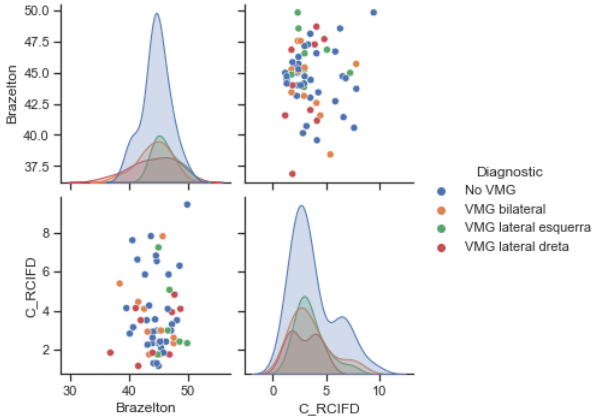

Brazelton
A_LOPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

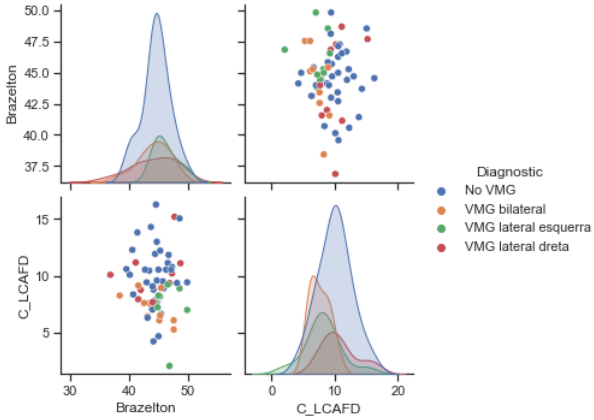

Brazelton
A_ROPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

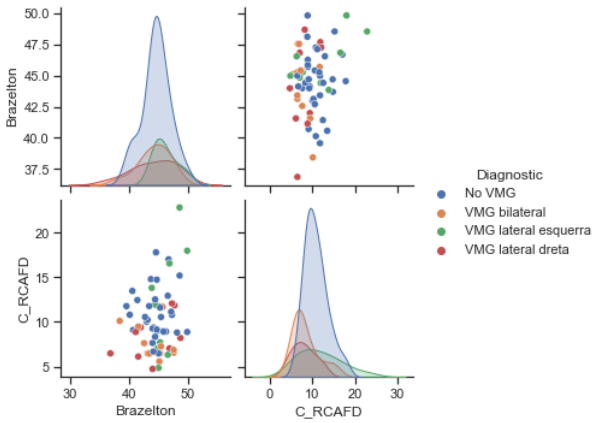

Brazelton
A_LPOVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

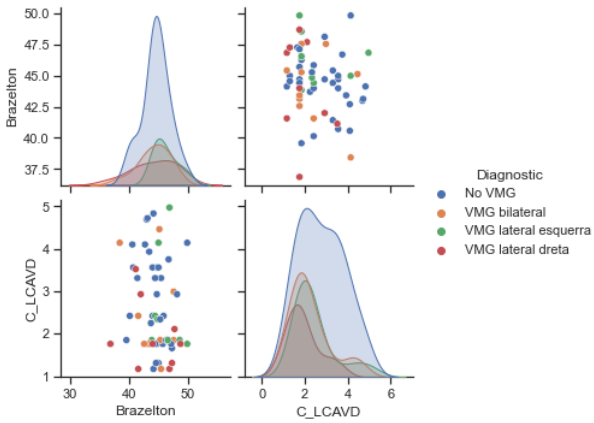

Brazelton
A_RPOVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

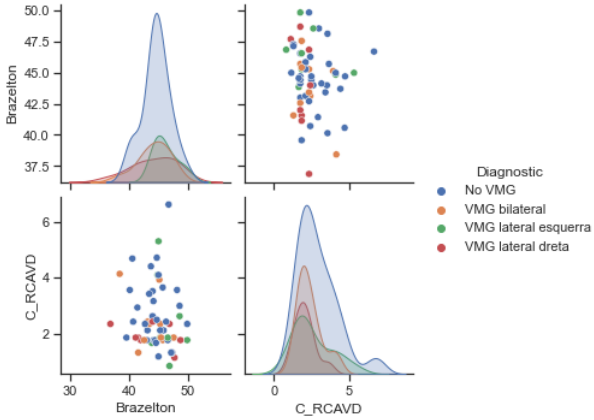

Brazelton
C_LCIFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

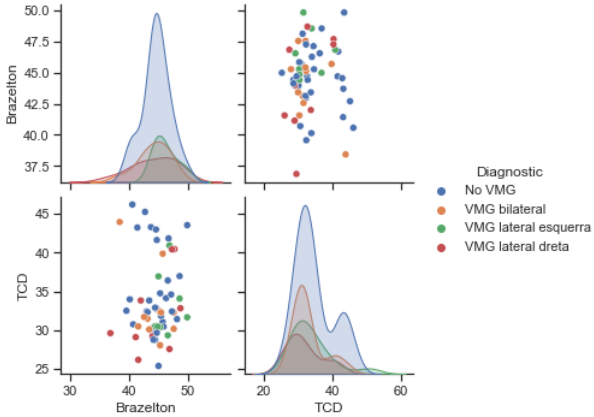

Brazelton
C_RCIFD
Brazelton
C_RCIFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

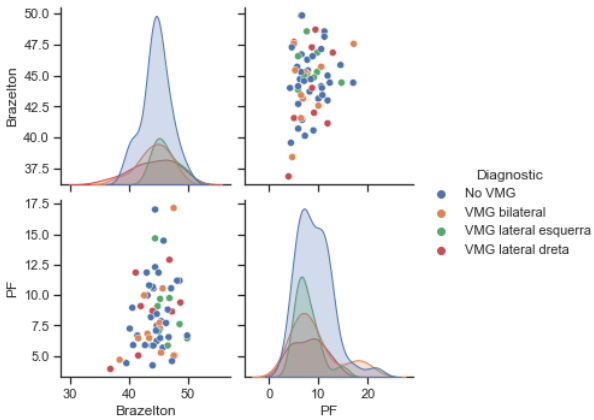

Brazelton
C_LCAFD
Brazelton
C_LCAFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

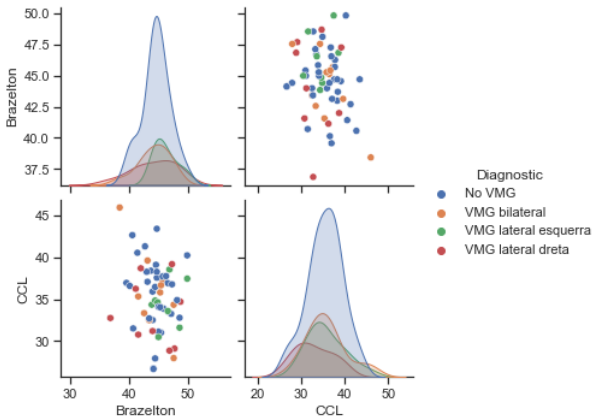

Brazelton
C_RCAFD
Brazelton
C_RCAFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

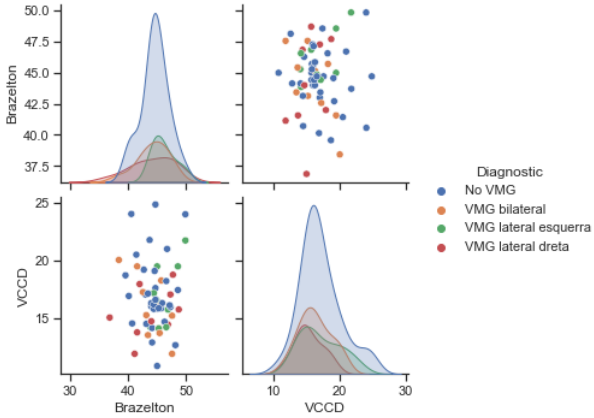

Brazelton
C_LCAVD
Brazelton
C_LCAVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

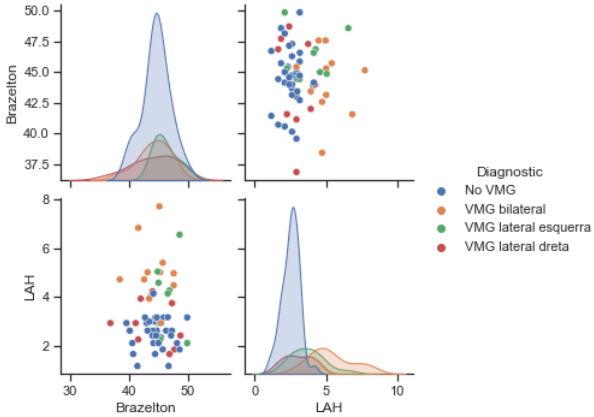

Brazelton
C_RCAVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

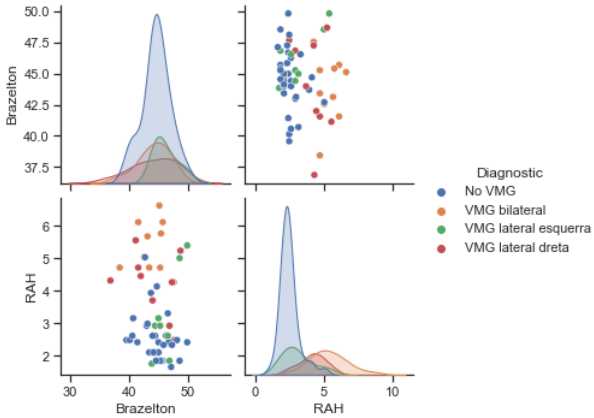

Brazelton
TCD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta


Brazelton
PF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta


Brazelton
CCL
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta



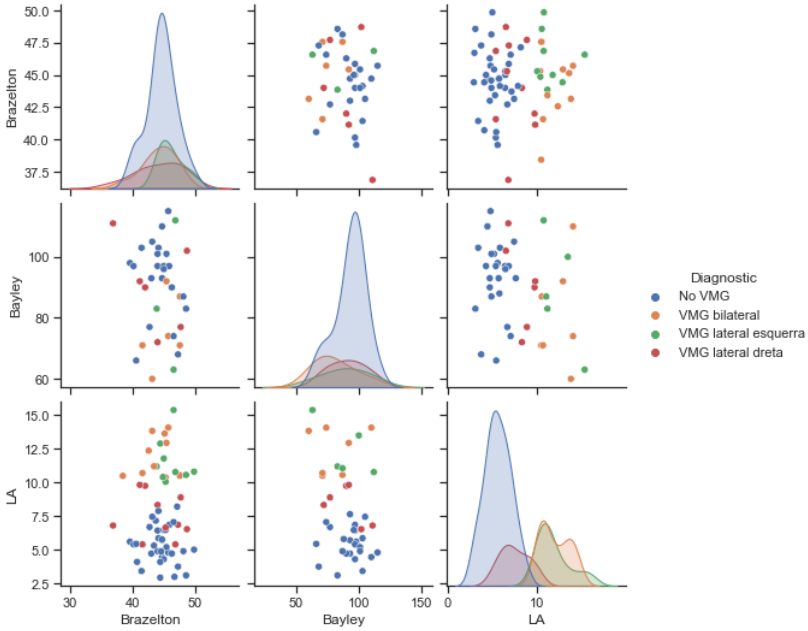


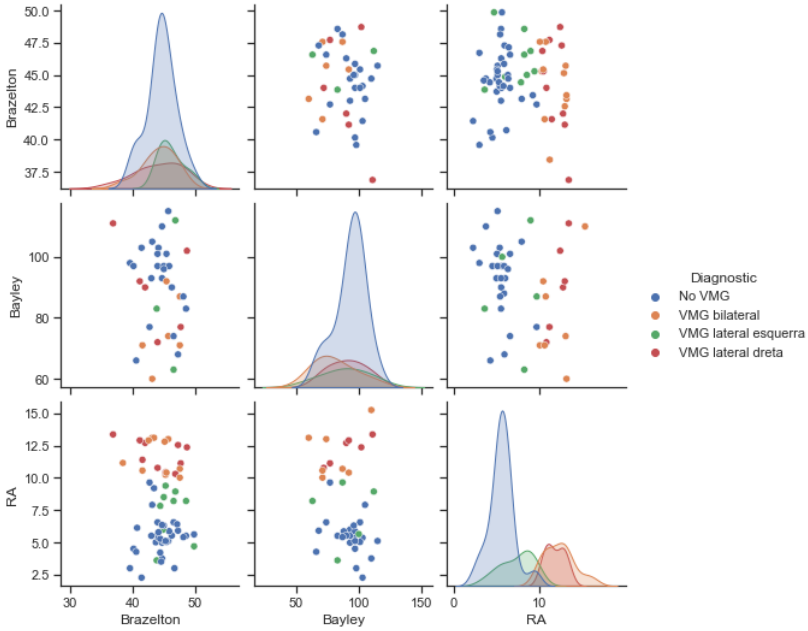


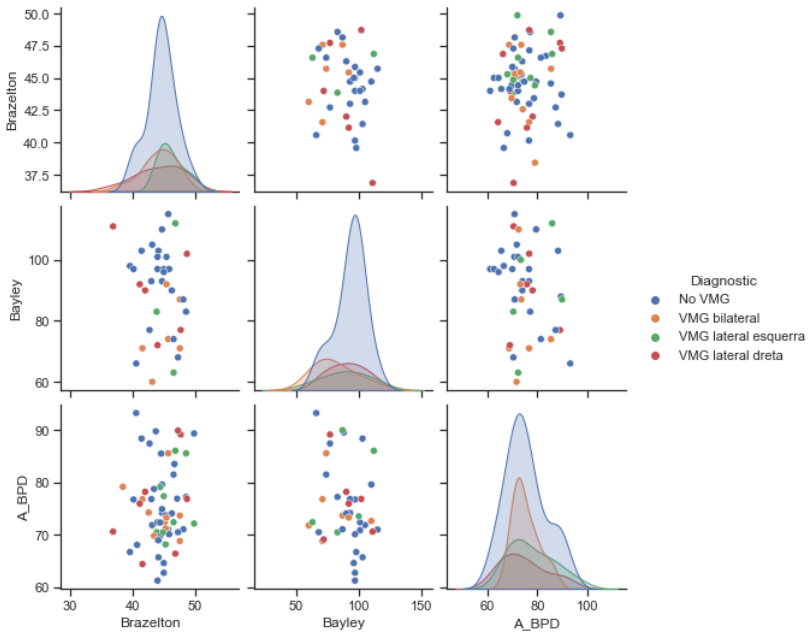
ANÀLISI DELS DOS TESTS JUNTS

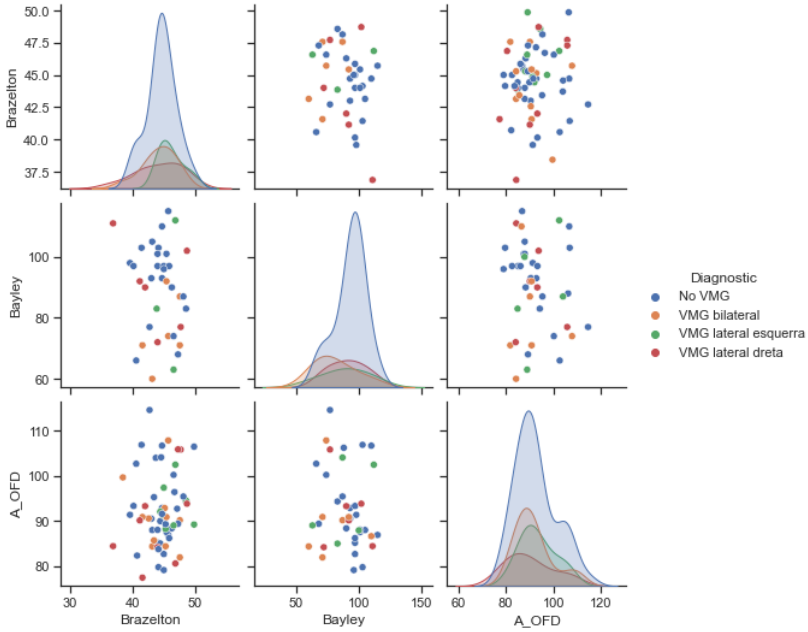
In [34]:

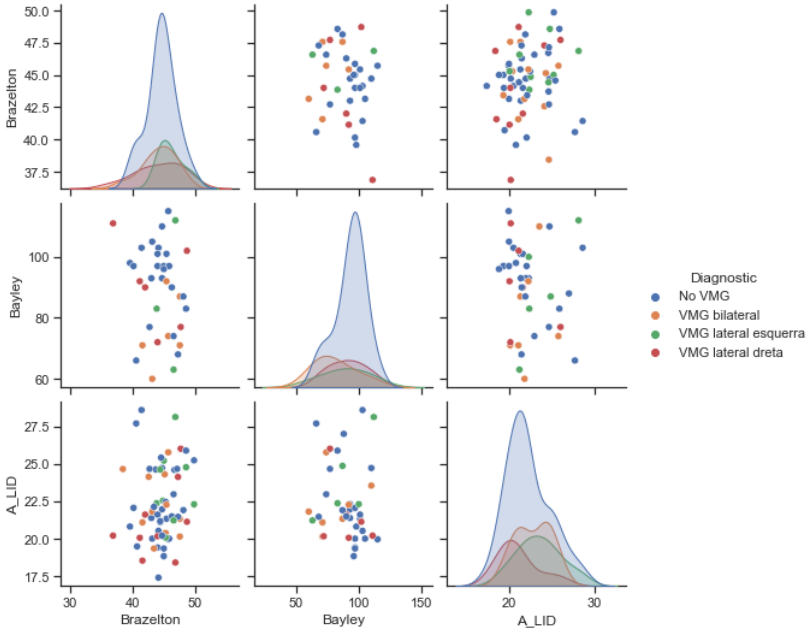
```
# relació de tots els paràmetres amb els dos tests
for parametre in parametres:
    sns.set_theme(style="ticks")
    sns.pairplot(data[["Diagnostic","Brazelton","Bayley", parametre]], hue="Diagnostic")
    plt.show()
```

