Universitat Oberta de Catalunya

uoc.edu

Reconeixement d’esdeveniments d’encefalogrames (REE)

Francisco García Caparrós
Estudis d'Informàtica, Multimèdia i Telecomunicació
Intel·ligència artificial

Consultor: Dr. David Isern Alarcón
Professor responsable: Dr. Friman Sánchez Castaño
06/2024

FITXA DEL TREBALL FINAL

Títol del treball:	*Reconeixement d'esdeveniments d'encefalogrames (REE)*
Nom de l'autor:	*Francisco García Caparrós*
Nom del consultor/a:	*Dr. David Isern Alarcón*
Nom del PRA:	*Dr. Friman Sánchez Castaño*
Data de lliurament (mm/aaaa):	23/06/2024
Titulació:	Estudis d'Informàtica, Multimèdia i Telecomunicació
Àrea del Treball Final:	Intel·ligència artificial
Idioma del treball:	Català
Nombre de crèdits:	12
Paraules clau	*EEG, BCI, REE*

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

Reconeixement d'esdeveniments d'encefalogrames, també anomenat REE, és un producte dissenyat amb l'objectiu de tractar, analitzar i predir esdeveniments que una persona pot pensar durant una sessió de lectura de la seva activitat cerebral. Des del punt de vista funcional, l'aplicació ha estat concebuda per poder tractar conjunts de dades relacionats amb esdeveniments d'imaginació motora produïts durant un període de temps determinat. Per exemple, pensaments relacionats amb els moviments de mans cap a l'esquerra o dreta generaran uns esdeveniments en el temps que, mitjançant uns lectors de senyals d'encefalogrames, produiran unes dades que l'aplicació haurà d'analitzar i classificar.

REE és una solució implementada amb Python i ofereix una versió d'escriptori amb una interfície gràfica que permet seleccionar conjunts de dades de la BCI Competition [1] per fer el tractament i aplicació d'algorismes d'aprenentatge computacional per obtenir les prediccions. El resultat de les prediccions generaran diverses mètriques com les matrius de confusió o corbes ROC per poder veure la qualitat de les prediccions obtingudes pel model.

La solució esmentada anteriorment s'ha dissenyat mitjançant un cicle de vida incremental, on cada part s'ha anat dissenyant de manera iterativa, cosa que ha generat una solució que ha anat madurant amb el temps. En aquest sentit, s'han realitzat diferents proves en cada fase amb diversos conjunts de dades que han permès assegurar la qualitat dels lliurables.

En conclusió, el producte és capaç de detectar diversos esdeveniments relacionats amb la imaginació motora amb un nivell d'encert d'entre el 50% i el 100% aproximadament.

Francisco García Caparrós

Abstract (in English, 250 words or less):

Encephalograms event recognition, known as EER, is a tool developed to manage and predict events human thoughts in an electroencephalogram (EEG) [2] reading session. This is an application designed to process motor imagery events within a time period. For instance, thoughts about moving the left or right hand may produce some specific events and data that will be registered through encephalogram signal readers. Those events must be analysed and classified by the application to make the predictions.

EER has been made with Python code and it has a graphic interface that gives the opportunity to select the datasets to work with. Those datasets are from the BCI Competition [1] and they will be processed to apply machine learning algorithms to get the predictions. As a result of that, some metrics will be generated, such as confusion matrices or ROC Curves. Those metrics will be useful to check que quality of the predictions obtained by the model.

This software has been made using the incremental life cicle methodology and every part of it has been developed iteratively. Therefore, we have been working with a functional program almost every phase of the project and several tests have been performed with the aim of ensure the quality all the deliveries.

To sum up, we have developed an application that is able to predict events about motor imagery with a precision between 50% and 100%. Moreover, a report with some metrics will be available to help to determine the quality of the prediction.

Índex

Llista de figures

Llista de taules

1. Resum

El projecte titulat Reconeixent d'esdeveniments d'encefalogrames amb l'acrònim REE, és un projecte que està basat en l'anàlisi i interpretació dels senyals EEG [2] per tal de poder fer prediccions dels pensaments que un ésser humà va tenir en una sessió de lectures d'activitat cerebral. Aquests pensaments estarien relacionats amb imaginació motora, com per exemple, moviment d'articulacions. De manera resumida, podríem destacar els següents conceptes:

- **Reconeixement:** És la fita principal del projecte i estaria relacionada amb l'anàlisi i predicció de les dades.
- **Esdeveniments:** Durant la lectura de dades es produeixen esdeveniment que caldrà detectar i analitzar. Aquests acostumen a ser moviments d'articulacions, encara que poden haver-hi d'altres com els artefactes o d'altres relacionats amb els tests.
- **Encefalogrames:** Els encefalogrames ens indicaran en un període de temps l'activitat elèctrica cerebral d'un individu. D'aquesta manera, s'obtindran unes dades que contindran els esdeveniments que caldrà detectar.

1.1. Abast

Com s'ha vist anteriorment, el producte que s'haurà de lliurar haurà de ser capaç de reconèixer una sèrie d'esdeveniments en lectures d'encefalogrames. Per tal d'aconseguir aquesta fita, l'abast del projecte haurà de contenir els següents apartats.

1.1.1. Font de dades

Caldrà escollir una font de dades fiable per poder dur a terme les prediccions. Aquesta font de dades pot ser en temps real mitjançant un lector d'encefalogrames, o de dades de sessions de lectura ja realitzades, com per exemple, les dades de la BCI Competition [1].

1.1.2. Tractament de dades

Un cop es disposi de les dades de les lectures d'encefalogrames, caldrà fer el tractament adequat per a la següent fase. Caldrà aplicar tècniques de mineria de dades per poder deixar les dades netes i únicament amb les característiques i observacions necessàries per a les prediccions.

1.1.3. Predicció d'esdeveniments

Amb les dades enllestides, caldrà que s'apliquin algorismes d'aprenentatge supervisat o no supervisat per obtenir les prediccions. Per exemple, classificadors com KNN, SVM, decission tree, etc.

1.1.4. Publicació de resultats

Per acabar, caldrà publicar els resultats obtinguts de les prediccions. Caldrà mostrar mitjançant alguna interfície gràfica els resultats per veure la precisió, exactitud, matriu de confusió, etc. A més, serà necessari incloure alguna representació gràfica que indiqui el rendiment del model, com seria el cas de la corba ROC [6].

1.1.5. Integració amb tercers

Finalment, els resultats obtinguts podrien ser aprofitables per generar una interfície que permeti a altres sistemes connectar-se al nostre i així obtenir prediccions. Aquesta interfície es va considerar opcional dins del projecte i no es va arribar a desenvolupar.

2. Introducció

Tal com s'ha indicat anteriorment, el producte que s'entregarà en aquest treball de final s'anomenarà **Reconeixent d'esdeveniments d'encefalogrames** (REE). D'aquesta manera, es lliurarà una solució capaç de reconèixer els esdeveniments de les lectures d'encefalogrames per poder fer una posterior interpretació. A grans trets, l'aplicació estarà composta d'aquestes parts:

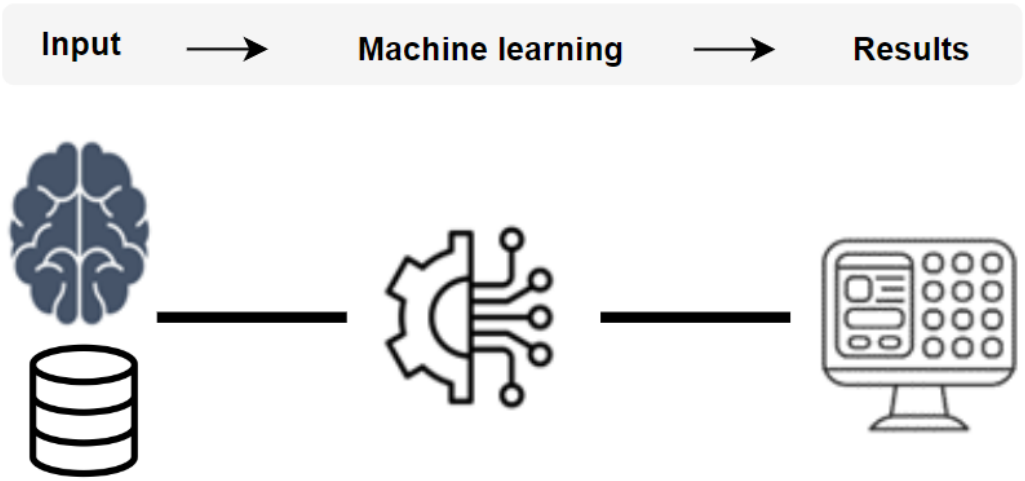


Figura 1. Fases generals de la solució

Per tant, tindrem una primera part on es generaran les dades d'entrada que posteriorment es tractaran amb aprenentatge computacional. Durant aquest tractament s'aplicaran algorismes d'aprenentatge computacionals KNN, MLP, SVM, etc. Finalment, caldrà mostrar els resultats obtinguts per tal d'identificar correctament els esdeveniments.

2.1. Esdeveniments

La gestió i predicció dels esdeveniments és un dels punts més importats de la solució. En aquest sentit, el producte haurà de ser capaç de detectar esdeveniments **voluntaris** fets o pensats per l'ésser humà, com per exemple, moviments d'ulls, articulacions, músculs, etc. Bàsicament, es tractaran dades que en un període de temps determinat produeixin aquests esdeveniments per tal de poder entrenar el corresponent model per futures prediccions. Així mateix, caldrà tractar altres esdeveniments que no seran aprofitables pel model, ja que seran eliminats, com els artefactes. Aquests esdeveniments es produeixen involuntàriament pel sistema cardiovascular, incorrectes mesures, altres músculs del cos, etc.

2.2. Diagrama d'alt nivell

En el diagrama següent mostrem el detall d'algunes tasques que es faran a les tres fases que hem explicat anteriorment.

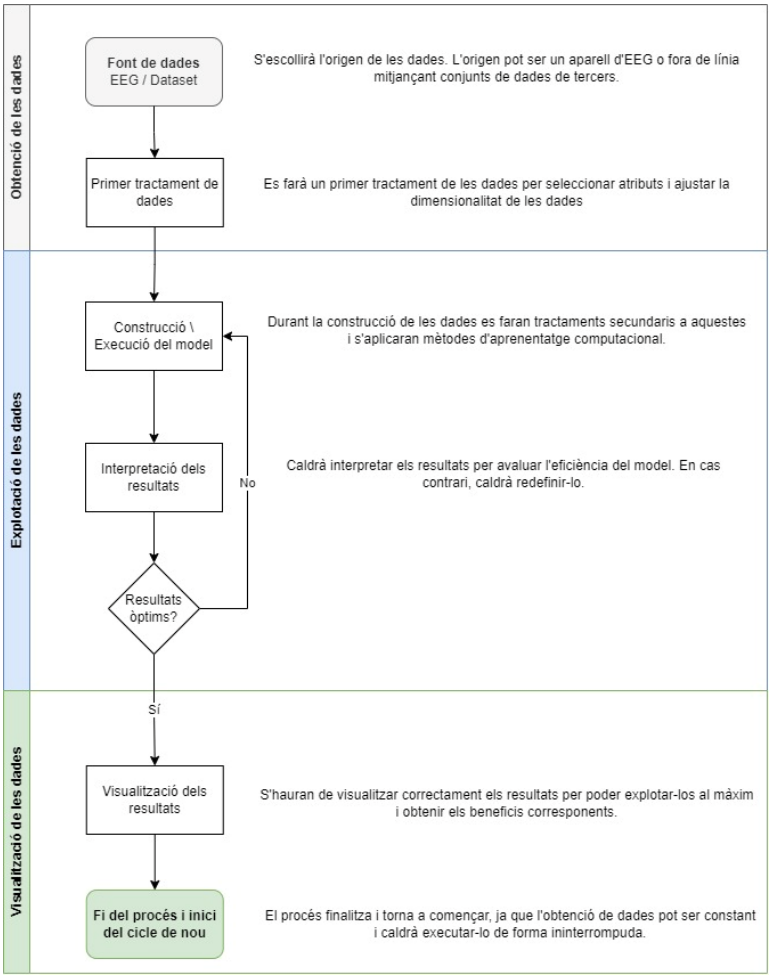


Figura 2. Detall de les fases generals de la solució

Obtenció de les dades

Aquesta fase correspondria a la part d'*Input* de la imatge anterior. Serà molt important seleccionar les dades adequades i fer el primer tractament d'aquestes. Altrament, la construcció del model de la fase posterior serà més complex.

Explotació de les dades

Aquesta part correspondria a la de *Machine learning*, i correspondrà al posterior tractament de les dades per acabar de fer els filtratges definitius per eliminar artefactes. Posteriorment, caldrà aplicar algorismes d'aprenentatge supervisat o no supervisat per poder obtenir els resultats, que s'avaluaran per esbrinar l'eficiència del model.

Visualització dels resultats

Per acabar, caldrà visualitzar els resultats correctament. A banda d'indicar els esdeveniments trobats, serà necessari descriure l'eficiència del model mitjançant taules de confusió, corbes ROC [6], o altres mètriques que demostrin la fiabilitat del producte.

2.3. Context i justificació del Treball

El projecte esmentat anteriorment és un projecte d'intel·ligència artificial on es farà servir algun tipus d'aprenentatge computacional per poder classificar senyals d'encefalogrames i poder fer prediccions. En principi, es faran servir tècniques d'aprenentatge supervisat, com per exemple, les xarxes neurals, KNN o SVM. Així mateix, el projecte englobarà diferents àrees científiques, com l'aprenentatge computacional, física, o matemàtiques. Malgrat l'estreta relació de la temàtica del projecte amb la medicina, no està inclòs en l'abast d'aquest treballar amb aquesta branca, ja que estaria fora de l'abast del projecte de final de grau.

Si descrivim el propòsit general d'aquest projecte, aquest estaria lligat a la tendència actual que hi ha amb la intel·ligència artificial, on nombrosos estudis intenten esbrinar mitjançant l'aprenentatge computacional els patrons que fa el cervell amb l'objectiu d'interpretar-los i poder fer accions que millorin la salut de les persones. Per exemple, retornar la mobilitat a persones tetraplègiques, capacitat de fer servir mentalment extremitats artificials, o comunicació per a persones sordmudes, serien exemples d'ús d'aquests estudis. Així mateix, els sistemes d'aquest tipus podrien ser útils per altres usos, com la gestió de dispositius en remot, o jugar als videojocs sense la necessitat de dispositius físics.

En definitiva, la idea, o **problemàtica a resoldre** d'aquest projecte és poder detectar correctament aquests patrons i interpretar-los adequadament, ja que malgrat que s'han fet nombrosos estudis, no tots els classificadors aconsegueixen percentatges d'èxit elevats.

2.4. Objectius del Treball

Els objectius del treball es classificaran en quatre categories, generals, funcionals, de projecte, i personals.

2.4.1. Generals

Un dels propòsits generals del projecte és poder predir els moviments fets o que està pensant una persona. D'aquesta manera, l'objectiu general és poder construir un producte que pugui interpretar correctament els senyals d'encefalogrames per poder predir els esdeveniments.

2.4.2. Funcionals

Els objectius funcionals serien aquells objectius més específics que ens permetrien poder aconseguir l'objectiu general. Aquests objectius seran els següents:

- Carregar els conjunts de dades que contenen els senyals d'EEG [2].
- Poder llegir i interpretar correctament els senyals d'EEG [2].
- Fer el tractament adequat de les dades dels senyals.
- Aplicació d'algorismes d'aprenentatge computacional adients a les dades que s'estan tractant.
- De manera opcional, poder oferir alguna mena d'integració amb altres sistemes per aprofitar el potencial del model.
- Poder fer prediccions i interpretar aquestes adequadament. D'aquesta manera, poder saber quan s'ha fet o pensat un moviment voluntari per una persona.

2.4.3. Projecte

Per poder assolir l'objectiu principal i els funcionals, haurem d'assolir també una sèrie d'objectius de projecte. Concretament, seran els següents:

- Aconseguir una font de dades fiable i amb les característiques adients per poder fer prediccions.
- Construir una aplicació amb llenguatge Python o Java que permeti el tractament de dades i l'aplicació d'algorismes d'aprenentatge computacional.
- Dissenyar una interfície o similar amb els llenguatges descrits anteriorment per poder mostrar els resultats del model. Aquesta part pot ser la sortida d'algun script o programa ja existent per aquest fet.
- Poder entregar tots els lliurables requerits en l'assignatura amb el corresponent informe d'avanç.
- Redactar la memòria del projecte segons els estàndards de la UOC i els requisits de l'assignatura.

2.4.4. Personals

Per últim, cal destacar aquells objectius personals que van més enllà de l'entrega del TFG. En aquest sentit, destacaríem els següents:

- Aprendre com funcionen els senyals d'EEG [2].
- Aprendre conceptes físics relacionats amb els senyals d'EEG [2].
- Poder aplicar tots els conceptes que s'han après de les assignatures de programació i gestió de projectes.
- Poder entregar un producte funcional amb una documentació correcte. Si fos possible, que el producte fos publicat al repositori de la UOC.

- Millorar el coneixement assolit en l'àrea d'intel·ligència artificial i que aquest serveixi per millorar la carrera professional de l'estudiant.

2.5. Enfocament i mètode seguit

Tal com s'ha pogut veure anteriorment, aquest TFG s'ha dividit en tres parts: obtenció de les dades, explotació de les dades i visualització de les dades. A continuació explicarem breument l'enfocament i mètode seguit en cadascuna d'aquestes parts.

2.5.1. Obtenció de les dades

Durant les primeres fases del projecte, es va avaluar l'adquisició d'un lector de senyals EEG [2]. No obstant això, es va decidir fer servir conjunts de dades ja existents, ja que treballar amb un lector podria endarrerir el projecte. Per tant, es van escollir una sèrie de conjunts de dades per tal que el producte fos capaç de realitzar prediccions adients. En aquest cas, es van escollir conjunts de dades de la BCI Competition [1], que estan verificades per estudis anteriors i contenen la informació necessària per poder desenvolupar el projecte. En aquest sentit, es faran servir els conjunts de dades següents:

- BCI Competition 2A [12]: Imaginació motora amb classificació multiclasse.
- BCI Competition 2B [13]: Imaginació motora amb classificació binària.

2.5.2. Explotació de les dades

Durant aquesta fase es van tractar tots els conjunts de dades per tal d'extreure la informació necessària per predir els esdeveniments. A grans trets, es van fer les següents tasques:

- Creació d'una interfície per carregar les dades i aplicar opcions sobre aquestes.
- Filtratge dels senyals.
- Reducció de la dimensionalitat.
- Creació dels conjunts de dades d'entrenament i prova.
- Tractament de valors nuls.
- Normalització de les dades.
- Altres: Aplicació d'extracció de components principals (PCA [16]) i independents (ICA [17]).
- Aplicació d'algorismes d'aprenentatge computacional.
- Càlcul de mètriques com matrius de confusió, corbes ROC [6], etc.

2.5.3. Visualització de les dades

Finalment, en aquesta fase es va crear una interfície per poder visualitzar els resultats. Concretament, es va complementar la interfície creada a la fase anterior per veure mètriques com la matriu de confusió, coeficient Kappa [7], precisió etc. A més a més, la interfície permet exportar en PDF els resultats obtinguts.

2.6. Planificació del Treball

En aquesta secció no mostrarem únicament el calendari amb les tasques del projecte, sinó que parlarem d'altres conceptes que s'han planificat per poder entregar el TFG. En concret, tractarem els requisits, les fites del projecte, i els riscos. En alguns casos, es farà llistat sense entrar al detall, ja que això es farà en les seccions posteriors.

2.6.1. Requisits

Els requisits generals que es van identificar per poder executar el projecte van ser els següents:

- Disposar d'una font de dades de senyals d'EEG que continguin esdeveniments com els que hem descrit anteriorment.
- Disposar del coneixement físics i matemàtics necessaris per a la interpretació dels senyals.
- Disposar dels entorns de desenvolupament necessaris per poder programar l'aplicació. Per exemple, entorns del tipus PyCharm o Eclipse.
- Disposar del maquinari adient per poder processar les dades. Per exemple, en cas de processar un volum important de característiques, seria necessari un entorn amb molta computació i memòria.

La satisfacció dels requisits anteriors garantiran l'èxit del projecte. Ara bé, caldrà tenir en compte altres temes com els riscos, que els comentarem més endavant.

2.6.2. Fites i calendari

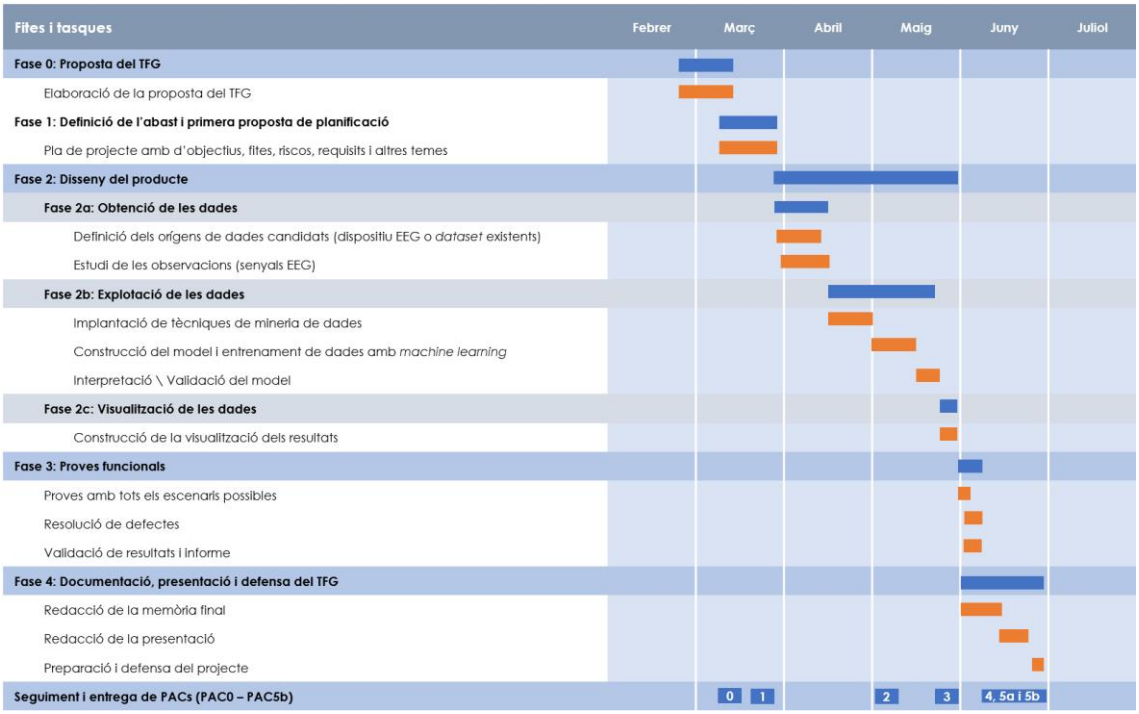
Les fites es caracteritzen per l'entrega de les proves d'avaluació continuada i els lliurables que es van lliurar en el projecte. Com es podrà veure, a partir de la fase 2, ja va començar a entregar lliurables. En qualsevol cas, les fites del projecte van ser les següents:

Fita	**Data d'entrega aprox.**	**Lliurable entregat**
Fase 0: Proposta del TFG	11/03/2024	No
Fase 1: Definició de l'abast i primera proposta de planificació	26/03/2024	No
Fase 2: Disseny del producte	**31/05/2024**	No
Fase 2a: Obtenció de les dades	15/04/2024	No
Fase 2b: Explotació de les dades	20/05/2024	Sí, PAC2
Fase 2c: Visualització de les dades	31/05/2024	No
Fase 3: Proves funcionals	**09/06/2024**	Sí, PAC3
Fase 4: Documentació del projecte	**30/06/2024**	Sí, la PAC4
Fase 4: Memòria final	16/06/2024	Sí, PAC4
Fase 4: Presentació	23/06/2024	Sí, PAC5a
Fase 4: Preparació i defensa	30/06/2024	Sí, PAC5b
Seguiment i entrega de PACs (PAC0 – PAC5b)	Durant tot el projecte	

Taula 1. Taula de fites del projecte

S'han marcat en negreta les fites que eren més significatives del projecte. D'altra banda, en el cas de la fase 4, s'han tipificat les dates d'entrega de la documentació que hi ha definida a l'aula.

Finalment, mostrarem el calendari amb les fites anteriors i les subtasques corresponents a cada fase.



Taula 2. Calendari del projecte

Com s'ha pogut veure, les fases del calendari i les fites encaixen amb les fases generals indicades en el punt 2 d'aquest document.

2.6.3. Riscos

Finalitzarem aquesta secció amb l'anàlisi dels riscos que es va fer durant tot el projecte. A la taula següent, podrem veure els riscos que es van planificar a l'inici del projecte amb el nivell d'aquell moment.

Codi	Nom	Causa	Descripció	Conseqüència	Possible mitigació	Nivell inicial	Situació actual			Observacions
							Prob.	Impacte	Nivell	
R01	Manca de coneixement d'EEG	Desconeixement dels sistemes EEG	Desconeixement de conceptes físics i tècnics de sistemes EEG	Endarreriment de totes les fases del projecte.	Cercar informació de sistemes existents i teoria per poder entendre més ràpidament els sistemes EEG	Mitjà	Mitjana	Alt	Alt	S'ha fet un repàs del funcionament dels sistemes EEG
R02	Adquisició d'un aparell d'EEG	Necessitat d'un aparell d'EEG per obtenir dades	L'obtenció d'un aparell d'EEG pot ser complicat i costos.	No es podran obtenir dades	No adquirir l'aparell	Baix	Baixa	Baix	Baix	Finalment, es faran servir conjunts de dades existents.
R03	Integració amb un aparell EEG	Necessitat d'integrar-se amb un aparell EEG per obtenir dades	La integració amb un aparell d'EEG pot ser complexa i impactar en el temps	No es podran obtenir dades	No adquirir l'aparell	Baix	Baixa	Baix	Baix	Segurament que es desestimarà i es faran servir conjunts de dades existents
R04	Fonts d'informació	Necessitat de fons d'informació per no fer servir aparells EEG	Les fonts de dades poden ser poc fiables i no garantir l'objectiu	Resultats poc fiables i endarreriment.	Consultar amb el professor responsable quines fonts d'informació serien les més adequades, o buscar-ne per internet	Baix	Baixa	Alta	Mitjà	S'ha acordat amb el professor responsable les fonts que es faran servir.
R05	Desenvolupament del model	Necessitat d'un model per aplicar l'aprenentatge computacional	La construcció del model des de zero pot impactar en el temps	Les tasques relacionades s'endarreriran	Reutilitzar el codi del TFG anterior	Alt	Alta	Alt	Alt	Acordat amb el professor responsable.
R06	Resultats incorrectes	Avaluació dels resultats del model	El model obtingut pot provocar falsos positius si no es validen tots els escenaris	Els resultats obtinguts seran incorrectes	Fer més proves i comparar els resultats	Baix	Alta	Alt	Alt	Ja es va produir el semestre anterior.
R07	Interpretacions incorrectes	Avaluació dels resultats del model	La interpretació dels resultats pot ser ineficient, cosa que pot provocar falsos positius	Els resultats obtinguts seran incorrectes	Acotar la interpretació o depurar-la perquè sigui més eficient	Mitjà	Mitjana	Alt	Mitjà	El producte no donarà la qualitat esperada.
R08	Interfície gràfica poc acurada	Creació de la interfície de visualització dels resultats	El mòdul de visualització pot ser poc entenedor o compatible amb poques plataformes	No es podran veure els resultats	Fer servir patrons o llibreries existents per millorar la visualització	Baix	Mitjana	Alt	Alt	Per exemple, crear un PDF amb dades i gràfics.
R09	Integració amb altres sistemes	Integrar el sistema amb altres serveis per compartir els resultats	La integració de resultats amb tercers pot impactar en el temps	No es podrà fer servir l'aplicació per tercers	Canviar l'abast per no implantar aquest requisit	Baix	Mitjana	Baix	Baix	L'impacte es reduiria si es treu de l'abast aquest requisit
R10	Manca de temps per endarreriment d'algunes fases	Endarreriment en finalitzar algunes fases	El retard en la finalització d'algunes tasques pot portar a no finalitzar el projecte amb les dates acordades	No es podrà finalitzar algunes tasques o el projecte segons la planificació	Avançar les fases posteriors sempre i quan sigui viable	Alt	Alta	Alt	Alt	Actualment, s'ha produit aquest risc

Taula 3. Taula de riscos del projecte

Exceptuant els riscs R06 i R10, tots els altres es van poder mitigar i no van impactar en el projecte. Sobre els riscos que varen impactar, es va comunicar al professor responsable del corresponent endarreriment i l'impacte que tenien aquests sobre el lliurable d'aquell moment.

2.7. Breu sumari de contribucions i productes obtinguts

Els productes obtinguts en aquest TFG són els de les següents seccions.

2.7.1. Dades de lectures de senyals EEG

S'han obtingut dades de senyals EEG en format GDF [8] que contenen diferents tests amb informació necessària pel model que s'havia construir en aquest projecte. Concretament, contenen informació detallada dels esdeveniments i per a cada tipus de conjunt de dades, hi ha una documentació on es detalla com s'han realitzat els tests i la informació dels esdeveniments.

2.7.2. Aplicació en Python per tractar els fitxers GDF i predir els esdeveniments

S'ha desenvolupat una aplicació en Python que és capaç de carregar, analitzar i predir els esdeveniments dels fitxers GDF [8] relacionats amb la imaginació motora. L'aplicació està basada en el model vista controlador (MVC) i disposa d'una interfície gràfica per carregar els conjunts de dades, aplicar una sèrie d'opcions i finalment, executar les prediccions per obtenir els resultats.

2.7.3. Sortida en Python i PDF dels resultats obtinguts

Per acabar, l'aplicació anterior mostrarà els resultats obtinguts amb la mateixa interfície que s'ha creat en Python. Els resultats mostraran una sèrie de mètriques com matrius de confusió o corbes ROC [6] i es podran guardar els resultats en format PDF. Així mateix, els resultats incorporaran una comparació entre les classes originals i les prediccions fetes pel classificador corresponent.

2.7.4. Interfície gràfica per poder executar l'aplicació

Per poder fer funcionar l'aplicació amb els requisits que hem comentat anteriorment, s'ha dissenyat una interfície gràfica que permet a l'usuari interaccionar amb l'aplicació fàcilment i sense necessitat d'escriure comandes per una consola. D'aquesta manera, l'obtenció dels resultats que hem comentat en el punt anterior és més ràpida i còmoda, cosa que proporciona un entorn orientat a l'usuari. Per acabar, la interfície desenvolupada no té dependència del sistema operatiu, així que hauria de funcionar en qualsevol plataforma que tingués instal·lat el programari Python.

2.8. Breu descripció dels altres capítols de la memòria

Es finalitzarà aquesta secció amb la descripció dels capítols d'aquesta memòria.

2.8.1. Resum

Aquest apartat fa una primera aproximació del projecte sense entrar al detall però amb la identificació dels aspectes més bàsics.

2.8.2. Introducció

Introdueix la temàtica principal del projecte i entra al detall amb alguns diagrames o imatges que mostrin les fases de la solució que s'entregarà. Concretament, aquest apartat disposarà dels següents subapartats.

- Context i justificació del treball: fa una descripció de la temàtica que està basat el projecte i justifica la problemàtica a resoldre

- Objectius del treball: es realitza una classificació dels objectius del projecte classificats segons la seva tipologia.
- Enfocament i mètode seguit: entra al detall de com s'ha orientat el projecte i descriu les accions realitzades per obtenir el resultat final.
- Planificació del treball: es descriuen les planificacions realitzades. A banda de les fites i calendari es descriuen requisits i riscos.
- Breu sumari de contribucions i productes obtinguts: descriu l'aplicació realitzada amb la seva estructura i composició.

2.8.3. Estat de l'art

S'analitza la situació actual d'altres iniciatives semblant a aquestes i els aspectes positius aportats en aquest projecte.

2.8.4. Metodologia

S'expliquen quines metodologies s'han aplicat, la seva justificació i es conclou amb els aspectes més importants aportats del seguiment d'aquestes.

2.8.5. Resultats

De manera detallada, es mostren els resultats del treball i es relacionen a cada fase de la planificació mencionada anteriorment.

2.8.6. Discussió

Es valoren els resultats obtinguts per tal d'esbrinar si han resolt la problemàtica plantejada i si aquests s'han ajustat a l'abast de la temàtica del TFG.