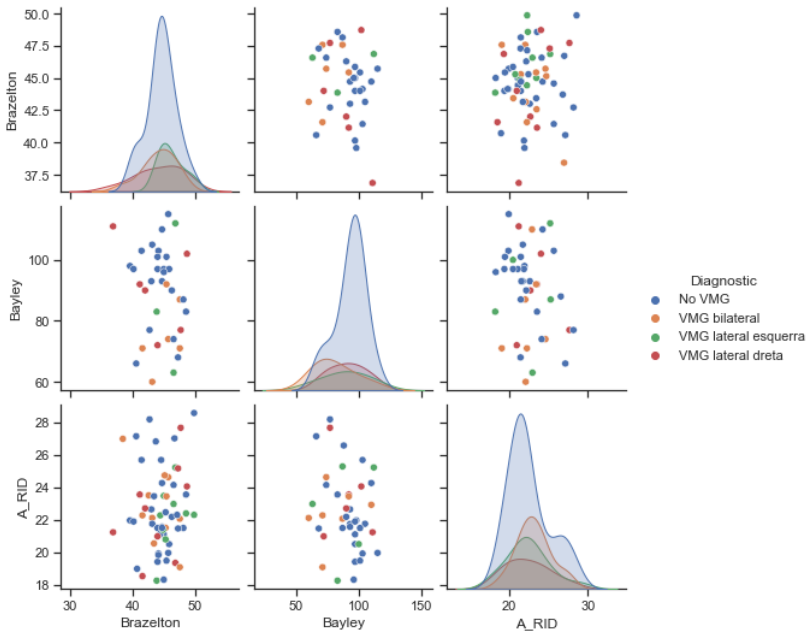

Brazelton
Bayley
LA
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

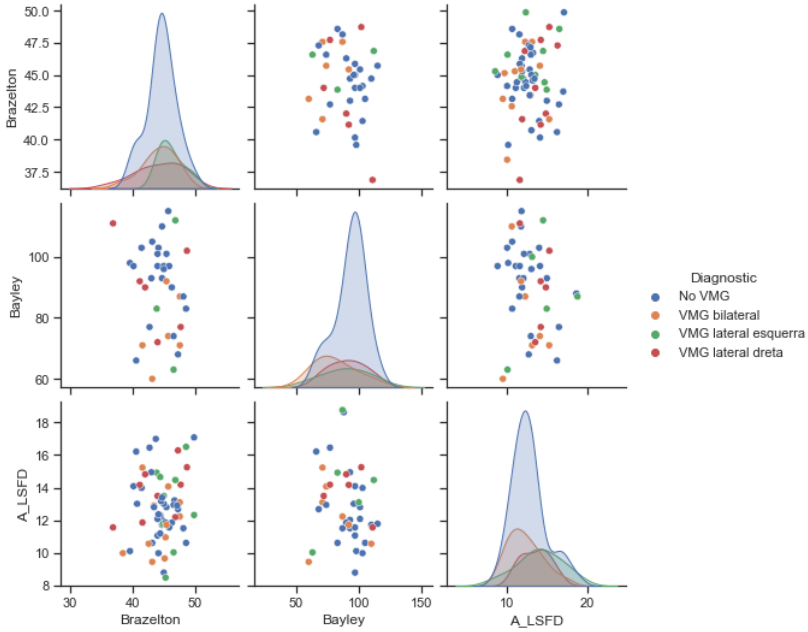

Brazelton
Bayley
RA
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

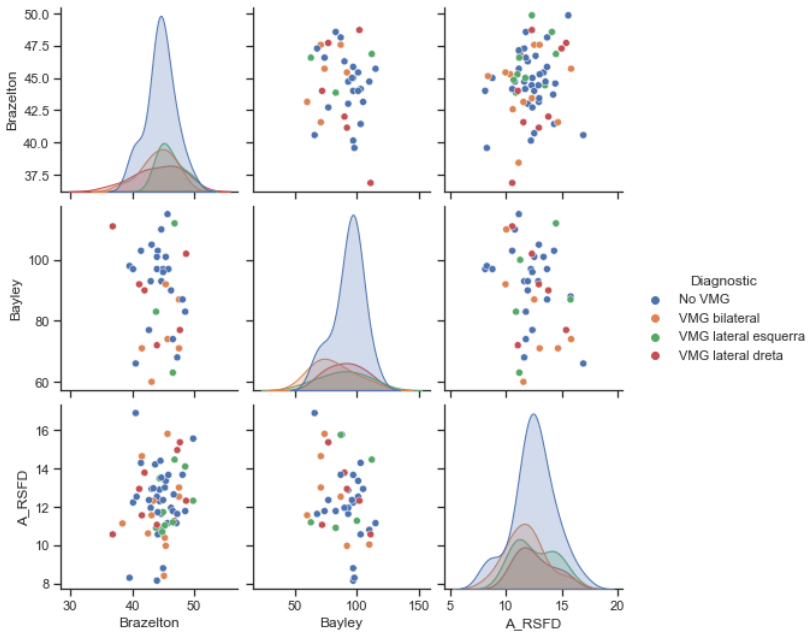

Brazelton
Bayley
A_BPD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

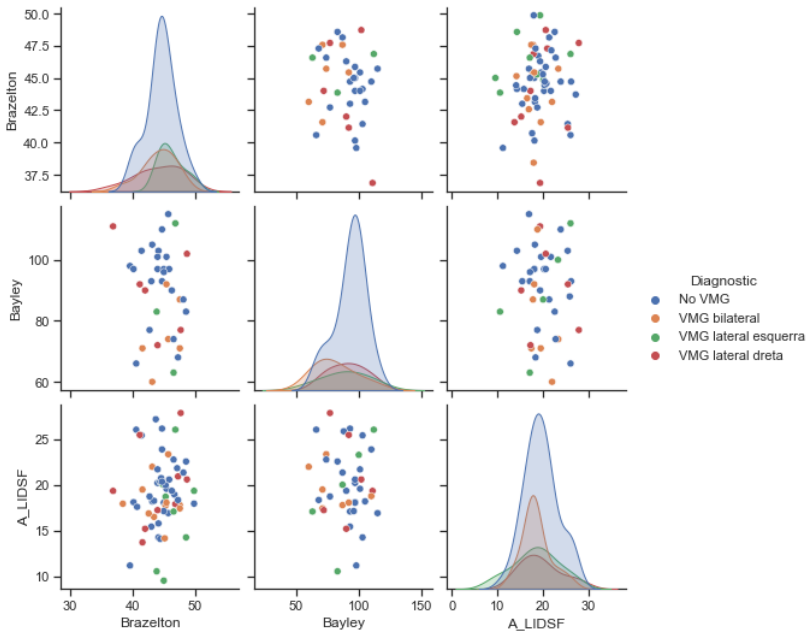

Brazelton
Bayley
A_OFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

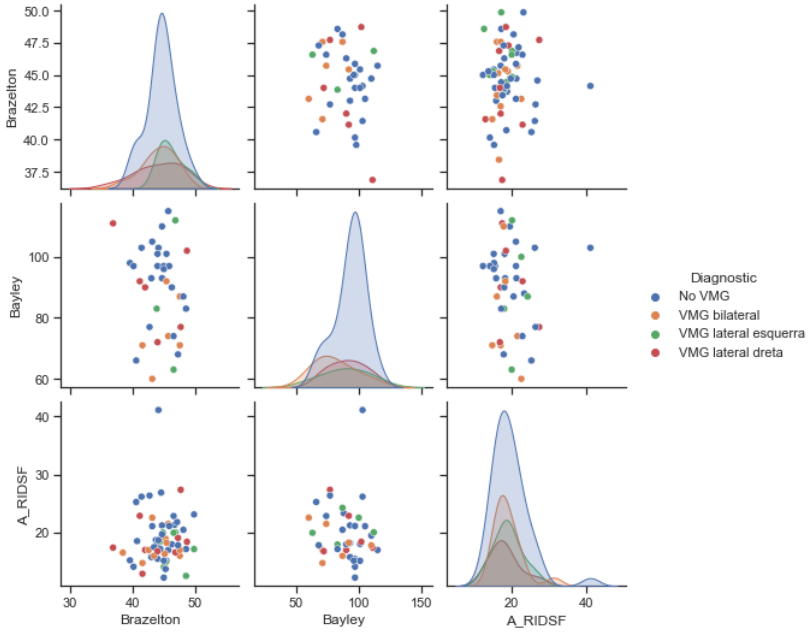

Brazelton
Bayley
A_LID
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

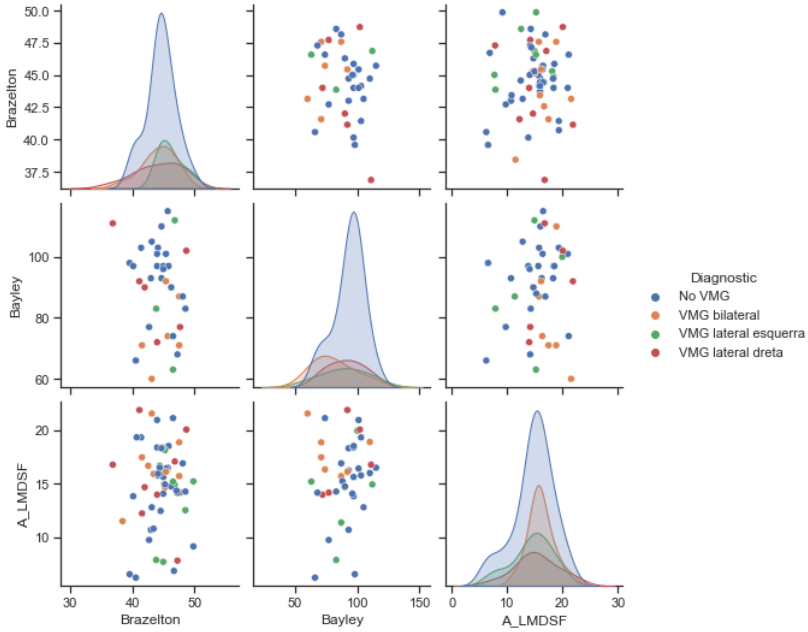

Brazelton
Bayley
A_RID
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

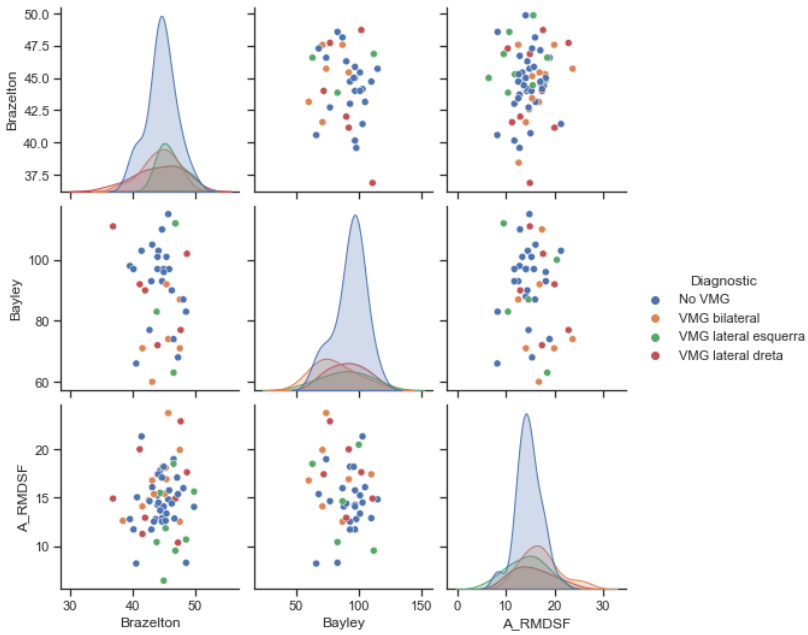

Brazelton
Bayley
A_LSFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

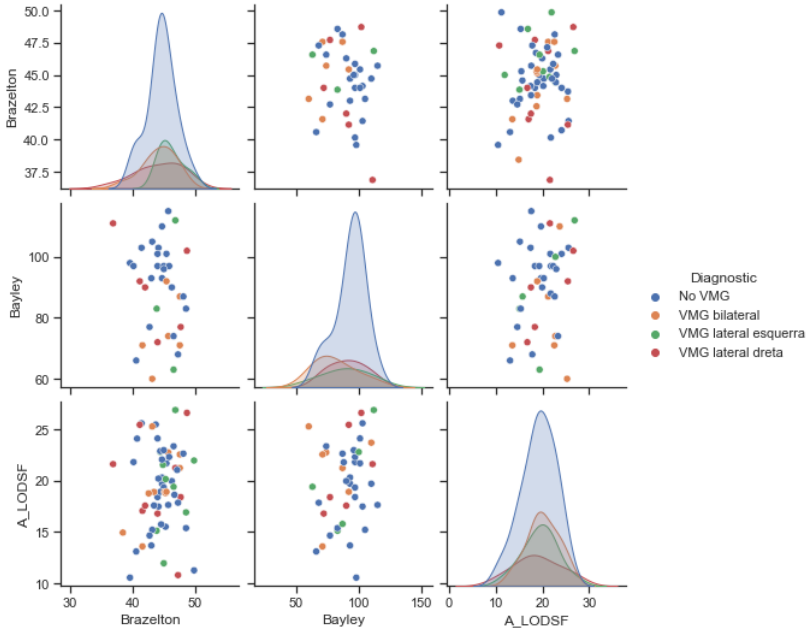

Brazelton
Bayley
A_RSFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

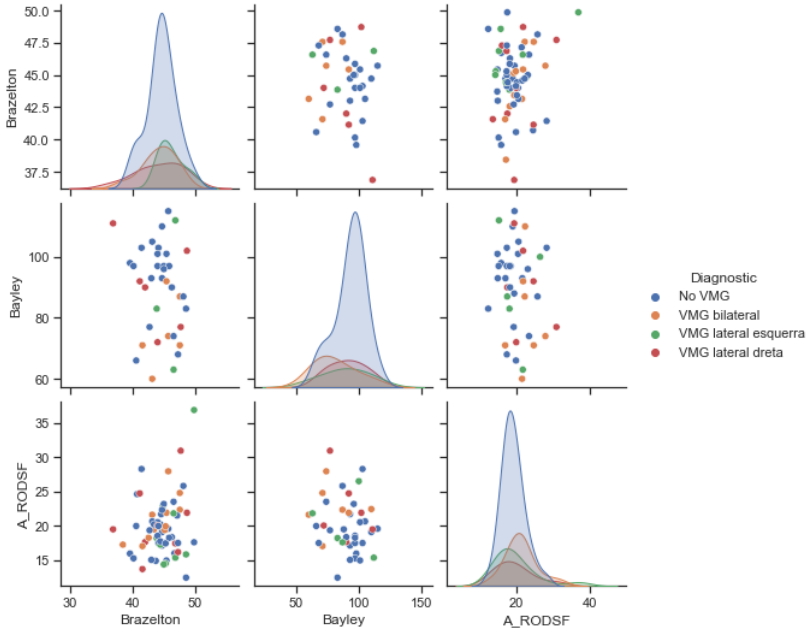

Brazelton
Bayley
A_LIDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

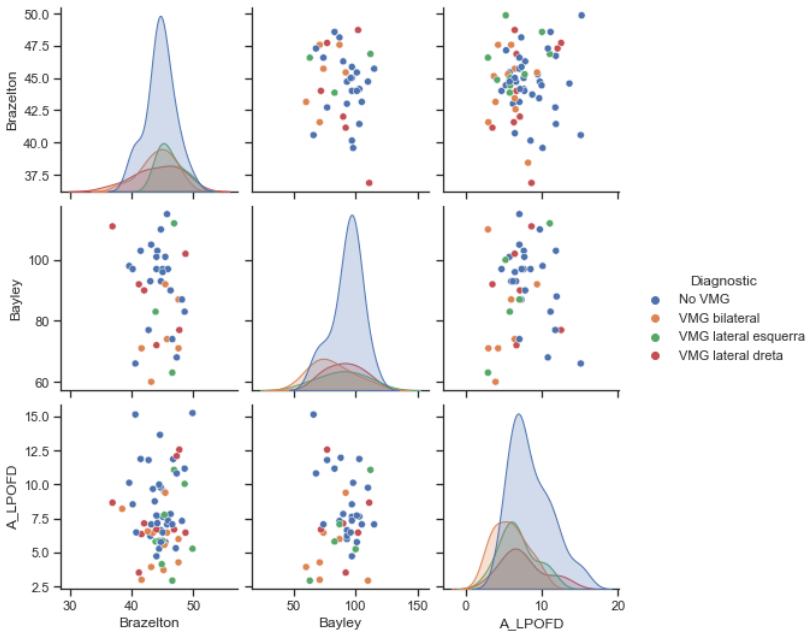

Brazelton
Bayley
A_RIDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

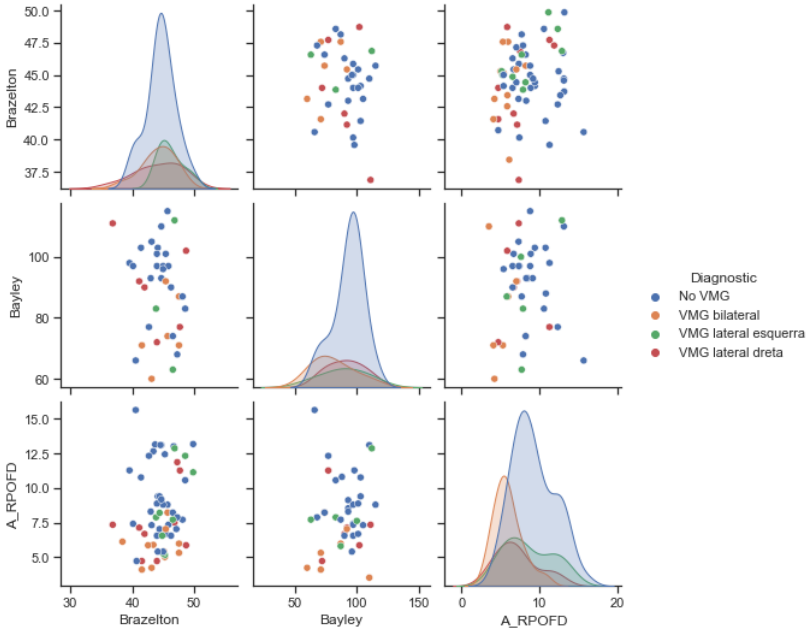

Brazelton
Bayley
A_LMDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

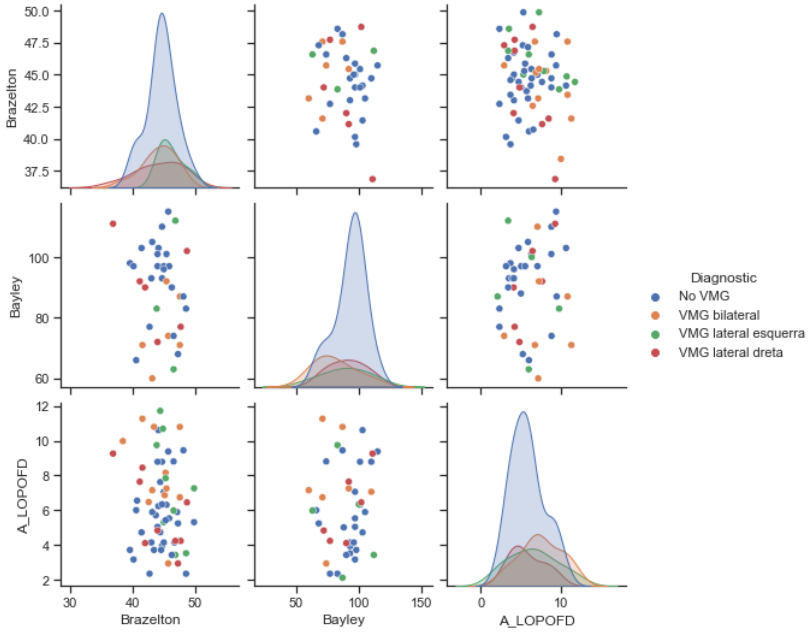

Brazelton
Bayley
A_RMDSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

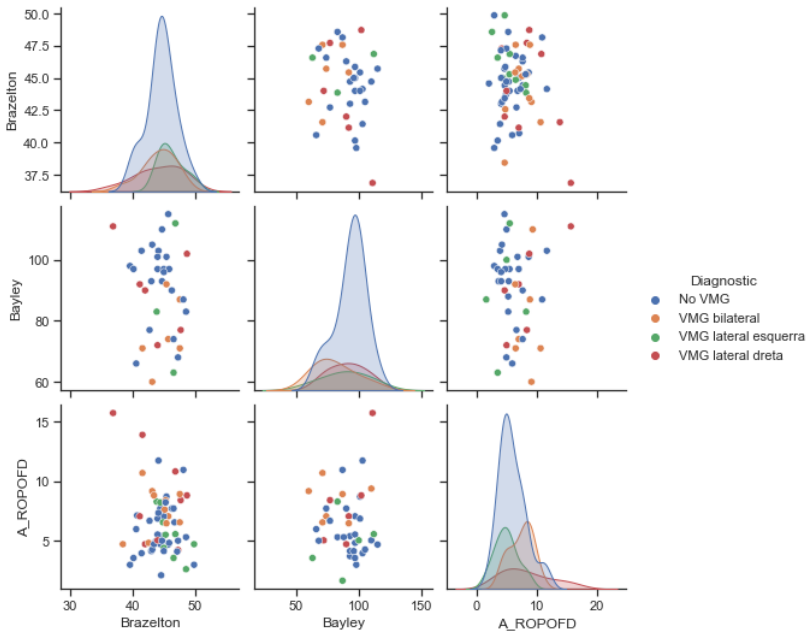

Brazelton
Bayley
A_LODSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

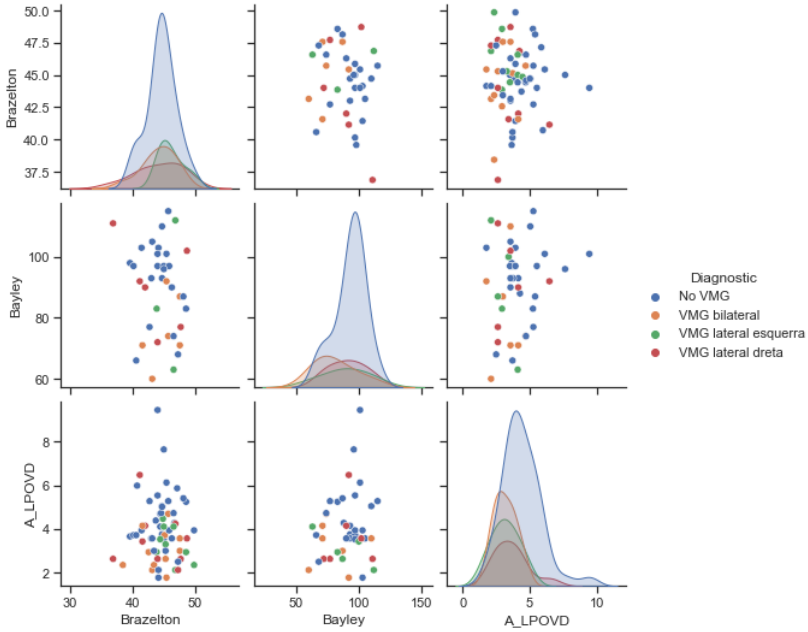

Brazelton
Bayley
A_RODSF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

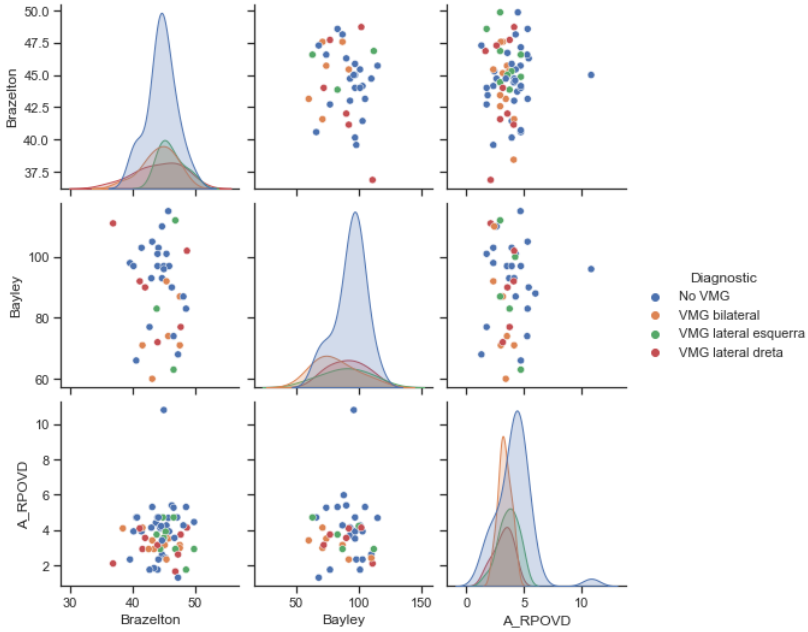

Brazelton
Bayley
A_LPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

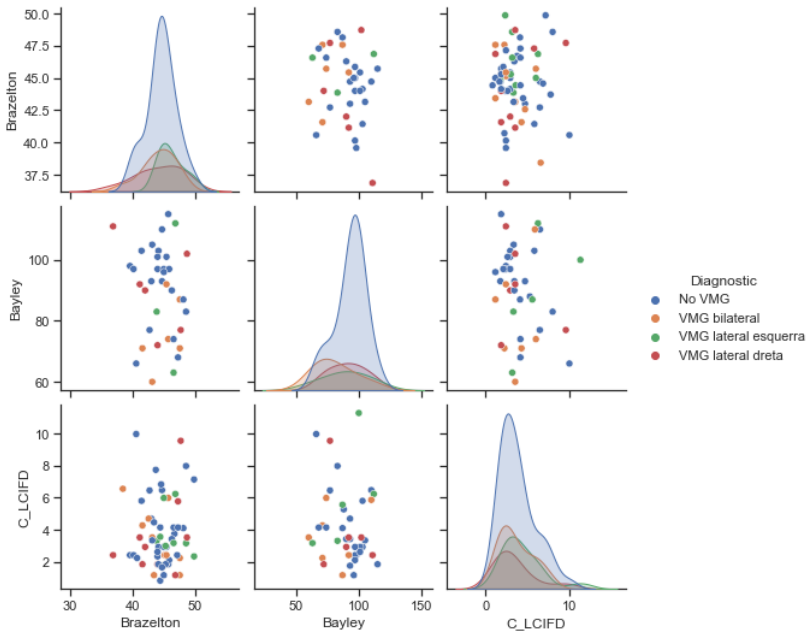

Brazelton
Bayley
A_RPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

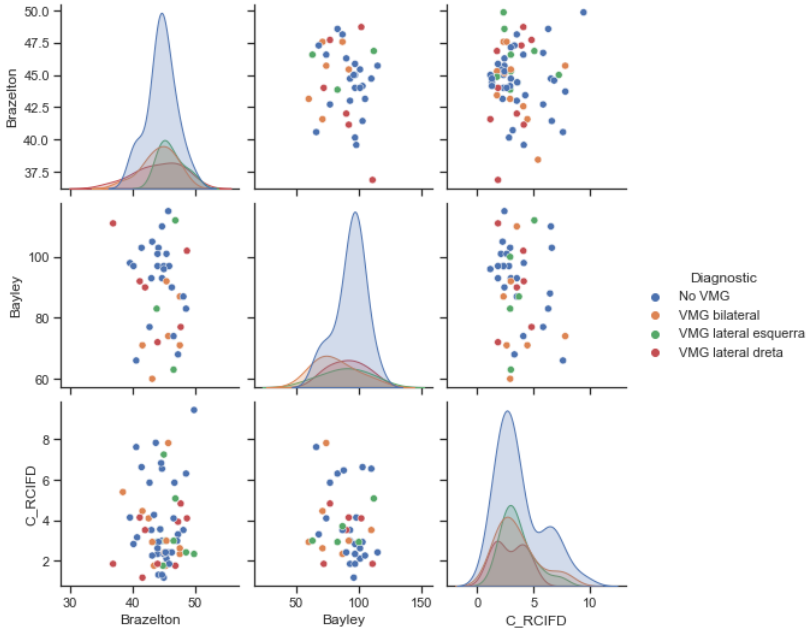

Brazelton
Bayley
A_LOPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

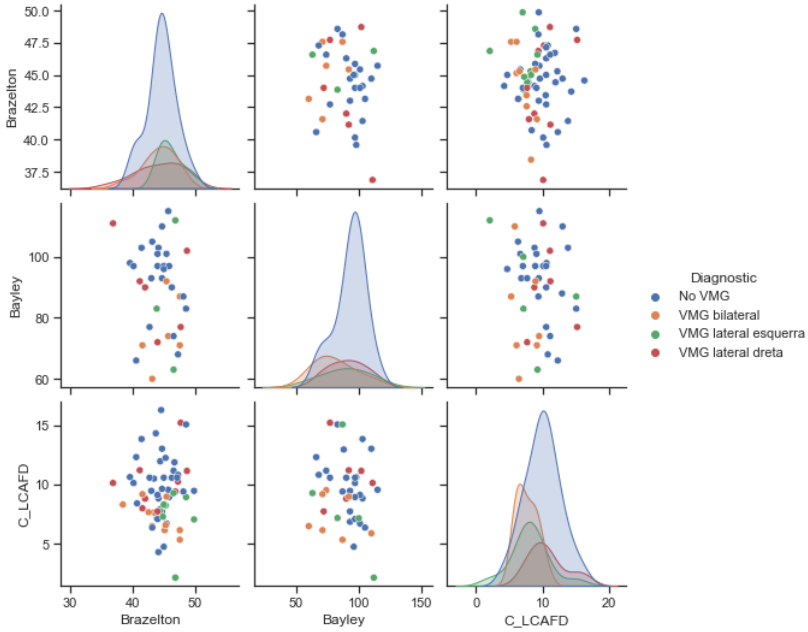

Brazelton
Bayley
A_ROPOFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

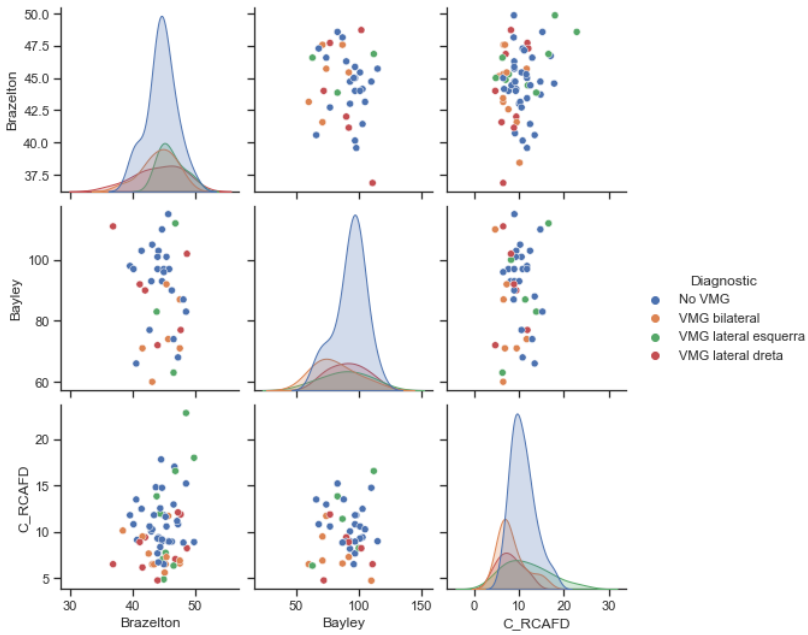

Brazelton
Bayley
A_LPOVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

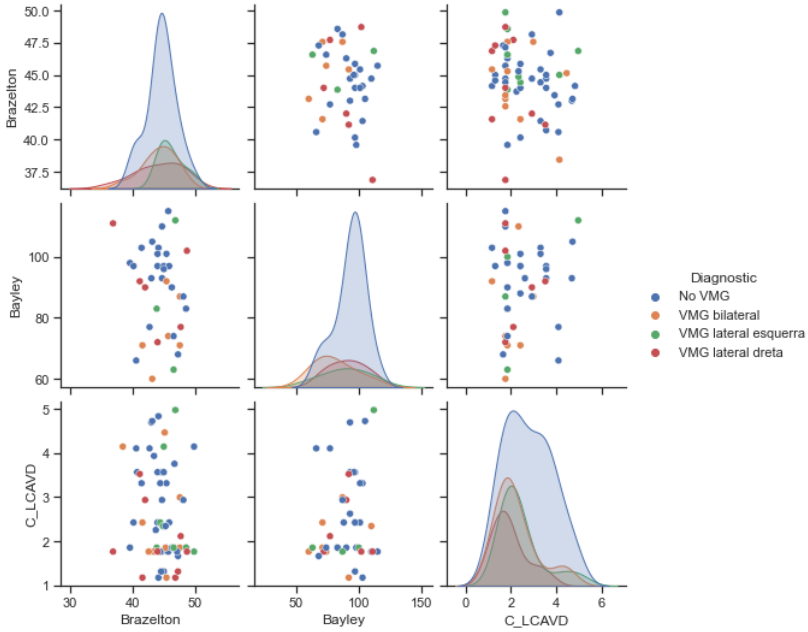

Brazelton
Bayley
A_RPOVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

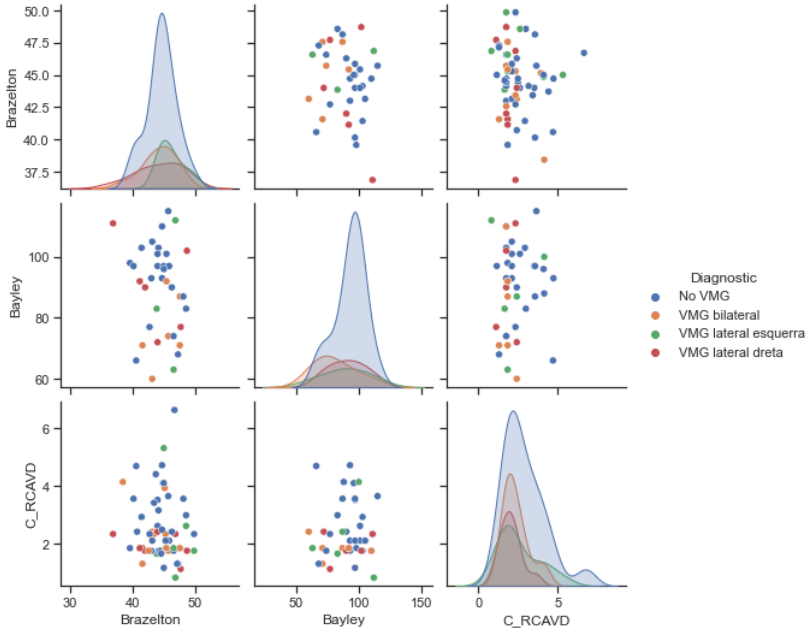

Brazelton
Bayley
C_LCIFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

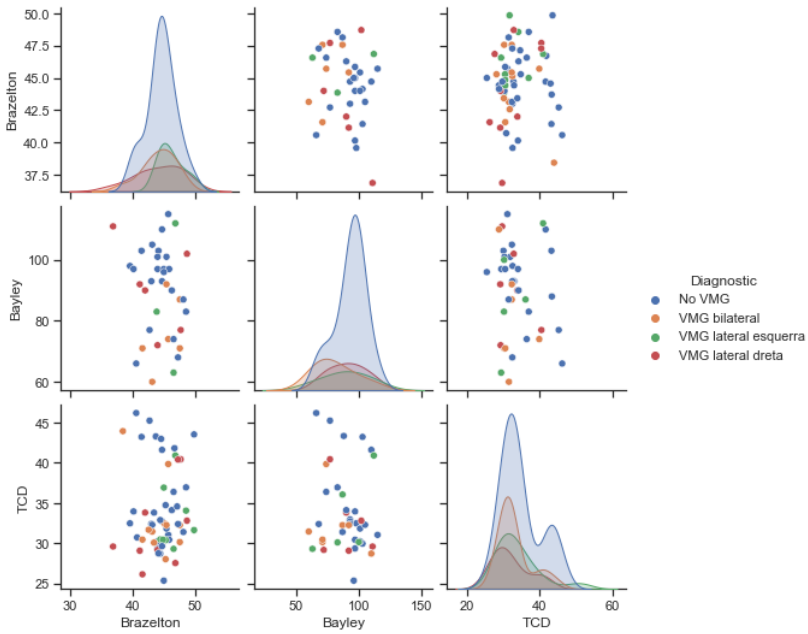

Brazelton
Bayley
C_RCIFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

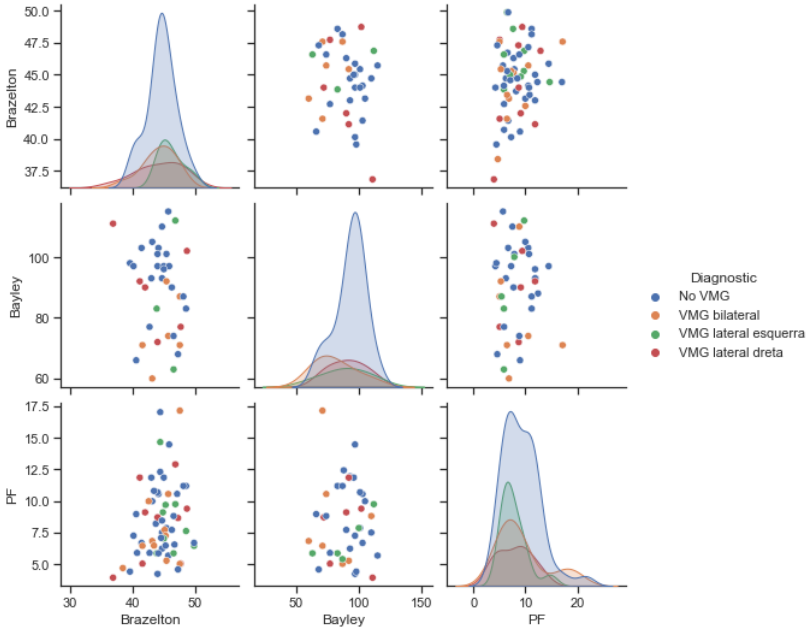

Brazelton
Bayley
C_LCAFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

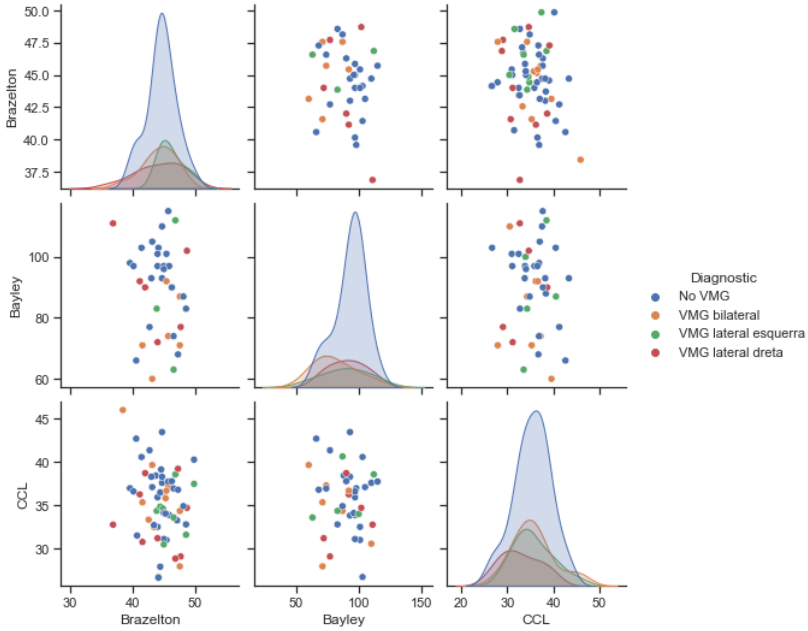

Brazelton
Bayley
C_RCAFD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

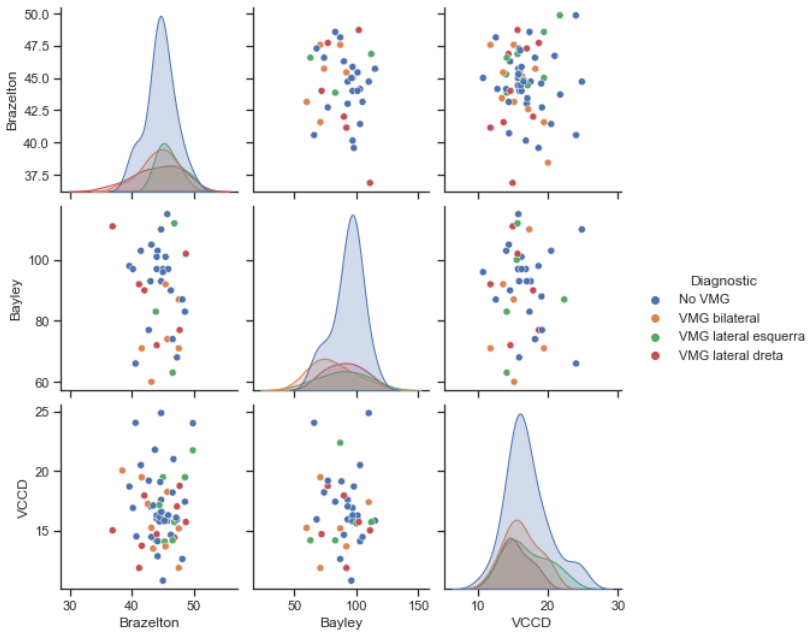

Brazelton
Bayley
C_LCAVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

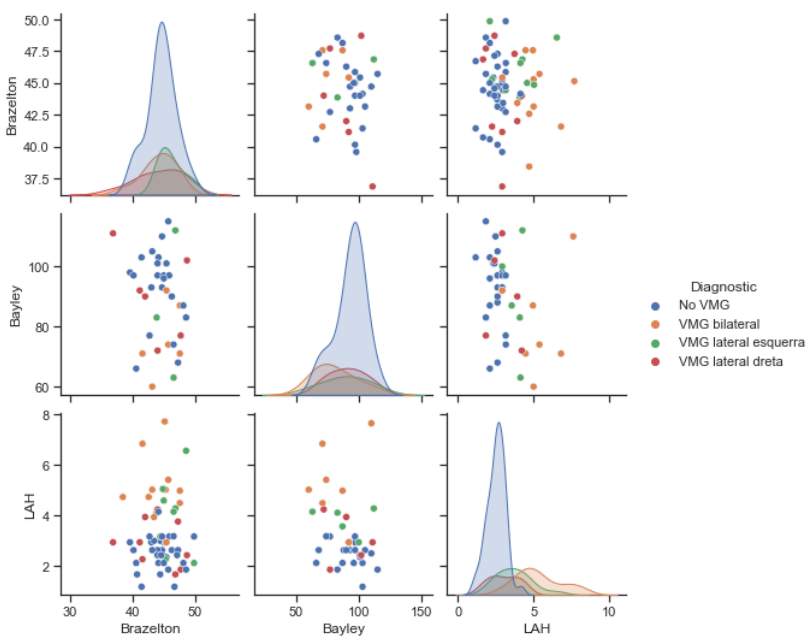

Brazelton
Bayley
C_RCAVD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

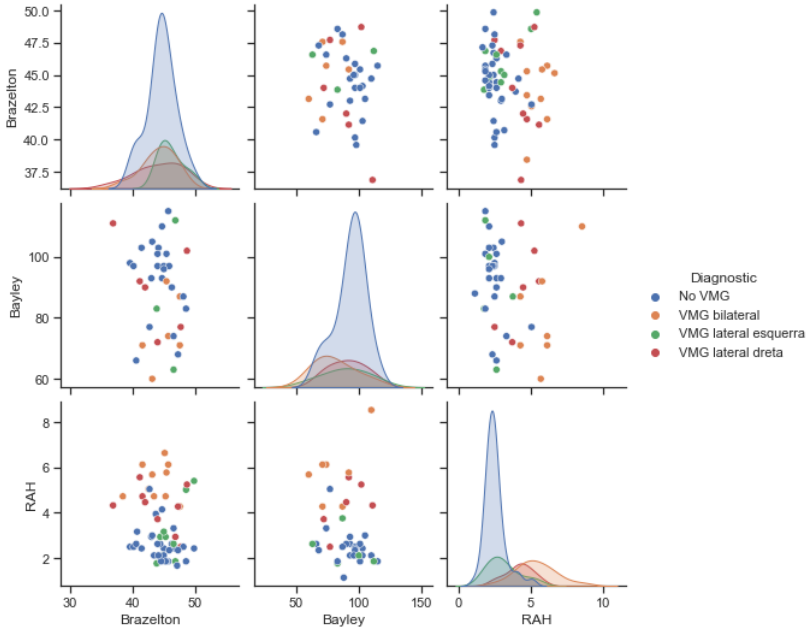

Brazelton
Bayley
TCD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta


Brazelton
Bayley
PF
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta


Brazelton
Bayley
CCL
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta

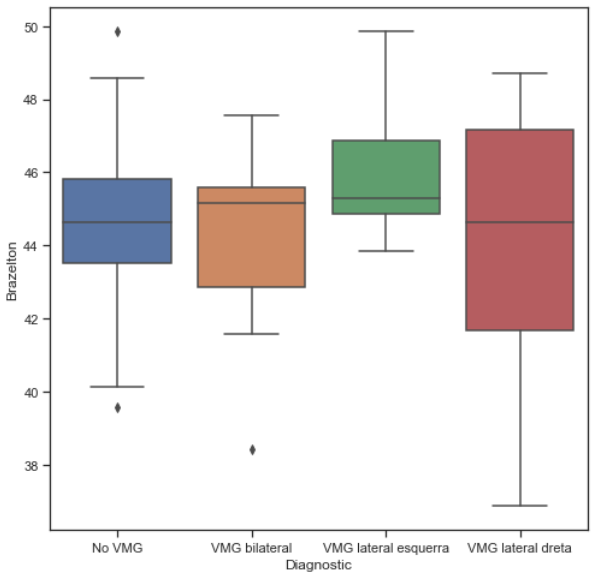

Brazelton
Bayley
VCCD
Diagnostic
No VMG
VMG bilateral
VMG lateral esquerra
VMG lateral dreta





In [35]:

```
# boxplot del test brazelton per diagnòstic
fig, ax = plt.subplots(figsize=(8,8))
sns.boxplot(y='Brazelton', x='Diagnostic', data=data)
plt.show()
```



In [ ]:

ANNEX 4: CODI PYTHON: MODEL DE REGRESSIÓ BAYLEY

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler, RobustScaler, PolynomialFeatures

from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
data = pd.read_pickle("./data.pkl")
```

3. MODEL DE REGRESSIÓ BAYLEY

In [3]:
```
# joc de dades per a aquest model. eliminació de dades post-natals i no numèriques
data_regression_Ba = data.drop(['ID', 'Sexe', 'Diagnostic', 'Vineland', 'Brazelton'],1) #no voldria treure Vineland i Brazelton, pero hi ha
data_regression_Ba = data_regression_Ba.dropna()

data_regression_Ba.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41 entries, 1 to 80
Data columns (total 35 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   LA              41 non-null     float64
 1   RA              41 non-null     float64
 2   Bayley          41 non-null     float64
 3   A_BPD           41 non-null     float64
 4   A_OFD           41 non-null     float64
 5   A_LID           41 non-null     float64
 6   A_RID           41 non-null     float64
 7   A_LSFD          41 non-null     float64
 8   A_RSFD          41 non-null     float64
 9   A_LIDSF         41 non-null     float64
 10  A_RIDSF         41 non-null     float64
 11  A_LMDSF         41 non-null     float64
 12  A_RMDSF         41 non-null     float64
 13  A_LODSF         41 non-null     float64
 14  A_RODSF         41 non-null     float64
 15  A_LPOFD         41 non-null     float64
 16  A_RPOFD         41 non-null     float64
 17  A_LOPOFD        41 non-null     float64
 18  A_ROPOFD        41 non-null     float64
 19  A_LPOVD         41 non-null     float64
 20  A_RPOVD         41 non-null     float64
 21  C_LCIFD         41 non-null     float64
 22  C_RCIFD         41 non-null     float64
 23  C_LCAFD         41 non-null     float64
 24  C_RCAFD         41 non-null     float64
 25  C_LCAVD         41 non-null     float64
 26  C_RCAVD         41 non-null     float64
 27  TCD             41 non-null     float64
 28  PF              41 non-null     float64
 29  CCL             41 non-null     float64
 30  VCCD            41 non-null     float64
 31  LAH             41 non-null     float64
 32  RAH             41 non-null     float64
 33  Sexe_cod        41 non-null     object
 34  Diagnostic_cod  41 non-null     object
dtypes: float64(33), object(2)
memory usage: 11.5+ KB
```

In [4]:
```
#pd.set_option('display.max_rows', None)
#display(data_regression_Ba)
#pd.set_option('display.max_rows', 10)
```

3.1 VALORS SENSE NORMALITZAR

In [5]:
```
# separació de dades i atribut objectiu
X = np.array(data_regression_Ba.drop(['Bayley'],1))
y = np.array(data_regression_Ba['Bayley'])
```

In [6]:
```
# model de regressió linial
model_Ba = linear_model.LinearRegression()
```

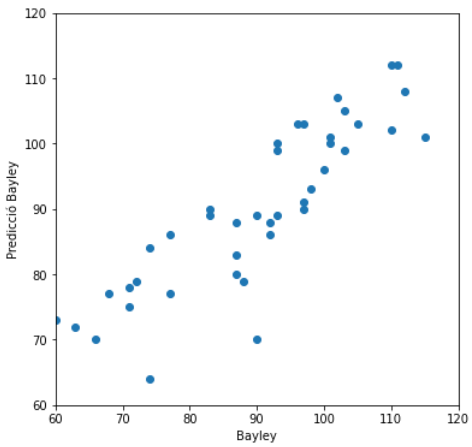
In [7]:
```
# entrenament del model
model_Ba.fit(X,y)
y_pred = np.round(model_Ba.predict(X),0)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Cuadrado Medio
```

```
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
      y   pred
0    74.0   64.0
1    77.0   77.0
2   110.0  112.0
3    87.0   80.0
4    93.0   99.0
5    83.0   90.0
6   103.0  105.0
7    93.0  100.0
8   115.0  101.0
9    97.0   90.0
10   83.0   89.0
11  111.0  112.0
12   88.0   79.0
13   98.0   93.0
14   97.0  103.0
15   93.0   89.0
16  103.0   99.0
17   66.0   70.0
18  101.0  100.0
19   71.0   78.0
20   60.0   73.0
21  112.0  108.0
22   90.0   89.0
23  105.0  103.0
24  101.0  101.0
25   90.0   70.0
26   97.0   91.0
27   96.0  103.0
28   72.0   79.0
29   77.0   86.0
30  100.0   96.0
31  110.0  102.0
32   71.0   75.0
33   92.0   88.0
34  102.0  107.0
35   87.0   88.0
36   74.0   84.0
37   92.0   86.0
38   68.0   77.0
39   63.0   72.0
40   87.0   83.0
Mean squared error: 50.4878
R2 (coeficient de determinacio): 0.7534
```

In [8]:

```
# visualització de l'ajust
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(prediccio['y'],prediccio['pred'])
ax.set_xlim(60, 120)
ax.set_ylim(60, 120)
plt.xlabel('Bayley')
plt.ylabel('Predicció Bayley')
plt.show()
```



In [9]:

```
## definició d'una funció per a avaluar els models
def resultats_CV (model, x, y, cv):
    scoring = ['r2', 'neg_mean_squared_error']
    resultat = model_selection.cross_validate(model, x, y, scoring=scoring, cv=cv)
    print("Model Mean squared error és: {:0.2f} amb una desviació de +/- {:0.2f}".
          format(resultat['test_neg_mean_squared_error'].mean(), resultat['test_neg_mean_squared_error'].std()))
    print("Model R2 (Coef det.) és: {:0.2f} amb una desviació de +/- {:0.2f}".
          format(resultat['test_r2'].mean(), resultat['test_r2'].std()))
```

In [10]:

```
# avaluació de resultats

cv= 10 #es proven diferents valors

resultats_CV (model_Ba, X, y, cv=cv)
```

```
Model Mean squared error és: -7969.80 amb una desviació de +/- 7075.51
Model R2 (Coef det.) és: -202.34 amb una desviació de +/- 413.89
```

3.2 VALORS SENSE NORMALITZAR I AMB CONJUNT TRAIN-TEST

In [11]:

```
# loop de valors diferents de test_size i avaluació dels resultats

for i in np.arange(0.05, 0.50, 0.05):
    print("****Valor de test_size", np.round(i,3))
```

```
    X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = i , random_state = 13)
    model_Ba.fit(X_train,y_train)
    # prediccions sobre train per avaluar el model
    y_pred = model_Ba.predict(X_train)
    # Error Cuadrado Medio
    print("Train Mean squared error: %.4f" % mean_squared_error(y_train, y_pred))
    # Puntaje de Varianza. El mejor puntaje es un 1.0
    print('Train R2 (coeficient de determinacio): %.4f' % r2_score(y_train, y_pred))
    # uso cross_val_score per avaluar el model
    resultats_CV (model_Ba, X_train, y_train, cv=5)

    y_pred_test = model_Ba.predict(X_test)
    prediccio = pd.DataFrame({"y": y_test, "pred": y_pred_test})
    print(prediccio)
    # Error Cuadrado Medio
    print("Test Mean squared error: %.4f" % mean_squared_error(y_test, y_pred_test))
    # Puntaje de Varianza. El mejor puntaje es un 1.0
    print('Test R2 (coeficient de determinacio): %.4f' % r2_score(y_test, y_pred_test))
    print('\n\n')
```

```
****Valor de test_size 0.05
Train Mean squared error: 24.8209
Train R2 (coeficient de determinacio): 0.8741
Model Mean squared error és: -4613.00 amb una desviació de +/- 2402.33
Model R2 (Coef det.) és: -37.67 amb una desviació de +/- 27.56
       y        pred
0  100.0  -18.903322
1   66.0  258.549783
2  105.0  230.519941
Test Mean squared error: 22322.8915
Test R2 (coeficient de determinacio): -73.3546



****Valor de test_size 0.1
Train Mean squared error: 24.6384
Train R2 (coeficient de determinacio): 0.8754
Model Mean squared error és: -2781.94 amb una desviació de +/- 991.13
Model R2 (Coef det.) és: -22.00 amb una desviació de +/- 16.70
       y        pred
0  100.0  -65.055690
1   66.0  180.484417
2  105.0  244.554225
3   71.0   68.196064
4   93.0   42.488255
Test Mean squared error: 12476.9485
Test R2 (coeficient de determinacio): -49.8848



****Valor de test_size 0.15
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -1345.99 amb una desviació de +/- 281.37
Model R2 (Coef det.) és: -13.35 amb una desviació de +/- 12.49
       y        pred
0  100.0   92.259525
1   66.0 -135.229948
2  105.0 -112.061132
3   71.0  138.695615
4   93.0   64.565738
5   68.0  152.849361
6   92.0 -150.209339
Test Mean squared error: 22703.5605
Test R2 (coeficient de determinacio): -99.3314



****Valor de test_size 0.2
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -1761.27 amb una desviació de +/- 365.27
Model R2 (Coef det.) és: -14.05 amb una desviació de +/- 9.70
       y        pred
0  100.0  -11.891772
1   66.0   92.576583
2  105.0   88.982923
3   71.0   -6.685824
4   93.0   49.435395
5   68.0  111.964091
6   92.0  -71.625068
7   97.0  154.309401
8   87.0  101.697508
Test Mean squared error: 5957.9979
Test R2 (coeficient de determinacio): -30.3538



****Valor de test_size 0.25
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -1481.96 amb una desviació de +/- 383.07
Model R2 (Coef det.) és: -12.32 amb una desviació de +/- 6.78
       y        pred
0   100.0  156.117859
1    66.0  195.193729
2   105.0   60.063876
3    71.0  143.654604
4    93.0  134.233469
5    68.0  120.170508
6    92.0    3.723260
7    97.0  111.260521
8    87.0   78.824503
9    93.0  133.756213
10   83.0  110.566630
Test Mean squared error: 3822.1921
Test R2 (coeficient de determinacio): -22.8394



****Valor de test_size 0.3
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -1520.27 amb una desviació de +/- 570.21
```

```
Model R2 (Coef det.) és: -8.01 amb una desviació de +/- 4.59
        y        pred
0   100.0  177.923589
1    66.0  205.339658
2   105.0   78.804152
3    71.0   72.332078
4    93.0  139.564779
5    68.0   82.950934
6    92.0    7.067771
7    97.0  112.343983
8    87.0   57.141974
9    93.0  134.764634
10   83.0  114.439296
11   93.0  103.500555
12   96.0  165.853562
Test Mean squared error: 3433.1039
Test R2 (coeficient de determinacio): -22.8921


****Valor de test_size 0.35
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -769.75 amb una desviació de +/- 385.28
Model R2 (Coef det.) és: -3.71 amb una desviació de +/- 3.11
        y        pred
0   100.0   88.230644
1    66.0   90.711433
2   105.0   67.271860
3    71.0   28.217154
4    93.0   81.559711
5    68.0  116.136082
6    92.0   52.054041
7    97.0  114.381554
8    87.0   66.358543
9    93.0  127.206528
10   83.0   98.681226
11   93.0   80.507297
12   96.0  101.505301
13  103.0  109.041499
14  110.0   19.699045
Test Mean squared error: 1237.8609
Test R2 (coeficient de determinacio): -6.4698


****Valor de test_size 0.4
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -696.07 amb una desviació de +/- 516.12
Model R2 (Coef det.) és: -6.41 amb una desviació de +/- 8.71
        y        pred
0   100.0   62.335267
1    66.0  113.510087
2   105.0   64.314570
3    71.0   45.558509
4    93.0   83.214723
5    68.0   97.893394
6    92.0   48.634138
7    97.0  105.249443
8    87.0   73.510076
9    93.0  114.224156
10   83.0  113.601037
11   93.0   67.287143
12   96.0  115.485444
13  103.0  101.101340
14  110.0   27.832380
15   88.0  114.522540
16  115.0   96.777149
Test Mean squared error: 1136.2807
Test R2 (coeficient de determinacio): -5.3023