2.8.7. Conclusions

En aquesta secció es fa una cloenda dels aspectes més importants, els aspectes que no s'han pogut treballar, i es conclourà amb el detall de compliment de la planificació i l'eficiència d'aquesta.

2.8.8. Glossari

Inclourà els termes i acrònims fets servir en aquest document.

2.8.9. Bibliografia

Es farà un llistat de les fonts d'informació consultades per aquest projecte.

2.8.10. Annexos

Es llisten els documents annexos a la memòria del TFG. Per exemple, manuals de posada en marxa l'aplicació o el codi d'aquesta.

3. Estat de l'art

En aquest apartat farem una breu descripció de la temàtica principal del projecte per tal de posar en context tots els conceptes que s'esmentaran en capítols posteriors. De totes maneres, no entrarem al detall dels temes principals, sinó que farem una descripció dels conceptes més importants per assegurar que en els futurs capítols no hi ha cap aspecte desconegut o ambigu.

3.1. Electroencefalografia

Electroencefalografia o electroencefalograma (EEG) [2], és la tècnica no invasiva que es fa servir per registrar l'activitat elèctrica del cervell en un temps i condicions determinades. En aquest sentit, es capten les ondes cerebrals, que tenen la funció de transportar informació dins del cervell mitjançant el sistema nerviós. Tota aquesta informació es captura mitjançant un lector de senyals EEG [2], el qual mitjançant uns elèctrodes que es connecten al cap del pacient, permet capturar la informació per la seva posterior anàlisis mitjançant un plànol amb coordenades cartesianes.

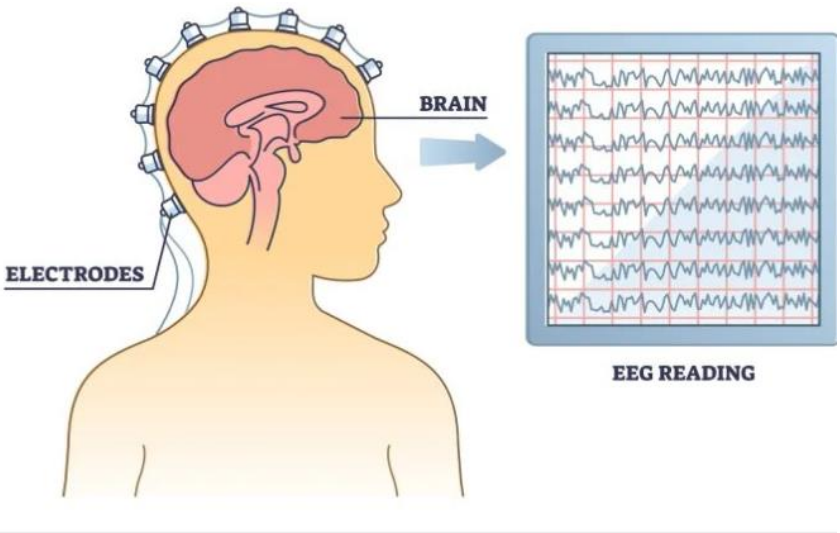


Figura 3. Imatge d'un lector de senyals EEG

3.1.1. Tipus d'ones

En funció de l'activitat cerebral i l'estat del pacient, les ones que viatjaran pel sistema nerviós tindran una freqüència i una amplitud diferent. Normalment, les freqüències varien entre 1 i 30 Hz, mentre que les amplituds entre 10 i 100 μV. Els tipus d'ones es classifiquen segons la seva banda de freqüència, així que podrem distingir entre cinc grups.

Banda de freqüència	Freqüència (Hz)
Alpha	8 – 12
Beta	12 – 30
Delta	0.5 – 4

Theta	4 – 8
Gamma	> 30

Taula 4. Taula de bandes d'ona

Les ones Alpha estarien relacionades amb els estats de relaxació sense que l'individu estigui concentrat i poden tenir una amplitud d'ona de fins a 60 μV. D'altra banda, les ones Beta es podrien capturar quan una persona està desperta i concentrada. Acostumen a ser ones amb una amplitud inferior als 30 μV. Pel que fa a les ones Theta, poden tenir una gran amplitud i estan relacionades amb les primeres etapes del son. Aquest també seria el mateix cas que les ones Delta, que estarien presents en etapes de somni profundes. Finalment, tenim les ones Gamma, que tenen una alta freqüència perquè es poden trobar en etapes d'alta intensitat del cervell. A causa de la seva alta freqüència i baixa amplitud, són difícils d'analitzar, ja que poden tenir molt de soroll.

A continuació, posarem una imatge d'exemple on es pot veure cada tipus d'ona.

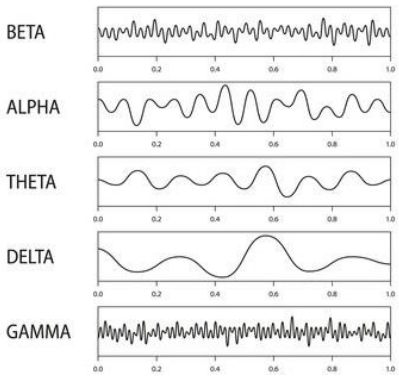


Figura 4. Imatge d'exemple dels tipus d'ones dels senyals d'EEG

3.1.2. Elèctrodes

Per poder capturar les ones adequadament, caldrà un lector de senyals EEG que estarà compost d'una sèrie d'elèctrodes que s'hauran de posar al cap del pacient. Es descriurà el sistema 10-20 [10], que és el que s'ha fet servir pels conjunts de dades d'aquest projecte. Sense entrar al detall, el sistema 10-20 [10] indica que els elèctrodes han d'estar separats entre ells entre un 10 o 20% d'una distància prèviament prefixada.

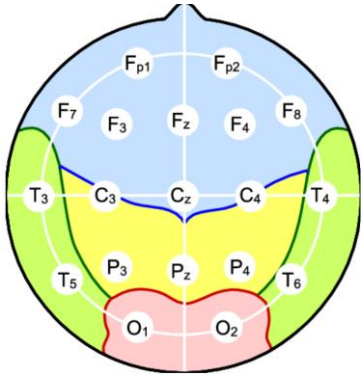


Figura 5. Imatge del sistema 10-20 de posició d'elèctrodes

3.1.3. Soroll i artefactes

Cal destacar que durant les sessions de lectures de senyals EEG [2], es poden produir lectures referents a senyals que contaminen el senyal original. Aquest soroll, també conegut com artefactes, es pot produir pels següents motius.

- Moviments oculars o també anomenat Electrooculograma (EOG) [14].
- Activitat muscular o també anomenada Electromiografia (EMG) [18].
- Activitat cardiovascular o també anomenada Electrocardiografia (ECG) [19].

A continuació mostrem una imatge d'exemple on es pot veure artefactes marcats en blau relacionats amb moviments oculars.

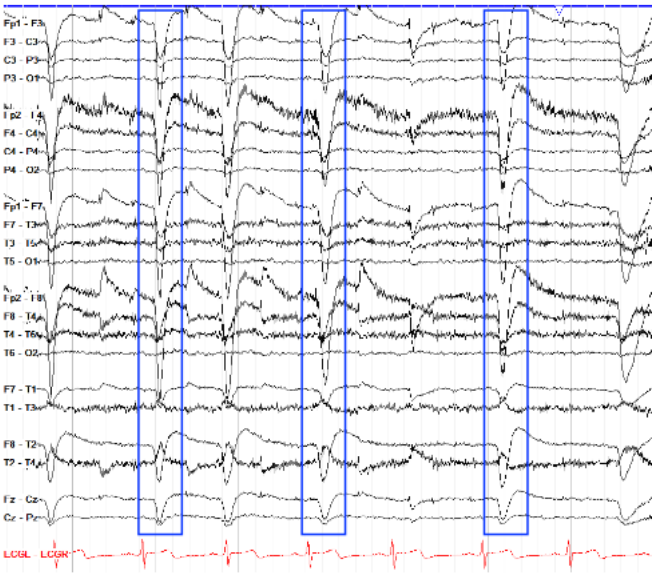


Figura 6. Imatge d'exemple d'artefactes per moviments oculars

3.1.4. Usos de l'encefalografia

Per acabar, farem un petit llistat dels usos més freqüents de l'encefalografia. Bàsicament, es fa servir per detectar alguns trastorns com els següents:

- Danys cerebrals
- Epilèpsia
- Demències
- Trastorns de la son
- Inflamacions cerebrals
- Altres patologies cerebrals

3.2. Brain Computer Interface (BCI)

Un cop hem entrat al detall de què és l'encefalografia, podem descriure el següent terme, que és el de Brain Computer Interface (BCI) [9]. Una interfície cervell-ordinador és un sistema que s'encarrega de capturar ones cerebrals d'una persona i s'envien a un dispositiu extern per la seva anàlisi i posterior ús, com per exemple, activar moviments d'extremitats. La majoria d'estudis es centren a aconseguir dispositius que millorin la vida dels pacients amb problemes de mobilitat, cosa que implica que a banda d'un dispositiu que capturi i processi els senyals, haurà d'haver-hi un tercer aparell que s'encarregui de

fer els moviments que la persona discapacitada no pot realitzar. No obstant a això, també hi ha experiments relacionat amb jocs o detecció de mentides.

Existeixen dos tipus d'interfícies cervell-ordinador, les invasives i les no invasives. Les primeres s'anomenen invasives perquè mitjançant la cirurgia, s'han d'instal·lar els sensors dins del cervell de la persona. D'altra banda, les no invasives es caracteritzen per portar els sensors en la superfície del cuir cabellut de la persona.

A continuació, mostrarem una imatge per resumir tot el que hem parlat anteriorment.

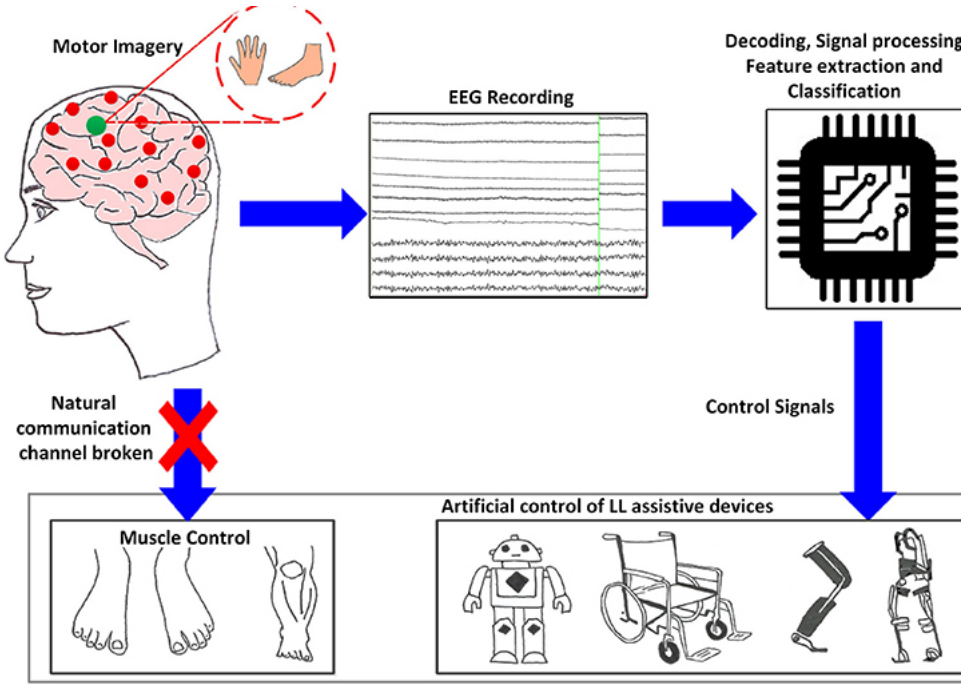


Figura 7. Exemple de funcionament d'una interfície cervell-ordinador.
Font: www.frontiersin.org

En aquest TFG es faran servir conjunts de dades de lectures no invasives i basades en imaginació motora. És a dir, la persona havia de pensar un moviment d'una extremitat en una sessió de lectura amb elèctrodes en el seu cuir cabellut.

3.3. EEG i BCI en el TFG

Finalitzarem aquesta secció amb els aspectes més importants d'aquest projecte amb relació al que s'ha explicat anteriorment.

- Existeix una font de dades que s'ha fet mitjançant una interfície cervell-ordinador no invasiva.
- Es realitza la càrrega de dades i es filtra segons el tipus d'ona. En el filtratge quedaran fora els artefactes.
- Es detecten i s'analitzen els esdeveniments que prèviament s'han obtingut d'un conjunt de dades d'entrenament.
- Es filtren els sensors que han capturat les dades per millorar els resultats.
- S'apliquen algorismes d'aprenentatge computacional per descodificar les dades i així predir els esdeveniments.
- Es produeix un resultat que pot ser utilitzat per un tercer per altres usos.

Per tant, com es podrà veure en apartats posteriors, s'ha aconseguit construir una mena d'interfície cervell-ordinador com les que s'han explicat en la secció anterior. No obstant

això, aquesta interfície no s'ha connectat a cap altre dispositiu o servei que pugui fer-ne ús, ja que únicament donarà servei en aquest projecte.

4. Metodologia

En aquest apartat definirem la metodologia que s'ha fet servir en aquest projecte per aconseguir el producte final. De la mateixa manera que hem fet a les seccions anteriors, mencionarem les tres fases principals i n'afegirem de complementàries, com seria la gestió de projectes. Per acabar, farem una descripció dels conceptes que s'hauran tractat en cada secció sense entrar al detall.

4.1. Gestió del projecte

En aquest TFG s'han seguit les recomanacions dels conceptes apresos a l'assignatura de Gestió de Projectes [3] de la UOC. En aquest sentit, s'ha seguit un model híbrid, on es va definir una planificació en cascada. Ara bé, també s'ha aplicat un cicle de vida incremental a cada fase del projecte, cosa que ens ha permès definir la solució de manera parcial i aplicar canvis quan ha sigut necessari. A més, a cada fase hem obtingut una versió del producte que ens ha permès refinar els requisits i el disseny del producte. D'aquesta manera, a les proves d'avaluació continuades dos i tres es van poder entregar prototips de l'aplicació que hi havia en aquell moment.

4.2. Disseny de la solució

En aquesta secció on descriurem les tres fases principals al detall i es posaran evidències.

4.2.1. Obtenció de les dades

En aquest projecte s'han fet servir dades de dos *datasets* diferents per poder avaluar el funcionament de l'eina desenvolupada. Ambdós conjunts de dades provenen de la BCI Competition [1], que es tracta d'una mena de competició a internet per tal de validar processament de senyals i mètodes de classificació per interfícies cervell-computadora. En el nostre cas, vàrem escollir la competició IV, i més concretament, els conjunts de dades 2A i 2B que estan basats en dades d'imaginació motora. Abans d'entrar al detall d'aquests conjunts de dades, farem una descripció del tipus de fitxers en el que ens hem trobat les dades. Seguidament, explicarem les altres tasques que es van fer en aquesta fase.

Fitxers GDF

En els dos conjunts de dades ens hem trobat els conjunts d'entrenament i avaluació en fitxers del format GDF [22]. Aquests fitxers estan dissenyats per guardar informació mèdica de senyals del tipus EEG, ECG, EMG, etc. S'utilitzen per treballar amb interfícies cervell-ordinador i es van crear per resoldre les limitacions dels fitxers EDF [11], versió Europea dels fitxers GDF creada per un grup d'enginyers mèdics. No entrarem al detall d'aquests tipus de fitxers, sinó que llistarem les característiques que ens han aportat en aquest projecte:

- Contenen les dades de les lectures EEG i EOG realitzades als individus que van participar en els tests.
- Contenen atributs com la freqüència, el temps, esdeveniments, etc.
- Es poden obrir amb aplicacions com el EEGLAB o llibreries de Python MNE.
- Permeten visualitzar gràficament les lectures realitzades.

A la següent pantalla mostrarem un fitxer del tipus GDF obert amb l'aplicació EEGLAB:

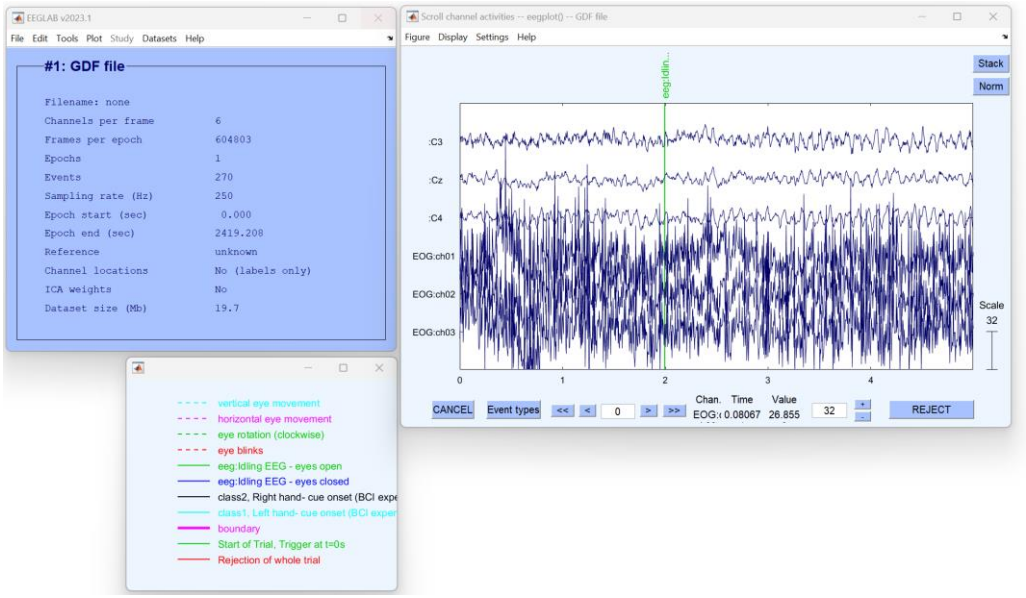


Figura 8. Exemple de fitxer GDF obert amb l'aplicació EEGLAB

Fitxers Matlab

Així mateix, s'ha treballat amb fitxers de l'aplicació Matlab per tal d'avaluar els resultats del model. Aquests fitxers contenen les correctes prediccions de cada conjunt de dades i s'han comparat amb les prediccions realitzades pel nostre del model per generar les mètriques que ens indiquen la qualitat d'aquest.

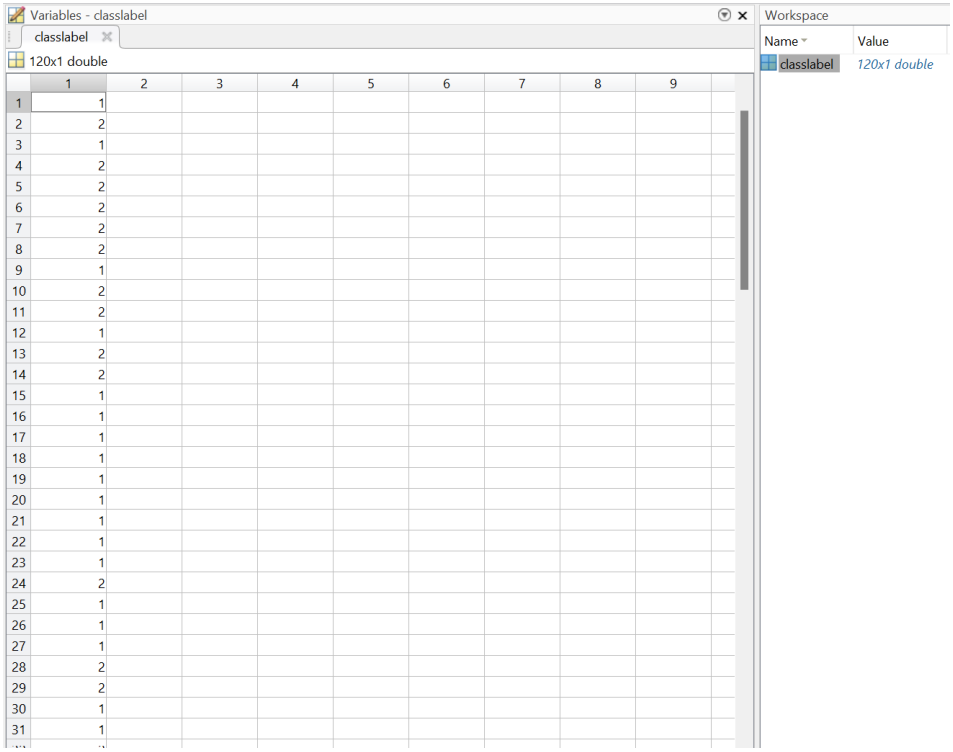


Figura 9. Exemple de fitxer de Matlab

Un cop s'han descrit els tipus de fitxers, es procedirà a descriure els dos tipus de conjunts de dades. Així mateix, a l'annex número 2 d'aquest document hi ha un resum de les característiques principals.

Conjunt de dades 2A

El conjunt de dades 2A [12] és un conjunt de dades de sessions de lectures de senyals de 9 individus on es van avaluar quatre classes de tipus d'imaginació motora. Concretament, van ser moviment de la mà esquerra, dreta, ambdós peus, i llengua. En els tests es van recollir senyals EEG [2] per avaluar les prediccions sobre les classes i EOG [14] per avaluar la detecció i eliminació d'artefactes. Cada sessió de lectura està composta de 6 execucions, i cada execució contindrà 48 tests, que seran 12 per a cadascuna de les quatre classes. D'aquesta manera, hi haurà 288 tests per sessió. Cada test tindrà *N* observacions que correspondran a una classe i s'hauran de predir de manera separada. És a dir, l'objectiu principal serà encertar la classe de cadascun d'aquests 288 tests.

Pel que fa a les execucions dels tests, començaran de la següent forma:

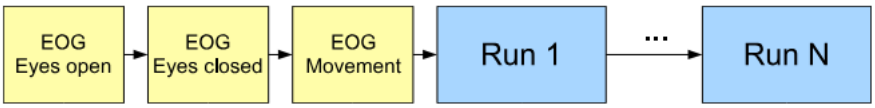


Figura 10. Descripció de les execucions d'una sessió de lectura dels datasets 2A

D'altra banda, cada test començarà de la següent manera:

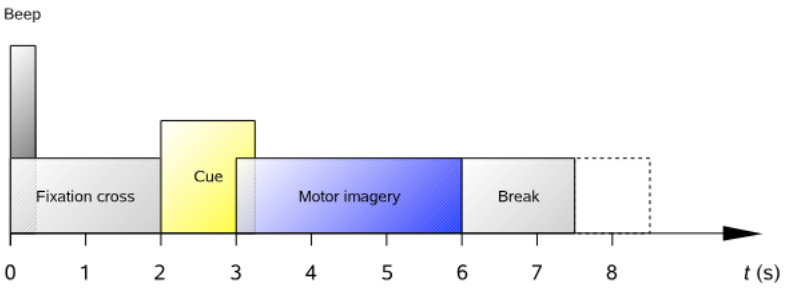


Figura 11. Composició de cada *test* del conjunt de dades 2A

Com s'ha pogut veure, a l'inici de cada test hi ha un senyal sonor, seguit d'una imatge visual d'una creu. Posteriorment, surt una imatge corresponent a una de les quatre classes i seguidament hi ha el temps on l'individu ha d'imaginar la classe que ha vist prèviament. Finalment, abans de començar la següent execució, hi ha un curt temps de descans.

Per a cada sessió, tindrem un fitxer d'entrenament i un fitxer d'avaluació, ambdós fitxers s'han enregistrat de la mateixa manera, però amb la diferència que el fitxer d'avaluació no conté els esdeveniments exactes del que ha imaginat l'individu, sinó que per a cada test conté un esdeveniment que està catalogat com desconegut i que és el que s'ha de predir.

Pel que fa als esdeveniments, tindrem els següents:

Esdeveniment	Descripció
276	Ulls oberts
277	Ulls tancats

768	Inici del test
769	Classe 1, esquerra
770	Classe 2, dreta
771	Classe 3, peu
772	Classe 4, llengua
783	Classe desconeguda
1023	Test cancel·lat
1072	Moviment d'ulls
3276	Inici d'una nova execució

Taula 5. Taula d'esdeveniments del conjunt 2A

Sobre la lectura dels senyals, es van fer servir 22 elèctrodes per captar senyals EEG [2] i 3 per senyals EOG [14]. El sistema de posició dels elèctrodes que es va fer servir és el 10-20 [10].

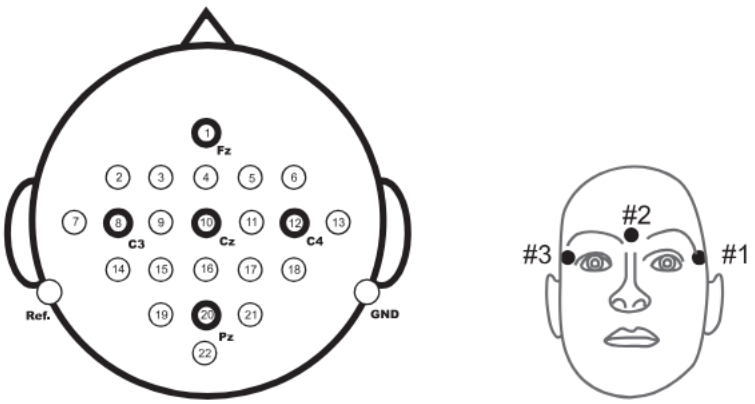


Figura 12. Imatge dels lectors EEG a l'esquerra i lectors de senyals EOG a la dreta

Finalment, els senyals es van capturar amb una freqüència de 250 Hz i filtrades entre 0.5 i 100 Hz. Així mateix, es va aplicar un filtre Notch per reduir el soroll.

Conjunt de dades 2B

Pel que fa al conjunt de dades 2B [13], també es van fer lectures de senyals de 9 individus, però només es van avaluar dues classes d'imaginació motora. Exactament, van ser

moviment de la mà esquerra i la dreta. De la mateixa manera que el cas anterior, en els tests es van recollir senyals EEG [2] per avaluar les prediccions sobre les classes i EOG [14] per avaluar la detecció i eliminació d'artefactes. Cada sessió de lectura està composta de 6 execucions, i cada execució contindrà 20 tests, que seran 10 per a cadascuna de les dues classes. En total, hi haurà 120 tests per sessió. Igual que el cas anterior, cada test s'haurà de predir per separat i contindrà *N observacions. C*ada sessió d'execució començarà de la següent manera:

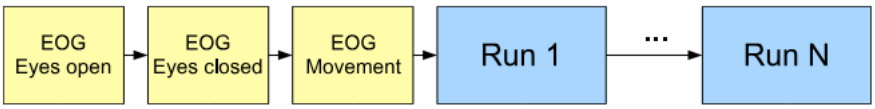


Figura 13. Descripció de les execucions d'una sessió de lectura dels datasets 2B

A diferència del cas anterior, tindrem dos tipus de sessions. Una molt semblant a la del conjunt 2A, i una segona en què l'individu haurà de respondre amb el somriure en funció de l'opció mostrada per pantalla. És a dir, si es presenta una opció que indica la dreta, l'individu haurà de moure el somriure a la dreta. A la següent imatge podrem veure el detall de les dues sessions:

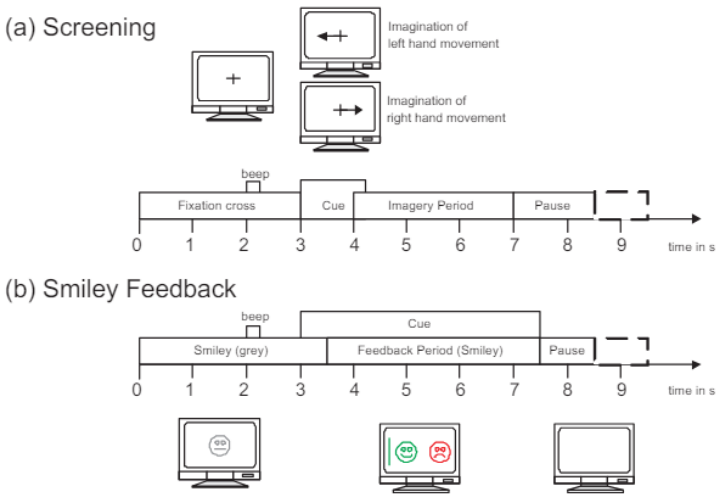


Figura 14. Descripció dels dos tipus de sessions del conjunt de dades 2B

A causa d'això, per a cada individu tindrem cinc tipus de conjunt de dades. Dos del tipus *Screening* i 3 del *Smiley Feedback*. Sobre aquests últims, tindrem un total de quatre execucions amb 20 trials per classe, cosa que farà un total de 160 tests. D'aquests cinc conjunts de dades, els tres primers es consideraran d'entrenament mentre que els dos últims d'avaluació. Pel que fa als esdeveniments, tindrem els següents:

Esdeveniment	Descripció
276	Ulls oberts
277	Ulls tancats

768	Inici del test
769	Classe 1, esquerra
770	Classe 2, dreta
783	Classe desconeguda
1023	Test cancel·lat
1073	Moviment horitzontal d'ulls
1078	Moviments vertical d'ulls
1079	Rotació d'ulls
1081	Parpelleig d'ulls
3276	Inici d'una nova execució

Taula 6. Taula d'esdeveniments del conjunt 2B

Quant a la lectura, es van fer servir 3 elèctrodes per captar senyals EEG [2] i 3 per senyals EOG [14]. Per acabar, igual que en el cas anterior, els senyals es van capturar amb una freqüència de 250 Hz i filtrades entre 0.5 i 100 Hz. Així mateix, es va aplicar un filtre Notch per reduir el soroll.

Càrrega de dades mitjançant la interfície i primer tractament

Un cop descrits els dos conjunts de dades que es van fer servir, es procedirà a descriure la càrrega i primer tractament de dades. En primer lloc, mostrarem un diagrama del flux de càrrega:

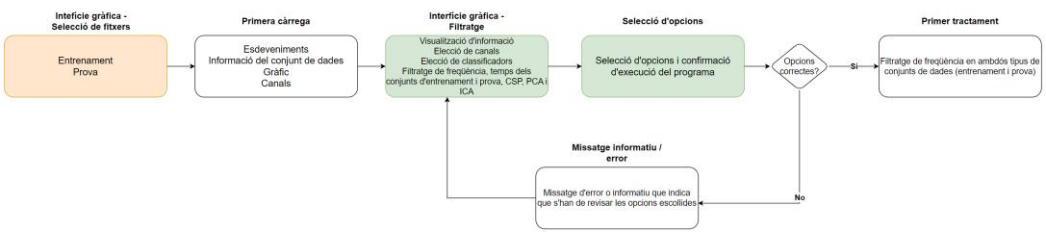


Figura 15. Diagrama de càrrega i primer tractament de dades

La interfície gràfica que veurà l'usuari per carregar i que correspondria a la caixa taronja del diagrama anterior serà la següent:

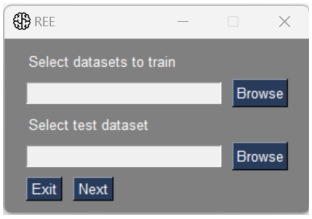


Figura 16. Interfície gràfica per carregar els conjunts de dades

La finestra anterior permetrà a l'usuari carregar *N* conjunts d'entrenament i un únic conjunt de prova.

D'altra banda, a la següent imatge podrem veure la interfície que li permetrà ajustar les opcions abans de fer les prediccions. Aquesta finestra correspondria a les caixes de color verd del diagrama anterior. Per a cada opció, hi haurà una petita descripció a la dreta de la imatge.