****Valor de test_size 0.45
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -707.93 amb una desviació de +/- 486.81
Model R2 (Coef det.) és: -6.67 amb una desviació de +/- 8.32
        y        pred
0   100.0   43.379870
1    66.0   98.470454
2   105.0   74.687963
3    71.0   43.506440
4    93.0   83.019306
5    68.0   99.629155
6    92.0   44.977232
7    97.0   94.322659
8    87.0   72.920549
9    93.0  111.553429
10   83.0   97.590734
11   93.0   71.821203
12   96.0  120.989447
13  103.0   99.426980
14  110.0   23.883833
15   88.0  113.766414
16  115.0   99.392596
17   72.0   78.452276
18   87.0   74.615239
Test Mean squared error: 1032.2552
Test R2 (coeficient de determinacio): -4.6903
```

In [12]:

```
# creació del conjunt d'entrenament i test

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.5, random_state=13)
#print("X_train:", X_train.shape)
#print("X_test:", X_test.shape)
```

In [13]:

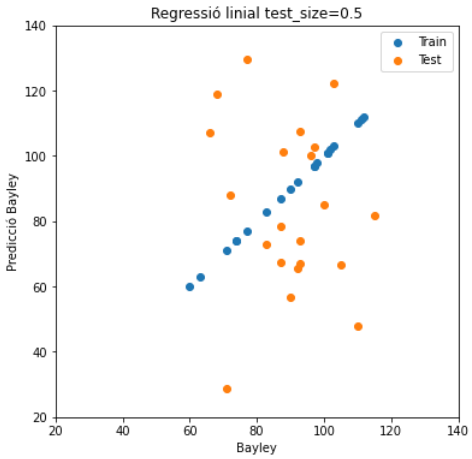
```
X_test.shape
```

Out[13]: (21, 34)

In [14]:
```
# definició de model i entrenament
model_Ba = linear_model.LinearRegression()
model_Ba.fit(X_train,y_train)
```

Out[14]: LinearRegression()

In [15]:
```
# visualització de l'ajust
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y_train,np.round(model_Ba.predict(X_train),2))
plt.scatter(y_test,np.round(model_Ba.predict(X_test),2))
ax.set_xlim(20, 140)
ax.set_ylim(20, 140)
ax.legend(['Train', 'Test'])
plt.xlabel('Bayley')
plt.ylabel('Predicció Bayley')
plt.title('Regressió linial test_size=0.5')
plt.show()
```

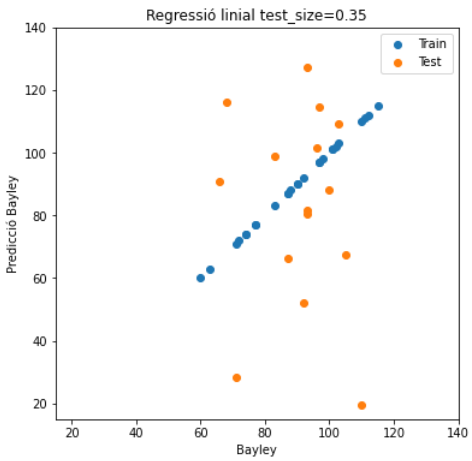


In [16]:
```
# repetir amb un altre valor de test_size
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.35, random_state=13)
#print("X_train:", X_train.shape)
#print("X_test:", X_test.shape)
```

In [17]:
```
model_Ba = linear_model.LinearRegression()
model_Ba.fit(X_train,y_train)
```

Out[17]: LinearRegression()

In [18]:
```
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y_train,np.round(model_Ba.predict(X_train),2))
plt.scatter(y_test,np.round(model_Ba.predict(X_test),2))
ax.set_xlim(15, 140)
ax.set_ylim(15, 140)
ax.legend(['Train', 'Test'])
plt.xlabel('Bayley')
plt.ylabel('Predicció Bayley')
plt.title('Regressió linial test_size=0.35')
plt.show()
```

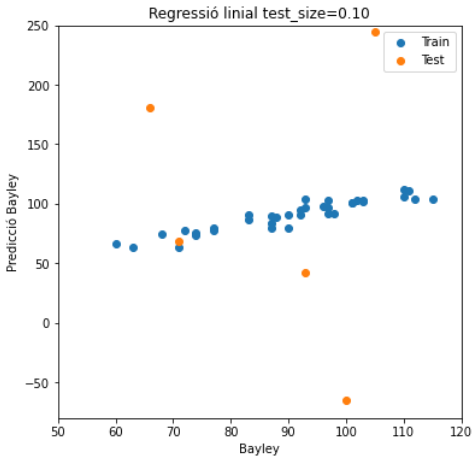


In [19]:
```
# repetir amb un tercer valor de test_size
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.10, random_state=13)
#print("X_train:", X_train.shape)
#print("X_test:", X_test.shape)
```

In [20]:
```
model_Ba = linear_model.LinearRegression()
model_Ba.fit(X_train,y_train)
```

Out[20]:
```
LinearRegression()
```

In [21]:
```
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y_train,np.round(model_Ba.predict(X_train),2))
plt.scatter(y_test,np.round(model_Ba.predict(X_test),2))
ax.set_xlim(50, 120)
ax.set_ylim(-80, 250)
ax.legend(['Train', 'Test'])
plt.xlabel('Bayley')
plt.ylabel('Predicció Bayley')
plt.title('Regressió linial test_size=0.10')
plt.show()
```



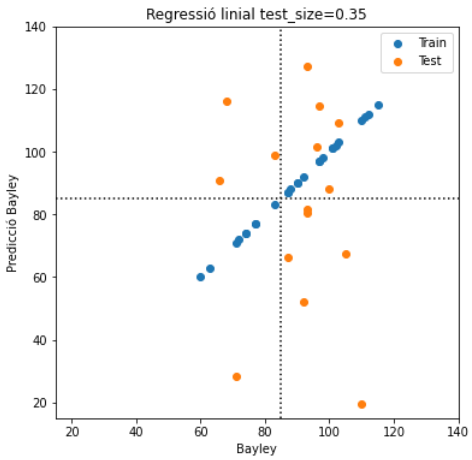
In [22]:
```
## split analitzat: escollim el valor intermig
```

In [23]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.35, random_state=13)
#print("X_train:", X_train.shape)
#print("X_test:", X_test.shape)
```

In [24]:
```
model_Ba = linear_model.LinearRegression()
model_Ba.fit(X_train,y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
# visualització ajust amb els límits Bayley=85
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y_train,np.round(model_Ba.predict(X_train),2))
plt.scatter(y_test,np.round(model_Ba.predict(X_test),2))
ax.set_xlim(15, 140)
ax.set_ylim(15, 140)
ax.legend(['Train', 'Test'])
plt.xlabel('Bayley')
plt.ylabel('Predicció Bayley')
plt.axvline(x=85, color="black", linestyle=":")
plt.axhline(y=85, color="black", linestyle=":")
plt.title('Regressió linial test_size=0.35')
plt.show()
```



3.3 VALORS NORMALITZATS

In [26]:
```
# es repetexi el procés amb valors normalitzats

# normalitzacio
columns_Ba = data_regression_Ba.columns
scaler_Ba = StandardScaler()
scaler_Ba.fit(data_regression_Ba)
data_regression_Ba_norm = scaler_Ba.transform(data_regression_Ba)

# torno a muntar el df
```

```
data_regression_Ba_norm = pd.DataFrame(data_regression_Ba_norm)
data_regression_Ba_norm.columns = columns_Ba
```

In [27]:
```
#pd.set_option('display.max_rows', None)
#display(data_regression_Ba_norm)
#pd.set_option('display.max_rows', 10)
```

In [28]:
```
# separació de parametres i valor objectiu
X = np.array(data_regression_Ba_norm.drop(['Bayley'],1))
y = np.array(data_regression_Ba_norm['Bayley'])
```

In [29]:
```
model_Ba_norm = linear_model.LinearRegression()
```

In [30]:
```
# entrenament
model_Ba_norm.fit(X,y)
y_pred = model_Ba_norm.predict(X)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)
# Error Cuadrado Medio
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
           y      pred
0  -1.116451 -1.802099
1  -0.906797 -0.910684
2   1.399399  1.526441
3  -0.207950 -0.682322
4   0.211359  0.600676
5  -0.487489 -0.017722
6   0.910206  1.061009
7   0.211359  0.726618
8   1.748822  0.742973
9   0.490898 -0.026275
10 -0.487489 -0.059401
11  1.469284  1.530158
12 -0.138065 -0.779441
13  0.560782  0.178042
14  0.490898  0.881004
15  0.211359 -0.101585
16  0.910206  0.618278
17 -1.675529 -1.373537
18  0.770436  0.727703
19 -1.326105 -0.842080
20 -2.094837 -1.166422
21  1.539168  1.248967
22  0.001705 -0.062461
23  1.049975  0.926160
24  0.770436  0.765555
25  0.001705 -1.390909
26  0.490898  0.100470
27  0.421013  0.910678
28 -1.256220 -0.737499
29 -0.906797 -0.303250
30  0.700552  0.393058
31  1.399399  0.832725
32 -1.326105 -1.055893
33  0.141474 -0.162778
34  0.840321  1.187975
35 -0.207950 -0.131352
36 -1.116451 -0.435061
37  0.141474 -0.257942
38 -1.535759 -0.915061
39 -1.885183 -1.279470
40 -0.207950 -0.465247
Mean squared error: 0.2473
R2 (coeficient de determinacio): 0.7527
```

In [31]:
```
# avaluació de resultats amb la funció propia
resultats_CV (model_Ba_norm, X, y, cv=5)
```

```
Model Mean squared error és: -54.20 amb una desviació de +/- 46.80
Model R2 (Coef det.) és: -71.57 amb una desviació de +/- 54.52
```

3.4 VALORS NORMALITZATS AMB ROBUSTSCALER

In [32]:
```
# Anàlisi amb un nou model de normalització
columns_Ba = data_regression_Ba.columns
scaler_Ba = RobustScaler()
scaler_Ba.fit(data_regression_Ba)
data_regression_Ba_norm = scaler_Ba.transform(data_regression_Ba)

# torno a muntar el df
data_regression_Ba_norm = pd.DataFrame(data_regression_Ba_norm)
data_regression_Ba_norm.columns = columns_Ba
```

In [33]:
```
X = np.array(data_regression_Ba_norm.drop(['Bayley'],1))
y = np.array(data_regression_Ba_norm['Bayley'])
```

In [34]:
```
model_Ba_norm = linear_model.LinearRegression()
```

In [35]:
```
model_Ba_norm.fit(X,y)
y_pred = model_Ba_norm.predict(X)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)
# Error Cuadrado Medio
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
           y      pred
0  -0.750000 -1.158797
1  -0.625000 -0.627318
2   0.750000  0.825745
3  -0.208333 -0.491163
4   0.041667  0.273786
5  -0.375000 -0.094916
6   0.458333  0.548245
7   0.041667  0.348874
8   0.958333  0.358626
9   0.208333 -0.100015
10 -0.375000 -0.119766
11  0.791667  0.827961
12 -0.166667 -0.549068
13  0.250000  0.021803
14  0.208333  0.440923
15  0.041667 -0.144916
16  0.458333  0.284280
17 -1.083333 -0.903280
18  0.375000  0.349522
19 -0.875000 -0.586415
20 -1.333333 -0.779794
21  0.833333  0.660309
22 -0.083333 -0.121590
23  0.541667  0.467846
24  0.375000  0.372090
25 -0.083333 -0.913638
26  0.208333 -0.024447
27  0.166667  0.458615
28 -0.833333 -0.524061
29 -0.625000 -0.265153
30  0.333333  0.149999
31  0.750000  0.412138
32 -0.875000 -0.713894
33  0.000000 -0.181401
34  0.416667  0.623945
35 -0.208333 -0.162664
36 -0.750000 -0.343742
37  0.000000 -0.238140
38 -1.000000 -0.629927
39 -1.208333 -0.847195
40 -0.208333 -0.361739
Mean squared error: 0.0879
R2 (coeficient de determinacio): 0.7527
```

In [36]:
```
resultats_CV (model_Ba_norm, X, y, cv=5)
```

```
Model Mean squared error és: -19.07 amb una desviació de +/- 16.41
Model R2 (Coef det.) és: -70.85 amb una desviació de +/- 53.81
```

3.5 REGRESSIÓ NO-LINIAL

In [37]:
```
# Separació de dades de nou
X = np.array(data_regression_Ba.drop(['Bayley'],1))
y = np.array(data_regression_Ba['Bayley'])
```

In [38]:
```
# GRAU 4
pf = PolynomialFeatures(degree = 4)    # es provaran graus de 2 a 4
X = pf.fit_transform(X)  # es transforma l'entrada en polinòmica
```

In [39]:
```
model_Ba = linear_model.LinearRegression()
```

In [40]:
```
# entrenament
model_Ba.fit(X,y)
y_pred = np.round(model_Ba.predict(X),0)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Quadratic Mig
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Coeficient de determinació
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
        y   pred
0    74.0   74.0
1    77.0   77.0
2   110.0  110.0
3    87.0   87.0
4    93.0   93.0
5    83.0   83.0
6   103.0  103.0
7    93.0   93.0
8   115.0  115.0
9    97.0   97.0
10   83.0   83.0
11  111.0  111.0
12   88.0   88.0
13   98.0   98.0
14   97.0   97.0
15   93.0   93.0
16  103.0  103.0
17   66.0   66.0
18  101.0  101.0
19   71.0   71.0
20   60.0   60.0
21  112.0  112.0
22   90.0   90.0
23  105.0  105.0
24  101.0  101.0
25   90.0   90.0
26   97.0   97.0
27   96.0   96.0
28   72.0   72.0
29   77.0   77.0
30  100.0  100.0
31  110.0  110.0
32   71.0   71.0
```

```
33   92.0   92.0
34  102.0  102.0
35   87.0   87.0
36   74.0   74.0
37   92.0   92.0
38   68.0   68.0
39   63.0   63.0
40   87.0   87.0
Mean squared error: 0.0000
R2 (coeficient de determinacio): 1.0000
```

In [41]:
```
#avaluació amb funció pròpia
resultats_CV (model_Ba, X, y, cv=5)
```

```
Model Mean squared error és: -2261.28 amb una desviació de +/- 1814.17
Model R2 (Coef det.) és: -10.41 amb una desviació de +/- 3.44
```

In [42]:
```
# REPETICIÓ AMB GRAU 3
X = np.array(data_regression_Ba.drop(['Bayley'],1))
y = np.array(data_regression_Ba['Bayley'])
```

In [43]:
```
pf = PolynomialFeatures(degree = 3)
X = pf.fit_transform(X)
```

In [44]:
```
model_Ba = linear_model.LinearRegression()
```

In [45]:
```
model_Ba.fit(X,y)
y_pred = np.round(model_Ba.predict(X),0)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Quadrat mig
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Coeficient de determinació
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
        y   pred
0    74.0   74.0
1    77.0   77.0
2   110.0  110.0
3    87.0   87.0
4    93.0   93.0
5    83.0   83.0
6   103.0  103.0
7    93.0   93.0
8   115.0  115.0
9    97.0   97.0
10   83.0   83.0
11  111.0  111.0
12   88.0   88.0
13   98.0   98.0
14   97.0   97.0
15   93.0   93.0
16  103.0  103.0
17   66.0   66.0
18  101.0  101.0
19   71.0   71.0
20   60.0   60.0
21  112.0  112.0
22   90.0   90.0
23  105.0  105.0
24  101.0  101.0
25   90.0   90.0
26   97.0   97.0
27   96.0   96.0
28   72.0   72.0
29   77.0   77.0
30  100.0  100.0
31  110.0  110.0
32   71.0   71.0
33   92.0   92.0
34  102.0  102.0
35   87.0   87.0
36   74.0   74.0
37   92.0   92.0
38   68.0   68.0
39   63.0   63.0
40   87.0   87.0
Mean squared error: 0.0000
R2 (coeficient de determinacio): 1.0000
```

In [46]:
```
resultats_CV (model_Ba, X, y, cv=10)
```

```
Model Mean squared error és: -2818.49 amb una desviació de +/- 2343.44
Model R2 (Coef det.) és: -32.16 amb una desviació de +/- 39.80
```

In [47]:
```
# REPETICIO AMB GRAU 2
X = np.array(data_regression_Ba.drop(['Bayley'],1))
y = np.array(data_regression_Ba['Bayley'])
```

In [48]:
```
pf = PolynomialFeatures(degree = 2)
X = pf.fit_transform(X)
```

In [49]:
```
model_Ba = linear_model.LinearRegression()
```

In [50]:
```
model_Ba.fit(X,y)
y_pred = np.round(model_Ba.predict(X),0)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Quadrat Mig
```

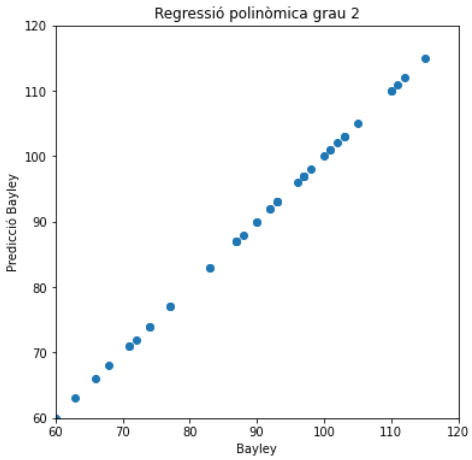
```
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Coeficient de determinació
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
      y   pred
0    74.0   74.0
1    77.0   77.0
2   110.0  110.0
3    87.0   87.0
4    93.0   93.0
5    83.0   83.0
6   103.0  103.0
7    93.0   93.0
8   115.0  115.0
9    97.0   97.0
10   83.0   83.0
11  111.0  111.0
12   88.0   88.0
13   98.0   98.0
14   97.0   97.0
15   93.0   93.0
16  103.0  103.0
17   66.0   66.0
18  101.0  101.0
19   71.0   71.0
20   60.0   60.0
21  112.0  112.0
22   90.0   90.0
23  105.0  105.0
24  101.0  101.0
25   90.0   90.0
26   97.0   97.0
27   96.0   96.0
28   72.0   72.0
29   77.0   77.0
30  100.0  100.0
31  110.0  110.0
32   71.0   71.0
33   92.0   92.0
34  102.0  102.0
35   87.0   87.0
36   74.0   74.0
37   92.0   92.0
38   68.0   68.0
39   63.0   63.0
40   87.0   87.0
Mean squared error: 0.0000
R2 (coeficient de determinacio): 1.0000
```

In [51]:
```
resultats_CV (model_Ba, X, y, cv=10)
```

```
Model Mean squared error és: -2856.52 amb una desviació de +/- 1823.69
Model R2 (Coef det.) és: -41.81 amb una desviació de +/- 66.32
```

In [52]:
```
# representació gràfica dels resultats del grau 2
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y,model_Ba.predict(X))
ax.set_xlim(60, 120)
ax.set_ylim(60, 120)
plt.xlabel('Bayley')
plt.ylabel('Predicció Bayley')
plt.title('Regressió polinòmica grau 2')
plt.show()
```



3.6 AMB CONJUNT TRAIN I TEST

In [53]:
```
# repetició de l'anàlisi del grau 2 amb un conjunt d'entrenament i test
```

In [54]:
```
X = np.array(data_regression_Ba.drop(['Bayley'],1))
y = np.array(data_regression_Ba['Bayley'])
```

In [55]:
```
pf = PolynomialFeatures(degree = 2)    # usarem grau 2
X = pf.fit_transform(X)
```

In [56]:
```
# el valor del test_size es determina per testeig
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.5, random_state=13)
#print("X_train:", X_train.shape)
#print("X_test:", X_test.shape)
```

In [57]:
```
model_Ba = linear_model.LinearRegression()
```

In [58]:
```
model_Ba.fit(X_train,y_train)
y_pred = np.round(model_Ba.predict(X),2)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Cuadrado Medio
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
        y    pred
0    74.0   74.00
1    77.0  126.55
2   110.0  110.00
3    87.0   68.22
4    93.0   74.29
5    83.0   72.56
6   103.0  118.36
7    93.0  105.13
8   115.0   83.11
9    97.0   97.00
10   83.0   83.00
11  111.0  111.00
12   88.0   99.49
13   98.0   98.00
14   97.0  101.71
15   93.0   70.27
16  103.0  103.00
17   66.0  100.10
18  101.0  101.00
19   71.0   71.00
20   60.0   60.00
21  112.0  112.00
22   90.0   62.48
23  105.0   72.11
24  101.0  101.00
25   90.0   90.00
26   97.0   97.00
27   96.0   99.92
28   72.0   88.19
29   77.0   77.00
30  100.0   75.88
31  110.0   51.69
32   71.0   37.80
33   92.0   92.00
34  102.0  102.00
35   87.0   87.00
36   74.0   74.00
37   92.0   67.48
38   68.0  116.13
39   63.0   63.00
40   87.0   75.08
Mean squared error: 408.8079
R2 (coeficient de determinacio): -0.9966
```

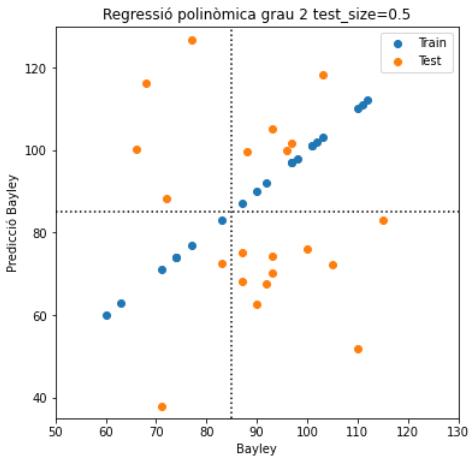
In [59]:
```
resultats_CV (model_Ba, X, y, cv=5)
```

```
Model Mean squared error és: -2003.93 amb una desviació de +/- 1077.98
Model R2 (Coef det.) és: -10.31 amb una desviació de +/- 3.16
```

In [60]:
```
resultats_CV (model_Ba, X_train,y_train, cv=5)
```

```
Model Mean squared error és: -623.58 amb una desviació de +/- 430.75
Model R2 (Coef det.) és: -4.76 amb una desviació de +/- 5.93
```

In [61]:
```
# representació gràfica amb els limits de Bayley=85
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y_train,np.round(model_Ba.predict(X_train),2))
plt.scatter(y_test,np.round(model_Ba.predict(X_test),2))
ax.set_xlim(50, 130)
ax.set_ylim(35, 130)
ax.legend(['Train', 'Test'])
plt.xlabel('Bayley')
plt.ylabel('Predicció Bayley')
plt.axvline(x=85, color="black", linestyle=":")
plt.axhline(y=85, color="black", linestyle=":")
plt.title('Regressió polinòmica grau 2 test_size=0.5')
plt.show()
```



ANNEX 5: CODI PYTHON: MODEL DE REGRESSIÓ BRAZELTON

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler, PolynomialFeatures

from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
# recuperació del joc de dades
data = pd.read_pickle("./data.pkl")
```

4. MODEL DE REGRESSIÓ BRAZELTON

In [3]:
```
# creació de joc de dades específic
data_regression_Br = data.drop(['ID', 'Sexe', 'Diagnostic', 'Vineland', 'Bayley'],1) #no voldria treure Vineland i Bayley, pero hi ha massa
data_regression_Br = data_regression_Br.dropna()

data_regression_Br.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62 entries, 0 to 78
Data columns (total 35 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   LA              62 non-null     float64
 1   RA              62 non-null     float64
 2   Brazelton       62 non-null     float64
 3   A_BPD           62 non-null     float64
 4   A_OFD           62 non-null     float64
 5   A_LID           62 non-null     float64
 6   A_RID           62 non-null     float64
 7   A_LSFD          62 non-null     float64
 8   A_RSFD          62 non-null     float64
 9   A_LIDSF         62 non-null     float64
 10  A_RIDSF         62 non-null     float64
 11  A_LMDSF         62 non-null     float64
 12  A_RMDSF         62 non-null     float64
 13  A_LODSF         62 non-null     float64
 14  A_RODSF         62 non-null     float64
 15  A_LPOFD         62 non-null     float64
 16  A_RPOFD         62 non-null     float64
 17  A_LOPOFD        62 non-null     float64
 18  A_ROPOFD        62 non-null     float64
 19  A_LPOVD         62 non-null     float64
 20  A_RPOVD         62 non-null     float64
 21  C_LCIFD         62 non-null     float64
 22  C_RCIFD         62 non-null     float64
 23  C_LCAFD         62 non-null     float64
 24  C_RCAFD         62 non-null     float64
 25  C_LCAVD         62 non-null     float64
 26  C_RCAVD         62 non-null     float64
 27  TCD             62 non-null     float64
 28  PF              62 non-null     float64
 29  CCL             62 non-null     float64
 30  VCCD            62 non-null     float64
 31  LAH             62 non-null     float64
 32  RAH             62 non-null     float64
 33  Sexe_cod        62 non-null     object
 34  Diagnostic_cod  62 non-null     object
dtypes: float64(33), object(2)
memory usage: 17.4+ KB
```

In [4]:
```
#pd.set_option('display.max_rows', None)
#display(data_regression_Br)
#pd.set_option('display.max_rows', 10)
```

4.1 VALORS SENSE NORMALITZAR

In [5]:
```
# separació de paràmetres i atribut objectiu
X = np.array(data_regression_Br.drop(['Brazelton'],1))
y = np.array(data_regression_Br['Brazelton'])
```

In [6]:
```
X.shape
```

Out[6]:
```
(62, 34)
```

In [7]:
```
# model de regressió linial
model_Br = linear_model.LinearRegression()
```

In [8]:
```
# entrenament i avaluació de resultats
model_Br.fit(X,y)
y_pred = model_Br.predict(X)
```

```
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Quadrat Mig
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Coeficient de determinació
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
            y       pred
0   43.714286  44.179426
1   45.714286  45.483251
2   49.857143  49.817201
3   47.714286  47.073650
4   44.714286  42.982346
..        ...        ...
57  47.285714  43.530147
58  47.142857  45.795625
59  44.142857  44.712133
60  46.571429  46.647080
61  44.714286  42.888843

[62 rows x 2 columns]
Mean squared error: 2.7996
R2 (coeficient de determinacio): 0.6047
```

In [9]:
```
## FUNCIO PER AVALUAR MODELS
def resultats_CV (model, x, y, cv):
    scoring = ['r2', 'neg_mean_squared_error']
    resultat = model_selection.cross_validate(model, x, y, scoring=scoring, cv=cv)
    print("Model Mean squared error és: {:0.2f} amb una desviació de +/- {:0.2f}".
          format(resultat['test_neg_mean_squared_error'].mean(), resultat['test_neg_mean_squared_error'].std()))
    print("Model R2 (Coef det.) és: {:0.2f} amb una desviació de +/- {:0.2f}".
          format(resultat['test_r2'].mean(), resultat['test_r2'].std()))
```

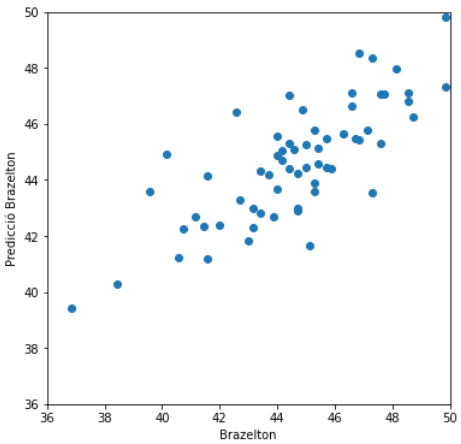
In [10]:
```
resultats_CV (model_Br, X, y, cv=9)
```

```
Model Mean squared error és: -21.84 amb una desviació de +/- 9.93
Model R2 (Coef det.) és: -6.42 amb una desviació de +/- 6.31
```

In [11]:
```
# prediccions
data_regression_Br['Brazelton_pred']= model_Br.predict(X)
```

In [12]:
```
# llistat de prediccions
#pd.set_option('display.max_rows', None)
#display(data_regression_Br[['Brazelton','Brazelton_pred']])
#pd.set_option('display.max_rows', 10)
```

In [13]:
```
# visualització de l'ajust
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(data_regression_Br['Brazelton'],data_regression_Br['Brazelton_pred'])
ax.set_xlim(36, 50)
ax.set_ylim(36, 50)
plt.xlabel('Brazelton')
plt.ylabel('Predicció Brazelton')
plt.show()
```



4.2 VALORS SENSE NORMALITZAR I AMB CONJUNT TRAIN-TEST

In [14]:
```
# loop per a determinar l'efecte de la mida del conjunt de test

for i in np.arange(0.05, 0.50, 0.05):
    print("****Valor de test_size", np.round(i,3))
    X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = i , random_state = 17)
    model_Br.fit(X_train,y_train)
    # prediccions sobre train per avaluar el model
    y_pred = model_Br.predict(X_train)
    # Error Cuadrado Medio
    print("Train Mean squared error: %.4f" % mean_squared_error(y_train, y_pred))
    # Puntaje de Varianza. El mejor puntaje es un 1.0
    print('Train R2 (coeficient de determinacio): %.4f' % r2_score(y_train, y_pred))
    # uso cross_val_score per avaluar el model
    resultats_CV (model_Br, X_train, y_train, cv=5)

    y_pred_test = model_Br.predict(X_test)
    prediccio = pd.DataFrame({"y": y_test, "pred": y_pred_test})
    print(prediccio)
    # Error Cuadrado Medio
    print("Test Mean squared error: %.4f" % mean_squared_error(y_test, y_pred_test))
```

```
    # Puntaje de Varianza. El mejor puntaje es un 1.0
    print('Test R2 (coeficient de determinacio): %.4f' % r2_score(y_test, y_pred_test))
    print('\n\n')
```

```
****Valor de test_size 0.05
Train Mean squared error: 2.0688
Train R2 (coeficient de determinacio): 0.6947
Model Mean squared error és: -36.49 amb una desviació de +/- 22.29
Model R2 (Coef det.) és: -7.89 amb una desviació de +/- 6.15
          y       pred
0  39.571429  48.685670
1  44.857143  46.768269
2  46.571429  49.673112
3  48.714286  43.179535
Test Mean squared error: 31.7439
Test R2 (coeficient de determinacio): -1.7764



****Valor de test_size 0.1
Train Mean squared error: 1.8770
Train R2 (coeficient de determinacio): 0.7306
Model Mean squared error és: -33.01 amb una desviació de +/- 15.93
Model R2 (Coef det.) és: -4.77 amb una desviació de +/- 2.14
          y       pred
0  39.571429  47.977734
1  44.857143  47.571617
2  46.571429  48.658755
3  48.714286  43.614319
4  44.142857  48.767258
5  44.714286  45.441948
6  41.571429  47.658937
Test Mean squared error: 23.9105
Test R2 (coeficient de determinacio): -2.0456



****Valor de test_size 0.15
Train Mean squared error: 1.8738
Train R2 (coeficient de determinacio): 0.7365
Model Mean squared error és: -58.88 amb una desviació de +/- 35.50
Model R2 (Coef det.) és: -10.84 amb una desviació de +/- 9.23
          y       pred
0  39.571429  47.634856
1  44.857143  47.303965
2  46.571429  48.892312
3  48.714286  44.533277
4  44.142857  48.926034
5  44.714286  45.981054
6  41.571429  49.313413
7  45.428571  45.459309
8  45.285714  45.850871
9  41.142857  45.464336
Test Mean squared error: 19.7290
Test R2 (coeficient de determinacio): -1.9416



****Valor de test_size 0.2
Train Mean squared error: 1.6930
Train R2 (coeficient de determinacio): 0.7746
Model Mean squared error és: -90.52 amb una desviació de +/- 31.88
Model R2 (Coef det.) és: -14.02 amb una desviació de +/- 3.14
           y       pred
0   39.571429  44.980724
1   44.857143  46.379445
2   46.571429  48.708151
3   48.714286  45.858612
4   44.142857  48.586187
5   44.714286  45.885113
6   41.571429  49.197765
7   45.428571  44.077941
8   45.285714  44.050005
9   41.142857  45.691505
10  43.714286  46.457699
11  45.142857  37.935779
12  45.428571  45.288213
Test Mean squared error: 15.9310
Test R2 (coeficient de determinacio): -1.9805



****Valor de test_size 0.25
Train Mean squared error: 1.4683
Train R2 (coeficient de determinacio): 0.8158
Model Mean squared error és: -913.42 amb una desviació de +/- 1144.00
Model R2 (Coef det.) és: -152.42 amb una desviació de +/- 180.02
           y       pred
0   39.571429  44.288099
1   44.857143  44.328977
2   46.571429  44.977516
3   48.714286  44.108172
4   44.142857  50.578902
5   44.714286  46.571661
6   41.571429  51.321858
7   45.428571  43.599482
8   45.285714  43.444739
9   41.142857  43.639643
10  43.714286  43.437718
11  45.142857  35.784136
12  45.428571  45.849763
13  45.857143  43.848905
14  44.714286  40.955894
15  45.000000  35.277824
Test Mean squared error: 24.9821
Test R2 (coeficient de determinacio): -4.5517



****Valor de test_size 0.3
Train Mean squared error: 1.4282
Train R2 (coeficient de determinacio): 0.7956
Model Mean squared error és: -1304.95 amb una desviació de +/- 1815.57
Model R2 (Coef det.) és: -718.34 amb una desviació de +/- 1281.70
           y       pred
```

```
0   39.571429  45.754129
1   44.857143  43.998421
2   46.571429  42.573910
3   48.714286  43.348712
4   44.142857  46.008001
5   44.714286  45.016996
6   41.571429  48.961814
7   45.428571  44.954014
8   45.285714  44.226514
9   41.142857  41.496268
10  43.714286  40.372878
11  45.142857  36.855675
12  45.428571  44.706091
13  45.857143  42.363951
14  44.714286  39.001360
15  45.000000  38.296482
16  43.142857  40.719402
17  36.857143  40.612834
18  44.142857  46.214173
Test Mean squared error: 17.7791
Test R2 (coeficient de determinacio): -1.6467



****Valor de test_size 0.35
Train Mean squared error: 1.4084
Train R2 (coeficient de determinacio): 0.8060
Model Mean squared error és: -169.06 amb una desviació de +/- 73.74
Model R2 (Coef det.) és: -57.03 amb una desviació de +/- 72.51
            y       pred
0   39.571429  46.489724
1   44.857143  43.963241
2   46.571429  43.485632
3   48.714286  41.952492
4   44.142857  46.896548
5   44.714286  45.824292
6   41.571429  49.757104
7   45.428571  46.602408
8   45.285714  45.707419
9   41.142857  41.894164
10  43.714286  41.387160
11  45.142857  37.220894
12  45.428571  44.774642
13  45.857143  40.830794
14  44.714286  38.673353
15  45.000000  36.130092
16  43.142857  41.087585
17  36.857143  40.609826
18  44.142857  48.258450
19  44.000000  44.983402
20  44.428571  42.224589
21  42.000000  38.102642
Test Mean squared error: 20.3245
Test R2 (coeficient de determinacio): -2.3950



****Valor de test_size 0.4
Train Mean squared error: 0.7824
Train R2 (coeficient de determinacio): 0.8845
Model Mean squared error és: -70.02 amb una desviació de +/- 25.05
Model R2 (Coef det.) és: -16.29 amb una desviació de +/- 8.17
            y       pred
0   39.571429  47.334456
1   44.857143  47.626240
2   46.571429  58.322754
3   48.714286  46.666036
4   44.142857  53.310728
5   44.714286  45.357552
6   41.571429  62.406179
7   45.428571  36.585226
8   45.285714  42.508126
9   41.142857  51.768210
10  43.714286  59.433678
11  45.142857  35.872980
12  45.428571  54.559053
13  45.857143  49.298516
14  44.714286  44.829982
15  45.000000  39.850948
16  43.142857  45.648620
17  36.857143  30.496702
18  44.142857  51.905286
19  44.000000  51.727027
20  44.428571  46.989646
21  42.000000  39.857747
22  41.428571  20.657579
23  46.285714  44.032393
24  40.142857  44.842333
Test Mean squared error: 80.7518
Test R2 (coeficient de determinacio): -11.8402



****Valor de test_size 0.45
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -69.47 amb una desviació de +/- 33.90
Model R2 (Coef det.) és: -14.54 amb una desviació de +/- 6.68
            y       pred
0   39.571429  50.019994
1   44.857143  49.563076
2   46.571429  45.636912
3   48.714286  34.497214
4   44.142857  50.577395
5   44.714286  61.771201
6   41.571429  60.016703
7   45.428571  43.957200
8   45.285714  43.014008
9   41.142857  39.338089
10  43.714286  33.067788
11  45.142857  41.541044
12  45.428571  49.411313
13  45.857143  42.079966
14  44.714286  40.328945
15  45.000000  26.520273
16  43.142857  44.926213
```

```
17  36.857143  44.953237
18  44.142857  45.744278
19  44.000000  56.185698
20  44.428571  34.261451
21  42.000000  41.565714
22  41.428571  37.878259
23  46.285714  41.982296
24  40.142857  40.409536
25  43.000000  37.053754
26  45.285714  61.357648
27  47.571429  34.990619
Test Mean squared error: 83.6118
Test R2 (coeficient de determinacio): -12.4690
```

In [15]:
```
# selecció d'un valor i separació dels conjunts d'entrenament i test
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = 0.30, random_state = 17)
```

In [16]:
```
model_Br.fit(X_train,y_train)
```

Out[16]:
```
LinearRegression()
```

In [17]:
```
# prediccions sobre train per avaluar el model
y_pred = model_Br.predict(X_train)
prediccio = pd.DataFrame({"y": y_train, "pred": y_pred})
print(prediccio)

# Error Cuadrado Medio
print("Train Mean squared error: %.4f" % mean_squared_error(y_train, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Train R2 (coeficient de determinacio): %.4f' % r2_score(y_train, y_pred))
```

```
            y       pred
0   44.000000  44.130452
1   44.428571  43.963441
2   42.000000  40.993284
3   41.428571  40.806022
4   46.285714  44.232594
5   40.142857  42.894644
6   43.000000  42.155765
7   45.285714  46.859997
8   47.571429  46.573572
9   46.714286  46.271790
10  44.428571  45.665853
11  48.142857  47.897508
12  48.571429  47.581198
13  49.857143  49.717455
14  47.571429  45.630987
15  44.428571  44.828544
16  46.571429  47.827495
17  43.142857  42.939083
18  48.571429  46.919324
19  45.285714  44.843892
20  46.857143  46.915928
21  47.142857  45.695890
22  47.714286  47.123892
23  43.428571  42.348868
24  45.714286  43.917353
25  40.714286  43.117250
26  49.857143  49.431937
27  44.000000  45.353341
28  45.714286  45.967011
29  43.857143  45.108050
30  42.571429  44.668969
31  46.857143  46.700503
32  45.000000  44.206936
33  44.714286  44.486306
34  44.000000  44.527487
35  47.285714  45.462168
36  43.428571  44.378724
37  44.571429  46.021829
38  40.571429  40.855380
39  41.571429  41.856731
40  38.428571  39.471444
41  47.285714  48.996620
42  42.714286  42.083052
Train Mean squared error: 1.4282
Train R2 (coeficient de determinacio): 0.7956
```

In [18]:
```
# prediccions sobre test
y_pred_test = model_Br.predict(X_test)