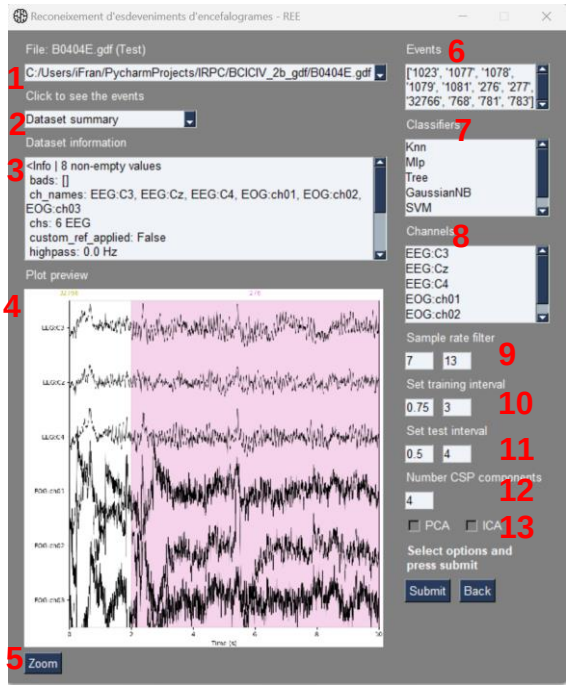


Figura 17: Interfície gràfica per filtrar les dades

1. Llista amb els fitxers carregats.
2. Desplegable per canviar entre la informació del conjunt de dades i els esdeveniments.
3. Camp d'informació del conjunt de dades.
4. Gràfic visual de les dades
5. Botó per veure ampliat el gràfic amb el visualitzador d'imatges del sistema
6. Esdeveniments presents en el conjunt de dades.
7. Llista de classificadors a escollir.
8. Llista de canals a escollir.
9. Filtratge mínim i màxim de banda.
10. Temps mínim i màxim de l'interval de filtratge del conjunt d'entrenament.
11. Temps mínim i màxim de l'interval de filtratge del conjunt de proves.
12. Nombre de components de l'algorisme CSP
13. Caselles opcionals per aplicar PCA o ICA

Pel que fa al primer tractament de les dades, aquest es realitzarà amb els paràmetres que hem indicat anteriorment. Concretament, realitza un filtratge de freqüència amb l'opció número nou del punt anterior. Aquest filtratge es realitza amb la llibreria MNE de Python. A continuació, posem la part del codi que realitza aquest filtratge:

```
data_frame.copy().filter(l_freq=self.__low_r, h_freq=self.__high_r, verbose=True)
```

On **self.__low_r** és el valor inferior i **self.__high_r** el valor superior de la freqüència a filtrar. Per exemple, es podria filtrar el conjunt de dades entre 8 i 28 Hz, cosa podria ajudar a reduir el soroll i artefactes. D'aquesta manera, tindríem els dos conjunts de dades filtrats per la freqüència i enllestits pel segon tractament de dades, que es veurà a continuació.

4.2.2. Explotació de les dades

En aquesta fase l'usuari que executa l'aplicació no veurà res de nou, ja que tot s'executarà en segon pla. No serà fins a la fase de visualització quan podrà veure el resultat de l'execució. Descriurem aquesta fase mitjançant diagrames de flux. En primer lloc, tindríem el tractament posterior al filtratge.

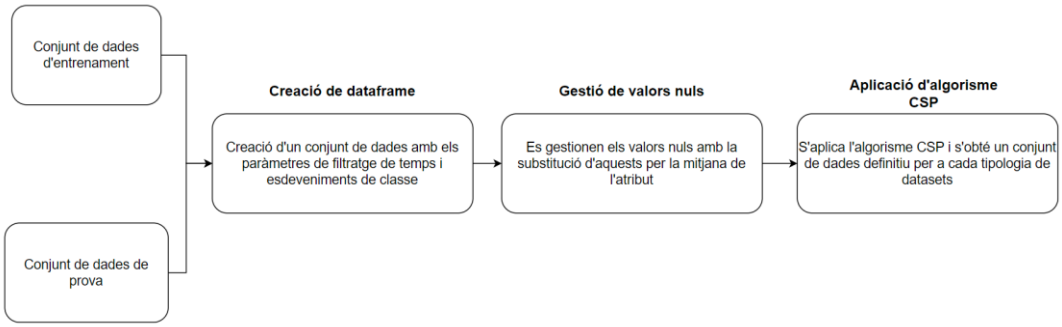


Figura 18. Segon tractament de dades

Ara ja tenim les dades preparades per aplicar els algorismes d'aprenentatge computacional. En aquest sentit, es faran els últims tractaments de les dades abans d'executar els algorismes dels classificadors per obtenir els resultats.

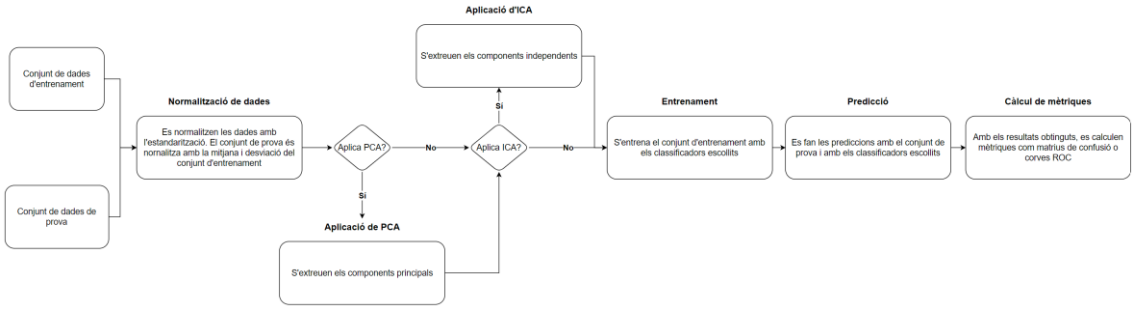


Figura 19. Últim tractament i obtenció de resultats

Al finalitzar aquesta fase ja tenim els resultats de l'execució, els quals s'hauran de mostrar per pantalla.

4.2.3. Visualització de les dades

Finalment, amb els resultats de la fase anterior, es podran veure els resultats per pantalla i exportar-los en format PDF. Tot això es farà amb la interfície gràfica que hem vist anteriorment.

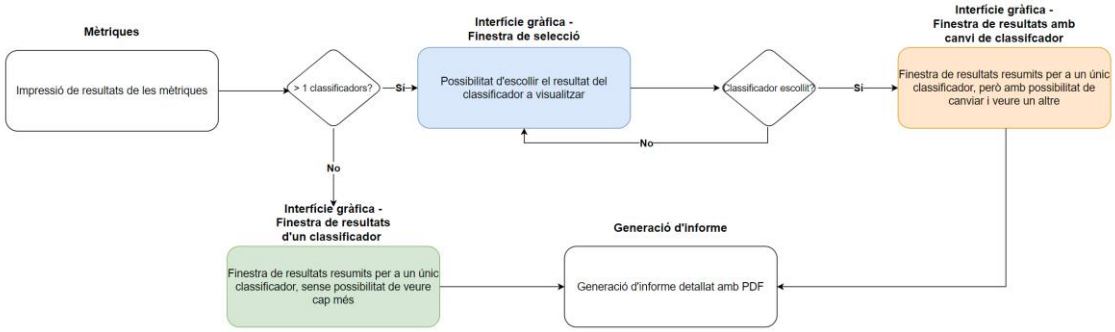


Figura 20. Diagrama d'impressió de resultats

La interfície gràfica corresponent la caixa blava seria la següent:

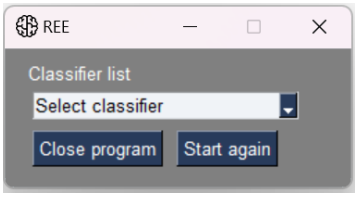


Figura 21. Finestra de selecció de classificador per veure els resultats

Pel que fa a la caixa verda, es veuria de la següent manera:

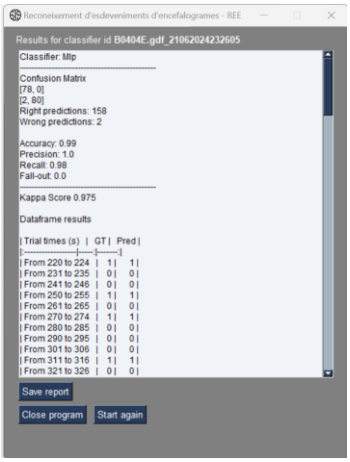


Figura 22. Finestra de resultats sense l'opció de veure altres classificadors

L'identificador que surt a principi de la finestra està compost pel fitxer que conté el conjunt de dades de prova i un codi format per la data d'execució i l'hora.

Per acabar, la caixa taronja, que correspondria als resultats de l'execució de diversos classificadors. Com es podrà veure a la imatge següent, la interfície és pràcticament idèntica a l'anterior, però amb la diferència que hi ha un desplegable per escollir el classificador.

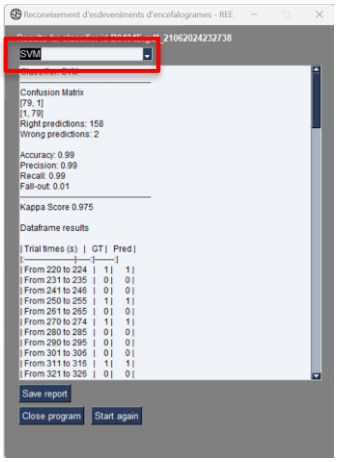


Figura 23. Finestra de resultats amb opció de veure altres classificadors

4.3. Definició de conceptes importants

Per tal de clarificar el flux de funcionament de la solució, a les seccions anteriors s'han posat conceptes en els diagrames que no s'han detallat. A causa d'això, s'ha creat aquesta secció per detallar tots aquells conceptes que s'han indicat sense entrar al detall.

4.3.1. Creació dels conjunts de dades en funció del temps

A la fase d'obtenció de les dades s'ha mencionat el concepte **Temps mínim i màxim de l'interval de filtratge del conjunt d'entrenament** a la figura número 17. Aquesta opció, que també està disponible per al conjunt de proves, serveix per seleccionar les observacions que es capturaran en un temps determinat. Per exemple, si tinguéssim un conjunt de dades del tipus 2A i volguéssim seleccionar només 3 segons dels conjunts d'entrenament i proves, es seleccionarien les observacions que estarien marcades en el període temporal en la següent imatge. Concretament, del requadre amb vores vermelles que s'ha superposat a la imatge.

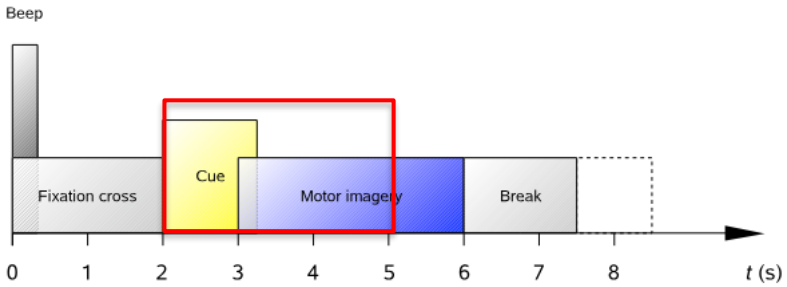


Figura 24. Detall de la selecció de les observacions d'un període de temps determinat

4.3.2. Selecció dels esdeveniments de classe

Tal com s'ha explicat anteriorment, hi ha dos tipus de conjunts de dades, el 2A i el 2B. En el primer cas, hi ha quatre classes per predir, mentre que en el segon, només dues. A causa d'això, l'aplicació farà la selecció de manera automàtica dels següents esdeveniments.

Conjunts de dades 2A:

Esdeveniment	Descripció

769	Classe 1, mà esquerra
770	Classe 2, mà dreta
771	Classe 3, peu
772	Classe 4, llengua

Taula 7. Taula de classes del conjunt 2A

Conjunts de dades 2B:

Esdeveniment	Descripció
769	Classe 1, mà esquerra
770	Classe 2, mà dreta

Taula 8. Taula de classes del conjunt 2B

4.3.3. Gestió de les classes múltiple

A banda de la selecció d'esdeveniments que s'ha indicat anteriorment, durant la fase d'explotació s'ha fet un tractament de les classes per tal de poder generar prediccions i mètriques adequadament. Aquest procés ha sigut més complex pel cas dels conjunts de dades del tipus 2A, que disposa de quatre classes diferents. A causa d'això, s'han creat diversos mètodes diferenciats dels mètodes per a les classes binàries per poder gestionar els *datasets* multiclasse.

4.3.4. Gestió dels valors nuls

Tots els atributs que tinguin valors nuls es substituiran per la mitjana de tots els altres valors que té l'atribut. No obstant a això, si es detecta que hi ha > 50% de valors nuls, l'execució s'atura i es mostra un missatge d'error.

4.3.5. Gestió d'artefactes i soroll dels senyals

L'eliminació dels artefactes i soroll es podrà fer mitjançant el filtratge de freqüència que es pot veure marcat amb l'opció número 9 de la figura 15. Per exemple, si seleccionem una freqüència d'entre 8 i 28 Hz, s'agafaran les ones de les bandes Alfa i Beta, que és quan l'individu està relaxat i concentrat.

4.3.6. Common spatial pattern

En primer lloc, farem una breu descripció de la tècnica Common spatial pattern i després descriurem com funciona en la nostra solució.

El mètode Common spatial pattern (CSP) [15] és una tècnica molt utilitzada en les interfícies cervell-ordinador per extreure característiques d'un senyal multicanal mitjançant filtres espacials que diferencien entre classes segons les seves matrius de covariància. Fa servir mètodes de multi variància per transformar les dades des del domini temporal a

l'espacial amb l'aplicació de filtres que aconsegueixen nous components que són combinacions lineals dels canals originals. D'aquesta manera, si tenim dues classes en un conjunt de dades de senyals EEG [2], CSP [15] trobarà un conjunt de valors que discriminaran al màxim les dues classes dins del conjunt de dades EEG.

No entrarem al detall matemàtic d'aquesta tècnica, sinó que mostrarem en un diagrama com s'ha fet servir en el nostre projecte.

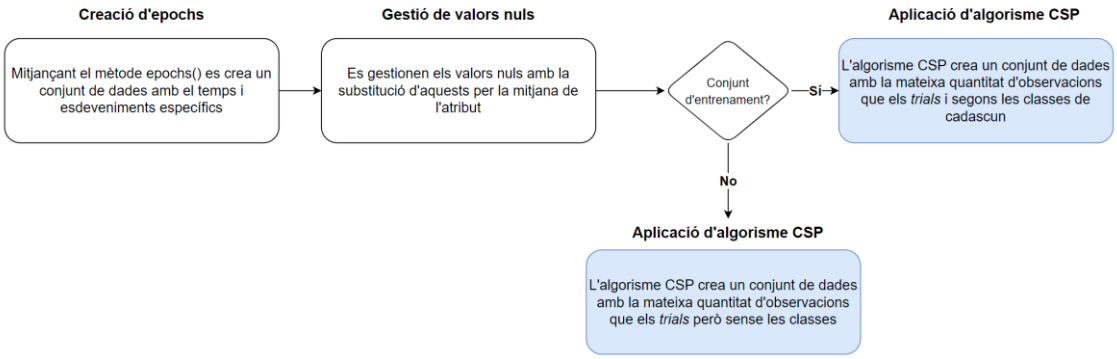


Figura 25. Funcionament d'extracció de característiques amb CSP

Com hem pogut veure, en funció del tipus de conjunt, la tècnica CSP [15] crearà un nou conjunt de dades diferent. En el cas dels conjunts d'entrenament, crearà un nou *dataset* amb el mateix nombre d'observacions que les execucions de la sessió de lectura d'EEG [2] amb l'aplicació dels filtres espacials i segons les classes establertes. Una cosa semblant passaria amb els conjunts de proves, amb la diferència que no tenim les classes, així que s'aplicarà la tècnica però sense aquestes. El resultat serà un conjunt amb les mateixes observacions que les sessions d'execució, però sense estar relacionats amb les classes, ja que aquestes s'hauran de predir més endavant. Per exemple, si tenim un conjunt de dades del tipus amb 120.000 observacions d'un conjunt del tipus 2B que corresponen a 160 execucions[1], CSP [15] crearà un nou *dataset* amb 160 observacions per característica que serà el resultat de l'aplicació dels algorismes espacials que hem descrit al principi d'aquesta sessió. A la següent imatge mostrem gràficament la transformació de dades que hem descrit.

[1] Cada execució pot durar *N* segons. Per exemple, si cada execució dura tres segons i tenim una freqüència de 250 Hz, tindrem 250 x 3 = 750 observacions per execució. Com hi haurà 160 execucions, el total d'observacions serà de 120.000.

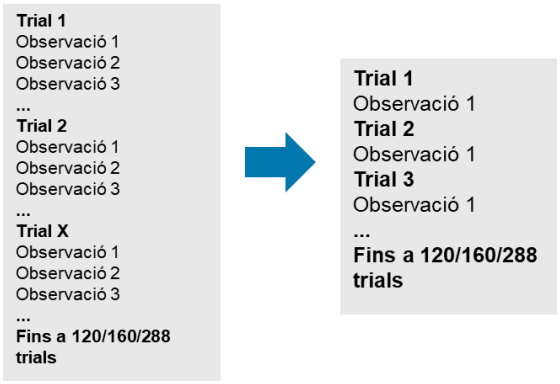


Figura 26. Transformació de les dades amb la tècnica del CSP

La taula de l'esquerra mostra el resultat del mètode epochs(), que crea un conjunt de dades amb *N* observacions agrupades en 120/160/288 grups corresponents als *trials*. És a dir, cada *trial* tindrà 120/160/288 grups on cadascú tindrà *N* observacions referents a l'esdeveniment que s'ha seleccionat. Pel que fa a la taula de la dreta, és la transformació de la tècnica CSP, que crearà un únic conjunt amb 120/160/288 grups, i cadascun d'aquests grups només tindrà una observació amb els corresponents atributs després d'aplicar la transformació.

Finalment, com es podrà veure en el capítol de resultats, hi haurà un paràmetre per indicar el nombre de paràmetres de CSP [15]. Aquest paràmetre servirà descompondre el senyal en el nombre de components i aplicar les tècniques que hem esmentat anteriorment.

4.3.7. Normalització de les dades

La normalització de les dades es farà a la fase d'explotació i sempre es produirà just abans d'entrenar i predir les dades. Anteriorment, s'ha pogut veure que s'aplica una normalització per estandardització, però en cap cas es fa amb els dos conjunts per separat. Concretament, es fa la normalització íntegra amb el conjunt d'entrenament i el conjunt de prova es normalitzarà amb la desviació i mitjana del conjunt d'entrenament. D'aquesta manera, ens evitem tenir dades molt diferenciades si els normalitzem per separat.

4.3.8. Principal component analysis

De manera opcional, es pot aplicar una anàlisi de components principals (PCA) [16] just abans de fer l'entrenament i prediccions de les dades. L'anàlisi de components principals permet obtenir un conjunt de dades amb una dimensionalitat més reduïda per minimitzar l'error quadràtic. És una tècnica lineal que obtindrà nou conjunt de dades reduït que serà una combinació lineal de les variables originals que seran independents entre elles. Matemàticament, es calcula de la següent manera:

1. Es normalitzen i centren les dades
2. Es calcularà la matriu de covariància del conjunt de dades.
3. Amb la matriu anterior, es calculen els valors i vectors propis.
4. Amb els valors propis es calcula la variància acumulada.
5. S'escullen els vectors propis que junts permetin conservar el percentatge de la variància desitjat.
6. Es fa el producte de matrius entre els vectors propis escollits i les dades originals. És a dir, $\boldsymbol{S} = \boldsymbol{PX}$ on P seran els vectors propis amb major valor propi associat a la matriu de covariància.

Per tant, el resultat serà un vector de projecció que serà el corresponent al major valor propi i que tindrà l'error quadràtic mínim

4.3.9. Independent component analysis

De la mateixa manera que el punt anterior, es pot escollir opcionalment aplicar la tècnica d'anàlisi de components independents (ICA) [17]. Aquesta tècnica ens permet trobar una representació lineal de dades no gaussianes de manera que els components siguin estadísticament independents. La diferència amb el cas anterior és que cerca components independents i que no estiguin correlacionats. L'algorisme de càlcul és semblant a l'anterior amb la diferència que s'hauran de blanquejar les dades perquè no tinguin correlacions, cosa que acabarà amb una projecció semblant a aquesta:

$$s = Wz$$

On s són els components principals, W les dades blanquejades i z el producte entre la matriu inversa dels vectors propis i les dades blanquejades.

4.3.10. Algorismes d'aprenentatge computacional

S'han aplicat algorismes d'aprenentatge supervisat per poder obtenir les prediccions de cada conjunt d'avaluació. A continuació descriurem sense entrar al detall els algorismes que es poden escollir en aquest projecte.

- **K-Nearest Neighbor**: Classificació segons els veïns més propers mitjançant centroides.
- **Multilayer perceptron**: Classificació neuronal mitjançant múltiples capes.
- **Decission Tree**: Arbre de classificació que desglossa les dades en nodes i fulles fins a trobar la classe.
- **Gaussian Naive Bayes**: Classificació mitjançant la distribució Gaussiana.
- **Support vector machine**: Classificació dels objecgtes mitjançant la construcció d'un hiperplà en l'espai de les dades que separa els objectes entre classes.
- **Linear Disciminant Analysis**: Troba la combinació lineal de semblança o separació de les classes.

4.3.11. Càlcul de mètriques

Un cop obtinguts els resultats dels classificadors, es calcularan una sèrie de mètriques per avaluar la qualitat del model. Concretament, es calcularan les següents dades:

- **Matriu de confusió:** Avalua el rendiment del classificador
- **Coeficient Kappa de Cohen [7]:** Mostra la mesura de concordança entre dues mostres
- **Exactitud:** Total de correctes classificacions dividides pel total de mostres.
- **Precisió:** Els positius vertaders dividits entre aquests més els falsos positius.
- **Recall:** Els positius vertaders dividits entre aquests més els falsos negatius.
- **Fall-out:** Els falsos positius dividits entre aquests i els negatius vertaders.
- **Receiver operating characteristic (ROC) [6]:** Mostra gràficament el rendiment del classificador mitjançant la ràtio de positius vertaders i la taxa de falsos positius. Contra més a prop de l'un estigui el valor de l'àrea de la corba, millor serà el resultat del classificador.

A banda, s'indicaran les prediccions correctes, incorrectes i el conjunt de dades final amb la comparació de les classes originals i les predites.

En seccions anteriors ja s'ha vist la sortida del programa amb algunes dades indicades anteriorment. No obstant a això, mostrarem una imatge amb algunes d'elles en format d'informe.

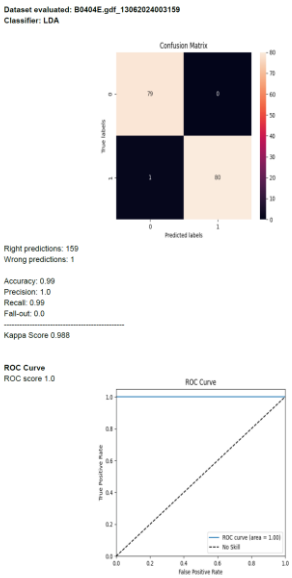


Figura 27. Imatge d'exemple de mètriques després d'una execució

4.3.12. Gestió d'errors i excepcions

Finalitzarem la secció amb la menció als errors i excepcions que ens podem trobar durant l'execució del programa. La majoria d'errors es donaran per la introducció de paràmetres incorrectes. A continuació mostrem un parell d'exemples.

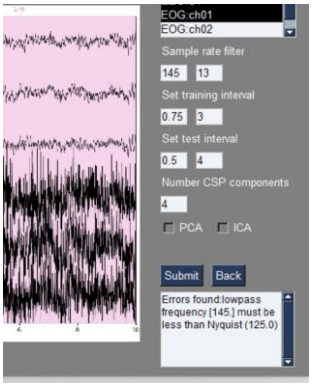


Figura 28. Error amb el paràmetre del filtratge de freqüència

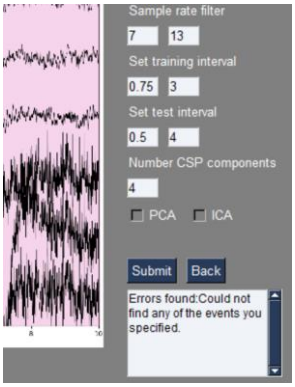


Figura 29. Error per carregar dos fitxers d'avaluació sense conjunt d'entrenament

5. Estructura de l'aplicació

En aquesta secció es descriurà sense entrar al detall el patró en què s'ha basat el disseny de l'aplicació, les llibreries, i les classes de l'aplicació. No es mostraran els diagrames de classes de la solució.

5.1. Patró Model, vista i controlador (MVC)

Totes les classes de l'aplicació seguiran el patró model, vista i controlador per desacoblar la interfície gràfica de la resta de classes del model. No es farà una descripció detallada del model, però es posarà un petit exemple on es mostra la crida a un mètode per obtenir una sèrie d'informació relacionada amb els esdeveniments d'un conjunt de dades.

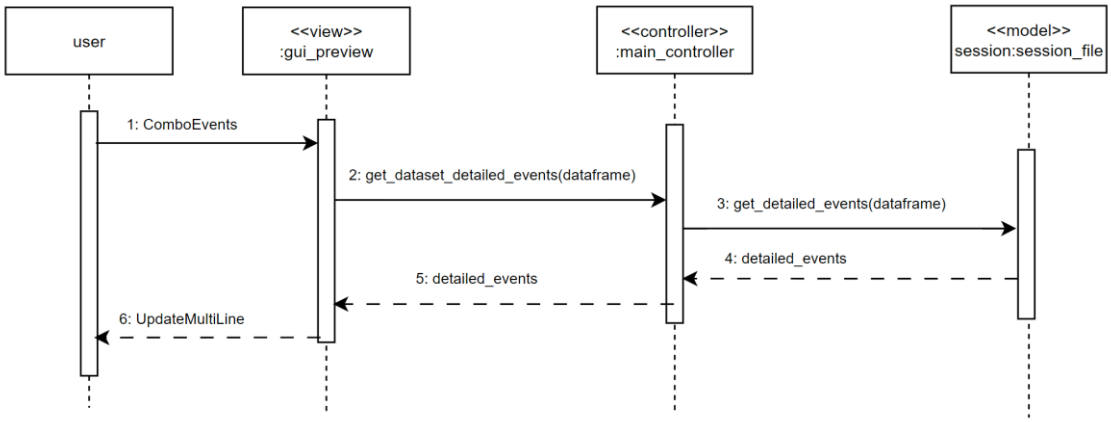


Figura 30. Exemple de crida per l'obtenció de dades amb el model MVC

5.2. Llibreries de l'aplicació

Les següents llibreries de Python s'entregaran amb el codi font i seran necessàries pel funcionament de l'aplicació:

- **Numpy 1.26.4**
- Pandas
- **Tabulate**
- abstractmethod
- Mne
- **Pysimplegui 4.60.5**
- Scipy
- Datetime
- Os
- Pathlib
- Sklearn (GaussianNB, KNeighborsClassifier, , MLPClassifier, DecisionTreeClassifier, LinearDiscriminantAnalys, svm, confusion_matrix, FastICA, roc_curve)
- FPDF
- Matplotlib
- sys
- subprocess
- seaborn

Pel que fa a la llibreria PySimpleGui i Numpy, es recomana la instal·lació de les versions indicades. No obstant a això, la llibreria PySimpleGui i la Tabulate, s'inclouran en el repositori del projecte i en l'entrega del RAC. D'altra banda, totes les llibreries de la llista poden instal·lar-ne d'altres que no figurin en aquest apartat.

Per acabar, es recomana llegir el document de l'annex nº 1 relacionat amb la instal·lació de l'aplicació, ja que en funció de la versió de l'entorn de desenvolupament, poden aparèixer errors que impedeixin l'execució de l'aplicació.

5.3. Classes de l'aplicació

L'aplicació està composta per les següents classes i llibreries:

- **controller**
  - controller_run.py: Llibreria amb mètodes per executar l'aplicació, gestionar les classes de les diferents finestres visuals de càrrega i cridar a la mostra de resultats.
  - main_controller: Classe que fa d'enllaç entre la interfície gràfica i el model.
- **gui**
  - gui: Super classe de totes les classes d'aquest tipus. És la classe principal que genera la interfície gràfica.
  - gui_main: Subclasse de gui, mostra la finestra per escollir els conjunts de dades.
  - gui_preview: Subclasse de gui, mostra les dades dels conjunts de dades i permet fer el filtre i executar l'aplicació per fer les prediccions.
  - gui_results: Classe que s'encarrega de mostrar els resultats per pantalla.
- **model**
  - session_file.py: Classe que s'encarrega d'obtenir informació del primer conjunt de dades carregat i ho envia a la classe processed_file pel seu processament. També retorna resultats informatius quan s'han carregat els conjunts de dades per primer cop.
  - processed_file.py: Aquesta classe processa els fitxers, aplica els filtratges i crea els conjunts de dades temporals que enviarà a la classe classifier.
  - classifier.py: Classe encarregada d'executar els diferents algorismes de classificació i calcular els resultats.
- **util**
  - preprocessing_data.py: Llibreria amb funcions de càlcul com la normalització, PCA, gestió de valors nuls, etc.
  - printing_data.py: Llibreria encarrega de proporcionar mètodes per imprimir resultats i generar els informes.

A l'annex 3 d'aquest document es podrà trobar el codi en Python de cadascuna de les classes i llibreries que s'han esmentat anteriorment.

5.4. Execució de l'aplicació

Es recomana executar l'aplicació en un entorn de desenvolupament PyCharm. En aquest sentit, i com s'ha comentat anteriorment, s'ha adjuntat un document adjunt que detalla la posada en marxa de l'aplicació amb aquest programari.

6. Resultats

Un cop s'ha descrit el funcionament de la solució i la seva estructura, es procedirà a exposar els resultats obtinguts de la seva execució. En aquest sentit, es provaran tots els conjunts de dades de les dues tipologies amb l'objectiu de trobar els millors resultats possibles. No obstant a això, cal destacar que el nombre de combinacions de paràmetres possibles és molt elevada, així que no s'han de considerar aquests resultats com a definitius. Per a cada tipologia de conjunt de dades, escollirem uns paràmetres que seran comuns per a totes les execucions. Per acabar, les conclusions de les execucions es comentaran al capítol següent.

6.1. Conjunts del tipus 2A

Els conjunts del tipus 2A [12] tenen quatre classes a predir. Per intentar encertar al màxim les prediccions, treballarem amb els següents paràmetres.

- Classificadors: Tots.
- Canals: Només els del tipus EEG.
- Filtratge de banda: Entre 8 i 28 Hz.
- Interval de temps del conjunt d'entrenament: Entre 0 i 4 segons.
- Interval de temps del conjunt de proves: Entre 0.5 i 3 segons.
- Nombre de components CSP [15]: 6.
- PCA [16]: Sí.
- ICA [17]: No.

D'aquesta manera, tindrem la següent finestra amb les opcions anteriors:

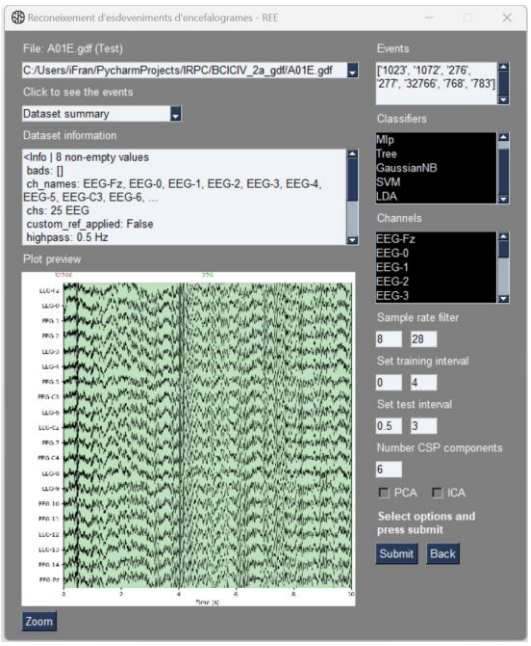


Figura 31. Imatge de l'aplicació amb filtres establerts previs a l'execució

El detall dels resultats de l'execució són els següents:

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score

1	LDA	A01T	A01E	0.68	270	18	0.94	0.89	0.86	0.04	0.97
2					**272**	**16**	**0.94**	**0.9**	**0.88**	**0.03**	**0.98**
3					237	51	0.82	0.82	0.38	0.03	0.87
4					237	51	0.82	0.59	0.94	0.22	0.93

Taula 9. Sessió 1A: resultats

Hem escollit el classificador que ha tingut millor coeficient de Kappa. De fet, així ho farem d'aquí en endavant. En aquest sentit, mostrarem la matriu de confusió i la corva ROC [6] de la classe amb millors resultats, que és la 2.