# Error Quadrat Mig
print("Test Mean squared error: %.4f" % mean_squared_error(y_test, y_pred_test))
# Coeficient de determinació
print('Test R2 (coeficient de determinacio): %.4f' % r2_score(y_test, y_pred_test))
```

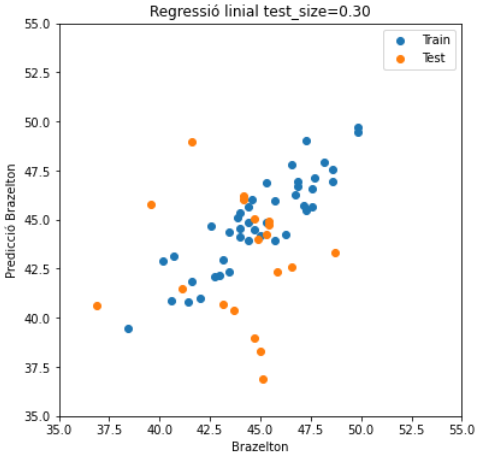
```
Test Mean squared error: 17.7791
Test R2 (coeficient de determinacio): -1.6467
```

In [19]:
```
# uso la funció pròpia per avaluar el model
resultats_CV (model_Br, X_train, y_train, cv=5)
```

```
Model Mean squared error és: -1304.95 amb una desviació de +/- 1815.57
Model R2 (Coef det.) és: -718.34 amb una desviació de +/- 1281.70
```

In [20]:
```
# visualització de l'ajust
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y_train,np.round(model_Br.predict(X_train),2))
plt.scatter(y_test,np.round(model_Br.predict(X_test),2))
ax.set_xlim(35, 55)
ax.set_ylim(35, 55)
ax.legend(['Train', 'Test'])
plt.xlabel('Brazelton')
plt.ylabel('Predicció Brazelton')
```

```
plt.title('Regressió linial test_size=0.30')
plt.show()
```



In [21]:
```
# generació de prediccions
data_regression_Br['Brazelton_pred_TT']= model_Br.predict(X)
```

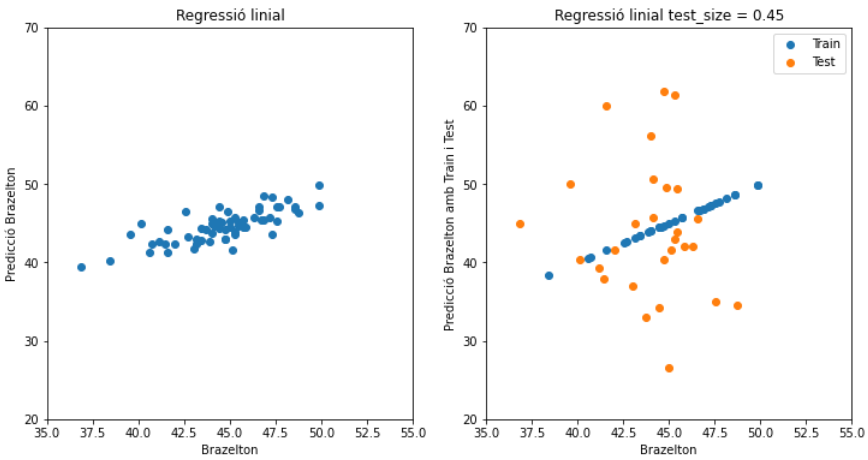
In [22]:
```
# llistat de prediccions
#pd.set_option('display.max_rows', None)
#display(data_regression_Br[['Brazelton','Brazelton_pred', 'Brazelton_pred_TT']])
#pd.set_option('display.max_rows', 10)
```

In [23]:
```
# selecció d'una mida de conjunt de test
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = 0.45, random_state = 17)
```

In [24]:
```
model_Br.fit(X_train,y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
# comparativa del model sense conjunt de test i amb un tamany de test que provoca sobreentrenament
fig, ax = plt.subplots(1,2,figsize=(12, 6))
ax[0].scatter(data_regression_Br['Brazelton'],data_regression_Br['Brazelton_pred'])
ax[0].set_xlim(35, 55)
ax[0].set_ylim(20, 70)
ax[0].set_xlabel('Brazelton')
ax[0].set_ylabel('Predicció Brazelton')
ax[0].set_title('Regressió linial')
ax[1].scatter(y_train,np.round(model_Br.predict(X_train),2))
ax[1].scatter(y_test,np.round(model_Br.predict(X_test),2))
ax[1].set_xlim(35, 55)
ax[1].set_ylim(20, 70)
ax[1].set_xlabel('Brazelton')
ax[1].legend(['Train', 'Test'])
ax[1].set_ylabel('Predicció Brazelton amb Train i Test')
ax[1].set_title('Regressió linial test_size = 0.45')
plt.show()
```



4.3 VALORS NORMALITZATS

In [26]:
```
# faig una copia del df treient les prediccions ja efectuades
data_regression_Br_norm =data_regression_Br.copy()
data_regression_Br_norm= data_regression_Br_norm.drop(['Brazelton_pred', 'Brazelton_pred_TT'],1)
```

In [27]:
```
# normalització
columns_Br_norm = data_regression_Br_norm.columns
scaler_Br = StandardScaler()
scaler_Br.fit(data_regression_Br_norm)
data_regression_Br_norm = scaler_Br.transform(data_regression_Br_norm)

# torno a muntar el df
```

```
data_regression_Br_norm = pd.DataFrame(data_regression_Br_norm)
data_regression_Br_norm.columns = columns_Br_norm
```

In [28]:
```
# conjunt de dades i d'atribut objectiu
X = np.array(data_regression_Br_norm.drop(['Brazelton'],1))
y = np.array(data_regression_Br_norm['Brazelton'])
```

In [29]:
```
X.shape
```

Out[29]: (62, 34)

In [30]:
```
model_Br_norm = linear_model.LinearRegression()
```

In [31]:
```
#entrenament
model_Br_norm.fit(X,y)
y_pred = model_Br_norm.predict(X)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)
# Error Quadrat Mig
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Coeficient de determinació
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
           y      pred
0  -0.346330 -0.171545
1   0.405206  0.318390
2   1.961958  1.946950
3   1.156742  0.916011
4   0.029438 -0.621369
..       ...       ...
57  0.995698 -0.415523
58  0.942017  0.435771
59 -0.185286  0.028629
60  0.727293  0.755720
61  0.029438 -0.656505

[62 rows x 2 columns]
Mean squared error: 0.3953
R2 (coeficient de determinacio): 0.6047
```

In [32]:
```
# avaluació amb funció pròpia
resultats_CV (model_Br_norm, X, y, cv=5)
```

```
Model Mean squared error és: -3.43 amb una desviació de +/- 0.93
Model R2 (Coef det.) és: -5.28 amb una desviació de +/- 5.12
```

In [33]:
```
# prediccions
data_regression_Br_norm['Brazelton_pred_norm']= model_Br_norm.predict(X)
```

In [34]:
```
# llistat de prediccions
#data_regression_Br_norm
```

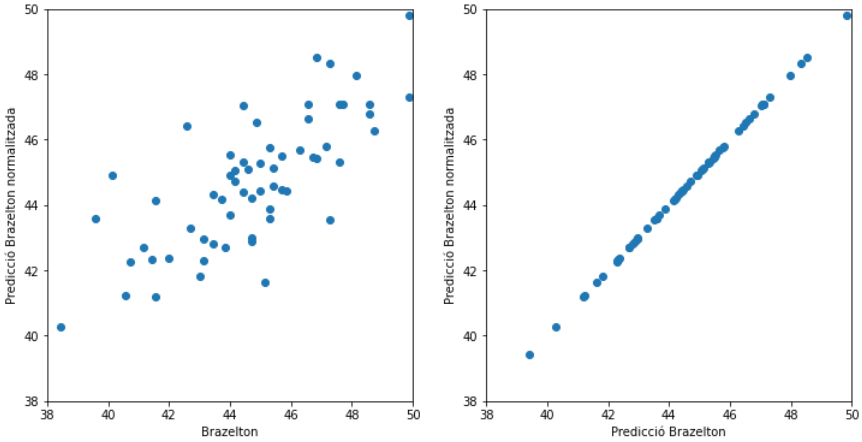
In [35]:
```
# procés de desnormalització
Des_scaler_Br = StandardScaler()
```

In [36]:
```
Des_scaler_Br.scale_,Des_scaler_Br.mean_,Des_scaler_Br.var_,Des_scaler_Br.n_samples_seen_= scaler_Br.scale_[2],scaler_Br.mean_[2],scaler_Br
```

In [37]:
```
data_regression_Br_norm['Brazelton_pred_norm'] = Des_scaler_Br.inverse_transform(data_regression_Br_norm['Brazelton_pred_norm'])
data_regression_Br_norm['Brazelton_desnorm'] = Des_scaler_Br.inverse_transform(data_regression_Br_norm['Brazelton'])
```

In [38]:
```
# llistat de prediccions desnormalitzades
#pd.set_option('display.max_rows', None)
#display(data_regression_Br_norm[['Brazelton_pred_norm','Brazelton_desnorm']])
#pd.set_option('display.max_rows', 10)
```

In [39]:
```
# visualització de les prediccions normalitzades i la relació amb la prediccio sense normalitzar
fig, ax = plt.subplots(1,2,figsize=(12, 6))
ax[0].scatter(data_regression_Br['Brazelton'],data_regression_Br_norm['Brazelton_pred_norm'])
ax[0].set_xlim(38, 50)
ax[0].set_ylim(38, 50)
ax[0].set_xlabel('Brazelton')
ax[0].set_ylabel('Predicció Brazelton normalitzada')
ax[1].scatter(data_regression_Br['Brazelton_pred'],data_regression_Br_norm['Brazelton_pred_norm'])
ax[1].set_xlim(38, 50)
ax[1].set_ylim(38, 50)
ax[1].set_xlabel('Predicció Brazelton')
ax[1].set_ylabel('Predicció Brazelton normalitzada')
plt.show()
```



4.4 VALORS NORMALITZATS I AMB CONJUNT TEST

In [40]:

```
#loop per determinar la mida del conjunt de test

for i in np.arange(0.05, 0.50, 0.05):
    print("****Valor de test_size", np.round(i,3))
    X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = i , random_state = 17)
    model_Br_norm.fit(X_train,y_train)
    # prediccions sobre train per avaluar el model
    y_pred = model_Br_norm.predict(X_train)
    # Error Cuadrado Medio
    print("Train Mean squared error: %.4f" % mean_squared_error(y_train, y_pred))
    # Puntaje de Varianza. El mejor puntaje es un 1.0
    print('Train R2 (coeficient de determinacio): %.4f' % r2_score(y_train, y_pred))
    # uso cross_val_score per avaluar el model
    resultats_CV (model_Br_norm, X_train, y_train, cv=5)

    y_pred_test = model_Br_norm.predict(X_test)
    prediccio = pd.DataFrame({"y": y_test, "pred": y_pred_test})
    print(prediccio)
    # Error Cuadrado Medio
    print("Test Mean squared error: %.4f" % mean_squared_error(y_test, y_pred_test))
    # Puntaje de Varianza. El mejor puntaje es un 1.0
    print('Test R2 (coeficient de determinacio): %.4f' % r2_score(y_test, y_pred_test))
    print('\n\n')
```

```
****Valor de test_size 0.05
Train Mean squared error: 0.2921
Train R2 (coeficient de determinacio): 0.6947
Model Mean squared error és: -5.15 amb una desviació de +/- 3.15
Model R2 (Coef det.) és: -7.89 amb una desviació de +/- 6.15
          y      pred
0 -1.903082  1.521756
1  0.083119  0.801259
2  0.727293  1.892805
3  1.532509 -0.547272
Test Mean squared error: 4.4823
Test R2 (coeficient de determinacio): -1.7764



****Valor de test_size 0.1
Train Mean squared error: 0.2650
Train R2 (coeficient de determinacio): 0.7306
Model Mean squared error és: -4.66 amb una desviació de +/- 2.25
Model R2 (Coef det.) és: -4.77 amb una desviació de +/- 2.14
          y      pred
0 -1.903082  1.255737
1  0.083119  1.103131
2  0.727293  1.511643
3  1.532509 -0.383894
4 -0.185286  1.552415
5  0.029438  0.302870
6 -1.151547  1.135943
Test Mean squared error: 3.3762
Test R2 (coeficient de determinacio): -2.0456



****Valor de test_size 0.15
Train Mean squared error: 0.2646
Train R2 (coeficient de determinacio): 0.7365
Model Mean squared error és: -8.31 amb una desviació de +/- 5.01
Model R2 (Coef det.) és: -10.84 amb una desviació de +/- 9.23
          y      pred
0 -1.903082  1.126895
1  0.083119  1.002556
2  0.727293  1.599406
3  1.532509 -0.038579
4 -0.185286  1.612078
5  0.029438  0.505449
6 -1.151547  1.757642
7  0.297844  0.309394
8  0.244163  0.456530
9 -1.312590  0.311283
Test Mean squared error: 2.7858
Test R2 (coeficient de determinacio): -1.9416



****Valor de test_size 0.2
Train Mean squared error: 0.2391
Train R2 (coeficient de determinacio): 0.7746
Model Mean squared error és: -12.78 amb una desviació de +/- 4.50
Model R2 (Coef det.) és: -14.02 amb una desviació de +/- 3.14
          y      pred
0  -1.903082  0.129557
```

```
1   0.083119  0.655151
2   0.727293  1.530204
3   1.532509  0.459439
4  -0.185286  1.484374
5   0.029438  0.469397
6  -1.151547  1.714186
7   0.297844 -0.209680
8   0.244163 -0.220177
9  -1.312590  0.396645
10 -0.346330  0.684557
11  0.190481 -2.517707
12  0.297844  0.245101
Test Mean squared error: 2.2495
Test R2 (coeficient de determinacio): -1.9805



****Valor de test_size 0.25
Train Mean squared error: 0.2073
Train R2 (coeficient de determinacio): 0.8158
Model Mean squared error és: -128.98 amb una desviació de +/- 161.53
Model R2 (Coef det.) és: -152.42 amb una desviació de +/- 180.02
           y      pred
0  -1.903082 -0.130709
1   0.083119 -0.115349
2   0.727293  0.128352
3   1.532509 -0.198320
4  -0.185286  2.233172
5   0.029438  0.727380
6  -1.151547  2.512351
7   0.297844 -0.389469
8   0.244163 -0.447617
9  -1.312590 -0.374378
10 -0.346330 -0.450255
11  0.190481 -3.326225
12  0.297844  0.456114
13  0.458887 -0.295744
14  0.029438 -1.382845
15  0.136800 -3.516481
Test Mean squared error: 3.5275
Test R2 (coeficient de determinacio): -4.5517



****Valor de test_size 0.3
Train Mean squared error: 0.2017
Train R2 (coeficient de determinacio): 0.7956
Model Mean squared error és: -182.73 amb una desviació de +/- 256.72
Model R2 (Coef det.) és: -716.01 amb una desviació de +/- 1282.63
           y      pred
0  -1.903082  0.420178
1   0.083119 -0.239561
2   0.727293 -0.774846
3   1.532509 -0.483701
4  -0.185286  0.515575
5   0.029438  0.143187
6  -1.151547  1.625523
7   0.297844  0.119520
8   0.244163 -0.153851
9  -1.312590 -1.179790
10 -0.346330 -1.601923
11  0.190481 -2.923575
12  0.297844  0.026359
13  0.458887 -0.853742
14  0.029438 -2.117296
15  0.136800 -2.382166
16 -0.561054 -1.471711
17 -2.923024 -1.511756
18 -0.185286  0.593047
Test Mean squared error: 2.5104
Test R2 (coeficient de determinacio): -1.6467



****Valor de test_size 0.35
Train Mean squared error: 0.1989
Train R2 (coeficient de determinacio): 0.8060
Model Mean squared error és: -14.21 amb una desviació de +/- 1.72
Model R2 (Coef det.) és: -26.39 amb una desviació de +/- 23.20
           y      pred
0  -1.903082  0.696591
1   0.083119 -0.252780
2   0.727293 -0.432250
3   1.532509 -1.008355
4  -0.185286  0.849462
5   0.029438  0.446543
6  -1.151547  1.924367
7   0.297844  0.738934
8   0.244163  0.402626
9  -1.312590 -1.030273
10 -0.346330 -1.220789
11  0.190481 -2.786338
12  0.297844  0.052118
13  0.458887 -1.429853
14  0.029438 -2.240550
15  0.136800 -3.196226
16 -0.561054 -1.333360
17 -2.923024 -1.512886
18 -0.185286  1.361221
19 -0.238968  0.130563
20 -0.077924 -0.906110
21 -0.990503 -2.455005
Test Mean squared error: 2.8698
Test R2 (coeficient de determinacio): -2.3950



****Valor de test_size 0.4
Train Mean squared error: 0.1105
Train R2 (coeficient de determinacio): 0.8845
Model Mean squared error és: -9.45 amb una desviació de +/- 3.76
Model R2 (Coef det.) és: -14.06 amb una desviació de +/- 5.05
           y      pred
0  -1.903082  1.014014
1   0.083119  1.123657
2   0.727293  5.143063
```

```
3   1.532509  0.762843
4  -0.185286  3.259705
5   0.029438  0.271157
6  -1.151547  6.677483
7   0.297844 -3.025201
8   0.244163 -0.799566
9  -1.312590  2.680076
10 -0.346330  5.560512
11  0.190481 -3.292840
12  0.297844  3.728785
13  0.458887  1.752045
14  0.029438  0.072913
15  0.136800 -1.798048
16 -0.561054  0.380531
17 -2.923024 -5.313073
18 -0.185286  2.731585
19 -0.238968  2.664601
20 -0.077924  0.884445
21 -0.990503 -1.795493
22 -1.205228 -9.010299
23  0.619930 -0.226795
24 -1.688358  0.077554
Test Mean squared error: 11.4023
Test R2 (coeficient de determinacio): -11.8402


****Valor de test_size 0.45
Train Mean squared error: 0.0000
Train R2 (coeficient de determinacio): 1.0000
Model Mean squared error és: -8.55 amb una desviació de +/- 3.49
Model R2 (Coef det.) és: -12.92 amb una desviació de +/- 5.03
           y      pred
0  -1.903082  1.992619
1   0.083119  1.828197
2   0.727293 -0.092301
3   1.532509 -4.049038
4  -0.185286  2.358770
5   0.029438  6.487486
6  -1.151547  5.475420
7   0.297844 -0.115179
8   0.244163 -0.623638
9  -1.312590 -2.379304
10 -0.346330 -4.982118
11  0.190481 -1.180542
12  0.297844  1.597204
13  0.458887 -1.140937
14  0.029438 -1.872618
15  0.136800 -6.991264
16 -0.561054 -0.002476
17 -2.923024  0.374234
18 -0.185286  0.301811
19 -0.238968  4.241897
20 -0.077924 -4.094135
21 -0.990503 -1.206478
22 -1.205228 -2.201102
23  0.619930 -1.124091
24 -1.688358 -1.764380
25 -0.614735 -3.100037
26  0.244163  6.299412
27  1.103060 -3.691167
Test Mean squared error: 12.2783
Test R2 (coeficient de determinacio): -13.0077
```

In [41]:
```
# generacio de conjunt d'entrenament i de test
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = 0.3, random_state = 17)
```

In [42]:
```
X_test.shape
```

Out[42]:
```
(19, 34)
```

In [43]:
```
model_Br_norm.fit(X_train,y_train)
```

Out[43]:
```
LinearRegression()
```

In [44]:
```
# prediccions sobre train per avaluar el model
y_pred = model_Br_norm.predict(X_train)
prediccio = pd.DataFrame({"y": y_train, "pred": y_pred})
print(prediccio)

# Error Quadrat Mig
print("Train Mean squared error: %.4f" % mean_squared_error(y_train, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Train R2 (coeficient de determinacio): %.4f' % r2_score(y_train, y_pred))
```

```
           y      pred
0  -0.238968 -0.189948
1  -0.077924 -0.252705
2  -0.990503 -1.368795
3  -1.205228 -1.439162
4   0.619930 -0.151566
5  -1.688358 -0.654325
6  -0.614735 -0.931972
7   0.244163  0.835727
8   1.103060  0.728098
9   0.780974  0.614698
10 -0.077924  0.387007
11  1.317785  1.225591
12  1.478828  1.106731
13  1.961958  1.909468
14  1.103060  0.373905
15 -0.077924  0.072373
16  0.727293  1.199282
17 -0.561054 -0.637626
18  1.478828  0.858021
19  0.244163  0.078140
```

```
20  0.834655  0.856744
21  0.942017  0.398293
22  1.156742  0.934891
23 -0.453692 -0.859410
24  0.405206 -0.270024
25 -1.473633 -0.570677
26  1.961958  1.802180
27 -0.238968  0.269575
28  0.405206  0.500172
29 -0.292649  0.177402
30 -0.775779  0.012410
31  0.834655  0.775795
32  0.136800 -0.161208
33  0.029438 -0.056229
34 -0.238968 -0.040755
35  0.995698  0.310468
36 -0.453692 -0.096655
37 -0.024243  0.520771
38 -1.527315 -1.420615
39 -1.151547 -1.044339
40 -2.332531 -1.940653
41  0.995698  1.638602
42 -0.722098 -0.959295
Train Mean squared error: 0.2017
Train R2 (coeficient de determinacio): 0.7956
```

In [45]:
```
# uso la funció pròpia per avaluar el model
resultats_CV (model_Br_norm, X_train, y_train, cv=5)
```

```
Model Mean squared error és: -182.73 amb una desviació de +/- 256.72
Model R2 (Coef det.) és: -716.01 amb una desviació de +/- 1282.63
```

In [46]:
```
# prediccions
y_pred_test = model_Br_norm.predict(X_test)

prediccio = pd.DataFrame({"y": y_test, "pred": y_pred_test})
print(prediccio)
```

```
           y      pred
0  -1.903082  0.420178
1   0.083119 -0.239561
2   0.727293 -0.774846
3   1.532509 -0.483701
4  -0.185286  0.515575
5   0.029438  0.143187
6  -1.151547  1.625523
7   0.297844  0.119520
8   0.244163 -0.153851
9  -1.312590 -1.179790
10 -0.346330 -1.601923
11  0.190481 -2.923575
12  0.297844  0.026359
13  0.458887 -0.853742
14  0.029438 -2.117296
15  0.136800 -2.382166
16 -0.561054 -1.471711
17 -2.923024 -1.511756
18 -0.185286  0.593047
```

In [47]:
```
# Error Quadrat Mig
print("Test Mean squared error: %.4f" % mean_squared_error(y_test, y_pred_test))
# coeficient de determinacio
print('Test R2 (coeficient de determinacio): %.4f' % r2_score(y_test, y_pred_test))
```

```
Test Mean squared error: 2.5104
Test R2 (coeficient de determinacio): -1.6467
```

In [48]:
```
#prediccions
data_regression_Br_norm['Brazelton_pred_norm_TT']= model_Br_norm.predict(X)
```

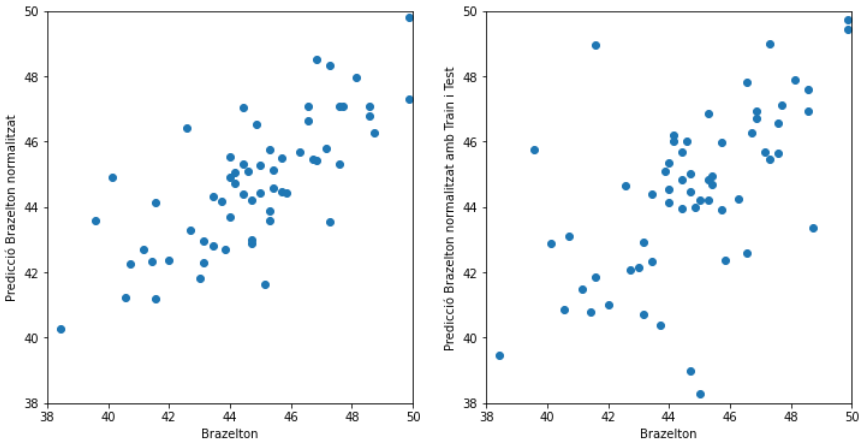
In [49]:
```
# desnormalitzar
Des_scaler_Br = StandardScaler()
```

In [50]:
```
Des_scaler_Br.scale_,Des_scaler_Br.mean_,Des_scaler_Br.var_,Des_scaler_Br.n_samples_seen_= scaler_Br.scale_[2],scaler_Br.mean_[2],scaler_Br
```

In [51]:
```
data_regression_Br_norm['Brazelton_pred_norm_TT'] = Des_scaler_Br.inverse_transform(data_regression_Br_norm['Brazelton_pred_norm_TT'])
```

In [52]:
```
# llistat de prediccions desnormalitzades
#pd.set_option('display.max_rows', None)
#display(data_regression_Br_norm[['Brazelton_pred_norm_TT','Brazelton_desnorm']])
#pd.set_option('display.max_rows', 10)
```

In [53]:
```
# visualització de les prediccions i de l'efecte de la creació de conjunt d'entrenament
fig, ax = plt.subplots(1,2,figsize=(12, 6))
ax[0].scatter(data_regression_Br['Brazelton'],data_regression_Br_norm['Brazelton_pred_norm'])
ax[0].set_xlim(38, 50)
ax[0].set_ylim(38, 50)
ax[0].set_xlabel('Brazelton')
ax[0].set_ylabel('Predicció Brazelton normalitzat')
ax[1].scatter(data_regression_Br['Brazelton'],data_regression_Br_norm['Brazelton_pred_norm_TT'])
ax[1].set_xlim(38, 50)
ax[1].set_ylim(38, 50)
ax[1].set_xlabel('Brazelton')
ax[1].set_ylabel('Predicció Brazelton normalitzat amb Train i Test')
plt.show()
```



4.5 REGRESSIÓ NO-LINIAL

In [54]:
```
# model de regressió polinomial
# dades i valor objectiu
X = np.array(data_regression_Br.drop(['Brazelton'],1))
y = np.array(data_regression_Br['Brazelton'])
```

In [55]:
```
#GRAU 4
```

In [56]:
```
pf = PolynomialFeatures(degree = 4)    # usarem desde grau 2 a 4
X = pf.fit_transform(X)  # transformació
```

In [57]:
```
model_Br = linear_model.LinearRegression()
```

In [58]:
```
#entrenament
model_Br.fit(X,y)
y_pred = np.round(model_Br.predict(X),0)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Quadrat Mig
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# coeficient de determinacio
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
            y  pred
0   43.714286  44.0
1   45.714286  46.0
2   49.857143  50.0
3   47.714286  48.0
4   44.714286  45.0
..        ...   ...
57  47.285714  47.0
58  47.142857  47.0
59  44.142857  44.0
60  46.571429  47.0
61  44.714286  45.0

[62 rows x 2 columns]
Mean squared error: 0.0902
R2 (coeficient de determinacio): 0.9873
```

In [59]:
```
resultats_CV (model_Br, X, y, cv=5)
```

```
Model Mean squared error és: -35.58 amb una desviació de +/- 13.66
Model R2 (Coef det.) és: -6.64 amb una desviació de +/- 3.78
```

In [60]:
```
#GRAU 3
```

In [61]:
```
# repetició de tots el passos
X = np.array(data_regression_Br.drop(['Brazelton'],1))
y = np.array(data_regression_Br['Brazelton'])
```

In [62]:
```
pf = PolynomialFeatures(degree = 3)
X = pf.fit_transform(X)
```

In [63]:
```
model_Br = linear_model.LinearRegression()
```

In [64]:
```
model_Br.fit(X,y)
y_pred = np.round(model_Br.predict(X),0)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Quadrat Mig
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# coeficient de determinacio
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
            y  pred
0   43.714286  44.0
1   45.714286  46.0
```

```
2   49.857143  50.0
3   47.714286  48.0
4   44.714286  45.0
..        ...   ...
57  47.285714  47.0
58  47.142857  47.0
59  44.142857  44.0
60  46.571429  47.0
61  44.714286  45.0

[62 rows x 2 columns]
Mean squared error: 0.0902
R2 (coeficient de determinacio): 0.9873
```

In [65]:
```
resultats_CV (model_Br, X, y, cv=5)
```

```
Model Mean squared error és: -32.79 amb una desviació de +/- 11.02
Model R2 (Coef det.) és: -6.16 amb una desviació de +/- 3.58
```

In [66]:
```
#GRAU 2
```

In [67]:
```
# repetició del procediment per al grau 2
X = np.array(data_regression_Br.drop(['Brazelton'],1))
y = np.array(data_regression_Br['Brazelton'])
```

In [68]:
```
pf = PolynomialFeatures(degree = 2)    # grau 2
X = pf.fit_transform(X)  # transformació
```

In [69]:
```
model_Br = linear_model.LinearRegression()
```

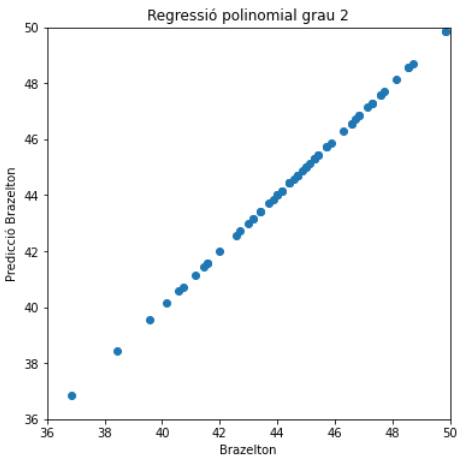
In [70]:
```
model_Br.fit(X,y)
y_pred = model_Br.predict(X)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Cuadrado Medio
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
            y       pred
0   43.714286  43.714286
1   45.714286  45.714286
2   49.857143  49.857143
3   47.714286  47.714286
4   44.714286  44.714286
..        ...        ...
57  47.285714  47.285714
58  47.142857  47.142857
59  44.142857  44.142857
60  46.571429  46.571429
61  44.714286  44.714286

[62 rows x 2 columns]
Mean squared error: 0.0000
R2 (coeficient de determinacio): 1.0000
```

In [71]:
```
resultats_CV (model_Br, X, y, cv=5)
```

```
Model Mean squared error és: -31.16 amb una desviació de +/- 9.35
Model R2 (Coef det.) és: -5.95 amb una desviació de +/- 3.60
```

In [72]:
```
# comprovació del sobreentrenament
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y,y_pred)
ax.set_xlim(36, 50)
ax.set_ylim(36, 50)
plt.xlabel('Brazelton')
plt.ylabel('Predicció Brazelton')
plt.title('Regressió polinomial grau 2')
plt.show()
```



4.6 AMB CONJUNT TRAIN I TEST

In [73]:
```
# separació de d'atributs i objectiu
```

```
X = np.array(data_regression_Br.drop(['Brazelton'],1))
y = np.array(data_regression_Br['Brazelton'])
```

In [74]:
```
# grau 2
pf = PolynomialFeatures(degree = 2)
X = pf.fit_transform(X)
```

In [75]:
```
# separació conjunt test. La mida es determina per testeig
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2, random_state=17)
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
```

```
X_train: (49, 703)
X_test: (13, 703)
```

In [76]:
```
model_Br = linear_model.LinearRegression()
```

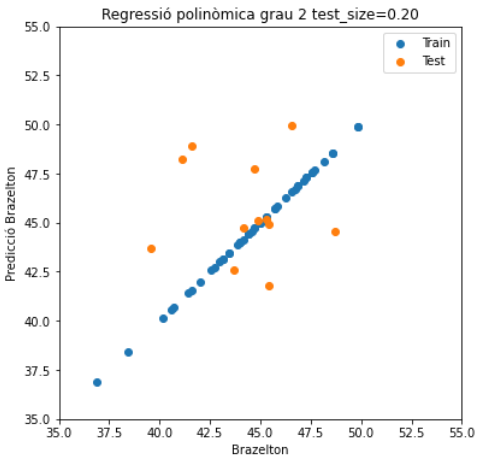
In [77]:
```
model_Br.fit(X_train,y_train)
y_pred = np.round(model_Br.predict(X),2)
prediccio = pd.DataFrame({"y": y, "pred": y_pred})
print(prediccio)

# Error Quadrat Mig
print("Mean squared error: %.4f" % mean_squared_error(y, y_pred))
# Coeficient de determinació
print('R2 (coeficient de determinacio): %.4f' % r2_score(y, y_pred))
```

```
            y   pred
0   43.714286  42.62
1   45.714286  45.71
2   49.857143  49.86
3   47.714286  47.71
4   44.714286  44.71
..        ...    ...
57  47.285714  47.29
58  47.142857  47.14
59  44.142857  44.14
60  46.571429  49.98
61  44.714286  44.71

[62 rows x 2 columns]
Mean squared error: 4.7673
R2 (coeficient de determinacio): 0.3268
```

In [78]:
```
resultats_CV (model_Br, X, y, cv=5)
```

```
Model Mean squared error és: -31.16 amb una desviació de +/- 9.35
Model R2 (Coef det.) és: -5.95 amb una desviació de +/- 3.60
```

In [79]:
```
# visualització de l'ajust
fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(y_train,np.round(model_Br.predict(X_train),2))
plt.scatter(y_test,np.round(model_Br.predict(X_test),2))
ax.set_xlim(35, 55)
ax.set_ylim(35, 55)
ax.legend(['Train', 'Test'])
plt.xlabel('Brazelton')
plt.ylabel('Predicció Brazelton')
plt.title('Regressió polinòmica grau 2 test_size=0.20')
plt.show()
```



In [ ]:

ANNEX 6: CODI PYTHON: MODEL DE CLASSIFICACIÓ DE LA PROGNOSI

TREBALL FINAL DE MASTER
MERITXELL BOSCH I GARCIA
2020/21-2 · Màster universitari en ciència de dades (*Data science*)
Estudis d'informàtica, multimèdia i telecomunicacions

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler, RobustScaler

from sklearn import linear_model, svm
from sklearn.svm import SVC, LinearSVC
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score, confusion_matrix, accuracy_score, f1_score, SCORERS

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline
```

In [2]:
```
# recuperar el joc de dades
data = pd.read_pickle("./data.pkl")
```

5. MODELS DE CLASSIFICACIO

In [3]:
```
# es discretitza el resultat del test Bayley entre els que són o no son anormals
data["Bayley_cod"]= np.where(data["Bayley"]<85,"1","0")
```

In [4]:
```
# comprovació
data[["Bayley","Bayley_cod"]]
```

Out[4]:

	Bayley	Bayley_cod
0	NaN	0
1	74.0	1
2	NaN	0
3	77.0	1
4	NaN	0
...	...	...
76	NaN	0
77	63.0	1
78	NaN	0
79	NaN	0
80	87.0	0

81 rows × 2 columns

In [5]:
```
# es crea la nova etiqueta objectiu: positiu en cas que un dels dos test doni positiu
data["B_o_V_cod"]= np.where((data["Vineland"]=="SI") | (data["Bayley_cod"]=="1"),"1","0")
```

In [6]:
```
# es crea un nou joc de dades eliminant totes les dades post-natals per a usar per classificar
data_logic = data.drop(['ID', 'Sexe', 'Diagnostic', 'Vineland', 'Bayley','Bayley_cod', 'Brazelton'],1)
data_logic = data_logic.dropna()
```

In [7]:
```
#pd.set_option('display.max_rows', None)
#display(data_logic)
#pd.set_option('display.max_rows', 10)
```

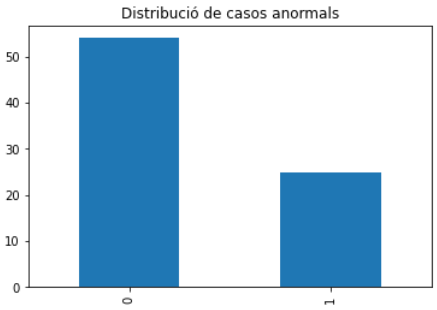
In [8]:
```
# guardar per a usos futurs
data_logic.to_pickle("./data_logic.pkl")
```

In [9]:
```
### analitzo el dataset
```

In [10]:
```
data_logic.value_counts(data['B_o_V_cod'])
```

Out[10]:
```
B_o_V_cod
0    54
1    25
dtype: int64
```

In [11]:
```
plot = data_logic['B_o_V_cod'].value_counts().plot(kind='bar',
                                        title='Distribució de casos anormals')
```



In [12]:
```
pd.crosstab(index=data_logic['B_o_V_cod'],
            columns=data_logic['Diagnostic_cod'] ,margins=True)
```

Out[12]:

Diagnostic_cod	0	1	2	3	All
B_o_V_cod					
0	33	7	6	8	54
1	6	7	4	8	25
All	39	14	10	16	79

In [13]:
```
# esta desbalancejada, caldrà tenir en compte no només accuracy sino F1
```

In [14]:
```
# separem les dades de l'etiqueta objectiu i es crea un joc de dades d'entrenament i de test
X = data_logic.iloc[:, :-1]
y = data_logic["B_o_V_cod"]

# mida test_size per testeig
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2, random_state=27)
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
```

```
X_train: (63, 34)
X_test: (16, 34)
```

In [15]:
```
# comprovem proporcionalitat entre entrenament i test
y_train.value_counts()
```

Out[15]:
```
0    44
1    19
Name: B_o_V_cod, dtype: int64
```

In [16]:
```
y_test.value_counts()
```

Out[16]:
```
0    10
1     6
Name: B_o_V_cod, dtype: int64
```

5.1 KNN

In [17]:
```
# definim model i graella per a determinar millors paràmetres
knn = KNeighborsClassifier()

param_grid = {"weights": ["uniform", "distance"]}

grid_search_knn = model_selection.GridSearchCV(knn, param_grid=param_grid, cv=5)

grid_search_knn.fit(X_train, y_train)

means = grid_search_knn.cv_results_["mean_test_score"]
stds = grid_search_knn.cv_results_["std_test_score"]
params = grid_search_knn.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors resultats s'obtenen amb: {}".format(grid_search_knn.best_params_))
knn_weights = grid_search_knn.best_params_["weights"]
```

```
Precisió mitjana: 66.79 +/- 5.07 amb paràmetres {'weights': 'uniform'}
Precisió mitjana: 66.79 +/- 5.07 amb paràmetres {'weights': 'distance'}
Els millors resultats s'obtenen amb: {'weights': 'uniform'}
```

In [18]:
```
# model amb millors paràmetres
model_knn = KNeighborsClassifier(n_neighbors=2, weights=knn_weights)
model_knn.fit(X_train, y_train)
```

Out[18]:
```
KNeighborsClassifier(n_neighbors=2)
```

In [19]:
```
# prediccions
y_pred_knn = model_knn.predict(X_test)
y_pred_knn
```

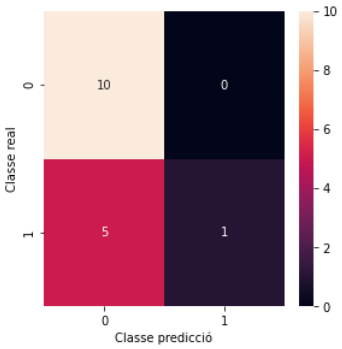
Out[19]:
```
array(['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0',
       '0', '0', '1'], dtype=object)
```

In [20]:
```
# mètriques
accuracy_knn = accuracy_score(y_test, y_pred_knn)
F1_score_knn = f1_score(y_test, y_pred_knn, average = "weighted")
print("L'accuracy del model knn és de:", accuracy_knn*100, "%")
print("La F1 del model knn és de:", np.round(F1_score_knn*100, 2), "%")
```

```
L'accuracy del model knn és de: 68.75 %
La F1 del model knn és de: 60.71 %
```

In [21]:

```
# matriu de confusió
knn_matrix = confusion_matrix(y_test, y_pred_knn)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(knn_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



5.2 DECISION TREE

In [22]:

```
# definim model i graella per a determinar millors paràmetres
dtc = DecisionTreeClassifier()

param_grid = {"min_samples_split": range(2, 10), "min_samples_leaf": range(1, 5),
              "max_depth": range(3, 6), "criterion": ["gini", "entropy"]}

grid_search_dtc = model_selection.GridSearchCV(dtc, param_grid=param_grid, cv=5)

grid_search_dtc.fit(X_train, y_train)

means = grid_search_dtc.cv_results_["mean_test_score"]
stds = grid_search_dtc.cv_results_["std_test_score"]
params = grid_search_dtc.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors resultats s'obtenen amb: {}".format(grid_search_dtc.best_params_))

dtc_criterion = grid_search_dtc.best_params_["criterion"]
dtc_min_samples_split = grid_search_dtc.best_params_["min_samples_split"]
dtc_min_samples_leaf = grid_search_dtc.best_params_["min_samples_leaf"]
dtc_max_depth = grid_search_dtc.best_params_["max_depth"]
```

```
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 3}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 4}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 5}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 6}
Precisió mitjana: 66.54 +/- 15.19 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 7}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 8}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 9}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 2}
Precisió mitjana: 71.54 +/- 18.22 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 3}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 4}
Precisió mitjana: 71.54 +/- 18.22 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 5}
Precisió mitjana: 66.54 +/- 15.19 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 6}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 7}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 8}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 9}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 2}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 3}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 4}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 5}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 6}
Precisió mitjana: 64.87 +/- 16.41 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 7}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 8}
Precisió mitjana: 69.87 +/- 19.68 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 9}
Precisió mitjana: 72.82 +/- 14.40 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 2}
Precisió mitjana: 72.82 +/- 14.40 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 3}
Precisió mitjana: 72.82 +/- 14.40 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 4}
Precisió mitjana: 72.82 +/- 14.40 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 5}
Precisió mitjana: 72.82 +/- 14.40 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 6}
Precisió mitjana: 72.82 +/- 14.40 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 7}
Precisió mitjana: 72.82 +/- 14.40 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 8}
Precisió mitjana: 72.82 +/- 14.40 amb paràmetres {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 9}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 2}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 3}
Precisió mitjana: 69.74 +/- 15.61 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 4}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 5}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 6}
Precisió mitjana: 69.74 +/- 12.66 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 7}
Precisió mitjana: 73.08 +/- 15.49 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 8}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 9}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 2}
Precisió mitjana: 74.49 +/- 16.65 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 3}
Precisió mitjana: 72.82 +/- 15.20 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 4}
Precisió mitjana: 69.74 +/- 12.66 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 5}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 6}
Precisió mitjana: 73.08 +/- 15.49 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 7}
Precisió mitjana: 71.41 +/- 17.32 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 8}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 9}
```

```
Precisió mitjana: 66.67 +/- 18.22 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 2}
Precisió mitjana: 66.67 +/- 18.22 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 3}
Precisió mitjana: 66.67 +/- 18.22 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 4}
Precisió mitjana: 63.33 +/- 16.29 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 5}
Precisió mitjana: 69.74 +/- 15.61 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 6}
Precisió mitjana: 63.33 +/- 16.29 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 7}
Precisió mitjana: 71.41 +/- 17.32 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 8}
Precisió mitjana: 66.41 +/- 14.05 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 9}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 2}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 3}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 4}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 5}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 6}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 7}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 8}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 9}
Precisió mitjana: 71.28 +/- 12.02 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2}
Precisió mitjana: 69.74 +/- 15.61 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 3}
Precisió mitjana: 69.74 +/- 12.66 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 4}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 6}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 7}
Precisió mitjana: 66.54 +/- 12.02 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 8}
Precisió mitjana: 68.08 +/- 14.47 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 9}
Precisió mitjana: 67.95 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}
Precisió mitjana: 67.95 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 3}
Precisió mitjana: 71.15 +/- 13.54 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 4}
Precisió mitjana: 66.54 +/- 12.02 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5}
Precisió mitjana: 66.54 +/- 12.02 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 6}
Precisió mitjana: 71.54 +/- 13.79 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 7}
Precisió mitjana: 69.87 +/- 15.66 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 8}
Precisió mitjana: 66.54 +/- 12.02 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 9}
Precisió mitjana: 66.41 +/- 14.05 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 2}
Precisió mitjana: 66.41 +/- 14.05 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 3}
Precisió mitjana: 66.41 +/- 14.05 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 4}
Precisió mitjana: 66.41 +/- 14.05 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 5}
Precisió mitjana: 66.41 +/- 14.05 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 6}
Precisió mitjana: 71.41 +/- 17.32 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 7}
Precisió mitjana: 66.41 +/- 14.05 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 8}
Precisió mitjana: 66.41 +/- 14.05 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 9}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 2}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 3}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 4}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 5}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 6}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 7}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 8}
Precisió mitjana: 71.28 +/- 15.10 amb paràmetres {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 9}
Precisió mitjana: 69.87 +/- 14.85 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2}
Precisió mitjana: 70.00 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 3}
Precisió mitjana: 73.21 +/- 15.61 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 4}
Precisió mitjana: 68.21 +/- 14.65 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 5}
Precisió mitjana: 71.54 +/- 15.77 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 6}
Precisió mitjana: 71.67 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 7}
Precisió mitjana: 70.00 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 8}
Precisió mitjana: 69.87 +/- 14.85 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 9}
Precisió mitjana: 68.21 +/- 14.65 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 2}
Precisió mitjana: 68.21 +/- 14.65 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 3}
Precisió mitjana: 71.54 +/- 15.77 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 4}
Precisió mitjana: 69.87 +/- 14.85 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 5}
Precisió mitjana: 71.67 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 6}
Precisió mitjana: 73.21 +/- 15.61 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 7}
Precisió mitjana: 70.00 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 8}
Precisió mitjana: 70.00 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 9}
Precisió mitjana: 66.67 +/- 17.09 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 2}
Precisió mitjana: 68.21 +/- 14.65 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 3}
Precisió mitjana: 68.33 +/- 17.41 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 4}
Precisió mitjana: 69.87 +/- 14.85 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 5}
Precisió mitjana: 71.67 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 6}
Precisió mitjana: 71.67 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 7}
Precisió mitjana: 66.67 +/- 17.09 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 8}
Precisió mitjana: 68.33 +/- 17.41 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 9}
Precisió mitjana: 71.67 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 2}
Precisió mitjana: 68.33 +/- 17.41 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 3}
Precisió mitjana: 69.87 +/- 14.85 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 4}
Precisió mitjana: 71.67 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 5}
Precisió mitjana: 68.33 +/- 17.41 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 6}
Precisió mitjana: 71.67 +/- 18.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 7}
Precisió mitjana: 73.21 +/- 15.61 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 8}
Precisió mitjana: 68.33 +/- 17.41 amb paràmetres {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 9}
Precisió mitjana: 74.62 +/- 10.13 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 2}
Precisió mitjana: 74.62 +/- 10.13 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 3}
Precisió mitjana: 74.62 +/- 10.13 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 4}
Precisió mitjana: 77.95 +/- 12.23 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 5}
Precisió mitjana: 74.49 +/- 10.51 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 6}
Precisió mitjana: 79.49 +/- 11.50 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 7}
Precisió mitjana: 76.15 +/- 11.11 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 8}
Precisió mitjana: 76.15 +/- 11.11 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 9}
Precisió mitjana: 74.49 +/- 10.51 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 2}
Precisió mitjana: 79.49 +/- 11.50 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 3}
Precisió mitjana: 79.49 +/- 11.50 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 4}
Precisió mitjana: 77.82 +/- 10.14 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 5}
Precisió mitjana: 74.49 +/- 10.51 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 6}
Precisió mitjana: 74.49 +/- 10.51 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 7}
Precisió mitjana: 77.95 +/- 12.23 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 8}
Precisió mitjana: 74.49 +/- 10.51 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 9}
Precisió mitjana: 74.62 +/- 10.13 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 2}
Precisió mitjana: 76.15 +/- 9.78 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 3}
Precisió mitjana: 76.15 +/- 9.78 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 4}
Precisió mitjana: 74.62 +/- 10.13 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 5}
Precisió mitjana: 76.15 +/- 11.11 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 6}
Precisió mitjana: 76.28 +/- 13.06 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 7}
Precisió mitjana: 77.95 +/- 12.23 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 8}
Precisió mitjana: 77.82 +/- 10.14 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 9}
Precisió mitjana: 74.36 +/- 11.07 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 2}
Precisió mitjana: 76.03 +/- 11.65 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 3}
Precisió mitjana: 74.36 +/- 11.07 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 4}
Precisió mitjana: 74.36 +/- 11.07 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 5}
Precisió mitjana: 72.82 +/- 11.14 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 6}
Precisió mitjana: 72.82 +/- 11.14 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 7}
Precisió mitjana: 74.49 +/- 11.93 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 8}
Precisió mitjana: 71.15 +/- 11.31 amb paràmetres {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'min_samples_split': 9}
Precisió mitjana: 74.87 +/- 12.10 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2}
Precisió mitjana: 68.21 +/- 8.52 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 3}
Precisió mitjana: 66.79 +/- 12.95 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 4}
Precisió mitjana: 71.67 +/- 12.12 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5}
```

```
Precisió mitjana: 71.54 +/- 8.89 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 6}
Precisió mitjana: 70.00 +/- 10.92 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 7}
Precisió mitjana: 68.46 +/- 13.36 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 8}
Precisió mitjana: 66.79 +/- 14.94 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 9}
Precisió mitjana: 69.87 +/- 8.86 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}
Precisió mitjana: 68.33 +/- 10.66 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 3}
Precisió mitjana: 70.13 +/- 16.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 4}
Precisió mitjana: 73.33 +/- 14.04 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5}
Precisió mitjana: 66.79 +/- 12.95 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 6}
Precisió mitjana: 73.21 +/- 12.54 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 7}
Precisió mitjana: 68.33 +/- 10.66 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 8}
Precisió mitjana: 71.67 +/- 12.12 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 9}
Precisió mitjana: 69.87 +/- 13.16 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 2}
Precisió mitjana: 71.54 +/- 13.18 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 3}
Precisió mitjana: 68.33 +/- 15.23 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 4}
Precisió mitjana: 69.87 +/- 13.16 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 5}
Precisió mitjana: 73.08 +/- 11.34 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 6}
Precisió mitjana: 71.54 +/- 14.19 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 7}
Precisió mitjana: 69.87 +/- 13.16 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 8}
Precisió mitjana: 68.33 +/- 15.23 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 9}
Precisió mitjana: 69.87 +/- 16.62 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 2}
Precisió mitjana: 69.74 +/- 13.56 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 3}
Precisió mitjana: 71.41 +/- 14.58 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 4}
Precisió mitjana: 69.74 +/- 13.56 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 5}
Precisió mitjana: 69.74 +/- 13.56 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 6}
Precisió mitjana: 68.21 +/- 15.57 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 7}
Precisió mitjana: 69.74 +/- 13.56 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 8}
Precisió mitjana: 68.21 +/- 15.57 amb paràmetres {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 9}
Els millors resultats s'obtenen amb: {'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 7}
```

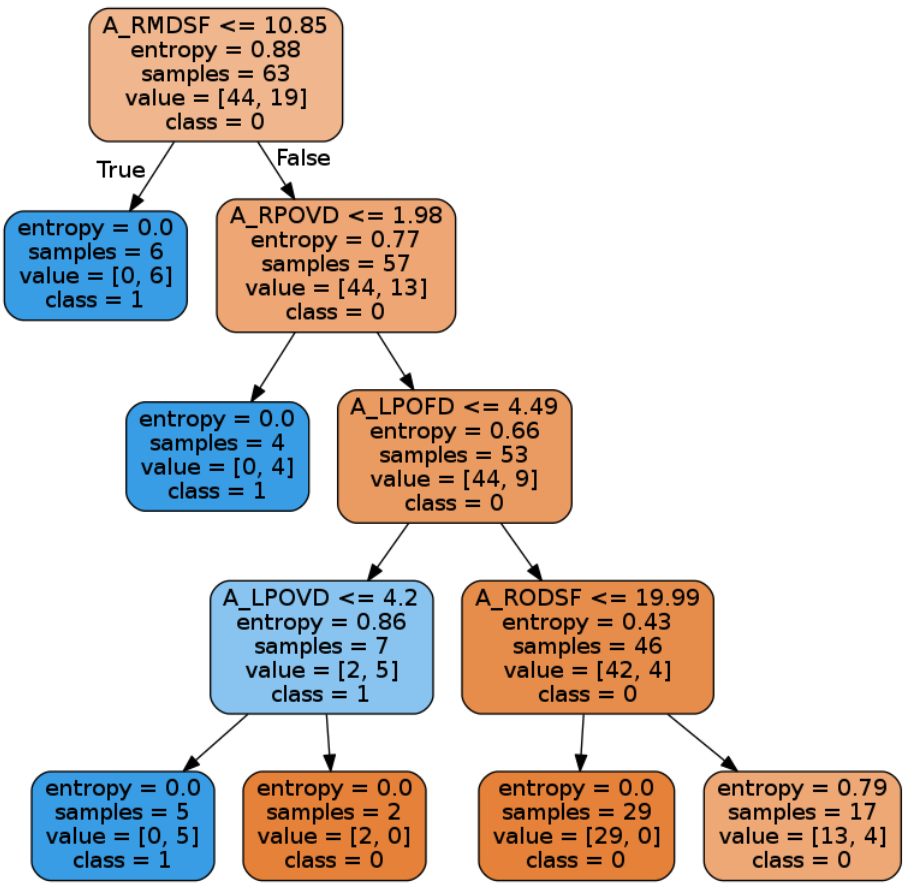
In [23]:

```
# model amb millors paràmetres

model_tree = DecisionTreeClassifier(criterion= dtc_criterion,
                                    min_samples_split=dtc_min_samples_split,
                                    min_samples_leaf=dtc_min_samples_leaf,
                                    max_depth = dtc_max_depth)
# Entreno el model
model_tree.fit(X_train, y_train)

parametres = data_logic.columns.values[:-1] # llista de tots els paràmetres a banda dels diagnostics

# S'exporta com a .dot file
export_graphviz(model_tree, out_file='tree.dot',
                feature_names = parametres,
                class_names = ['0','1'],
                rounded = True, proportion = False,
                precision = 2, filled = True)
```

In [24]:

```
# amb Graphviz es passa a png
#from subprocess import call
#call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

# Es retorna al notebook després de convertir el document a través de https://onlineconvertfree.com/complete/dot-png/
from IPython.display import Image
Image(filename = 'tree.png')
```

Out[24]:



In [25]:

```
# prediccions
y_pred_tree = model_tree.predict(X_test)
```

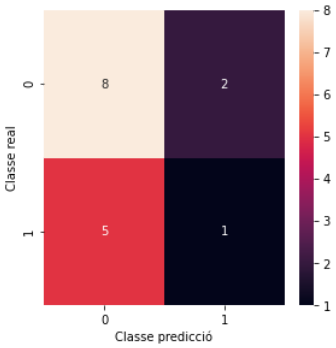
In [26]:

```
# mètriques
accuracy_tree = accuracy_score(y_test, y_pred_tree)
F1_score_tree = f1_score(y_test, y_pred_tree, average = "weighted")
print("L'accuracy del model decision tree és de:", accuracy_tree*100, "%")
print("La F1 del model decision tree és de:", np.round(F1_score_tree*100, 2), "%")
```

```
L'accuracy del model decision tree és de: 56.25 %
La F1 del model decision tree és de: 51.81 %
```

In [27]:

```
# matriu de confusió
tree_matrix = confusion_matrix(y_test, y_pred_tree)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(tree_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



5.3 RANDOM FOREST

In [28]:

```
# definim model i graella per a determinar millors paràmetres
rf= RandomForestClassifier()

param_grid = {"max_depth": range(2, 10), "n_estimators": range(2, 100, 5)}

grid_search_rf = model_selection.GridSearchCV(rf, param_grid=param_grid, cv=4)
grid_search_rf.fit(X_train, y_train)

means = grid_search_rf.cv_results_["mean_test_score"]
stds = grid_search_rf.cv_results_["std_test_score"]
params = grid_search_rf.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres per al Random Forest són: {}".format(grid_search_rf.best_params_))
rf_max_depth = grid_search_rf.best_params_["max_depth"]
rf_n_estimators = grid_search_rf.best_params_["n_estimators"]
```

```
Precisió mitjana: 65.10 +/- 5.18 amb paràmetres {'max_depth': 2, 'n_estimators': 2}
Precisió mitjana: 71.46 +/- 5.22 amb paràmetres {'max_depth': 2, 'n_estimators': 7}
Precisió mitjana: 66.77 +/- 6.36 amb paràmetres {'max_depth': 2, 'n_estimators': 12}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 2, 'n_estimators': 17}
Precisió mitjana: 71.46 +/- 2.77 amb paràmetres {'max_depth': 2, 'n_estimators': 22}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'max_depth': 2, 'n_estimators': 27}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 2, 'n_estimators': 32}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 2, 'n_estimators': 37}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 2, 'n_estimators': 42}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 2, 'n_estimators': 47}
Precisió mitjana: 71.56 +/- 4.87 amb paràmetres {'max_depth': 2, 'n_estimators': 52}
Precisió mitjana: 71.56 +/- 4.87 amb paràmetres {'max_depth': 2, 'n_estimators': 57}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 2, 'n_estimators': 62}
Precisió mitjana: 74.69 +/- 3.99 amb paràmetres {'max_depth': 2, 'n_estimators': 67}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'max_depth': 2, 'n_estimators': 72}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 2, 'n_estimators': 77}
Precisió mitjana: 71.56 +/- 4.87 amb paràmetres {'max_depth': 2, 'n_estimators': 82}
Precisió mitjana: 71.56 +/- 4.87 amb paràmetres {'max_depth': 2, 'n_estimators': 87}
Precisió mitjana: 74.69 +/- 5.95 amb paràmetres {'max_depth': 2, 'n_estimators': 92}
Precisió mitjana: 71.56 +/- 4.87 amb paràmetres {'max_depth': 2, 'n_estimators': 97}
Precisió mitjana: 68.54 +/- 11.36 amb paràmetres {'max_depth': 3, 'n_estimators': 2}
Precisió mitjana: 69.90 +/- 4.84 amb paràmetres {'max_depth': 3, 'n_estimators': 7}
Precisió mitjana: 71.46 +/- 2.77 amb paràmetres {'max_depth': 3, 'n_estimators': 12}
Precisió mitjana: 66.88 +/- 8.77 amb paràmetres {'max_depth': 3, 'n_estimators': 17}
Precisió mitjana: 71.56 +/- 9.07 amb paràmetres {'max_depth': 3, 'n_estimators': 22}
Precisió mitjana: 71.46 +/- 2.77 amb paràmetres {'max_depth': 3, 'n_estimators': 27}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'max_depth': 3, 'n_estimators': 32}
Precisió mitjana: 73.02 +/- 5.11 amb paràmetres {'max_depth': 3, 'n_estimators': 37}
Precisió mitjana: 77.81 +/- 2.85 amb paràmetres {'max_depth': 3, 'n_estimators': 42}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 3, 'n_estimators': 47}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 3, 'n_estimators': 52}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 3, 'n_estimators': 57}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 3, 'n_estimators': 62}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 3, 'n_estimators': 67}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 3, 'n_estimators': 72}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 3, 'n_estimators': 77}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 3, 'n_estimators': 82}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 3, 'n_estimators': 87}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 3, 'n_estimators': 92}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 3, 'n_estimators': 97}
Precisió mitjana: 61.98 +/- 5.78 amb paràmetres {'max_depth': 4, 'n_estimators': 2}
Precisió mitjana: 65.21 +/- 6.44 amb paràmetres {'max_depth': 4, 'n_estimators': 7}
Precisió mitjana: 65.10 +/- 6.81 amb paràmetres {'max_depth': 4, 'n_estimators': 12}
Precisió mitjana: 74.69 +/- 7.41 amb paràmetres {'max_depth': 4, 'n_estimators': 17}
Precisió mitjana: 65.31 +/- 15.94 amb paràmetres {'max_depth': 4, 'n_estimators': 22}
Precisió mitjana: 66.77 +/- 9.95 amb paràmetres {'max_depth': 4, 'n_estimators': 27}
Precisió mitjana: 73.12 +/- 7.83 amb paràmetres {'max_depth': 4, 'n_estimators': 32}
Precisió mitjana: 70.00 +/- 6.31 amb paràmetres {'max_depth': 4, 'n_estimators': 37}
```

```
Precisió mitjana: 65.31 +/- 10.84 amb paràmetres {'max_depth': 4, 'n_estimators': 42}
Precisió mitjana: 70.00 +/- 6.31 amb paràmetres {'max_depth': 4, 'n_estimators': 47}
Precisió mitjana: 74.69 +/- 5.95 amb paràmetres {'max_depth': 4, 'n_estimators': 52}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 4, 'n_estimators': 57}
Precisió mitjana: 73.12 +/- 6.47 amb paràmetres {'max_depth': 4, 'n_estimators': 62}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 4, 'n_estimators': 67}
Precisió mitjana: 69.90 +/- 4.84 amb paràmetres {'max_depth': 4, 'n_estimators': 72}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 4, 'n_estimators': 77}
Precisió mitjana: 73.12 +/- 7.83 amb paràmetres {'max_depth': 4, 'n_estimators': 82}
Precisió mitjana: 69.90 +/- 7.91 amb paràmetres {'max_depth': 4, 'n_estimators': 87}
Precisió mitjana: 74.69 +/- 5.95 amb paràmetres {'max_depth': 4, 'n_estimators': 92}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 4, 'n_estimators': 97}
Precisió mitjana: 63.75 +/- 13.14 amb paràmetres {'max_depth': 5, 'n_estimators': 2}
Precisió mitjana: 63.54 +/- 7.86 amb paràmetres {'max_depth': 5, 'n_estimators': 7}
Precisió mitjana: 66.77 +/- 7.74 amb paràmetres {'max_depth': 5, 'n_estimators': 12}
Precisió mitjana: 73.02 +/- 2.56 amb paràmetres {'max_depth': 5, 'n_estimators': 17}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 5, 'n_estimators': 22}
Precisió mitjana: 71.46 +/- 2.77 amb paràmetres {'max_depth': 5, 'n_estimators': 27}
Precisió mitjana: 74.69 +/- 3.99 amb paràmetres {'max_depth': 5, 'n_estimators': 32}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 5, 'n_estimators': 37}
Precisió mitjana: 71.56 +/- 9.07 amb paràmetres {'max_depth': 5, 'n_estimators': 42}
Precisió mitjana: 70.00 +/- 6.31 amb paràmetres {'max_depth': 5, 'n_estimators': 47}
Precisió mitjana: 73.02 +/- 6.75 amb paràmetres {'max_depth': 5, 'n_estimators': 52}
Precisió mitjana: 74.69 +/- 7.41 amb paràmetres {'max_depth': 5, 'n_estimators': 57}
Precisió mitjana: 73.12 +/- 7.83 amb paràmetres {'max_depth': 5, 'n_estimators': 62}
Precisió mitjana: 74.69 +/- 7.41 amb paràmetres {'max_depth': 5, 'n_estimators': 67}
Precisió mitjana: 74.69 +/- 3.99 amb paràmetres {'max_depth': 5, 'n_estimators': 72}
Precisió mitjana: 68.33 +/- 3.85 amb paràmetres {'max_depth': 5, 'n_estimators': 77}
Precisió mitjana: 73.02 +/- 2.56 amb paràmetres {'max_depth': 5, 'n_estimators': 82}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 5, 'n_estimators': 87}
Precisió mitjana: 70.00 +/- 6.31 amb paràmetres {'max_depth': 5, 'n_estimators': 92}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 5, 'n_estimators': 97}
Precisió mitjana: 63.75 +/- 10.68 amb paràmetres {'max_depth': 6, 'n_estimators': 2}
Precisió mitjana: 68.33 +/- 5.86 amb paràmetres {'max_depth': 6, 'n_estimators': 7}
Precisió mitjana: 70.00 +/- 6.31 amb paràmetres {'max_depth': 6, 'n_estimators': 12}
Precisió mitjana: 68.33 +/- 3.85 amb paràmetres {'max_depth': 6, 'n_estimators': 17}
Precisió mitjana: 71.46 +/- 2.77 amb paràmetres {'max_depth': 6, 'n_estimators': 22}
Precisió mitjana: 69.90 +/- 4.84 amb paràmetres {'max_depth': 6, 'n_estimators': 27}
Precisió mitjana: 71.56 +/- 4.87 amb paràmetres {'max_depth': 6, 'n_estimators': 32}
Precisió mitjana: 73.02 +/- 5.11 amb paràmetres {'max_depth': 6, 'n_estimators': 37}
Precisió mitjana: 70.00 +/- 11.73 amb paràmetres {'max_depth': 6, 'n_estimators': 42}
Precisió mitjana: 71.46 +/- 2.77 amb paràmetres {'max_depth': 6, 'n_estimators': 47}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 6, 'n_estimators': 52}
Precisió mitjana: 77.81 +/- 5.26 amb paràmetres {'max_depth': 6, 'n_estimators': 57}
Precisió mitjana: 73.12 +/- 6.47 amb paràmetres {'max_depth': 6, 'n_estimators': 62}
Precisió mitjana: 66.88 +/- 8.77 amb paràmetres {'max_depth': 6, 'n_estimators': 67}
Precisió mitjana: 68.44 +/- 7.15 amb paràmetres {'max_depth': 6, 'n_estimators': 72}
Precisió mitjana: 71.56 +/- 9.07 amb paràmetres {'max_depth': 6, 'n_estimators': 77}
Precisió mitjana: 71.46 +/- 2.77 amb paràmetres {'max_depth': 6, 'n_estimators': 82}
Precisió mitjana: 71.56 +/- 4.87 amb paràmetres {'max_depth': 6, 'n_estimators': 87}
Precisió mitjana: 73.12 +/- 6.47 amb paràmetres {'max_depth': 6, 'n_estimators': 92}
Precisió mitjana: 76.25 +/- 2.17 amb paràmetres {'max_depth': 6, 'n_estimators': 97}
Precisió mitjana: 63.65 +/- 8.77 amb paràmetres {'max_depth': 7, 'n_estimators': 2}
Precisió mitjana: 61.88 +/- 8.90 amb paràmetres {'max_depth': 7, 'n_estimators': 7}
Precisió mitjana: 63.44 +/- 5.48 amb paràmetres {'max_depth': 7, 'n_estimators': 12}
Precisió mitjana: 66.77 +/- 4.57 amb paràmetres {'max_depth': 7, 'n_estimators': 17}
Precisió mitjana: 70.00 +/- 6.31 amb paràmetres {'max_depth': 7, 'n_estimators': 22}
Precisió mitjana: 71.46 +/- 6.84 amb paràmetres {'max_depth': 7, 'n_estimators': 27}
Precisió mitjana: 71.56 +/- 4.87 amb paràmetres {'max_depth': 7, 'n_estimators': 32}
Precisió mitjana: 76.35 +/- 6.48 amb paràmetres {'max_depth': 7, 'n_estimators': 37}
Precisió mitjana: 66.77 +/- 9.95 amb paràmetres {'max_depth': 7, 'n_estimators': 42}
Precisió mitjana: 70.00 +/- 7.71 amb paràmetres {'max_depth': 7, 'n_estimators': 47}
Precisió mitjana: 73.12 +/- 7.83 amb paràmetres {'max_depth': 7, 'n_estimators': 52}
Precisió mitjana: 70.00 +/- 7.71 amb paràmetres {'max_depth': 7, 'n_estimators': 57}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 7, 'n_estimators': 62}
Precisió mitjana: 71.56 +/- 9.07 amb paràmetres {'max_depth': 7, 'n_estimators': 67}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 7, 'n_estimators': 72}
Precisió mitjana: 74.58 +/- 4.48 amb paràmetres {'max_depth': 7, 'n_estimators': 77}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 7, 'n_estimators': 82}
Precisió mitjana: 74.69 +/- 3.99 amb paràmetres {'max_depth': 7, 'n_estimators': 87}
Precisió mitjana: 71.56 +/- 10.09 amb paràmetres {'max_depth': 7, 'n_estimators': 92}
Precisió mitjana: 70.00 +/- 8.88 amb paràmetres {'max_depth': 7, 'n_estimators': 97}
Precisió mitjana: 73.12 +/- 10.02 amb paràmetres {'max_depth': 8, 'n_estimators': 2}
Precisió mitjana: 74.90 +/- 10.64 amb paràmetres {'max_depth': 8, 'n_estimators': 7}
Precisió mitjana: 74.69 +/- 9.70 amb paràmetres {'max_depth': 8, 'n_estimators': 12}
Precisió mitjana: 69.90 +/- 4.84 amb paràmetres {'max_depth': 8, 'n_estimators': 17}
Precisió mitjana: 66.88 +/- 8.77 amb paràmetres {'max_depth': 8, 'n_estimators': 22}
Precisió mitjana: 73.12 +/- 7.83 amb paràmetres {'max_depth': 8, 'n_estimators': 27}
Precisió mitjana: 70.00 +/- 8.88 amb paràmetres {'max_depth': 8, 'n_estimators': 32}
Precisió mitjana: 77.81 +/- 2.85 amb paràmetres {'max_depth': 8, 'n_estimators': 37}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 8, 'n_estimators': 42}
Precisió mitjana: 77.81 +/- 2.85 amb paràmetres {'max_depth': 8, 'n_estimators': 47}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 8, 'n_estimators': 52}
Precisió mitjana: 68.33 +/- 3.85 amb paràmetres {'max_depth': 8, 'n_estimators': 57}
Precisió mitjana: 74.69 +/- 3.99 amb paràmetres {'max_depth': 8, 'n_estimators': 62}
Precisió mitjana: 74.69 +/- 3.99 amb paràmetres {'max_depth': 8, 'n_estimators': 67}
Precisió mitjana: 73.12 +/- 6.47 amb paràmetres {'max_depth': 8, 'n_estimators': 72}
Precisió mitjana: 73.12 +/- 10.02 amb paràmetres {'max_depth': 8, 'n_estimators': 77}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 8, 'n_estimators': 82}
Precisió mitjana: 76.25 +/- 2.17 amb paràmetres {'max_depth': 8, 'n_estimators': 87}
Precisió mitjana: 71.46 +/- 5.22 amb paràmetres {'max_depth': 8, 'n_estimators': 92}
Precisió mitjana: 73.12 +/- 6.47 amb paràmetres {'max_depth': 8, 'n_estimators': 97}
Precisió mitjana: 69.69 +/- 6.15 amb paràmetres {'max_depth': 9, 'n_estimators': 2}
Precisió mitjana: 63.65 +/- 8.77 amb paràmetres {'max_depth': 9, 'n_estimators': 7}
Precisió mitjana: 66.77 +/- 7.74 amb paràmetres {'max_depth': 9, 'n_estimators': 12}
Precisió mitjana: 70.00 +/- 8.88 amb paràmetres {'max_depth': 9, 'n_estimators': 17}
Precisió mitjana: 74.58 +/- 4.48 amb paràmetres {'max_depth': 9, 'n_estimators': 22}
Precisió mitjana: 76.15 +/- 6.94 amb paràmetres {'max_depth': 9, 'n_estimators': 27}
Precisió mitjana: 74.69 +/- 7.41 amb paràmetres {'max_depth': 9, 'n_estimators': 32}
Precisió mitjana: 73.02 +/- 5.11 amb paràmetres {'max_depth': 9, 'n_estimators': 37}
Precisió mitjana: 71.46 +/- 5.22 amb paràmetres {'max_depth': 9, 'n_estimators': 42}
Precisió mitjana: 76.25 +/- 4.92 amb paràmetres {'max_depth': 9, 'n_estimators': 47}
Precisió mitjana: 73.02 +/- 2.56 amb paràmetres {'max_depth': 9, 'n_estimators': 52}
Precisió mitjana: 77.81 +/- 5.26 amb paràmetres {'max_depth': 9, 'n_estimators': 57}
Precisió mitjana: 73.12 +/- 4.72 amb paràmetres {'max_depth': 9, 'n_estimators': 62}
Precisió mitjana: 69.90 +/- 4.84 amb paràmetres {'max_depth': 9, 'n_estimators': 67}
Precisió mitjana: 73.02 +/- 2.56 amb paràmetres {'max_depth': 9, 'n_estimators': 72}
Precisió mitjana: 73.12 +/- 6.47 amb paràmetres {'max_depth': 9, 'n_estimators': 77}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 9, 'n_estimators': 82}
Precisió mitjana: 74.69 +/- 7.41 amb paràmetres {'max_depth': 9, 'n_estimators': 87}
Precisió mitjana: 71.56 +/- 6.58 amb paràmetres {'max_depth': 9, 'n_estimators': 92}
Precisió mitjana: 74.69 +/- 7.41 amb paràmetres {'max_depth': 9, 'n_estimators': 97}
Els millors paràmetres per al Random Forest són: {'max_depth': 3, 'n_estimators': 42}
```

In [29]:

```
# model amb millors paràmetres
```

```
model_rf = RandomForestClassifier(max_depth=rf_max_depth, n_estimators=rf_n_estimators, random_state=0)
model_rf.fit(X_train, y_train)
```

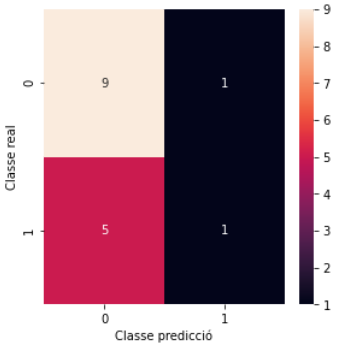
Out[29]:
```
RandomForestClassifier(max_depth=3, n_estimators=42, random_state=0)
```

In [30]:
```
#prediccions
y_pred_rf = model_rf.predict(X_test)
```

In [31]:
```
#mètriques
accuracy_rf = accuracy_score(y_test, y_pred_rf)
F1_score_rf = f1_score(y_test, y_pred_rf, average = "weighted")
print("L'accuracy del model random forest és de:", accuracy_rf*100, "%")
print("La F1 del model random forest és de:", np.round(F1_score_rf*100, 2), "%")
```

```
L'accuracy del model random forest és de: 62.5 %
La F1 del model random forest és de: 56.25 %
```

In [32]:
```
#matriu de confusio
rf_matrix = confusion_matrix(y_test, y_pred_rf)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(rf_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



5.4 SUPORT VECTOR MACHINE

5.4.1 KERNEL LINIAL

In [33]:
```
# definim model i graella per a determinar millors paràmetres
svm = SVC(kernel= 'linear')

param_grid = {"C": [0.05, 0.1, 1, 10, 50]}

grid_search_svm= model_selection.GridSearchCV(svm, param_grid=param_grid, cv=4)
grid_search_svm.fit(X_train, y_train)

means= grid_search_svm.cv_results_["mean_test_score"]
stds= grid_search_svm.cv_results_["std_test_score"]
params= grid_search_svm.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres són: {}".format(grid_search_svm.best_params_))
svm_C = grid_search_svm.best_params_["C"]
```

```
Precisió mitjana: 60.42 +/- 10.83 amb paràmetres {'C': 0.05}
Precisió mitjana: 62.19 +/- 12.82 amb paràmetres {'C': 0.1}
Precisió mitjana: 52.81 +/- 20.55 amb paràmetres {'C': 1}
Precisió mitjana: 51.15 +/- 18.44 amb paràmetres {'C': 10}
Precisió mitjana: 51.15 +/- 18.44 amb paràmetres {'C': 50}
Els millors paràmetres són: {'C': 0.1}
```

In [34]:
```
#model amb millors paràmetres
model_svm_linial = SVC(kernel= 'linear',C= svm_C)
model_svm_linial.fit(X_train, y_train)
```

Out[34]:
```
SVC(C=0.1, kernel='linear')
```

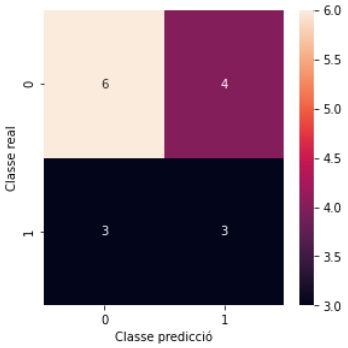
In [35]:
```
#prediccions
y_pred_svm = model_svm_linial.predict(X_test)
```

In [36]:
```
#mètriques
accuracy_svm = accuracy_score(y_test, y_pred_svm)
F1_score_svm = f1_score(y_test, y_pred_svm, average = "weighted")
print("L'accuracy del model SVM linial és de:", accuracy_svm*100, "%")
print("La F1 del model SVM linial és de:", np.round(F1_score_svm*100, 2), "%")
```