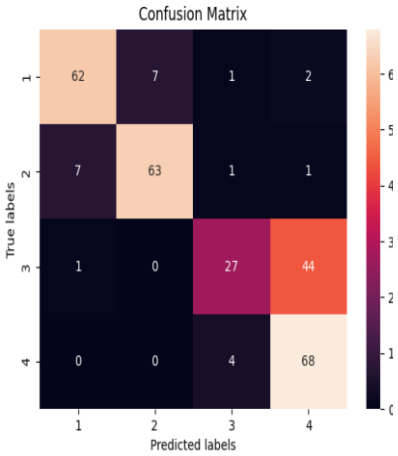


Figura 32. Sessió 1A: matriu de confusió

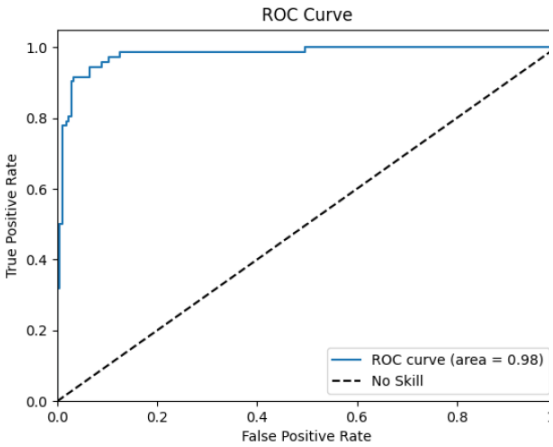


Figura 33. Sessió 1A: corba roc de la classe 2

A continuació, realitzarem el mateix exercici per les vuit execucions restants. Tal com s'ha comentat anteriorment, ens limitarem a posar els resultats sense fer cap consideració al respecte.

Sessió nº 2

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1	LDA	A02T	A02E	0.37	196	92	0.68	0.36	0.36	0.21	0.64
2					211	77	0.73	0.46	0.36	0.14	0.7
3					**250**	**38**	**0.87**	**0.7**	**0.82**	**0.12**	**0.95**
4					225	63	0.78	0.56	0.58	0.15	0.81

Taula 10. Sessió 2A: resultats

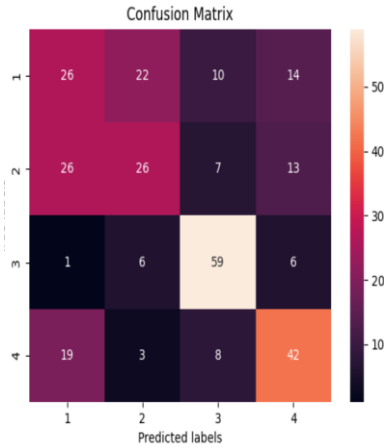


Figura 34. Sessió 2A: matriu de confusió

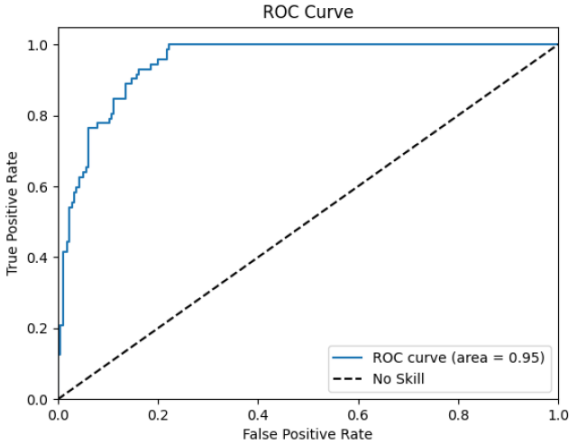


Figura 35. Sessió 2A: corba roc de la classe 3

Sessió nº 3

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1	MLP	A03T	A03E	0.77	274	14	0.95	0.91	0.89	0.03	0.99
2					**279**	**9**	**0.97**	**0.92**	**0.96**	**0.03**	**0.99**
3					240	48	0.83	0.68	0.64	0.1	0.9
4					245	43	0.85	0.69	0.72	0.11	0.91

Taula 11. Sessió 3A: resultats

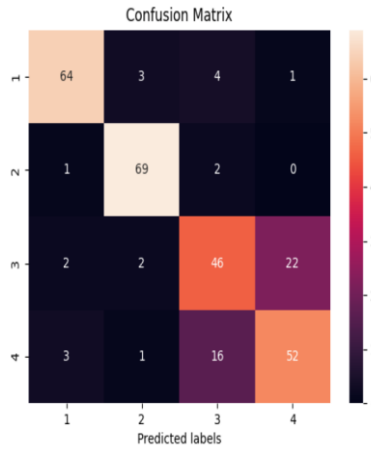


Figura 36. Sessió 3A: matriu de confusió

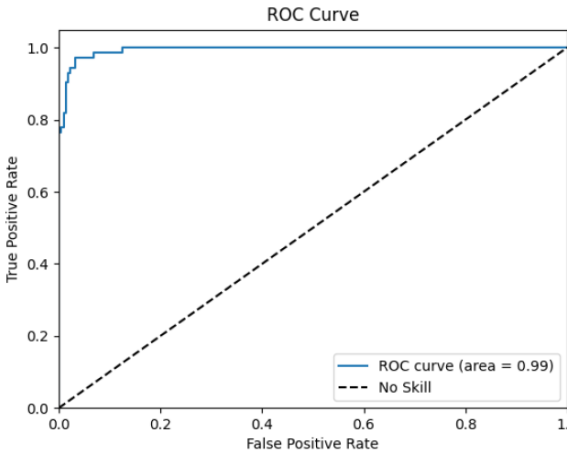


Figura 37. Sessió 3A: corba roc de la classe 2

Sessió nº 4

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1	LDA	A04T	A04E	0.5	238	50	0.83	0.64	0.69	0.13	0.87
2					233	55	0.81	0.6	0.71	0.16	0.83
3					**237**	**51**	**0.82**	**0.66**	**0.6**	**0.1**	**0.88**
4					230	58	0.8	0.62	0.51	0.11	0.84

Taula 12. Sessió 4A: resultats

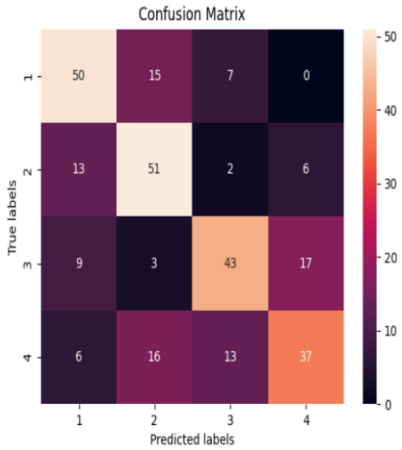


Figura 38. Sessió 4A: matriu de confusió

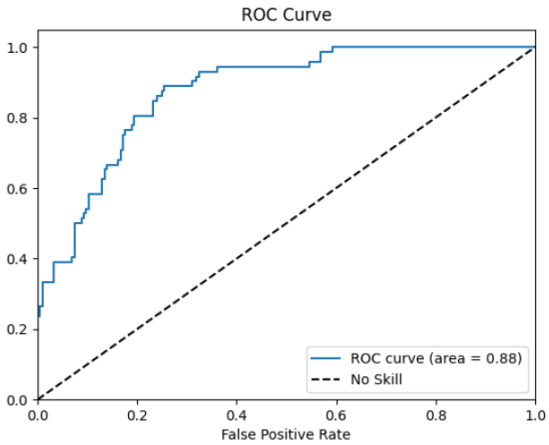


Figura 39. Sessió 4A: corba roc de la classe 3

Sessió nº 5

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1	SVM	A05T	A05E	0.5	194	94	0.67	0.37	0.43	0.25	0.58
2					194	94	0.67	0.36	0.4	0.24	0.63
3					**204**	**84**	**0.71**	**0.35**	**0.19**	**0.12**	**0.56**
4					200	88	0.69	0.4	0.47	0.23	0.65

Taula 13. Sessió 5A: resultats

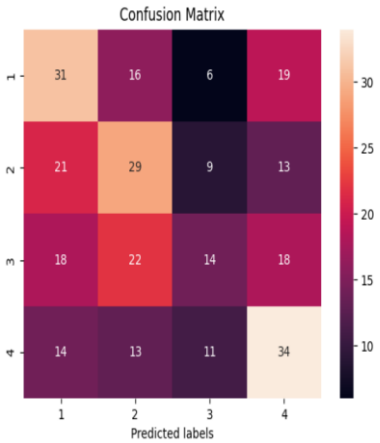


Figura 40. Sessió 5A: matriu de confusió

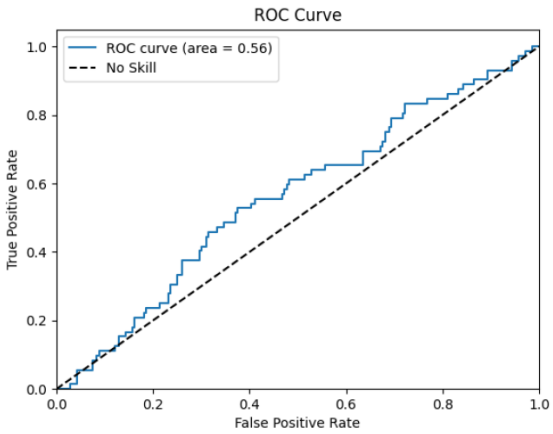


Figura 41. Sessió 5A: corba roc de la classe 3

Sessió nº 6

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1	LDA	A06T	A06E	0.28	**218**	**70**	**0.76**	**0.51**	**0.54**	**0.17**	**0.72**
2					206	82	0.72	0.43	0.46	0.2	0.66
3					215	73	0.75	0.49	0.39	0.13	0.65
4					203	85	0.7	0.42	0.46	0.21	0.72

Taula 14. Sessió 6A: resultats

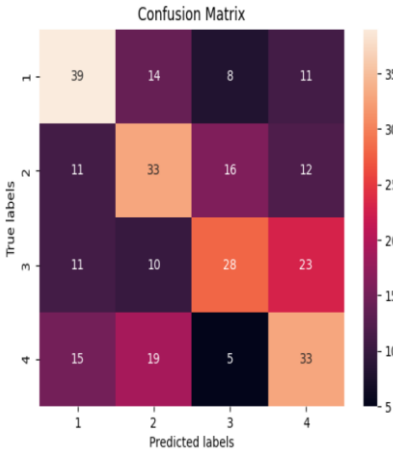


Figura 42. Sessió 6A: matriu de confusió

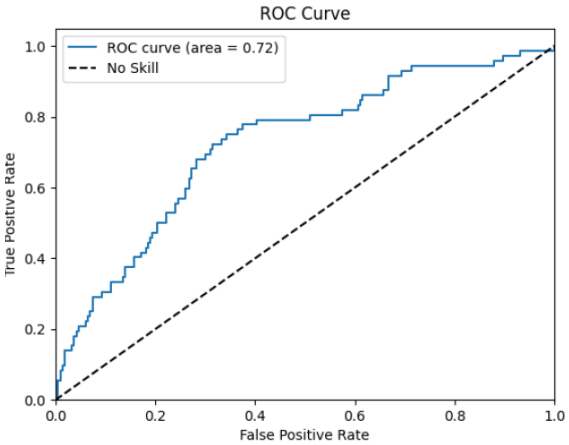


Figura 43. Sessió 6A: corba roc de la classe 1

Sessió nº 7

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1	LDA	A07T	A07E	0.62	236	52	0.82	0.62	0.69	0.14	0.89
2					233	55	0.81	0.63	0.57	0.11	0.87
3					**260**	**28**	**0.9**	**0.79**	**0.83**	**0.07**	**0.96**
4					257	31	0.89	0.81	0.75	0.06	0.91

Taula 15. Sessió 7A: resultats

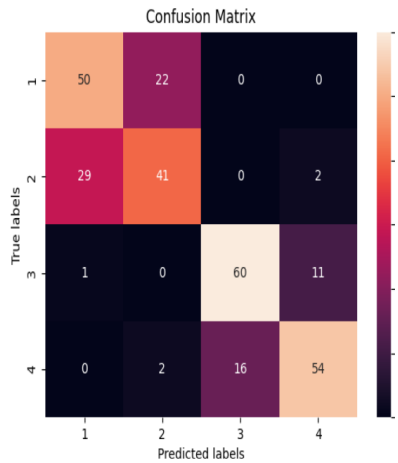


Figura 44. Sessió 7A: matriu de confusió

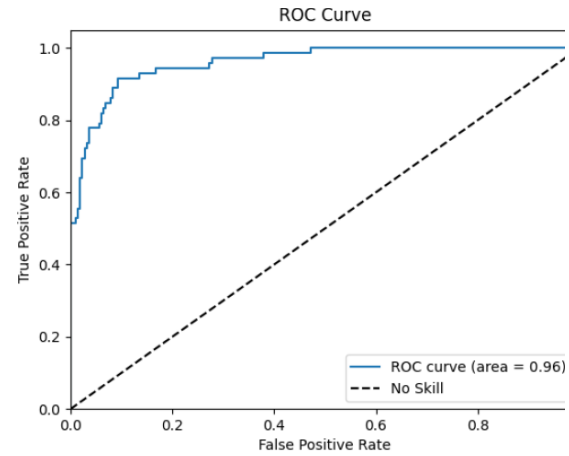


Figura 45. Sessió 7A: corba roc de la classe 3

Sessió nº 8

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1	LDA	A08T	A08E	0.74	**268**	**20**	**0.93**	**0.89**	**0.82**	**0.03**	**0.96**
2					265	23	0.92	0.83	0.86	0.06	0.96
3					247	41	0.86	0.73	0.68	0.08	0.89
4					262	26	0.91	0.79	0.88	0.08	0.94

Taula 16. Sessió 8A: resultats

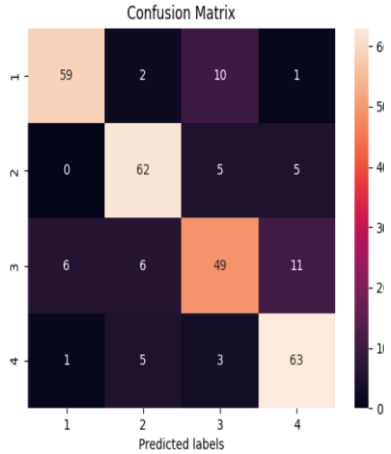


Figura 46. Sessió 8A: matriu de confusió

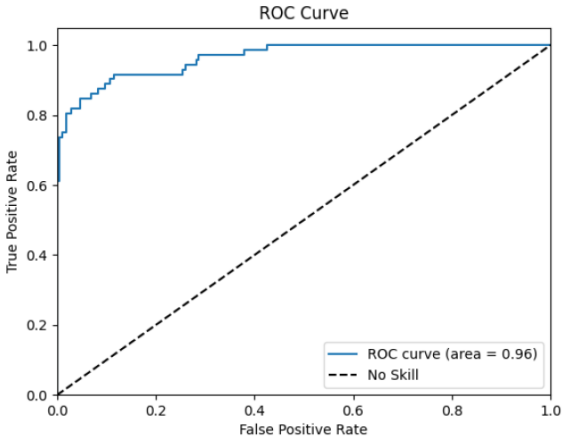


Figura 47. Sessió 8A: corba roc de la classe 1

Sessió nº 9

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1	LDA	A09T	A09E	0.53	**270**	**18**	**0.94**	**0.88**	**0.88**	**0.04**	**0.98**
2					219	69	0.76	0.53	0.32	0.09	0.66
3					218	70	0.76	0.52	0.47	0.15	0.7
4					243	45	0.84	0.63	0.93	0.19	0.96

Taula 17. Sessió 9A: resultats

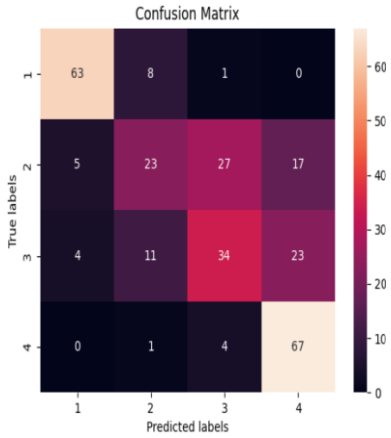


Figura 48. Sessió 9A: matriu de confusió

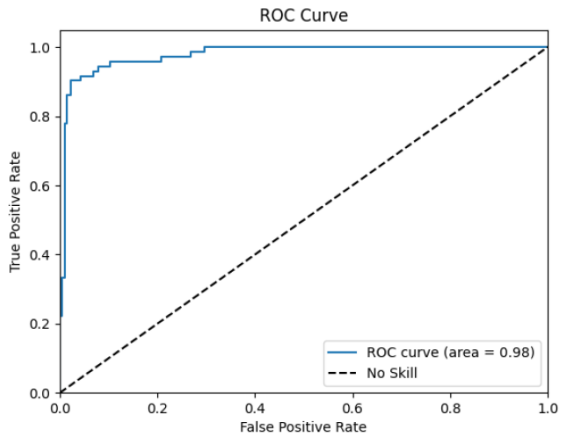


Figura 49. Sessió 9A: corba roc de la classe 1

6.2. Conjunts del tipus 2B

Per aquests tipus de conjunts, tindrem diversos *datasets* d'entrenament i prova per a cada sessió. D'altra banda, tindrem únicament dues classes. S'escollirà aquells conjunts de dades que donin el millor resultat mitjançant les següents opcions:

- Classificadors: Tots.
- Canals: Només els del tipus EEG [2].
- Filtratge de banda: Entre 8 i 28 Hz.
- Interval de temps del conjunt d'entrenament: Entre 0.75 i 3 segons.