```
L'accuracy del model SVM linial és de: 56.25 %
La F1 del model SVM linial és de: 56.78 %
```

In [37]:
```
#matriu de confusió
svm_matrix = confusion_matrix(y_test, y_pred_svm)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']
```

```
figure = plt.figure(figsize=(4, 4))
sns.heatmap(svm_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



5.4.2 KERNEL RADIAL

In [38]:
```
# definim model i graella per a determinar millors paràmetres
svm_rad = SVC(kernel= 'rbf')

param_grid = {"C": [0.05, 0.1, 1, 10, 50], "gamma": [0.001, 0.01, 0.1, 1, 10]}

grid_search_svm_rad= model_selection.GridSearchCV(svm_rad, param_grid=param_grid, cv=4)
grid_search_svm_rad.fit(X_train, y_train)

means= grid_search_svm_rad.cv_results_["mean_test_score"]
stds= grid_search_svm_rad.cv_results_["std_test_score"]
params= grid_search_svm_rad.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres són: {}".format(grid_search_svm_rad.best_params_))
svm_rad_C = grid_search_svm_rad.best_params_["C"]
svm_rad_gamma = grid_search_svm_rad.best_params_["gamma"]
```

```
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 10}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 10}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 10}
Precisió mitjana: 62.19 +/- 11.19 amb paràmetres {'C': 10, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 10}
Precisió mitjana: 55.83 +/- 11.03 amb paràmetres {'C': 50, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 10}
Els millors paràmetres són: {'C': 0.05, 'gamma': 0.001}
```

In [39]:
```
# model amb els millors paràmetres
model_svm_rad = SVC(kernel= 'rbf',C= svm_rad_C, gamma= svm_rad_gamma )
model_svm_rad.fit(X_train, y_train)
```

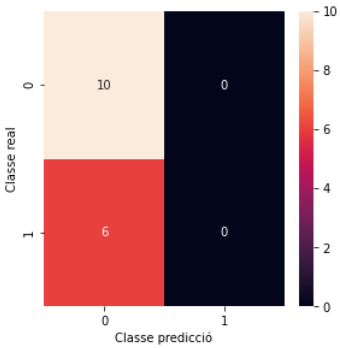
Out[39]:
```
SVC(C=0.05, gamma=0.001)
```

In [40]:
```
# prediccions
y_pred_svm_rad = model_svm_rad.predict(X_test)
```

In [41]:
```
# mètriques
accuracy_svm_rad = accuracy_score(y_test, y_pred_svm_rad)
F1_score_svm_rad = f1_score(y_test, y_pred_svm_rad, average = "weighted")
print("L'accuracy del model SVM radial és de:", accuracy_svm_rad*100, "%")
print("La F1 del model SVM radial és de:", np.round(F1_score_svm_rad*100, 2), "%")
```

```
L'accuracy del model SVM radial és de: 62.5 %
La F1 del model SVM radial és de: 48.08 %
```

In [42]:
```
# matriu de confusió
svm_rad_matrix = confusion_matrix(y_test, y_pred_svm_rad)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(svm_rad_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



5.4.3 KERNEL POLINOMIAL

In [43]:
```
# definim model i graella per a determinar millors paràmetres
svm_pol = SVC(kernel= 'poly')

param_grid = {"C": [0.01, 0.05, 0.1, 1, 10], "gamma": [0.001, 0.01, 0.1, 1, 10], "degree": range(3, 5)}

grid_search_svm_pol= model_selection.GridSearchCV(svm_pol, param_grid=param_grid, cv=4)
grid_search_svm_pol.fit(X_train, y_train)

means= grid_search_svm_pol.cv_results_["mean_test_score"]
stds= grid_search_svm_pol.cv_results_["std_test_score"]
params= grid_search_svm_pol.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres són: {}".format(grid_search_svm_pol.best_params_))
svm_pol_C = grid_search_svm_pol.best_params_["C"]
svm_pol_gamma = grid_search_svm_pol.best_params_["gamma"]
svm_pol_degree = grid_search_svm_pol.best_params_["degree"]
```

```
Precisió mitjana: 66.88 +/- 11.64 amb paràmetres {'C': 0.01, 'degree': 3, 'gamma': 0.001}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.01, 'degree': 3, 'gamma': 0.01}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.01, 'degree': 3, 'gamma': 0.1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.01, 'degree': 3, 'gamma': 1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.01, 'degree': 3, 'gamma': 10}
Precisió mitjana: 55.94 +/- 14.12 amb paràmetres {'C': 0.01, 'degree': 4, 'gamma': 0.001}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.01, 'degree': 4, 'gamma': 0.01}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.01, 'degree': 4, 'gamma': 0.1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.01, 'degree': 4, 'gamma': 1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.01, 'degree': 4, 'gamma': 10}
Precisió mitjana: 58.96 +/- 15.85 amb paràmetres {'C': 0.05, 'degree': 3, 'gamma': 0.001}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.05, 'degree': 3, 'gamma': 0.01}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.05, 'degree': 3, 'gamma': 0.1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.05, 'degree': 3, 'gamma': 1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.05, 'degree': 3, 'gamma': 10}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.05, 'degree': 4, 'gamma': 0.001}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.05, 'degree': 4, 'gamma': 0.01}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.05, 'degree': 4, 'gamma': 0.1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.05, 'degree': 4, 'gamma': 1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.05, 'degree': 4, 'gamma': 10}
Precisió mitjana: 55.94 +/- 14.12 amb paràmetres {'C': 0.1, 'degree': 3, 'gamma': 0.001}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.1, 'degree': 3, 'gamma': 0.01}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.1, 'degree': 3, 'gamma': 0.1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.1, 'degree': 3, 'gamma': 1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 0.1, 'degree': 3, 'gamma': 10}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.1, 'degree': 4, 'gamma': 0.001}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.1, 'degree': 4, 'gamma': 0.01}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.1, 'degree': 4, 'gamma': 0.1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.1, 'degree': 4, 'gamma': 1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 0.1, 'degree': 4, 'gamma': 10}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 1, 'degree': 3, 'gamma': 0.001}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 1, 'degree': 3, 'gamma': 0.01}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 1, 'degree': 3, 'gamma': 0.1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 1, 'degree': 3, 'gamma': 1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 1, 'degree': 3, 'gamma': 10}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 1, 'degree': 4, 'gamma': 0.001}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 1, 'degree': 4, 'gamma': 0.01}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 1, 'degree': 4, 'gamma': 0.1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 1, 'degree': 4, 'gamma': 1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 1, 'degree': 4, 'gamma': 10}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 10, 'degree': 3, 'gamma': 0.001}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 10, 'degree': 3, 'gamma': 0.01}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 10, 'degree': 3, 'gamma': 0.1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 10, 'degree': 3, 'gamma': 1}
Precisió mitjana: 49.58 +/- 14.63 amb paràmetres {'C': 10, 'degree': 3, 'gamma': 10}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 10, 'degree': 4, 'gamma': 0.001}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 10, 'degree': 4, 'gamma': 0.01}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 10, 'degree': 4, 'gamma': 0.1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 10, 'degree': 4, 'gamma': 1}
Precisió mitjana: 51.25 +/- 17.37 amb paràmetres {'C': 10, 'degree': 4, 'gamma': 10}
Els millors paràmetres són: {'C': 0.01, 'degree': 3, 'gamma': 0.001}
```

In [44]:
```
# model amb els millors paràmetres
model_svm_pol = SVC(kernel= 'poly',C= svm_pol_C, gamma= svm_pol_gamma, degree = svm_pol_degree )
model_svm_pol.fit(X_train, y_train)
```

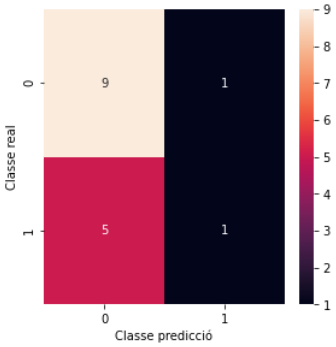
Out[44]:
```
SVC(C=0.01, gamma=0.001, kernel='poly')
```

In [45]:
```
# prediccions
y_pred_svm_pol = model_svm_pol.predict(X_test)
```

In [46]:
```
# mètriques
accuracy_svm_pol = accuracy_score(y_test, y_pred_svm_pol)
F1_score_svm_pol = f1_score(y_test, y_pred_svm_pol, average = "weighted")
print("L'accuracy del model SVM polinomial és de:", accuracy_svm_pol*100, "%")
print("La F1 del model SVM polinomial és de:", np.round(F1_score_svm_pol*100, 2), "%")
```

```
L'accuracy del model SVM polinomial és de: 62.5 %
La F1 del model SVM polinomial és de: 56.25 %
```

In [47]:
```
# matriu de confusió
svm_pol_matrix = confusion_matrix(y_test, y_pred_svm_pol)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(svm_pol_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



5.4.4 KERNEL SIGMOIDE

In [48]:
```
# definim model i graella per a determinar millors paràmetres
svm_sig = SVC(kernel= 'sigmoid')

param_grid = {"C": [0.05, 0.1, 1, 10, 50], "gamma": [0.001, 0.01, 0.1, 1, 10]}

grid_search_svm_sig= model_selection.GridSearchCV(svm_sig, param_grid=param_grid, cv=4)
grid_search_svm_sig.fit(X_train, y_train)

means= grid_search_svm_sig.cv_results_["mean_test_score"]
stds= grid_search_svm_sig.cv_results_["std_test_score"]
params= grid_search_svm_sig.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres són: {}".format(grid_search_svm_sig.best_params_))
svm_sig_C = grid_search_svm_sig.best_params_["C"]
svm_sig_gamma = grid_search_svm_sig.best_params_["gamma"]
```

```
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.05, 'gamma': 10}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 0.1, 'gamma': 10}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 1, 'gamma': 10}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 10, 'gamma': 10}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 0.001}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 0.01}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 0.1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 1}
Precisió mitjana: 69.90 +/- 1.98 amb paràmetres {'C': 50, 'gamma': 10}
Els millors paràmetres són: {'C': 0.05, 'gamma': 0.001}
```

In [49]:
```
# model amb els millors paràmetres
model_svm_sig = SVC(kernel= 'sigmoid',C= svm_sig_C, gamma= svm_sig_gamma )
model_svm_sig.fit(X_train, y_train)
```

Out[49]:
```
SVC(C=0.05, gamma=0.001, kernel='sigmoid')
```

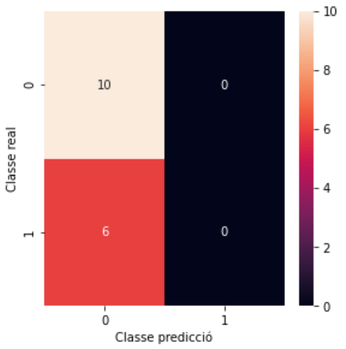
In [50]:
```
# prediccions
y_pred_svm_sig = model_svm_sig.predict(X_test)
```

In [51]:
```
# mètriques
accuracy_svm_sig = accuracy_score(y_test, y_pred_svm_sig)
f1_score_svm_sig = f1_score(y_test, y_pred_svm_sig, average = "weighted")
print("L'accuracy del model SVM radial és de:", accuracy_svm_sig*100, "%")
print("La F1 del model SVM radial és de:", np.round(f1_score_svm_sig*100, 2), "%")
```

```
L'accuracy del model SVM radial és de: 62.5 %
La F1 del model SVM radial és de: 48.08 %
```

In [52]:
```
# matriu de confusió
svm_sig_matrix = confusion_matrix(y_test, y_pred_svm_sig)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']
```

```
figure = plt.figure(figsize=(4, 4))
sns.heatmap(svm_sig_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



5.4.5 KERNEL LINIAL2

In [53]:
```
# definim model i graella per a determinar millors paràmetres
lin_svm = LinearSVC(max_iter=500000, dual = True)

param_grid = {"C": [0.05, 0.1, 1, 10, 50]}

grid_search_lin_svm= model_selection.GridSearchCV(lin_svm, param_grid=param_grid, cv=4)
grid_search_lin_svm.fit(X_train, y_train)

means= grid_search_lin_svm.cv_results_["mean_test_score"]
stds= grid_search_lin_svm.cv_results_["std_test_score"]
params= grid_search_lin_svm.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres són: {}".format(grid_search_lin_svm.best_params_))
lin_svm_C = grid_search_lin_svm.best_params_["C"]
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning: Liblinear faile
d to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning: Liblinear faile
d to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning: Liblinear faile
d to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
Precisió mitjana: 57.40 +/- 11.41 amb paràmetres {'C': 0.05}
Precisió mitjana: 52.71 +/- 12.18 amb paràmetres {'C': 0.1}
Precisió mitjana: 49.48 +/- 15.34 amb paràmetres {'C': 1}
Precisió mitjana: 51.15 +/- 17.34 amb paràmetres {'C': 10}
Precisió mitjana: 49.48 +/- 15.34 amb paràmetres {'C': 50}
Els millors paràmetres són: {'C': 0.05}
```

In [54]:
```
# model amb millors paràmetres
model_lin_svm = LinearSVC(max_iter=500000, dual = True, C= lin_svm_C)
model_lin_svm.fit(X_train, y_train)
```

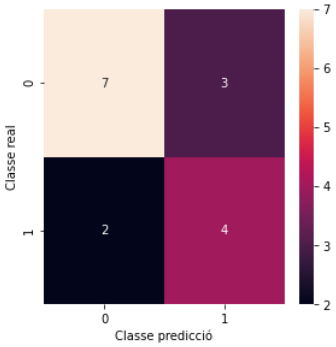
Out[54]:
```
LinearSVC(C=0.05, max_iter=500000)
```

In [55]:
```
# prediccions
y_pred_lin_svm = model_lin_svm.predict(X_test)
```

In [56]:
```
# mètriques
accuracy_lin_svm = accuracy_score(y_test, y_pred_lin_svm)
F1_score_lin_svm = f1_score(y_test, y_pred_lin_svm, average = "weighted")
print("L'accuracy del model linial SVM és de:", accuracy_lin_svm*100, "%")
print("La F1 del model linial SVM és de:", np.round(F1_score_lin_svm*100, 2), "%")
```

```
L'accuracy del model linial SVM és de: 68.75 %
La F1 del model linial SVM és de: 69.13 %
```

In [57]:
```
# matriu de confusió
lin_svm_matrix = confusion_matrix(y_test, y_pred_lin_svm)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(lin_svm_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



5.5 REGRESSIÓ LOGISTICA

In [58]:
```
# definim model i graella per a determinar millors paràmetres
log_reg = linear_model.LogisticRegression(max_iter = 10000)

param_grid = {"C": [0.01, 0.05, 0.1, 1, 10], "solver": ["newton-cg", "lbfgs", "liblinear", "sag", "saga"]}

grid_search_log_reg= model_selection.GridSearchCV(log_reg, param_grid=param_grid, cv=4)
grid_search_log_reg.fit(X_train, y_train)

means= grid_search_log_reg.cv_results_["mean_test_score"]
stds= grid_search_log_reg.cv_results_["std_test_score"]
params= grid_search_log_reg.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres són: {}".format(grid_search_log_reg.best_params_))
log_reg_C = grid_search_log_reg.best_params_["C"]
log_reg_solver = grid_search_log_reg.best_params_["solver"]
```

```
Precisió mitjana: 68.44 +/- 7.15 amb paràmetres {'C': 0.01, 'solver': 'newton-cg'}
Precisió mitjana: 68.44 +/- 7.15 amb paràmetres {'C': 0.01, 'solver': 'lbfgs'}
Precisió mitjana: 68.44 +/- 7.15 amb paràmetres {'C': 0.01, 'solver': 'liblinear'}
Precisió mitjana: 68.44 +/- 7.15 amb paràmetres {'C': 0.01, 'solver': 'sag'}
Precisió mitjana: 68.44 +/- 7.15 amb paràmetres {'C': 0.01, 'solver': 'saga'}
Precisió mitjana: 62.08 +/- 9.37 amb paràmetres {'C': 0.05, 'solver': 'newton-cg'}
Precisió mitjana: 62.08 +/- 9.37 amb paràmetres {'C': 0.05, 'solver': 'lbfgs'}
Precisió mitjana: 60.42 +/- 7.65 amb paràmetres {'C': 0.05, 'solver': 'liblinear'}
Precisió mitjana: 60.42 +/- 7.65 amb paràmetres {'C': 0.05, 'solver': 'sag'}
Precisió mitjana: 60.42 +/- 7.65 amb paràmetres {'C': 0.05, 'solver': 'saga'}
Precisió mitjana: 58.96 +/- 12.39 amb paràmetres {'C': 0.1, 'solver': 'newton-cg'}
Precisió mitjana: 58.96 +/- 12.39 amb paràmetres {'C': 0.1, 'solver': 'lbfgs'}
Precisió mitjana: 58.96 +/- 12.39 amb paràmetres {'C': 0.1, 'solver': 'liblinear'}
Precisió mitjana: 58.96 +/- 12.39 amb paràmetres {'C': 0.1, 'solver': 'sag'}
Precisió mitjana: 58.96 +/- 12.39 amb paràmetres {'C': 0.1, 'solver': 'saga'}
Precisió mitjana: 57.60 +/- 17.54 amb paràmetres {'C': 1, 'solver': 'newton-cg'}
Precisió mitjana: 57.60 +/- 17.54 amb paràmetres {'C': 1, 'solver': 'lbfgs'}
Precisió mitjana: 55.83 +/- 10.10 amb paràmetres {'C': 1, 'solver': 'liblinear'}
Precisió mitjana: 54.37 +/- 15.65 amb paràmetres {'C': 1, 'solver': 'sag'}
Precisió mitjana: 55.83 +/- 11.03 amb paràmetres {'C': 1, 'solver': 'saga'}
Precisió mitjana: 51.15 +/- 18.44 amb paràmetres {'C': 10, 'solver': 'newton-cg'}
Precisió mitjana: 51.15 +/- 18.44 amb paràmetres {'C': 10, 'solver': 'lbfgs'}
Precisió mitjana: 49.48 +/- 12.53 amb paràmetres {'C': 10, 'solver': 'liblinear'}
Precisió mitjana: 55.94 +/- 14.12 amb paràmetres {'C': 10, 'solver': 'sag'}
Precisió mitjana: 54.37 +/- 15.65 amb paràmetres {'C': 10, 'solver': 'saga'}
Els millors paràmetres són: {'C': 0.01, 'solver': 'newton-cg'}
```

In [59]:
```
# model amb els millors paràmetres
model_log_reg = linear_model.LogisticRegression(max_iter = 10000,C=log_reg_C, solver=log_reg_solver )
model_log_reg.fit(X_train, y_train)
```

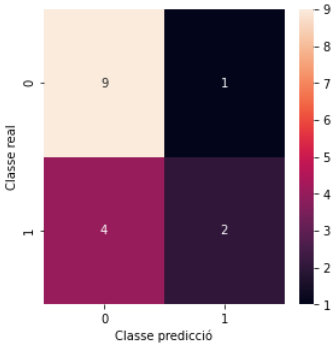
Out[59]:
```
LogisticRegression(C=0.01, max_iter=10000, solver='newton-cg')
```

In [60]:
```
# prediccions
y_pred_log_reg = model_log_reg.predict(X_test)
```

In [61]:
```
# mètriques
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
F1_score_log_reg = f1_score(y_test, y_pred_log_reg, average = "weighted")
print("L'accuracy del model de regressió logistica és de:", accuracy_log_reg*100, "%")
print("La F1 del model de regressió logistica és de:", np.round(F1_score_log_reg*100, 2), "%")
```

```
L'accuracy del model de regressió logistica és de: 68.75 %
La F1 del model de regressió logistica és de: 65.58 %
```

In [62]:
```
# matriu de confusió
log_reg_matrix = confusion_matrix(y_test, y_pred_log_reg)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(log_reg_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



PREPARATIUS PER A ETAPES POSTERIORS

In [63]:
```
# calcul de les prediccions i mètriques per al joc de dades complert
```

In [64]:
```
y_pred_knn_cascad = model_knn.predict(X)
accuracy_knn_cascad = accuracy_score(y, y_pred_knn_cascad)
F1_score_knn_cascad = f1_score(y, y_pred_knn_cascad, average = "weighted")
```

In [65]:
```
np.round(accuracy_knn_cascad*100, 2)
```

Out[65]: 70.89

In [66]:
```
np.round(F1_score_knn_cascad*100, 2)
```

Out[66]: 61.04

In [67]:
```
y_pred_lin_svm_cascad = model_lin_svm.predict(X)
accuracy_lin_svm_cascad = accuracy_score(y, y_pred_lin_svm_cascad)
F1_score_lin_svm_cascad = f1_score(y, y_pred_lin_svm_cascad, average = "weighted")
```

In [68]:
```
np.round(accuracy_lin_svm_cascad*100, 2)
```

Out[68]: 82.28

In [69]:
```
np.round(F1_score_lin_svm_cascad*100, 2)
```

Out[69]: 82.07

In [70]:
```
y_pred_log_reg_cascad = model_log_reg.predict(X)
accuracy_log_reg_cascad = accuracy_score(y, y_pred_log_reg_cascad)
F1_score_log_reg_cascad = f1_score(y, y_pred_log_reg_cascad, average = "weighted")
```

In [71]:
```
np.round(accuracy_log_reg_cascad*100, 2)
```

Out[71]: 75.95

In [72]:
```
np.round(F1_score_log_reg_cascad*100, 2)
```

Out[72]: 71.39

In [73]:
```
y_pred_tree_cascad = model_tree.predict(X)
accuracy_tree_cascad = accuracy_score(y, y_pred_tree_cascad)
F1_score_tree_cascad = f1_score(y, y_pred_tree_cascad, average = "weighted")
```

In [74]:
```
np.round(accuracy_tree_cascad*100, 2)
```

Out[74]: 86.08

In [75]:
```
np.round(F1_score_tree_cascad*100, 2)
```

Out[75]: 85.37

In [76]:
```
y_pred_rf_cascad = model_rf.predict(X)
accuracy_rf_cascad = accuracy_score(y, y_pred_rf_cascad)
F1_score_rf_cascad = f1_score(y, y_pred_rf_cascad, average = "weighted")
```

In [77]:
```
np.round(accuracy_rf_cascad*100, 2)
```

Out[77]: 84.81

In [78]:
```
np.round(F1_score_rf_cascad*100, 2)
```

Out[78]: 83.56

In [79]:
```
y_pred_svm_cascad = model_svm_linial.predict(X)
accuracy_svm_cascad = accuracy_score(y, y_pred_svm_cascad)
F1_score_svm_cascad = f1_score(y, y_pred_svm_cascad, average = "weighted")
```

In [80]:
```
np.round(accuracy_svm_cascad*100, 2)
```

Out[80]: 83.54

In [81]:
```
np.round(F1_score_svm_cascad*100, 2)
```

Out[81]: 83.24

In [82]:
```
y_pred_svm_rad_cascad = model_svm_rad.predict(X)
accuracy_svm_rad_cascad = accuracy_score(y, y_pred_svm_rad_cascad)
F1_score_svm_rad_cascad = f1_score(y, y_pred_svm_rad_cascad, average = "weighted")
```

In [83]:
```
np.round(accuracy_svm_rad_cascad*100, 2)
```

Out[83]: 68.35

In [84]:
```
np.round(F1_score_svm_rad_cascad*100, 2)
```

Out[84]: 55.51

In [85]:
```
y_pred_svm_pol_cascad = model_svm_pol.predict(X)
accuracy_svm_pol_cascad = accuracy_score(y, y_pred_svm_pol_cascad)
F1_score_svm_pol_cascad = f1_score(y, y_pred_svm_pol_cascad, average = "weighted")
```

In [86]:
```
np.round(accuracy_svm_pol_cascad*100, 2)
```

Out[86]: 79.75

In [87]:
```
np.round(F1_score_svm_pol_cascad*100, 2)
```

Out[87]: 76.97

In [88]:
```
y_pred_svm_sig_cascad = model_svm_sig.predict(X)
accuracy_svm_sig_cascad = accuracy_score(y, y_pred_svm_sig_cascad)
F1_score_svm_sig_cascad = f1_score(y, y_pred_svm_sig_cascad, average = "weighted")
```

In [89]:
```
np.round(accuracy_svm_sig_cascad*100, 2)
```

Out[89]: 68.35

In [90]:
```
np.round(F1_score_svm_sig_cascad*100, 2)
```

Out[90]: 55.51

In [91]:
```
# gravació de variables a l'entorn

%store y_pred_knn
%store accuracy_knn
%store F1_score_knn

%store y_pred_tree
%store accuracy_tree
%store F1_score_tree

%store y_pred_rf
%store accuracy_rf
%store F1_score_rf

%store y_pred_svm
%store accuracy_svm
%store F1_score_svm

%store y_pred_svm_rad
%store accuracy_svm_rad
%store F1_score_svm_rad

%store y_pred_svm_pol
%store accuracy_svm_pol
%store F1_score_svm_pol

%store y_pred_svm_sig
%store accuracy_svm_sig
%store F1_score_svm_sig

%store y_pred_lin_svm
%store accuracy_lin_svm
%store F1_score_lin_svm

%store y_pred_log_reg
%store accuracy_log_reg
%store F1_score_log_reg

%store y_pred_knn_cascad
%store accuracy_knn_cascad
%store F1_score_knn_cascad

%store y_pred_lin_svm_cascad
```

```
%store accuracy_lin_svm_cascad
%store F1_score_lin_svm_cascad

%store y_pred_log_reg_cascad
%store accuracy_log_reg_cascad
%store F1_score_log_reg_cascad
```

```
Stored 'y_pred_knn' (ndarray)
Stored 'accuracy_knn' (float64)
Stored 'F1_score_knn' (float64)
Stored 'y_pred_tree' (ndarray)
Stored 'accuracy_tree' (float64)
Stored 'F1_score_tree' (float64)
Stored 'y_pred_rf' (ndarray)
Stored 'accuracy_rf' (float64)
Stored 'F1_score_rf' (float64)
Stored 'y_pred_svm' (ndarray)
Stored 'accuracy_svm' (float64)
Stored 'F1_score_svm' (float64)
Stored 'y_pred_svm_rad' (ndarray)
Stored 'accuracy_svm_rad' (float64)
Stored 'F1_score_svm_rad' (float64)
Stored 'y_pred_svm_pol' (ndarray)
Stored 'accuracy_svm_pol' (float64)
Stored 'F1_score_svm_pol' (float64)
Stored 'y_pred_svm_sig' (ndarray)
Stored 'accuracy_svm_sig' (float64)
Stored 'F1_score_svm_sig' (float64)
Stored 'y_pred_lin_svm' (ndarray)
Stored 'accuracy_lin_svm' (float64)
Stored 'F1_score_lin_svm' (float64)
Stored 'y_pred_log_reg' (ndarray)
Stored 'accuracy_log_reg' (float64)
Stored 'F1_score_log_reg' (float64)
Stored 'y_pred_knn_cascad' (ndarray)
Stored 'accuracy_knn_cascad' (float64)
Stored 'F1_score_knn_cascad' (float64)
Stored 'y_pred_lin_svm_cascad' (ndarray)
Stored 'accuracy_lin_svm_cascad' (float64)
Stored 'F1_score_lin_svm_cascad' (float64)
Stored 'y_pred_log_reg_cascad' (ndarray)
Stored 'accuracy_log_reg_cascad' (float64)
Stored 'F1_score_log_reg_cascad' (float64)
```

ANNEX 7: CODI PYTHON: ANÀLISI PRÈVIA A LA XARXA NEURONAL

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler, RobustScaler

from sklearn import linear_model, svm
from sklearn.svm import SVC, LinearSVC
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score, confusion_matrix, accuracy_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam, SGD, Adadelta, Adagrad

from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.callbacks import TensorBoard
from tensorflow.python.keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.metrics import Accuracy, AUC, TruePositives, FalsePositives, TrueNegatives, FalseNegatives

import tensorflow
import tensorflow_addons as tfa

from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import cross_val_score
import keras.backend as K

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
Using TensorFlow backend.
```

In [2]:
```
# recuperació del joc de dades preparat per a classificacio
data_logic = pd.read_pickle("data_logic.pkl")
```

6. XARXA NEURONAL

In [3]:
```
# Carreguem les dades

classes = ['0','1']
parametres = data_logic.columns.values[:-1]
X = data_logic.iloc[:, :-1]
y = data_logic["B_o_V_cod"]


# One hot encoding
enc = OneHotEncoder()
Y = enc.fit_transform(y[:, np.newaxis]).toarray()

# Estandaritzem les dades (aquest pas és important per a la convergència de la xarxa neuronal)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Dividim les dades en train/test
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X_scaled, Y, test_size=0.2, random_state= 27)
n_features = X.shape[1]
n_classes = Y.shape[1]
```

In [4]:
```
#numero de 1 en test
np.count_nonzero(Y_test[:,0] == 1)
```

Out[4]: 10

In [5]:
```
#numero de 0 en test
np.count_nonzero(Y_test[:,1] == 1)
```

Out[5]: 6

In [6]:
```
# funció per crear model
def neural_model(capes,  opt, lr):

    model = Sequential()
    model.add(Dense(8, activation='relu', input_dim=n_features))

    model.add(Dense(2, activation='softmax'))
```

```
    model.compile(loss='categorical_crossentropy', optimizer=opt(lr=lr), metrics=['accuracy', tfa.metrics.F1Score(num_classes=2, average= '

    return model
```

In [7]:
```
# funció per a fer una graella d'entrenament

def train_grid(ns, opts, lrs):
    columnes = {"Neurones": [], "Optimitzador": [], "Velocitat aprenentatge": [], "Accuracy": [], "F1": [], "val_Accuracy": [],"val_F1": []

    for n in ns:
        for optimitzador in opts:
            for lr in lrs:
                columnes['Neurones'].append(n)
                model = neural_model(capes= n, opt= optimitzador, lr=lr)
                model.fit(X_train, Y_train, epochs=50, batch_size = 1, validation_data = (X_test, Y_test), verbose=0)
                columnes['Optimitzador'].append(optimitzador)
                columnes['Velocitat aprenentatge'].append(lr)
                columnes['Accuracy'].append(max(model.history.history['accuracy']))
                columnes['val_Accuracy'].append(max(model.history.history['val_accuracy']))
                columnes['F1'].append(max(model.history.history['f1_score']))
                columnes['val_F1'].append(max(model.history.history['val_f1_score']))
    return pd.DataFrame(columnes)
```

In [8]:
```
# parametres inicials de la graella

#ns= [2,4,6,8]
#lrs = [0,00005, 0.0001, 0.0005, 0.001, 0.005, 0.01]
#opts = [Adam, SGD, Adadelta, Adagrad]
#grid_per_lr_i_optim = train_grid(ns, opts, lrs)
```

In [9]:
```
# reducció de valors dels paràmetres
ns= [4,8]
lrs = [0.0001, 0.0005, 0.001, 0.005]
opts = [Adam, SGD, Adadelta, Adagrad]
grid_per_lr_i_optim = train_grid(ns, opts, lrs)
```

In [10]:
```
# adaptació per a millorar la visualització
grid_per_lr_i_optim['Optimitzador']= ["Adam","Adam","Adam","Adam","SGD", "SGD","SGD", "SGD", "Adadelta","Adadelta","Adadelta","Adadelta","A
```

In [11]:
```
# resultats
grid_per_lr_i_optim
```

Out[11]:

	Neurones	Optimitzador	Velocitat aprenentatge	Accuracy	F1	val_Accuracy	val_F1
0	4	Adam	0.0001	0.619048	0.580931	0.6875	0.686275
1	4	Adam	0.0005	0.888889	0.856305	0.8125	0.768116
2	4	Adam	0.0010	0.984127	0.980869	0.6250	0.600000
3	4	Adam	0.0050	1.000000	1.000000	0.8125	0.768116
4	4	SGD	0.0001	0.603175	0.558700	0.7500	0.733333
5	4	SGD	0.0005	0.777778	0.678571	0.9375	0.930736
6	4	SGD	0.0010	0.809524	0.724490	0.5625	0.458937
7	4	SGD	0.0050	0.984127	0.980869	0.6875	0.653680
8	4	Adadelta	0.0001	0.444444	0.382180	0.3750	0.365079
9	4	Adadelta	0.0005	0.523810	0.508836	0.6875	0.686275
10	4	Adadelta	0.0010	0.571429	0.498674	0.6875	0.653680
11	4	Adadelta	0.0050	0.428571	0.424949	0.5625	0.560784
12	4	Adagrad	0.0001	0.619048	0.497340	0.5625	0.360000
13	4	Adagrad	0.0005	0.714286	0.465094	0.6250	0.384615
14	4	Adagrad	0.0010	0.634921	0.545055	0.6250	0.500000
15	4	Adagrad	0.0050	0.746032	0.664894	0.7500	0.709091
16	8	Adam	0.0001	0.746032	0.664894	0.6875	0.653680
17	8	Adam	0.0005	0.936508	0.919437	0.8125	0.811765
18	8	Adam	0.0010	0.968254	0.961111	0.8750	0.854545
19	8	Adam	0.0050	1.000000	1.000000	0.6250	0.546559
20	8	SGD	0.0001	0.634921	0.572944	0.8750	0.866667
21	8	SGD	0.0005	0.730159	0.636333	0.5625	0.515152
22	8	SGD	0.0010	0.809524	0.709231	0.5625	0.360000
23	8	SGD	0.0050	0.984127	0.980869	0.6875	0.653680
24	8	Adadelta	0.0001	0.380952	0.338271	0.3750	0.333333
25	8	Adadelta	0.0005	0.539683	0.532138	0.2500	0.250000
26	8	Adadelta	0.0010	0.412698	0.412698	0.4375	0.417004
27	8	Adadelta	0.0050	0.539683	0.507945	0.7500	0.733333
28	8	Adagrad	0.0001	0.619048	0.418462	0.7500	0.666667
29	8	Adagrad	0.0005	0.666667	0.529684	0.3750	0.272727
30	8	Adagrad	0.0010	0.571429	0.541880	0.6875	0.686275
31	8	Adagrad	0.0050	0.825397	0.774194	0.6875	0.676113

ANNEX 8: CODI PYTHON: XARXA NEURONAL

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler, RobustScaler

from sklearn import linear_model, svm
from sklearn.svm import SVC, LinearSVC
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score, confusion_matrix, accuracy_score, f1_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam, Adagrad, SGD

from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.callbacks import TensorBoard
from tensorflow.python.keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.metrics import Accuracy, AUC, TruePositives, FalsePositives, TrueNegatives, FalseNegatives

import keras.backend as K

import tensorflow
import tensorflow_addons as tfa

from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import cross_val_score

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
Using TensorFlow backend.
```

In [2]:
```
# recuperació del joc de dades preparat per a classificacio
data_logic = pd.read_pickle("./data_logic.pkl")
```

6. XARXA NEURONAL

In [3]:
```
# Carreguem les dades

classes = ['0','1']
parametres = data_logic.columns.values[:-1]
X = data_logic.iloc[:, :-1]
y = data_logic["B_o_V_cod"]


# One hot encoding
enc = OneHotEncoder()
Y = enc.fit_transform(y[:, np.newaxis]).toarray()

# Estandaritzem les dades (aquest pas és important per a la convergència de la xarxa neuronal)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Dividim les dades en train/test
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X_scaled, Y, test_size=0.2, random_state= 27)
n_features = X.shape[1]
n_classes = Y.shape[1]
```

In [4]:
```
#numero de 1 en test
np.count_nonzero(Y_test[:,0] == 1)
```

Out[4]: 10

In [5]:
```
#numero de 0 en test
np.count_nonzero(Y_test[:,1] == 1)
```

Out[5]: 6

In [6]:
```
# Crear el model amb Keras
model_neuronal= Sequential()

model_neuronal.add(Dense(8, activation='relu', input_dim=n_features))

# Capa de sortida
```

```
model_neuronal.add(Dense(2, activation='softmax'))

model_neuronal.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 8)                 280
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 18
=================================================================
Total params: 298
Trainable params: 298
Non-trainable params: 0
_________________________________________________________________
```

6.1 TRES MODELS SEGONS OPTIMITZADOR

In [7]:
```
# creació de 3 models iguals
model_Adam = model_neuronal
model_Adagrad = model_neuronal
model_SGD = model_neuronal
```

In [8]:
```
# Compilació dels models
batch_size = 4
lr= 0.0005
epochs = 300

model_Adam.compile(loss='categorical_crossentropy', optimizer=Adam(lr=lr),
                       metrics=['accuracy', tfa.metrics.F1Score(num_classes=2, average= "macro", threshold= None)])

model_Adagrad.compile(loss='categorical_crossentropy', optimizer=Adagrad(lr=lr),
                       metrics=['accuracy', tfa.metrics.F1Score(num_classes=2, average= "macro", threshold= None)])

model_SGD.compile(loss='categorical_crossentropy', optimizer=SGD(lr=lr),
                       metrics=['accuracy', tfa.metrics.F1Score(num_classes=2, average= "macro", threshold= None)])
```

In [9]:
```
# mecanisme de parada
es = EarlyStopping(
    monitor="val_loss",
    min_delta=0,
    patience=15,
    verbose=1,
    mode="min",
    baseline=None,
    restore_best_weights=True)
```

In [10]:
```
# entreno el model Adam
history_Adam = model_Adam.fit(
    X_train,
    Y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=0,
    validation_data=(X_test, Y_test),
    callbacks=[es]
)
```

```
Restoring model weights from the end of the best epoch.
Epoch 00087: early stopping
```

In [11]:
```
# entreno el model Adagrad
history_Adagrad = model_Adagrad.fit(
    X_train,
    Y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=0,
    validation_data=(X_test, Y_test),
    callbacks=[es]
)
```

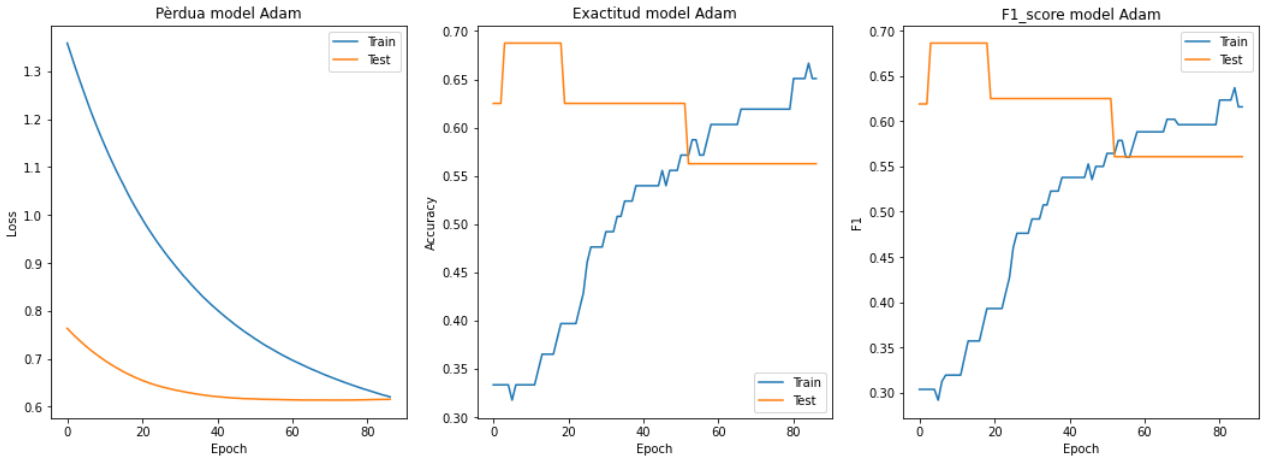
```
Restoring model weights from the end of the best epoch.
Epoch 00016: early stopping
```

In [12]:
```
# entreno el model Adam
history_SGD = model_SGD.fit(
    X_train,
    Y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=0,
    validation_data=(X_test, Y_test),
    callbacks=[es]
)
```

```
Restoring model weights from the end of the best epoch.
Epoch 00017: early stopping
```

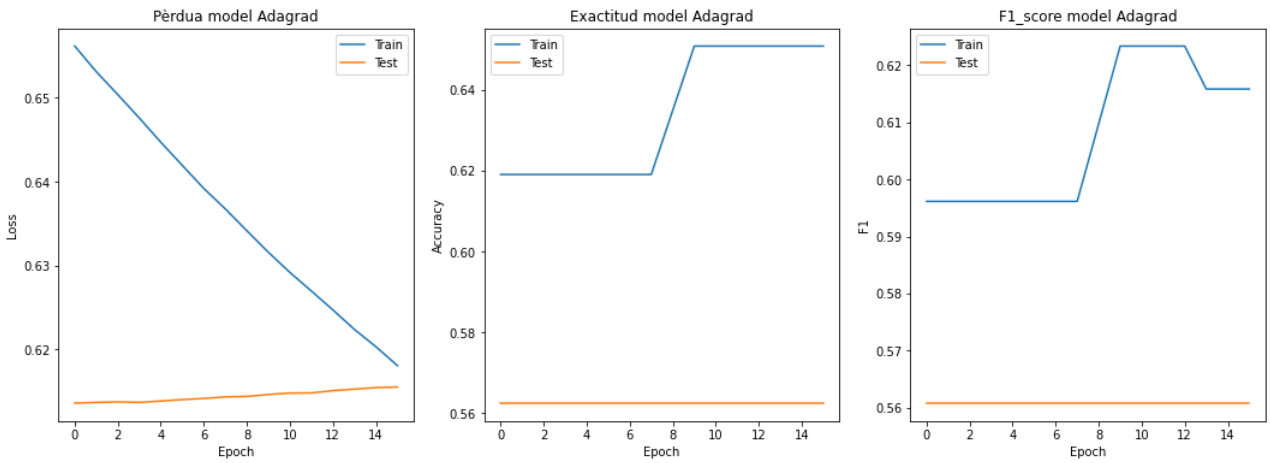
In [13]:
```
# representació gràfica de la història del Adam
fig, ax = plt.subplots(1, 3, figsize=(18, 6))
ax[0].plot(history_Adam.history["loss"])
ax[0].plot(history_Adam.history['val_loss'])
ax[0].set_title('Pèrdua model Adam')
ax[0].set_ylabel('Loss')
ax[0].set_xlabel('Epoch')
ax[0].legend(['Train', 'Test'])
ax[1].plot(history_Adam.history["accuracy"])
ax[1].plot(history_Adam.history['val_accuracy'])
ax[1].set_title('Exactitud model Adam')
ax[1].set_ylabel('Accuracy')
```

```
ax[1].set_xlabel('Epoch')
ax[1].legend(['Train', 'Test'])
ax[2].plot(history_Adam.history["f1_score"])
ax[2].plot(history_Adam.history['val_f1_score'])
ax[2].set_title('F1_score model Adam')
ax[2].set_ylabel('F1')
ax[2].set_xlabel('Epoch')
ax[2].legend(['Train', 'Test'])

plt.show()
```



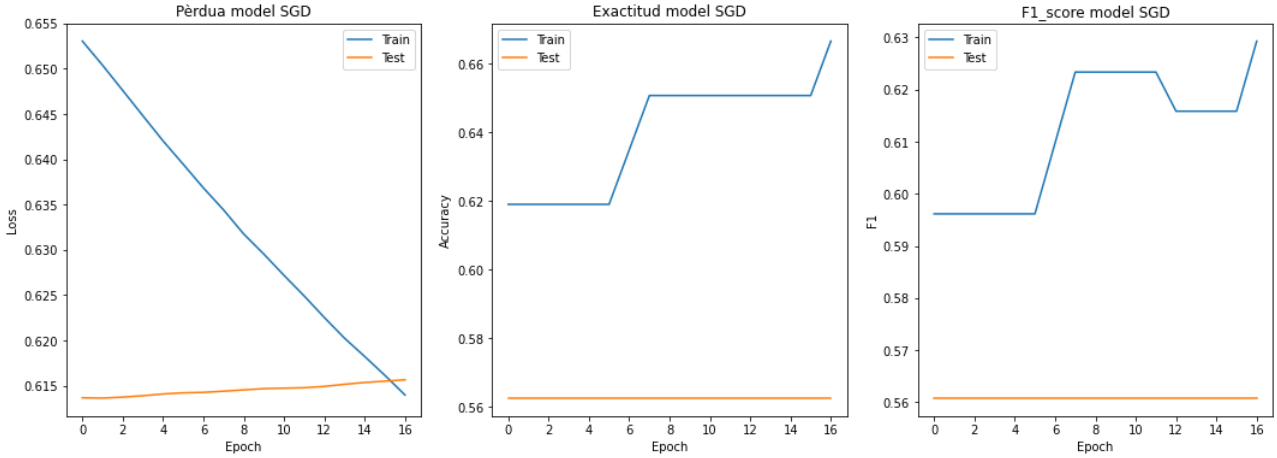
In [14]:

```
# representació gràfica de la història del Adagrad
fig, ax = plt.subplots(1, 3, figsize=(18, 6))
ax[0].plot(history_Adagrad.history["loss"])
ax[0].plot(history_Adagrad.history['val_loss'])
ax[0].set_title('Pèrdua model Adagrad')
ax[0].set_ylabel('Loss')
ax[0].set_xlabel('Epoch')
ax[0].legend(['Train', 'Test'])
ax[1].plot(history_Adagrad.history["accuracy"])
ax[1].plot(history_Adagrad.history['val_accuracy'])
ax[1].set_title('Exactitud model Adagrad')
ax[1].set_ylabel('Accuracy')
ax[1].set_xlabel('Epoch')
ax[1].legend(['Train', 'Test'])
ax[2].plot(history_Adagrad.history["f1_score"])
ax[2].plot(history_Adagrad.history['val_f1_score'])
ax[2].set_title('F1_score model Adagrad')
ax[2].set_ylabel('F1')
ax[2].set_xlabel('Epoch')
ax[2].legend(['Train', 'Test'])

plt.show()
```



In [15]:

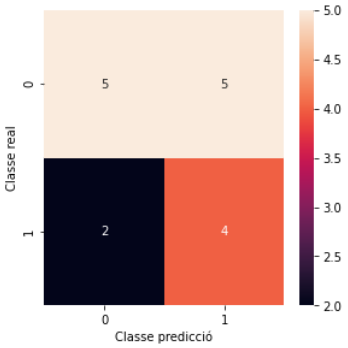
```
# representació gràfica de la història del SGD
fig, ax = plt.subplots(1, 3, figsize=(18, 6))
ax[0].plot(history_SGD.history["loss"])
ax[0].plot(history_SGD.history['val_loss'])
ax[0].set_title('Pèrdua model SGD')
ax[0].set_ylabel('Loss')
ax[0].set_xlabel('Epoch')
ax[0].legend(['Train', 'Test'])
ax[1].plot(history_SGD.history["accuracy"])
ax[1].plot(history_SGD.history['val_accuracy'])
ax[1].set_title('Exactitud model SGD')
ax[1].set_ylabel('Accuracy')
ax[1].set_xlabel('Epoch')
ax[1].legend(['Train', 'Test'])
ax[2].plot(history_SGD.history["f1_score"])
ax[2].plot(history_SGD.history['val_f1_score'])
ax[2].set_title('F1_score model SGD')
ax[2].set_ylabel('F1')
ax[2].set_xlabel('Epoch')
ax[2].legend(['Train', 'Test'])

plt.show()
```



6.2 PREDICCIONS AMB EL MODEL ADAM

In [16]:

```
#prediccions i matriu de confusió
y_pred_Adam = model_Adam.predict(X_test)

y_pred_Adam_class = np.argmax(y_pred_Adam, axis=-1)

Y_test_class = np.argmax(Y_test, axis=-1)

Adam_matrix = confusion_matrix(Y_test_class, y_pred_Adam_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(Adam_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



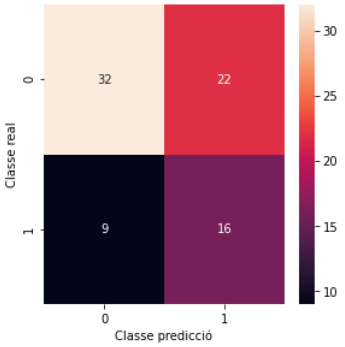
In [17]:

```
# mètriques
accuracy_Adam = accuracy_score(Y_test_class, y_pred_Adam_class)
F1_score_Adam = f1_score(Y_test_class, y_pred_Adam_class, average = "weighted")
print("L'accuracy del model Adam és de:", accuracy_Adam*100, "%")
print("La F1 del model Adam és de:", np.round(F1_score_Adam*100, 2), "%")
```

```
L'accuracy del model Adam és de: 56.25 %
La F1 del model Adam és de: 56.76 %
```

In [18]:

```
# valors per a models de combinació posteriors
y_pred_Adam_cascad = model_Adam.predict(X_scaled)

y_pred_Adam_cascad_class = np.argmax(y_pred_Adam_cascad, axis=-1)

Y_class = np.argmax(Y, axis=-1)

Adam_matrix_cascad = confusion_matrix(Y_class, y_pred_Adam_cascad_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(Adam_matrix_cascad, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```

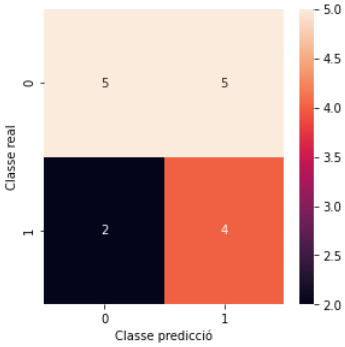


In [19]:
```
# mètriques per a models de combinació
accuracy_Adam_cascad = accuracy_score(Y_class, y_pred_Adam_cascad_class)
F1_score_Adam_cascad = f1_score(Y_class, y_pred_Adam_cascad_class, average = "weighted")
print("L'accuracy del model Adam és de:", accuracy_Adam_cascad*100, "%")
print("La F1 del model Adam és de:", np.round(F1_score_Adam_cascad*100, 2), "%")
```

```
L'accuracy del model Adam és de: 60.75949367088608 %
La F1 del model Adam és de: 62.12 %
```

6.3 PREDICCIONS AMB EL MODEL ADAGRAD

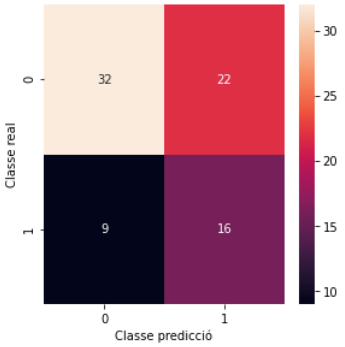
In [20]:
```
#prediccions i matriu de confusió
y_pred_Adagrad = model_Adagrad.predict(X_test)

y_pred_Adagrad_class = np.argmax(y_pred_Adagrad, axis=-1)

Y_test_class = np.argmax(Y_test, axis=-1)

Adagrad_matrix = confusion_matrix(Y_test_class, y_pred_Adagrad_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(Adagrad_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



In [21]:
```
#mètriques
accuracy_Adagrad = accuracy_score(Y_test_class, y_pred_Adagrad_class)
F1_score_Adagrad = f1_score(Y_test_class, y_pred_Adagrad_class, average = "weighted")
print("L'accuracy del model Adagrad és de:", accuracy_Adagrad*100, "%")
print("La F1 del model Adagrad és de:", np.round(F1_score_Adagrad*100, 2), "%")
```

```
L'accuracy del model Adagrad és de: 56.25 %
La F1 del model Adagrad és de: 56.76 %
```

In [22]:
```
# valors per a models posteriors
y_pred_Adagrad_cascad = model_Adagrad.predict(X_scaled)

y_pred_Adagrad_cascad_class = np.argmax(y_pred_Adagrad_cascad, axis=-1)

Y_class = np.argmax(Y, axis=-1)

Adagrad_matrix_cascad = confusion_matrix(Y_class, y_pred_Adagrad_cascad_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(Adagrad_matrix_cascad, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```

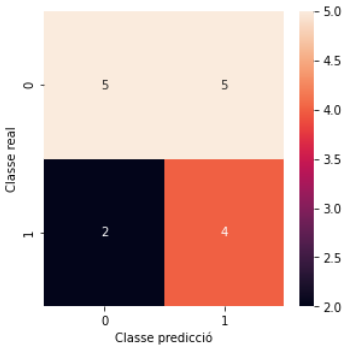


In [23]:
```
# mètriques per a models posteriors
accuracy_Adagrad_cascad = accuracy_score(Y_class, y_pred_Adagrad_cascad_class)
F1_score_Adagrad_cascad = f1_score(Y_class, y_pred_Adagrad_cascad_class, average = "weighted")
print("L'accuracy del model Adagrad és de:", accuracy_Adagrad_cascad*100, "%")
print("La F1 del model Adagrad és de:", np.round(F1_score_Adagrad_cascad*100, 2), "%")
```

```
L'accuracy del model Adagrad és de: 60.75949367088608 %
La F1 del model Adagrad és de: 62.12 %
```

6.4 PREDICCIONS AMB EL MODEL SGD

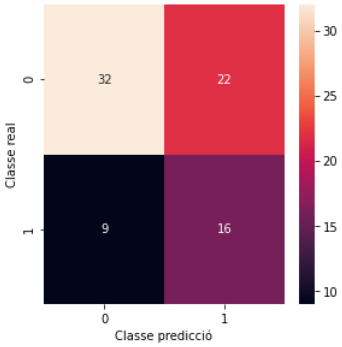
In [24]:
```
#prediccions i matriu de confusió
y_pred_SGD = model_SGD.predict(X_test)

y_pred_SGD_class = np.argmax(y_pred_SGD, axis=-1)

Y_test_class = np.argmax(Y_test, axis=-1)

SGD_matrix = confusion_matrix(Y_test_class, y_pred_SGD_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(SGD_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



In [25]:
```
#mètriques
accuracy_SGD = accuracy_score(Y_test_class, y_pred_SGD_class)
F1_score_SGD = f1_score(Y_test_class, y_pred_SGD_class, average = "weighted")
print("L'accuracy del model SGD és de:", accuracy_SGD*100, "%")
print("La F1 del model SGD és de:", np.round(F1_score_SGD*100, 2), "%")
```

```
L'accuracy del model SGD és de: 56.25 %
La F1 del model SGD és de: 56.76 %
```

In [26]:
```
# valors per a models de combinació posteriors
y_pred_SGD_cascad = model_SGD.predict(X_scaled)

y_pred_SGD_cascad_class = np.argmax(y_pred_SGD_cascad, axis=-1)

Y_class = np.argmax(Y, axis=-1)

SGD_matrix_cascad = confusion_matrix(Y_class, y_pred_SGD_cascad_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(SGD_matrix_cascad, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



In [27]:
```
# mètriques per a models posteriors
accuracy_SGD_cascad = accuracy_score(Y_class, y_pred_SGD_cascad_class)
F1_score_SGD_cascad = f1_score(Y_class, y_pred_SGD_cascad_class, average = "weighted")
print("L'accuracy del model SGD és de:", accuracy_SGD_cascad*100, "%")
print("La F1 del model SGD és de:", np.round(F1_score_SGD_cascad*100, 2), "%")
```

```
L'accuracy del model SGD és de: 60.75949367088608 %
La F1 del model SGD és de: 62.12 %
```

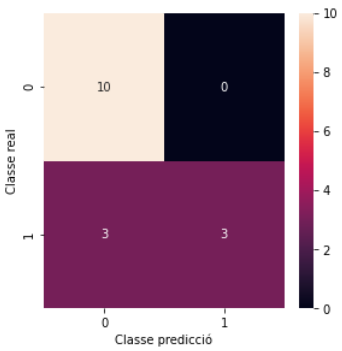
In [28]:
```
# gaurdar els models entrenats i les històries
model_Adam.save("model_Adam_rev.h5")
model_Adagrad.save("model_Adagrad_rev.h5")
model_SGD.save("model_SGD_rev.h5")
import json

with open('history_Adam_rev.json', 'w') as file:
    json.dump(history_Adam.history, file)

with open('history_Adagrad_rev.json', 'w') as file:
    json.dump(history_Adagrad.history, file)

with open('history_SGD_rev.json', 'w') as file:
    json.dump(history_SGD.history, file)
```

6.5 RECUPERACIÓ MILLOR MODEL ANTERIOR

In [29]:
```
###### el millor fins ara

from tensorflow.keras.models import load_model
model_Adam = load_model("model_Adam.h5")
model_Adagrad = load_model("model_Adagrad.h5")
model_SGD = load_model("model_SGD.h5")
```

PREDICCIONS AMB EL MODEL ADAM

In [30]:
```
#prediccions i matriu de confusió
y_pred_Adam = model_Adam.predict(X_test)

y_pred_Adam_class = np.argmax(y_pred_Adam, axis=-1)

Y_test_class = np.argmax(Y_test, axis=-1)

Adam_matrix = confusion_matrix(Y_test_class, y_pred_Adam_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(Adam_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```

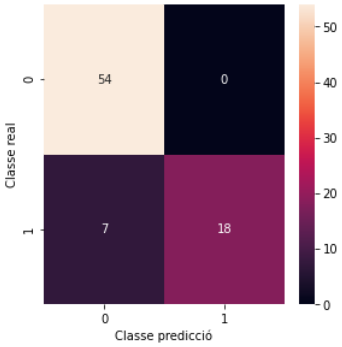


In [31]:
```
# mètriques
accuracy_Adam = accuracy_score(Y_test_class, y_pred_Adam_class)
F1_score_Adam = f1_score(Y_test_class, y_pred_Adam_class, average = "weighted")
print("L'accuracy del model Adam és de:", accuracy_Adam*100, "%")
print("La F1 del model Adam és de:", np.round(F1_score_Adam*100, 2), "%")
```

```
L'accuracy del model Adam és de: 81.25 %
La F1 del model Adam és de: 79.35 %
```

In [32]:
```
# valors per a models posteriors de combinació
```

```
y_pred_Adam_cascad = model_Adam.predict(X_scaled)

y_pred_Adam_cascad_class = np.argmax(y_pred_Adam_cascad, axis=-1)

Y_class = np.argmax(Y, axis=-1)

Adam_matrix_cascad = confusion_matrix(Y_class, y_pred_Adam_cascad_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(Adam_matrix_cascad, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```

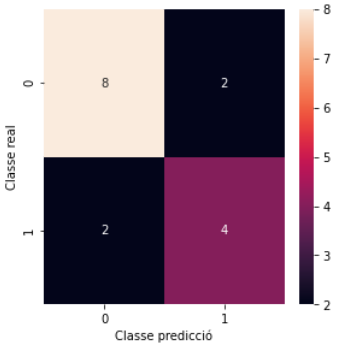


In [33]:
```
# mètriques per models de combinació posteriors
accuracy_Adam_cascad = accuracy_score(Y_class, y_pred_Adam_cascad_class)
F1_score_Adam_cascad = f1_score(Y_class, y_pred_Adam_cascad_class, average = "weighted")
print("L'accuracy del model Adam és de:", accuracy_Adam_cascad*100, "%")
print("La F1 del model Adam és de:", np.round(F1_score_Adam_cascad*100, 2), "%")
```

```
L'accuracy del model Adam és de: 91.13924050632912 %
La F1 del model Adam és de: 90.69 %
```

PREDICCIONS AMB EL MODEL ADAGRAD

In [34]:
```
#prediccions i matriu de confusió
y_pred_Adagrad = model_Adagrad.predict(X_test)

y_pred_Adagrad_class = np.argmax(y_pred_Adagrad, axis=-1)

Y_test_class = np.argmax(Y_test, axis=-1)

Adagrad_matrix = confusion_matrix(Y_test_class, y_pred_Adagrad_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(Adagrad_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```

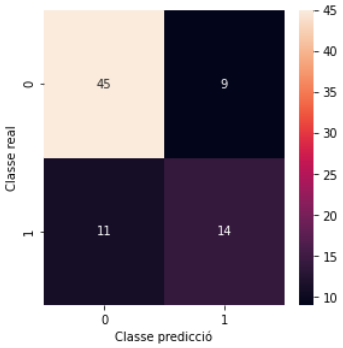


In [35]:
```
#mètriques
accuracy_Adagrad = accuracy_score(Y_test_class, y_pred_Adagrad_class)
F1_score_Adagrad = f1_score(Y_test_class, y_pred_Adagrad_class, average = "weighted")
print("L'accuracy del model Adagrad és de:", accuracy_Adagrad*100, "%")
print("La F1 del model Adagrad és de:", np.round(F1_score_Adagrad*100, 2), "%")
```

```
L'accuracy del model Adagrad és de: 75.0 %
La F1 del model Adagrad és de: 75.0 %
```

In [36]:
```
# prediccions per a combinació de models posteriors
y_pred_Adagrad_cascad = model_Adagrad.predict(X_scaled)

y_pred_Adagrad_cascad_class = np.argmax(y_pred_Adagrad_cascad, axis=-1)

Y_class = np.argmax(Y, axis=-1)

Adagrad_matrix_cascad = confusion_matrix(Y_class, y_pred_Adagrad_cascad_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']
```

```
figure = plt.figure(figsize=(4, 4))
sns.heatmap(Adagrad_matrix_cascad, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```

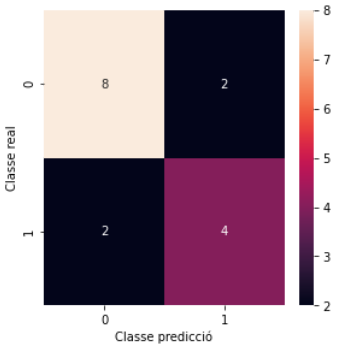


In [37]:
```
#mètriques
accuracy_Adagrad_cascad = accuracy_score(Y_class, y_pred_Adagrad_cascad_class)
F1_score_Adagrad_cascad = f1_score(Y_class, y_pred_Adagrad_cascad_class, average = "weighted")
print("L'accuracy del model Adagrad és de:", accuracy_Adagrad_cascad*100, "%")
print("La F1 del model Adagrad és de:", np.round(F1_score_Adagrad_cascad*100, 2), "%")
```

```
L'accuracy del model Adagrad és de: 74.68354430379746 %
La F1 del model Adagrad és de: 74.39 %
```

PREDICCIONS AMB EL MODEL SGD

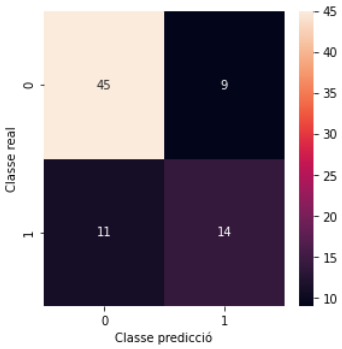
In [38]:
```
#prediccions i matriu de confusió
y_pred_SGD = model_SGD.predict(X_test)

y_pred_SGD_class = np.argmax(y_pred_SGD, axis=-1)

Y_test_class = np.argmax(Y_test, axis=-1)

SGD_matrix = confusion_matrix(Y_test_class, y_pred_SGD_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(SGD_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



In [39]:
```
#mètriques
accuracy_SGD = accuracy_score(Y_test_class, y_pred_SGD_class)
F1_score_SGD = f1_score(Y_test_class, y_pred_SGD_class, average = "weighted")
print("L'accuracy del model SGD és de:", accuracy_SGD*100, "%")
print("La F1 del model SGD és de:", np.round(F1_score_SGD*100, 2), "%")
```

```
L'accuracy del model SGD és de: 75.0 %
La F1 del model SGD és de: 75.0 %
```

In [40]:
```
# prediccions per a models de combinacio posteriors
y_pred_SGD_cascad = model_SGD.predict(X_scaled)

y_pred_SGD_cascad_class = np.argmax(y_pred_SGD_cascad, axis=-1)

Y_class = np.argmax(Y, axis=-1)

SGD_matrix_cascad = confusion_matrix(Y_class, y_pred_SGD_cascad_class)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(SGD_matrix_cascad, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



In [41]:
```
#mètriques per a models combinatius posteriors
accuracy_SGD_cascad = accuracy_score(Y_class, y_pred_SGD_cascad_class)
F1_score_SGD_cascad = f1_score(Y_class, y_pred_SGD_cascad_class, average = "weighted")
print("L'accuracy del model SGD és de:", accuracy_SGD_cascad*100, "%")
print("La F1 del model SGD és de:", np.round(F1_score_SGD_cascad*100, 2), "%")
```

```
L'accuracy del model SGD és de: 74.68354430379746 %
La F1 del model SGD és de: 74.39 %
```

In [42]:
```
# guardem valors per a models posteriors

%store y_pred_Adam
%store y_pred_Adam_cascad

%store y_pred_SGD
%store y_pred_SGD_cascad

%store y_pred_Adagrad
%store y_pred_Adagrad_cascad
```

```
Stored 'y_pred_Adam' (ndarray)
Stored 'y_pred_Adam_cascad' (ndarray)
Stored 'y_pred_SGD' (ndarray)
Stored 'y_pred_SGD_cascad' (ndarray)
Stored 'y_pred_Adagrad' (ndarray)
Stored 'y_pred_Adagrad_cascad' (ndarray)
```

In [ ]:

ANNEX 9: CODI PYTHON: COMBINACIÓ DE MODELS

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler, RobustScaler

from sklearn import linear_model, svm
from sklearn.svm import SVC, LinearSVC
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score, confusion_matrix, accuracy_score, f1_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.pipeline import make_pipeline
```

In [2]:
```
# recuperació del joc de dades preparat per a classificacio

data_logic = pd.read_pickle("./data_logic.pkl")
```

7. AGRUPACIO DE MODELS

In [3]:
```
# separem les dades de la variable objectiu i generem conjunt d'entrenament i de test
X = data_logic.iloc[:, :-1]
y = data_logic["B_o_V_cod"]
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2, random_state=27)
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
```

```
X_train: (63, 34)
X_test: (16, 34)
```

In [4]:
```
# comprovació de la proporció dels conjunts
y_train.value_counts()
```

Out[4]:
```
0    44
1    19
Name: B_o_V_cod, dtype: int64
```

In [5]:
```
y_test.value_counts()
```

Out[5]:
```
0    10
1     6
Name: B_o_V_cod, dtype: int64
```

7.1 BOOSTING (GRADIENT BOOSTING)

In [6]:
```
# definim model i graella per a determinar millors paràmetres

gb = GradientBoostingClassifier()

param_grid = {"max_depth": range(2, 10), "n_estimators": [10, 50, 100, 200, 300]}

grid_search_gb = model_selection.GridSearchCV(gb, param_grid=param_grid, cv=4)
grid_search_gb.fit(X_train, y_train)

means = grid_search_gb.cv_results_["mean_test_score"]
stds = grid_search_gb.cv_results_["std_test_score"]
params = grid_search_gb.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres per al Gradient Boosting són: {}".format(grid_search_gb.best_params_))
gb_max_depth = grid_search_gb.best_params_["max_depth"]
gb_n_estimators = grid_search_gb.best_params_["n_estimators"]
```

```
Precisió mitjana: 81.04 +/- 6.05 amb paràmetres {'max_depth': 2, 'n_estimators': 10}
Precisió mitjana: 79.48 +/- 7.92 amb paràmetres {'max_depth': 2, 'n_estimators': 50}
Precisió mitjana: 81.04 +/- 6.05 amb paràmetres {'max_depth': 2, 'n_estimators': 100}
Precisió mitjana: 82.60 +/- 5.00 amb paràmetres {'max_depth': 2, 'n_estimators': 200}
Precisió mitjana: 81.04 +/- 6.05 amb paràmetres {'max_depth': 2, 'n_estimators': 300}
Precisió mitjana: 79.38 +/- 5.12 amb paràmetres {'max_depth': 3, 'n_estimators': 10}
Precisió mitjana: 79.48 +/- 7.92 amb paràmetres {'max_depth': 3, 'n_estimators': 50}
Precisió mitjana: 79.48 +/- 7.92 amb paràmetres {'max_depth': 3, 'n_estimators': 100}
Precisió mitjana: 82.60 +/- 5.00 amb paràmetres {'max_depth': 3, 'n_estimators': 200}
Precisió mitjana: 81.04 +/- 6.05 amb paràmetres {'max_depth': 3, 'n_estimators': 300}
Precisió mitjana: 84.17 +/- 5.30 amb paràmetres {'max_depth': 4, 'n_estimators': 10}
Precisió mitjana: 79.38 +/- 9.21 amb paràmetres {'max_depth': 4, 'n_estimators': 50}
Precisió mitjana: 74.79 +/- 7.32 amb paràmetres {'max_depth': 4, 'n_estimators': 100}
Precisió mitjana: 71.67 +/- 10.05 amb paràmetres {'max_depth': 4, 'n_estimators': 200}
Precisió mitjana: 71.67 +/- 10.05 amb paràmetres {'max_depth': 4, 'n_estimators': 300}
Precisió mitjana: 74.69 +/- 14.53 amb paràmetres {'max_depth': 5, 'n_estimators': 10}
Precisió mitjana: 73.23 +/- 14.03 amb paràmetres {'max_depth': 5, 'n_estimators': 50}
Precisió mitjana: 71.46 +/- 9.26 amb paràmetres {'max_depth': 5, 'n_estimators': 100}
```

```
Precisió mitjana: 68.12 +/- 10.33 amb paràmetres {'max_depth': 5, 'n_estimators': 200}
Precisió mitjana: 74.90 +/- 15.81 amb paràmetres {'max_depth': 5, 'n_estimators': 300}
Precisió mitjana: 74.79 +/- 15.14 amb paràmetres {'max_depth': 6, 'n_estimators': 10}
Precisió mitjana: 73.12 +/- 10.02 amb paràmetres {'max_depth': 6, 'n_estimators': 50}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 6, 'n_estimators': 100}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 6, 'n_estimators': 200}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 6, 'n_estimators': 300}
Precisió mitjana: 76.46 +/- 14.79 amb paràmetres {'max_depth': 7, 'n_estimators': 10}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 7, 'n_estimators': 50}
Precisió mitjana: 73.12 +/- 10.02 amb paràmetres {'max_depth': 7, 'n_estimators': 100}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 7, 'n_estimators': 200}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 7, 'n_estimators': 300}
Precisió mitjana: 73.23 +/- 10.90 amb paràmetres {'max_depth': 8, 'n_estimators': 10}
Precisió mitjana: 73.12 +/- 10.02 amb paràmetres {'max_depth': 8, 'n_estimators': 50}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 8, 'n_estimators': 100}
Precisió mitjana: 73.12 +/- 10.02 amb paràmetres {'max_depth': 8, 'n_estimators': 200}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 8, 'n_estimators': 300}
Precisió mitjana: 74.69 +/- 11.54 amb paràmetres {'max_depth': 9, 'n_estimators': 10}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 9, 'n_estimators': 50}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 9, 'n_estimators': 100}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 9, 'n_estimators': 200}
Precisió mitjana: 71.56 +/- 12.67 amb paràmetres {'max_depth': 9, 'n_estimators': 300}
Els millors paràmetres per al Gradient Boosting són: {'max_depth': 4, 'n_estimators': 10}
```

In [7]:
```
# model amb millors paràmetres
model_gb = GradientBoostingClassifier(max_depth=gb_max_depth, n_estimators=gb_n_estimators)
model_gb.fit(X_train, y_train)
```

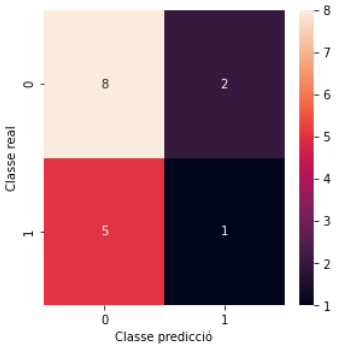
Out[7]:
```
GradientBoostingClassifier(max_depth=4, n_estimators=10)
```

In [8]:
```
# prediccions
y_pred_gb = model_gb.predict(X_test)
```

In [9]:
```
# mètriques
accuracy_gb = accuracy_score(y_test, y_pred_gb)
F1_score_gb = f1_score(y_test, y_pred_gb, average = "weighted")
print("L'accuracy del model Gradient Boosting és de:", accuracy_gb*100, "%")
print("La F1 del model Gradient Boosting és de:", np.round(F1_score_gb*100, 2), "%")
```

```
L'accuracy del model Gradient Boosting és de: 56.25 %
La F1 del model Gradient Boosting és de: 51.81 %
```

In [10]:
```
# matriu de confusió
gb_matrix = confusion_matrix(y_test, y_pred_gb)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(gb_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



In [11]:
```
# calcul de les prediccions i mètriques per al joc de dades complert
y_pred_gb_cascad = model_gb.predict(X)
accuracy_gb_cascad = accuracy_score(y, y_pred_gb_cascad)
F1_score_gb_cascad = f1_score(y, y_pred_gb_cascad, average = "weighted")
print("L'accuracy del model Gradient Boosting és de:", accuracy_gb_cascad*100, "%")
print("La F1 del model Gradient Boosting és de:", np.round(F1_score_gb_cascad*100, 2), "%")
```

```
L'accuracy del model Gradient Boosting és de: 86.07594936708861 %
La F1 del model Gradient Boosting és de: 85.37 %
```

7.2 STACKING

In [12]:
```
# recupero resultats models
%store -r y_pred_knn

%store -r y_pred_tree

%store -r y_pred_rf

%store -r y_pred_svm

%store -r y_pred_svm_rad

%store -r y_pred_svm_sig

%store -r y_pred_svm_pol

%store -r y_pred_lin_svm

%store -r y_pred_log_reg
```

```
%store -r accuracy_knn

%store -r F1_score_knn

%store -r accuracy_tree

%store -r F1_score_tree

%store -r accuracy_rf

%store -r F1_score_rf

%store -r accuracy_svm

%store -r F1_score_svm

%store -r accuracy_svm_rad

%store -r F1_score_svm_rad

%store -r accuracy_svm_sig

%store -r F1_score_svm_sig

%store -r accuracy_svm_pol

%store -r F1_score_svm_pol

%store -r accuracy_lin_svm

%store -r F1_score_lin_svm

%store -r accuracy_log_reg

%store -r F1_score_log_reg
```

In [13]:
```
#resultats obtinguts
```

In [14]:
```
print("model knn\n      accuracy: ", accuracy_knn*100, "%  | F1_score: " , np.round(F1_score_knn*100, 2), "%")
print("model tree\n      accuracy: ", accuracy_tree*100, "%  | F1_score: " , np.round(F1_score_tree*100, 2), "%")
print("model random forest\n      accuracy: ", accuracy_rf*100, "%  | F1_score: " , np.round(F1_score_rf*100, 2), "%")
print("model SVM linial\n      accuracy: ", accuracy_svm*100, "%  | F1_score: " , np.round(F1_score_svm*100, 2), "%")
print("model SVM radial\n      accuracy: ", accuracy_svm_rad*100, "%  | F1_score: " , np.round(F1_score_svm_rad*100, 2), "%")
print("model SVM polinomic\n      accuracy: ", accuracy_svm_pol*100, "%  | F1_score: " , np.round(F1_score_svm_pol*100, 2), "%")
print("model SVM sigmoide\n      accuracy: ", accuracy_svm_sig*100, "%  | F1_score: " , np.round(F1_score_svm_sig*100, 2), "%")
print("model Linial SVM\n      accuracy: ", accuracy_lin_svm*100, "%  | F1_score: " , np.round(F1_score_lin_svm*100, 2), "%")
print("model Regressió Logistica\n      accuracy: ", accuracy_log_reg*100, "%  | F1_score: " , np.round(F1_score_log_reg*100, 2), "%")
```

```
model knn
      accuracy:  68.75 %  | F1_score:  60.71 %
model tree
      accuracy:  56.25 %  | F1_score:  51.81 %
model random forest
      accuracy:  62.5 %  | F1_score:  56.25 %
model SVM linial
      accuracy:  56.25 %  | F1_score:  56.78 %
model SVM radial
      accuracy:  62.5 %  | F1_score:  48.08 %
model SVM polinomic
      accuracy:  62.5 %  | F1_score:  56.25 %
model SVM sigmoide
      accuracy:  62.5 %  | F1_score:  48.08 %
model Linial SVM
      accuracy:  68.75 %  | F1_score:  69.13 %
model Regressió Logistica
      accuracy:  68.75 %  | F1_score:  65.58 %
```

In [15]:
```
#stacking de 2
X_test_stack = np.column_stack((y_pred_lin_svm, y_pred_log_reg))
```

In [16]:
```
# definim model i graella per a determinar millors paràmetres
S_gb = GradientBoostingClassifier()

param_grid = {"max_depth": range(1, 5), "n_estimators": [1, 2, 4, 6, 8, 10, 25]}

grid_search_S_gb = model_selection.GridSearchCV(S_gb, param_grid=param_grid, cv=4)
grid_search_S_gb.fit(X_test_stack, y_test)

means = grid_search_S_gb.cv_results_["mean_test_score"]
stds = grid_search_S_gb.cv_results_["std_test_score"]
params = grid_search_S_gb.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres per al Gradient Boosting són: {}".format(grid_search_S_gb.best_params_))
S_gb_max_depth = grid_search_S_gb.best_params_["max_depth"]
S_gb_n_estimators = grid_search_S_gb.best_params_["n_estimators"]
```

```
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 2}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 4}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 6}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 8}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 10}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 25}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 2}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 4}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 6}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 8}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 10}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 25}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 2}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 4}
```

```
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 6}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 8}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 10}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 25}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 2}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 4}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 6}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 8}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 10}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 25}
Els millors paràmetres per al Gradient Boosting són: {'max_depth': 1, 'n_estimators': 1}
```

In [17]:
```
# model amb millors paràmetres

model_S_gb = GradientBoostingClassifier(n_estimators=S_gb_n_estimators, max_depth=S_gb_max_depth)

# uso cross_val_score per avaluar el model
resultats_S_gb = model_selection.cross_val_score(model_S_gb, X_test_stack, y_test, cv=5)
print("La precisió mitjana amb cross_validation és: {:0.2f}% amb una desviació de +/- {:0.2f}%".
      format(100*resultats_S_gb.mean(), 100*resultats_S_gb.std()))
```

```
La precisió mitjana amb cross_validation és: 63.33% amb una desviació de +/- 6.67%
```

In [18]:
```
# model amb millors paràmetres
model_S_gb = GradientBoostingClassifier(max_depth=S_gb_max_depth, n_estimators=S_gb_n_estimators)
model_S_gb.fit(X_test_stack, y_test)
```

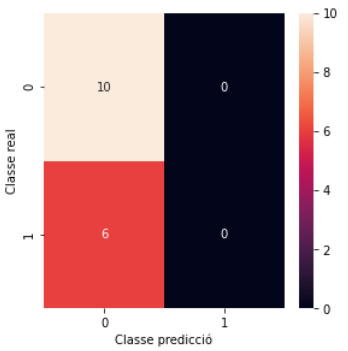
Out[18]:
```
GradientBoostingClassifier(max_depth=1, n_estimators=1)
```

In [19]:
```
# prediccions
y_pred_S_gb = model_S_gb.predict(X_test_stack)
```

In [20]:
```
# mètriques
accuracy_S_gb = accuracy_score(y_test, y_pred_S_gb)
F1_score_S_gb = f1_score(y_test, y_pred_S_gb, average = "weighted")
print("L'accuracy del Stacking de 2 prediccions és de:", accuracy_S_gb*100, "%")
print("La F1 del Stacking de 2 prediccions és de:", np.round(F1_score_S_gb*100, 2), "%")
```

```
L'accuracy del Stacking de 2 prediccions és de: 62.5 %
La F1 del Stacking de 2 prediccions és de: 48.08 %
```

In [21]:
```
# matriu de confusió
S_gb_matrix = confusion_matrix(y_test, y_pred_S_gb)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(S_gb_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



In [22]:
```
# calcul de les prediccions i mètriques per al joc de dades complert

accuracy_S_gb = accuracy_score(y_test, y_pred_S_gb)
F1_score_S_gb = f1_score(y_test, y_pred_S_gb, average = "weighted")
print("L'accuracy del Stacking de 2 prediccions és de:", accuracy_S_gb*100, "%")
print("La F1 del Stacking de 2 prediccions és de:", np.round(F1_score_S_gb*100, 2), "%")
```

```
L'accuracy del Stacking de 2 prediccions és de: 62.5 %
La F1 del Stacking de 2 prediccions és de: 48.08 %
```

In [23]:
```
#stacking de 3
X3_test_stack = np.column_stack((y_pred_knn, y_pred_lin_svm, y_pred_log_reg))
```

In [24]:
```
# definim model i graella per a determinar millors paràmetres

S3_gb = GradientBoostingClassifier()

param_grid = {"max_depth": range(1, 5), "n_estimators": [1, 2, 4, 6, 8, 10, 25]}

grid_search_S3_gb = model_selection.GridSearchCV(S3_gb, param_grid=param_grid, cv=4)
grid_search_S3_gb.fit(X3_test_stack, y_test)

means = grid_search_S3_gb.cv_results_["mean_test_score"]
stds = grid_search_S3_gb.cv_results_["std_test_score"]
params = grid_search_S3_gb.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))
```

```
print("Els millors paràmetres per al Gradient Boosting són: {}".format(grid_search_S3_gb.best_params_))
S3_gb_max_depth = grid_search_S3_gb.best_params_["max_depth"]
S3_gb_n_estimators = grid_search_S3_gb.best_params_["n_estimators"]
```

```
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 2}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 4}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 6}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 8}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 10}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 1, 'n_estimators': 25}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 2}
Precisió mitjana: 56.25 +/- 10.83 amb paràmetres {'max_depth': 2, 'n_estimators': 4}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 6}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 8}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 10}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 2, 'n_estimators': 25}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 2}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 4}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 6}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 8}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 10}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 3, 'n_estimators': 25}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 2}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 4}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 6}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 8}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 10}
Precisió mitjana: 50.00 +/- 0.00 amb paràmetres {'max_depth': 4, 'n_estimators': 25}
Els millors paràmetres per al Gradient Boosting són: {'max_depth': 1, 'n_estimators': 1}
```

In [25]:
```
# creo el model

model_S3_gb = GradientBoostingClassifier(n_estimators=S3_gb_n_estimators, max_depth=S3_gb_max_depth)

# uso cross_val_score per avaluar el model
resultats_S3_gb = model_selection.cross_val_score(model_S3_gb, X3_test_stack, y_test, cv=5)
print("La precisió mitjana amb cross_validation és: {:0.2f}% amb una desviació de +/- {:0.2f}%".
      format(100*resultats_S3_gb.mean(), 100*resultats_S3_gb.std()))
```

```
La precisió mitjana amb cross_validation és: 63.33% amb una desviació de +/- 6.67%
```

In [26]:
```
# model amb millors paràmetres
model_S3_gb = GradientBoostingClassifier(max_depth=S3_gb_max_depth, n_estimators=S3_gb_n_estimators)
model_S3_gb.fit(X3_test_stack, y_test)
```

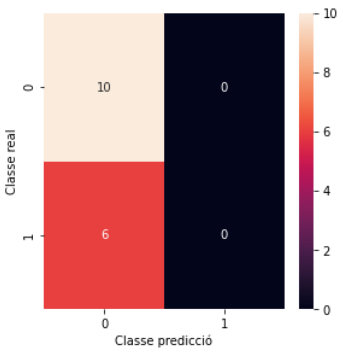
Out[26]:
```
GradientBoostingClassifier(max_depth=1, n_estimators=1)
```

In [27]:
```
# prediccions
y_pred_S3_gb = model_S3_gb.predict(X3_test_stack)
```

In [28]:
```
# mètriques
accuracy_S3_gb = accuracy_score(y_test, y_pred_S3_gb)
F1_score_S3_gb = f1_score(y_test, y_pred_S3_gb, average = "weighted")
print("L'accuracy del Stacking de 3 prediccions és de:", accuracy_S3_gb*100, "%")
print("La F1 del Stacking de 3 prediccions és de:", np.round(F1_score_S3_gb*100, 2), "%")
```

```
L'accuracy del Stacking de 3 prediccions és de: 62.5 %
La F1 del Stacking de 3 prediccions és de: 48.08 %
```

In [29]:
```
# matriu de confusió
S3_gb_matrix = confusion_matrix(y_test, y_pred_S3_gb)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(S3_gb_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



7.3 CASCADING

In [30]:
```
#cascading de 2
X_test_cascad = np.column_stack((y_pred_lin_svm, y_pred_log_reg, X_test))
```

In [31]:
```
#Faig un grid per determinar els millors parametres del nou GB
C_gb = GradientBoostingClassifier()
```

```
param_grid = {"max_depth": range(1, 10), "n_estimators": [1, 5, 10, 15, 20]}

grid_search_C_gb = model_selection.GridSearchCV(C_gb, param_grid=param_grid, cv=4)
grid_search_C_gb.fit(X_test_cascad, y_test)

means = grid_search_C_gb.cv_results_["mean_test_score"]
stds = grid_search_C_gb.cv_results_["std_test_score"]
params = grid_search_C_gb.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres per al Gradient Boosting són: {}".format(grid_search_C_gb.best_params_))
C_gb_max_depth = grid_search_C_gb.best_params_["max_depth"]
C_gb_n_estimators = grid_search_C_gb.best_params_["n_estimators"]
```

```
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 1}
Precisió mitjana: 56.25 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 5}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 10}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 15}
Precisió mitjana: 68.75 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 2, 'n_estimators': 5}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 2, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 2, 'n_estimators': 15}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 2, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 1}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 3, 'n_estimators': 5}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 3, 'n_estimators': 10}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 3, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 3, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 10.83 amb paràmetres {'max_depth': 4, 'n_estimators': 5}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 4, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 4, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 5, 'n_estimators': 5}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 5, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 5, 'n_estimators': 15}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 5, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 6, 'n_estimators': 5}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 6, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 6, 'n_estimators': 15}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 6, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 5}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 10}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 7, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 7, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 1}
Precisió mitjana: 56.25 +/- 10.83 amb paràmetres {'max_depth': 8, 'n_estimators': 5}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 8, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 8, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 8, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 9, 'n_estimators': 5}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 9, 'n_estimators': 10}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 9, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 9, 'n_estimators': 20}
Els millors paràmetres per al Gradient Boosting són: {'max_depth': 2, 'n_estimators': 10}
```

In [32]:

```
# creo el model de Gradient Boosting

model_C_gb = GradientBoostingClassifier(n_estimators=C_gb_n_estimators, max_depth=C_gb_max_depth)

# uso cross_val_score per avaluar el model
resultats_C_gb = model_selection.cross_val_score(model_C_gb, X_test_cascad, y_test, cv=5)
print("La precisió mitjana amb cross_validation és: {:0.2f}% amb una desviació de +/- {:0.2f}%".
      format(100*resultats_C_gb.mean(), 100*resultats_C_gb.std()))
```

```
La precisió mitjana amb cross_validation és: 81.67% amb una desviació de +/- 15.28%
```

In [33]:

```
# model amb millors paràmetres
model_C_gb = GradientBoostingClassifier(max_depth=C_gb_max_depth, n_estimators=C_gb_n_estimators)
model_C_gb.fit(X_test_cascad, y_test)
```

Out[33]:

```
GradientBoostingClassifier(max_depth=2, n_estimators=10)
```

In [34]:

```
# prediccions
y_pred_C_gb = model_C_gb.predict(X_test_cascad)
```

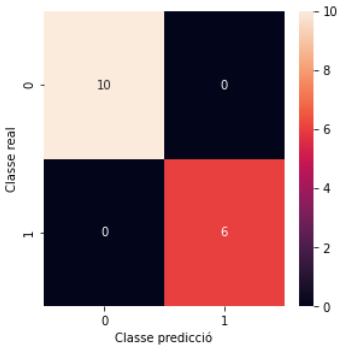
In [35]:

```
# mètriques
accuracy_C_gb = accuracy_score(y_test, y_pred_C_gb)
F1_score_C_gb = f1_score(y_test, y_pred_C_gb, average = "weighted")
print("L'accuracy del Cascading de 2 prediccions és de:", accuracy_C_gb*100, "%")
print("La F1 del Cascading de 2 prediccions és de:", np.round(F1_score_C_gb*100, 2), "%")
```

```
L'accuracy del Cascading de 2 prediccions és de: 100.0 %
La F1 del Cascading de 2 prediccions és de: 100.0 %
```

In [36]:

```
# matriu de confusió

C_gb_matrix = confusion_matrix(y_test, y_pred_C_gb)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(C_gb_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



In [37]:
```
#cascading de 3
X3_test_cascad = np.column_stack((y_pred_knn, y_pred_lin_svm, y_pred_log_reg, X_test))
```

In [38]:
```
#Faig un grid per determinar els millors parametres del nou GB
C3_gb = GradientBoostingClassifier()

param_grid = {"max_depth": range(1, 10), "n_estimators": [1, 5, 10, 15, 20]}

grid_search_C3_gb = model_selection.GridSearchCV(C3_gb, param_grid=param_grid, cv=4)
grid_search_C3_gb.fit(X3_test_cascad, y_test)

means = grid_search_C3_gb.cv_results_["mean_test_score"]
stds = grid_search_C3_gb.cv_results_["std_test_score"]
params = grid_search_C3_gb.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres per al Gradient Boosting són: {}".format(grid_search_C3_gb.best_params_))
C3_gb_max_depth = grid_search_C3_gb.best_params_["max_depth"]
C3_gb_n_estimators = grid_search_C3_gb.best_params_["n_estimators"]
```

```
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 1}
Precisió mitjana: 56.25 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 5}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 10}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 15}
Precisió mitjana: 68.75 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 2, 'n_estimators': 5}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 2, 'n_estimators': 10}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 2, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 2, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 3, 'n_estimators': 5}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 3, 'n_estimators': 10}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 3, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 3, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 4, 'n_estimators': 5}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 4, 'n_estimators': 10}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 4, 'n_estimators': 15}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 4, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 5}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 5, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 5, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 5, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 1}
Precisió mitjana: 56.25 +/- 10.83 amb paràmetres {'max_depth': 6, 'n_estimators': 5}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 6, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 6, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 6, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 1}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 7, 'n_estimators': 5}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 7, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 7, 'n_estimators': 15}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 7, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 1}
Precisió mitjana: 56.25 +/- 10.83 amb paràmetres {'max_depth': 8, 'n_estimators': 5}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 8, 'n_estimators': 10}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 8, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 8, 'n_estimators': 20}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 1}
Precisió mitjana: 56.25 +/- 10.83 amb paràmetres {'max_depth': 9, 'n_estimators': 5}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 9, 'n_estimators': 10}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 9, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 9, 'n_estimators': 20}
Els millors paràmetres per al Gradient Boosting són: {'max_depth': 2, 'n_estimators': 10}
```

In [39]:
```
# creo el model de Gradient Boosting

model_C3_gb = GradientBoostingClassifier(n_estimators=C3_gb_n_estimators, max_depth=C3_gb_max_depth)

# uso cross_val_score per avaluar el model
resultats_C3_gb = model_selection.cross_val_score(model_C3_gb, X3_test_cascad, y_test, cv=5)
print("La precisió mitjana amb cross_validation és: {:0.2f}% amb una desviació de +/- {:0.2f}%".
      format(100*resultats_C3_gb.mean(), 100*resultats_C3_gb.std()))
```

```
La precisió mitjana amb cross_validation és: 81.67% amb una desviació de +/- 15.28%
```

In [40]:
```
# model amb millors paràmetres
model_C3_gb = GradientBoostingClassifier(max_depth=C3_gb_max_depth, n_estimators=C3_gb_n_estimators)
model_C3_gb.fit(X3_test_cascad, y_test)
```

Out[40]:
```
GradientBoostingClassifier(max_depth=2, n_estimators=10)
```

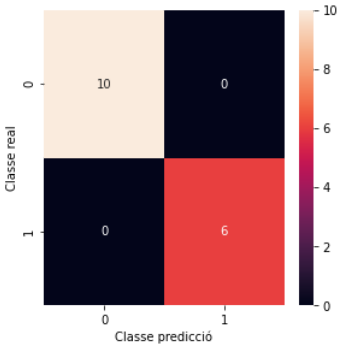
In [41]:
```
# prediccions
```

```
y_pred_C3_gb = model_C3_gb.predict(X3_test_cascad)
```

In [42]:
```
# mètriques
accuracy_C3_gb = accuracy_score(y_test, y_pred_C3_gb)
F1_score_C3_gb = f1_score(y_test, y_pred_C3_gb, average = "weighted")
print("L'accuracy del Cascading de 3 prediccions és de:", accuracy_C3_gb*100, "%")
print("La F1 del Cascading de 3 prediccions és de:", np.round(F1_score_C3_gb*100, 2), "%")
```

```
L'accuracy del Cascading de 3 prediccions és de: 100.0 %
La F1 del Cascading de 3 prediccions és de: 100.0 %
```

In [43]:
```
# matriu de confusió
C3_gb_matrix = confusion_matrix(y_test, y_pred_C3_gb)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(C3_gb_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```



7.4 VALIDACIO AMB TOT EL CONJUNT

In [44]:
```
%store -r y_pred_knn_cascad
%store -r accuracy_knn_cascad
%store -r F1_score_knn_cascad

%store -r y_pred_lin_svm_cascad
%store -r accuracy_lin_svm_cascad
%store -r F1_score_lin_svm_cascad

%store -r y_pred_log_reg_cascad
%store -r accuracy_log_reg_cascad
%store -r F1_score_log_reg_cascad
```

In [45]:
```
print("model knn GLOBAL\n      accuracy: ", accuracy_knn_cascad*100, "%  | F1_score: " , np.round(F1_score_knn_cascad*100, 2), "%")
print("model Linial SVM GLOBAL \n      accuracy: ", accuracy_lin_svm_cascad*100, "%  | F1_score: " , np.round(F1_score_lin_svm_cascad*100,
print("model Regressió Logistica GLOBAL \n      accuracy: ", accuracy_log_reg_cascad*100, "%  | F1_score: " , np.round(F1_score_log_reg_cas
```

```
model knn GLOBAL
      accuracy:  70.88607594936708 %  | F1_score:  61.04 %
model Linial SVM GLOBAL
      accuracy:  82.27848101265823 %  | F1_score:  82.07 %
model Regressió Logistica GLOBAL
      accuracy:  75.9493670886076 %  | F1_score:  71.39 %
```

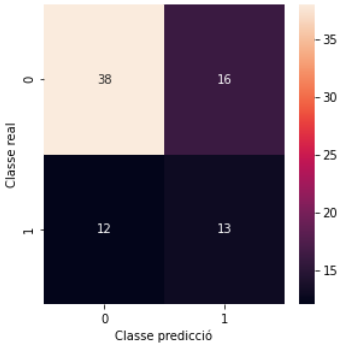
In [46]:
```
X3_cascad = np.column_stack((y_pred_knn_cascad, y_pred_lin_svm_cascad, y_pred_log_reg_cascad, X))
```

In [47]:
```
# prediccions
pred_X3_cascad = model_C3_gb.predict(X3_cascad)
```

In [48]:
```
# mètriques
accuracy_X3_cascad = accuracy_score(y, pred_X3_cascad)
F1_score_X3_cascad = f1_score(y, pred_X3_cascad, average = "weighted")
print("L'accuracy del model Cascading complert és de:", accuracy_X3_cascad*100, "%")
print("La F1 del model Cascading complert és de:", np.round(F1_score_X3_cascad*100, 2), "%")
```

```
L'accuracy del model Cascading complert és de: 64.55696202531645 %
La F1 del model Cascading complert és de: 65.19 %
```

In [49]:
```
# matriu de confusió
X3_cascad_matrix = confusion_matrix(y, pred_X3_cascad)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(X3_cascad_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Classe predicció')
plt.show()
```


38
16
12
13
0
1
Classe real
Classe predicció
35
30
25
20
15

ANNEX 10: CODI PYTHON: COMBINACIÓ DE MODELS AMB XARXA NEURONAL

Caracterització i detecció precoç de la ventriculomegalia aïllada a partir de dades pre i post-natals

In [1]:
```
# Importem llibreries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler, RobustScaler

from sklearn import linear_model, svm
from sklearn.svm import SVC, LinearSVC
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score, confusion_matrix, accuracy_score, f1_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.pipeline import make_pipeline
```

In [2]:
```
# recuperació del joc de dades preparat per a classificacio
data_logic = pd.read_pickle("./data_logic.pkl")
```

8. CASCADA AMB XARXA NEURONAL

In [3]:
```
# separem les dades de la variable objectiu i generem conjunt d'entrenament i de test
X = data_logic.iloc[:, :-1]
y = data_logic["B_o_V_cod"]
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2, random_state=27)
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
```

```
X_train: (63, 34)
X_test: (16, 34)
```

In [4]:
```
# comprovació de la proporció dels conjunts
y_train.value_counts()
```

Out[4]:
```
0    44
1    19
Name: B_o_V_cod, dtype: int64
```

In [5]:
```
y_test.value_counts()
```

Out[5]:
```
0    10
1     6
Name: B_o_V_cod, dtype: int64
```

In [6]:
```
# recupero resultats models

%store -r y_pred_lin_svm
%store -r accuracy_lin_svm
%store -r F1_score_lin_svm

%store -r y_pred_log_reg
%store -r accuracy_log_reg
%store -r F1_score_log_reg

%store -r y_pred_lin_svm_cascad
%store -r accuracy_lin_svm_cascad
%store -r F1_score_lin_svm_cascad

%store -r y_pred_log_reg_cascad
%store -r accuracy_log_reg_cascad
%store -r F1_score_log_reg_cascad

%store -r y_pred_knn
%store -r y_pred_knn_cascad

%store -r y_pred_Adam
%store -r y_pred_Adam_cascad

%store -r y_pred_SGD
%store -r y_pred_SGD_cascad

%store -r y_pred_Adagrad
%store -r y_pred_Adagrad_cascad
```

In [7]:
```
# cascading
X4N_test_cascad = np.column_stack((y_pred_Adam, y_pred_knn, y_pred_lin_svm, y_pred_log_reg, X_test))
```

In [8]:
```
# definim model i graella per a determinar millors paràmetres
C4N_gb = GradientBoostingClassifier()

param_grid = {"max_depth": range(1, 10), "n_estimators": range(1, 20)}
```

```
grid_search_C4N_gb = model_selection.GridSearchCV(C4N_gb, param_grid=param_grid, cv=4)
grid_search_C4N_gb.fit(X4N_test_cascad, y_test)

means = grid_search_C4N_gb.cv_results_["mean_test_score"]
stds = grid_search_C4N_gb.cv_results_["std_test_score"]
params = grid_search_C4N_gb.cv_results_['params']

for mean, std, pms in zip(means, stds, params):
    print("Precisió mitjana: {:.2f} +/- {:.2f} amb paràmetres {}".format(mean*100, std*100, pms))

print("Els millors paràmetres per al Gradient Boosting són: {}".format(grid_search_C4N_gb.best_params_))
C4N_gb_max_depth = grid_search_C4N_gb.best_params_["max_depth"]
C4N_gb_n_estimators = grid_search_C4N_gb.best_params_["n_estimators"]
```

```
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 1, 'n_estimators': 2}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 3}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 4}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 5}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 1, 'n_estimators': 6}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 7}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 8}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 1, 'n_estimators': 9}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 10}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 1, 'n_estimators': 11}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 12}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 13}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 14}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 15}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 16}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 1, 'n_estimators': 17}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 1, 'n_estimators': 18}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 1, 'n_estimators': 19}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 2, 'n_estimators': 2}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 3}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 2, 'n_estimators': 4}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 2, 'n_estimators': 5}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 2, 'n_estimators': 6}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 7}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 2, 'n_estimators': 8}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 2, 'n_estimators': 9}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 2, 'n_estimators': 10}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 2, 'n_estimators': 11}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 2, 'n_estimators': 12}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 13}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 2, 'n_estimators': 14}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 15}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 16}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 17}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 2, 'n_estimators': 18}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 2, 'n_estimators': 19}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 21.65 amb paràmetres {'max_depth': 3, 'n_estimators': 2}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 3}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 4}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 5}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 3, 'n_estimators': 6}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 3, 'n_estimators': 7}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 3, 'n_estimators': 8}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 9}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 10}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 3, 'n_estimators': 11}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 12}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 13}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 3, 'n_estimators': 14}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 3, 'n_estimators': 16}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 3, 'n_estimators': 17}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 18}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 3, 'n_estimators': 19}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 4, 'n_estimators': 2}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 4, 'n_estimators': 3}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 4}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 5}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 4, 'n_estimators': 6}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 4, 'n_estimators': 7}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 4, 'n_estimators': 8}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 4, 'n_estimators': 9}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 4, 'n_estimators': 10}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 4, 'n_estimators': 11}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 12}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 13}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 4, 'n_estimators': 14}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 15}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 16}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 4, 'n_estimators': 17}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 4, 'n_estimators': 18}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 4, 'n_estimators': 19}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 21.65 amb paràmetres {'max_depth': 5, 'n_estimators': 2}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 3}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 5, 'n_estimators': 4}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 5}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 5, 'n_estimators': 6}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 7}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 8}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 5, 'n_estimators': 9}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 5, 'n_estimators': 10}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 5, 'n_estimators': 11}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 12}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 5, 'n_estimators': 13}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 14}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 5, 'n_estimators': 16}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 5, 'n_estimators': 17}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 5, 'n_estimators': 18}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 5, 'n_estimators': 19}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 6, 'n_estimators': 2}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 6, 'n_estimators': 3}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 4}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 6, 'n_estimators': 5}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 6}
```

```
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 6, 'n_estimators': 7}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 6, 'n_estimators': 8}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 6, 'n_estimators': 9}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 10}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 6, 'n_estimators': 11}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 12}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 6, 'n_estimators': 13}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 14}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 6, 'n_estimators': 15}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 6, 'n_estimators': 16}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 17}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 18}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 6, 'n_estimators': 19}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 1}
Precisió mitjana: 56.25 +/- 10.83 amb paràmetres {'max_depth': 7, 'n_estimators': 2}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 7, 'n_estimators': 3}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 4}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 7, 'n_estimators': 5}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 6}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 7, 'n_estimators': 7}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 7, 'n_estimators': 8}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 9}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 7, 'n_estimators': 10}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 11}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 12}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 13}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 14}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 15}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 16}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 7, 'n_estimators': 17}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 7, 'n_estimators': 18}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 7, 'n_estimators': 19}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 1}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 2}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 8, 'n_estimators': 3}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 8, 'n_estimators': 4}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 5}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 6}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 8, 'n_estimators': 7}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 8}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 9}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 10}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 8, 'n_estimators': 11}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 8, 'n_estimators': 12}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 8, 'n_estimators': 13}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 8, 'n_estimators': 14}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 15}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 16}
Precisió mitjana: 75.00 +/- 17.68 amb paràmetres {'max_depth': 8, 'n_estimators': 17}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 18}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 8, 'n_estimators': 19}
Precisió mitjana: 62.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 1}
Precisió mitjana: 68.75 +/- 20.73 amb paràmetres {'max_depth': 9, 'n_estimators': 2}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 3}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 4}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 5}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 6}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 7}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 9, 'n_estimators': 8}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 9, 'n_estimators': 9}
Precisió mitjana: 81.25 +/- 20.73 amb paràmetres {'max_depth': 9, 'n_estimators': 10}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 9, 'n_estimators': 11}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 9, 'n_estimators': 12}
Precisió mitjana: 93.75 +/- 10.83 amb paràmetres {'max_depth': 9, 'n_estimators': 13}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 14}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 15}
Precisió mitjana: 81.25 +/- 10.83 amb paràmetres {'max_depth': 9, 'n_estimators': 16}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 17}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 18}
Precisió mitjana: 87.50 +/- 12.50 amb paràmetres {'max_depth': 9, 'n_estimators': 19}
Els millors paràmetres per al Gradient Boosting són: {'max_depth': 1, 'n_estimators': 5}
```

In [9]:
```
# model amb millors paràmetres

model_C4N_gb = GradientBoostingClassifier(n_estimators=C4N_gb_n_estimators, max_depth=C4N_gb_max_depth)

# uso cross_val_score per avaluar el model
resultats_C4N_gb = model_selection.cross_val_score(model_C4N_gb, X4N_test_cascad, y_test, cv=5)
print("La precisió mitjana amb cross_validation és: {:0.2f}% amb una desviació de +/- {:0.2f}%".
      format(100*resultats_C4N_gb.mean(), 100*resultats_C4N_gb.std()))
```

```
La precisió mitjana amb cross_validation és: 93.33% amb una desviació de +/- 13.33%
```

In [10]:
```
model_C4N_gb.fit(X4N_test_cascad, y_test)
```

Out[10]:
```
GradientBoostingClassifier(max_depth=1, n_estimators=5)
```

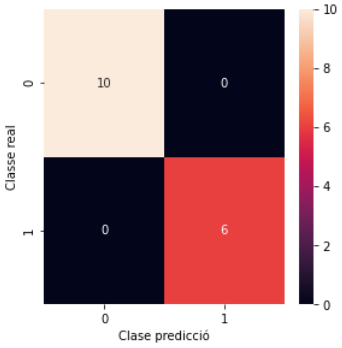
In [11]:
```
# prediccions
y_pred_C4N_gb = model_C4N_gb.predict(X4N_test_cascad)
```

In [12]:
```
# mètriques
accuracy_C4N_gb = accuracy_score(y_test, y_pred_C4N_gb)
F1_score_C4N_gb = f1_score(y_test, y_pred_C4N_gb, average = "weighted")
print("L'accuracy del Cascading de 6 prediccions és de:", accuracy_C4N_gb*100, "%")
print("La F1 del Cascading de 6 prediccions és de:", np.round(F1_score_C4N_gb*100, 2), "%")
```

```
L'accuracy del Cascading de 6 prediccions és de: 100.0 %
La F1 del Cascading de 6 prediccions és de: 100.0 %
```

In [13]:
```
# matriu de confusió
C4N_gb_matrix = confusion_matrix(y_test, y_pred_C4N_gb)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(C4N_gb_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
```

```
plt.xlabel('Clase predicció')
plt.show()
```



In [14]:
```
# usar tota la bbdd per a validar dades:
```

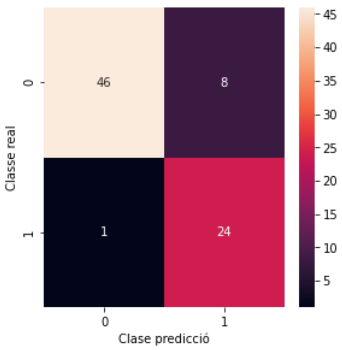
In [15]:
```
X4N_cascad = np.column_stack((y_pred_Adam_cascad, y_pred_knn_cascad ,y_pred_lin_svm_cascad, y_pred_log_reg_cascad, X))
```

In [16]:
```
# prediccions
pred_X4N_cascad = model_C4N_gb.predict(X4N_cascad)
```

In [17]:
```
# mètriques
accuracy_X4N_cascad = accuracy_score(y, pred_X4N_cascad)
F1_score_X4N_cascad = f1_score(y, pred_X4N_cascad, average = "weighted")
print("L'accuracy del model Cascading complert és de:", accuracy_X4N_cascad*100, "%")
print("La F1 del model Cascading complert és de:", np.round(F1_score_X4N_cascad*100, 2), "%")
```

```
L'accuracy del model Cascading complert és de: 88.60759493670885 %
La F1 del model Cascading complert és de: 88.91 %
```

In [18]:
```
# matriu de confusió
X4N_cascad_matrix = confusion_matrix(y, pred_X4N_cascad)
x_axis_labels = ['0','1']
y_axis_labels = ['0','1']

figure = plt.figure(figsize=(4, 4))
sns.heatmap(X4N_cascad_matrix, annot=True, xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.tight_layout()
plt.ylabel('Classe real')
plt.xlabel('Clase predicció')
plt.show()
```



9. BIBLIOGRAFIA

[1] Fetal Medicine Barcelona. [Internet]. Fundación Fetal Medicince Barcelona; 2018 [consultat: 21 de febrer de 2021]. Disponible a: medicinafetalbarcelona.org/el-feto-como-paciente/cerebro/ventriculomegalia.html

[2] Cardoza JD, Goldstein RB, Filly RA. Exclusion of fetal ventriculomegaly with a single measurement: the width of the lateral ventricular atrium. Radiology [Internet]. Radiological Society of North America (RSNA); 1988 Dec;169(3):711–4. [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1148/radiology.169.3.3055034.

[3] Hahner N. Assessment of cortical development in fetuses with isolated non-severe ventriculomegaly and its correlation with neurodevelopmental outcome. PhD THESIS. Erasmus Mundus Joint Doctorate in Fetal and Perinatal Medicine. Fetalmed-PhD; 2019 Dec.

[4] Ferreira C, Rocha I, Silva J, Sousa AO, Godinho C, Azevedo M, Brito C, Valente F. Mild to moderate fetal ventriculomegaly: obstetric and postnatal outcome. Acta Obstet Ginecol Port 2014; 8(3):246-251. 2014

[5] Cardoen L, De Catte L, Demaerel P, Devlieger R, Lewi L, Deprest J, Claus F. The role of magnetic resonance imaging in the diagnostic work-up of fetal ventriculomegaly. Facts Views Vis Obgyn. 2011;3(3):159-63. PMID: 24753861; PMCID: PMC3991458.

[6] Nabila M, Fatoni MH, Sardjono TA. Automated Cerebral Lateral Ventricle Ratio Measurement From 2-Dimensional Fetal Ultrasound Image to Predict Ventriculomegaly. 2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM) [Internet]. IEEE; 2020 Nov 17; [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1109/cenim51130.2020.9297908

[7] Vergani P, Locatelli A, Strobelt N, Cavallone M, Ceruti P, Paterlini G, et al. Clinical outcome of mild fetal ventriculomegaly. American Journal of Obstetrics and Gynecology [Internet]. Elsevier BV; 1998 Feb;178(2):218–22. [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1016/s0002-9378(98)80003-3

[8] Gaglioti P, Danelon D, Bontempo S, Mombrò M, Cardaropoli S, Todros T. Fetal cerebral ventriculomegaly: outcome in 176 cases. Ultrasound in Obstetrics and Gynecology [Internet]. Wiley; 2005 Mar 24;25(4):372–7. [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1002/uog.1857

[9] Gaglioti P, Oberto M, Todros T. The significance of fetal ventriculomegaly: etiology, short- and long-term outcomes. Prenatal Diagnosis [Internet]. Wiley; 2009 Apr;29(4):381–8. [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1002/pd.2195

[10] Ouahba J, Luton D, Vuillard E, Garel C, Gressens P, Blanc N, et al. Prenatal isolated mild ventriculomegaly: outcome in 167 cases. BJOG: An International Journal of Obstetrics & Gynaecology [Internet]. Wiley; 2006 Aug 25;113(9):1072–9. [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1111/j.1471-0528.2006.01050.x

[11] Melchiorre K, Bhide A, Gika AD, Pilu G, Papageorghiou AT. Counseling in isolated mild fetal ventriculomegaly. Ultrasound in Obstetrics and Gynecology [Internet]. Wiley; 2009 Aug;34(2):212–24. [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1002/uog.7307

[12] Hernández S M, Orribo M O, Martínez W I, Padilla P AI, Álvarez de la Rosa R M, Troyano L JM. Detección ecográfica y pronóstico de la ventriculomegalia fetal. Revista chilena de obstetricia y ginecología [Internet]. SciELO Agencia Nacional de Investigacion y Desarrollo (ANID); 2012;77(4):249–54. [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.4067/s0717-75262012000400002

[13] Kotsiantis SB, Zaharakis ID, Pintelas PE. Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering; Amsterdam; 2007; 160 (1):3-24.

[14] Osisanwo FY, Akinsola JET, Awodele O, Hinmikaiye JO, Olakanmi O, Akinjobi J. Supervised machine learning algorithms: classification and comparison.International Journal of Computer Trends and Technology (IJCTT); 2017; 48(3): 128-38.

[15] Singh A, Thakur N, Sharma A. A review of supervised machine learning algorithms. 2016. 3rd International Conference on Computing for Sustainable Global Development (INDIACom); IEEE; 2016; 1310-15.

[16] Bhavsar H, Ganatra A. A comparative study of training algorithms for supervised machine learning. International Journal of Soft Computing and Engineering (IJSCE);2012; 2(4): 2231-307.

[17] Sarwar MA, Kamal N, Hamid W, Shah MA. Prediction of Diabetes Using Machine Learning Algorithms in Healthcare. 2018 24th International Conference on Automation and Computing (ICAC) [Internet]. IEEE; 2018 Sep; [consultat: 13 d'abril de 2021]. Disponible a: http://dx.doi.org/10.23919/iconac.2018.8748992

[18] Li D-C, Chen H-Y, Shi Q-S. Learning from small datasets containing nominal attributes. Neurocomputing [Internet]. Elsevier BV; 2018 May;291:226–36. [consultat: 24 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1016/j.neucom.2018.02.069

[19] Romero M, Interian Y, Solberg T, Valdes G. Targeted transfer learning to improve performance in small medical physics datasets. Medical Physics [Internet]. Wiley; 2020 Oct 25;47(12):6246–56. [consultat: 24 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1002/mp.14507

[20] Menon A, Thompson-Colón JA, Washburn NR. Hierarchical Machine Learning Model for Mechanical Property Predictions of Polyurethane Elastomers From Small Datasets. Frontiers in Materials [Internet]. Frontiers Media SA; 2019 May 8;6. [consultat: 24 d'abril de 2021]. Disponible a: http://dx.doi.org/10.3389/fmats.2019.00087

[21] Abu Zohair LM. Prediction of Student's performance by modelling small dataset size. International Journal of Educational Technology in Higher Education [Internet]. Springer Science and Business Media LLC; 2019 Aug 2;16(1). [consultat: 24 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1186/s41239-019-0160-3

[22] Narayanan B, Saadeldin M, Albert P, McGuinness K, Mac Namee B. Extracting pasture phenotype and biomass percentages using weakly supervised multi-target deep learning on a small dataset;2021; arXiv preprint arXiv:2101.03198.

[23] Guvenir HA, Acar B, Demiroz G, Cekin A. A supervised machine learning algorithm for arrhythmia analysis. Computers in Cardiology 1997 [Internet]. IEEE; [consultat: 24 d'abril de 2021]. Disponible a: http://dx.doi.org/10.1109/cic.1997.647926.

[24] https://www.encyclopedia.com

[25] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine Learning in Python. JMLR: 2011; 12: 2825-30. Disponible a: https://scikit-learn.org/stable/index.html.

[26] Géron A. Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. Sebastopol, CA: O'Reilly Media; 2017.

[27] Gironés J, Casas J, Minguillón J, Caihuelas R. Minería de datos: modelos y algoritmos. Editorial UOC; 2017.

[28] Encyclopedia of Machine Learning and Data Mining [Internet]. Springer US; 2017. Disponible a: https://link.springer.com/referencework/10.1007/978-1-4899-7687-1.

[29] Bosch A, Casas J, Lozano T. Deep Learning: principios y fundamentos. Editorial UOC; 2019