- Interval de temps del conjunt de prova: Entre 0.5 i 4 segons.
- Nombre de components CSP [15]: 4.
- PCA [16]: Sí.
- ICA [17]: No.
- Conjunts de dades: Variarà en funció de l'individu.

Per tant, procedirem a fer les nou execucions i a exposar els resultats.

Sessió nº 1

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	LDA	B0102T i B0103T	B0104E	0.6	129	31	0.81	0.82	0.79	0.18	0.86

Taula 18. Sessió 1A: resultats

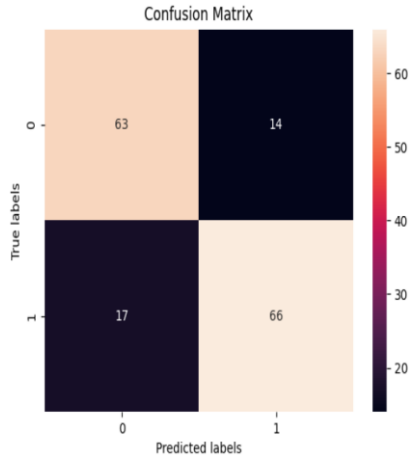


Figura 50. Sessió 1B: matriu de confusió

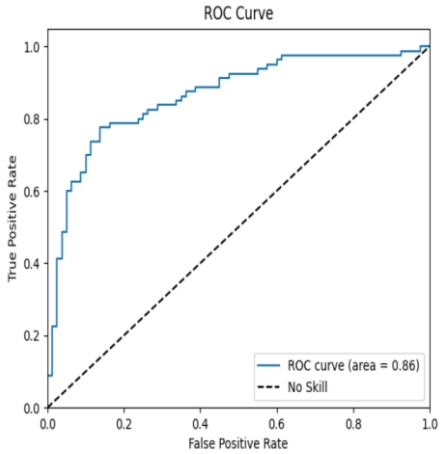


Figura 51. Sessió 1B: corba roc la classificació binària

Sessió nº 2

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	LDA	B0201T , B0202T i B0203T	B0205E	0.2	96	64	0.6	0.6	0.59	0.39	0.61

Taula 19. Sessió 2A: resultats

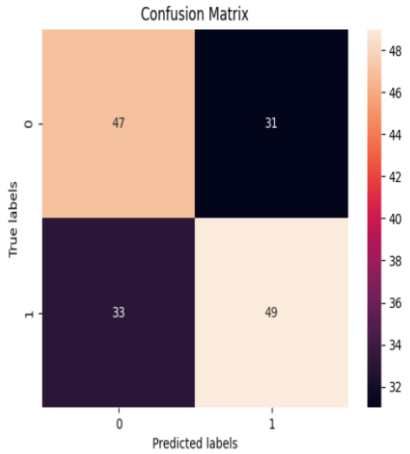


Figura 52. Sessió 2B: matriu de confusió

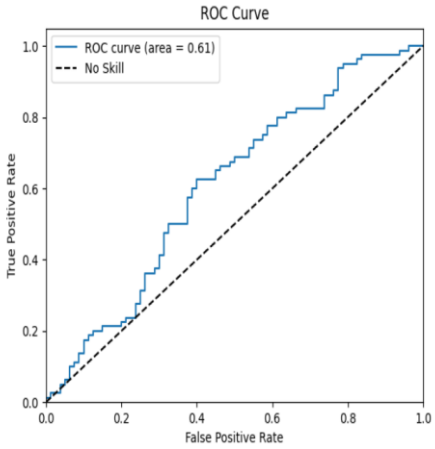


Figura 53. Sessió 2B: corba roc la classificació binària

Sessió nº 3

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	Naive Bayes	B0301T	B0304E	0.15	92	68	0.57	0.61	0.42	0.28	0.56

Taula 20. Sessió 3A: resultats

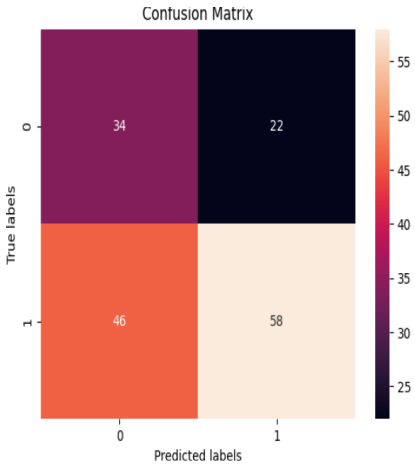


Figura 54. Sessió 3B: matriu de confusió

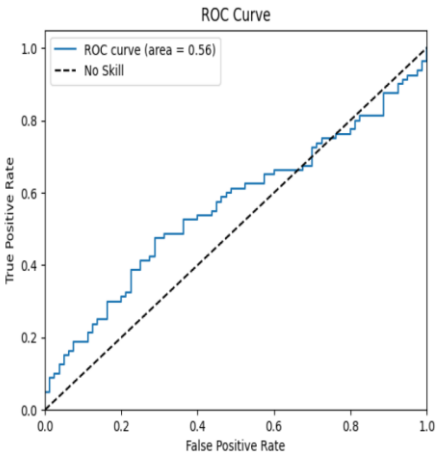


Figura 55. Sessió 3B: corba roc la classificació binària

Sessió nº 4

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	Naive Bayes	B0403T	B0403E	0.97	158	2	0.99	1	0.98	0	1

Taula 21. Sessió 4A: resultats

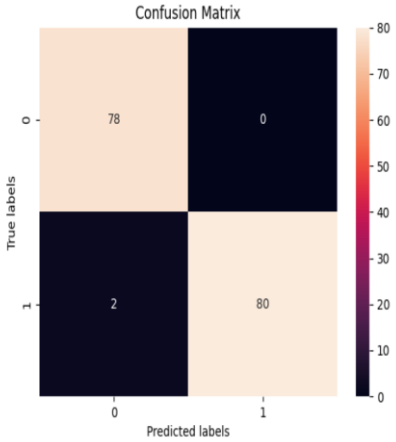


Figura 56. Sessió 4B: matriu de confusió

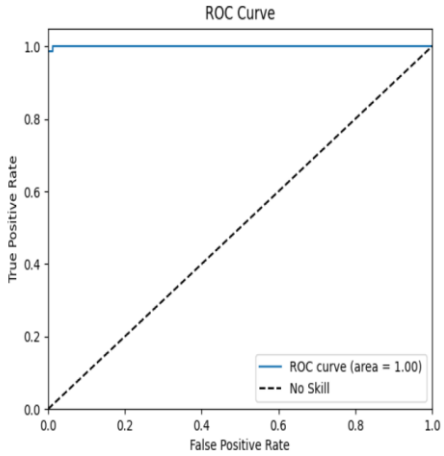


Figura 57. Sessió 4B: corba roc la classificació binària

Sessió nº 5

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	Naive Bayes	B0503T	B0504E	0.7	136	24	0.85	0.87	0.82	0.12	0.93

Taula 22. Sessió 5B: resultats

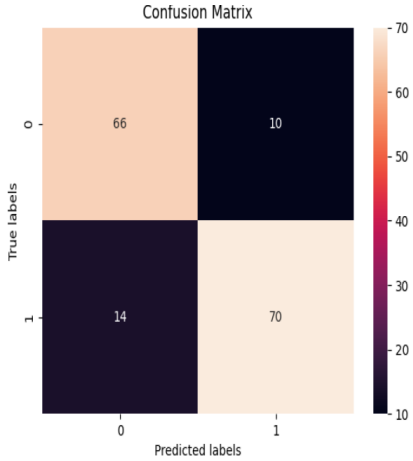


Figura 58. Sessió 5B: matriu de confusió

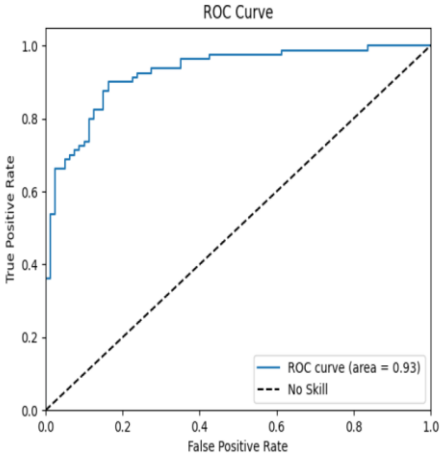


Figura 59. Sessió 5B: corba roc la classificació binària

Sessió nº 6

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	MLP	B0603T	B0604E	0.62	130	30	0.81	0.8	0.82	0.2	0.89

Taula 23. Sessió 6A: resultats

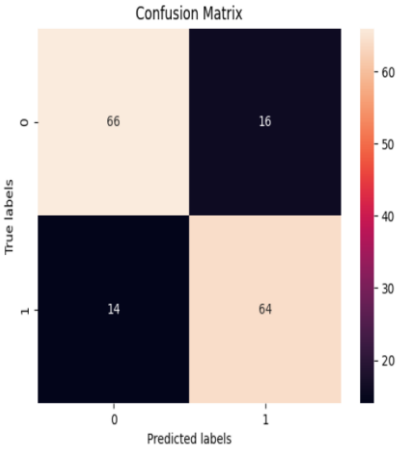


Figura 60. Sessió 6B: matriu de confusió

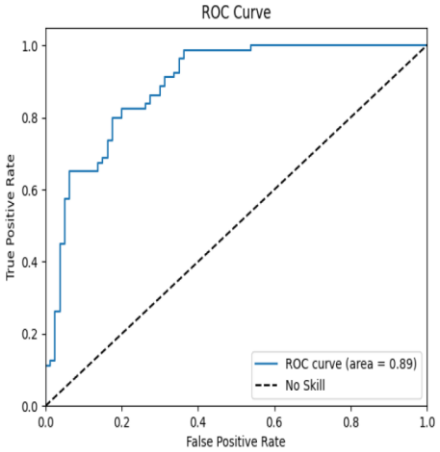


Figura 61. Sessió 6B: corba roc la classificació binària

Sessió nº 7

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	LDA	B0703T	B0704E	0.5	120	40	0.75	0.76	0.72	0.22	0.82

Taula 24. Sessió 7A: resultats

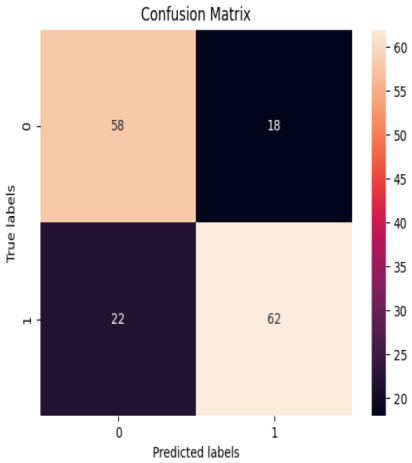


Figura 62. Sessió 7B: matriu de confusió

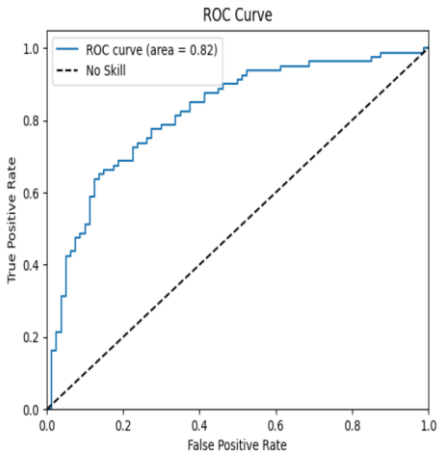


Figura 63. Sessió 7B: corba roc la classificació binària

Sessió nº 8

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	KNN	B0803T	B0805E	0.85	148	12	0.92	0.9	0.95	0.1	0.97

Taula 25. Sessió 8A: resultats

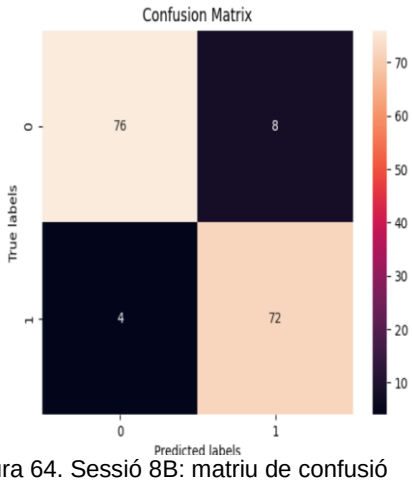


Figura 64. Sessió 8B: matriu de confusió

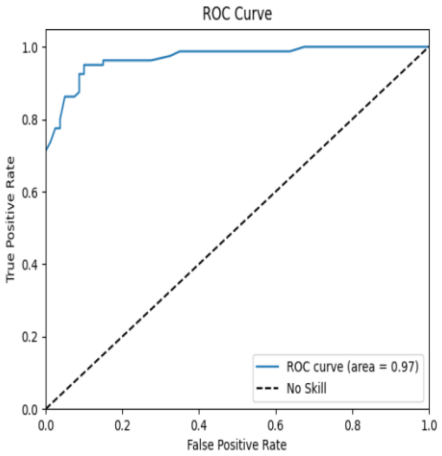


Figura 65. Sessió 8B: corba roc la classificació binària

Sessió nº 9

Class	Classifier	Train dataset	Test dataset	Kappa score	Right predictions	Wrong predictions	Accuracy	Precission	Recall	Fall-Out	ROC score
1 i 2	KNN	B0903T	B0905E	0.74	139	21	0.87	0.82	0.94	0.2	0.94

Taula 26. Sessió 9A: resultats

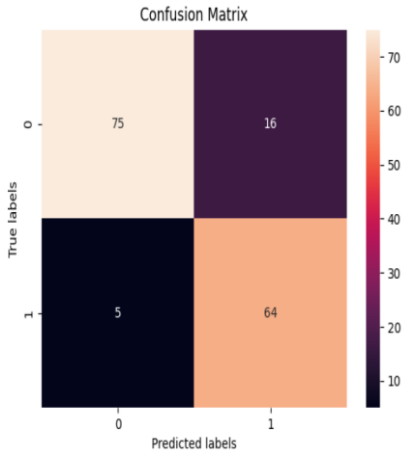


Figura 66. Sessió 9B: matriu de confusió

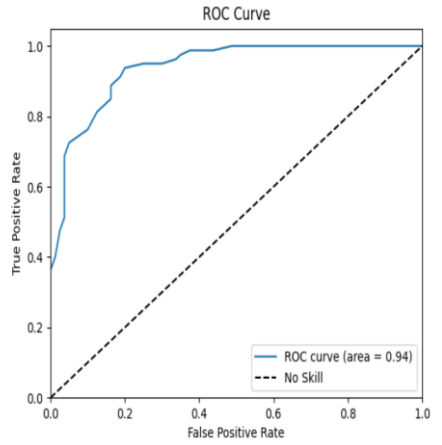


Figura 67. Sessió 9B: corba roc la classificació binària

7. Discussió

Dividirem aquest apartat en dos subapartats per donar resposta als resultats de les tres fases del projecte. Concretament, es parlarà de l'aplicació desenvolupada i dels resultats d'aquesta.

7.1. Interfície desenvolupada

Un dels resultats d'aquest projecte ha sigut la creació d'una interfície gràfica que permet carregar i analitzar conjunts de dades referents a lectures de senyals EEG [2]. Aquesta solució s'ha dissenyat segons els objectius específics indicats a la introducció d'aquest document i s'ajusta a les tres fases que hem indicat també en aquella secció. D'altra banda, l'aplicació desenvolupada permet ajustar un nombre d'opcions significatiu que poden fer variar el resultat obtingut.

D'aquesta manera, s'aconsegueix una flexibilitat que podria permetre a l'usuari variacions en el seu estudi, com barrejar conjunts de dades de diferents individus amb l'objectiu de comparar diferents lectures de diferents persones.

7.2. Resultats de les execucions

Els resultats de l'apartat anterior estan alineats amb l'objectiu general del projecte, que és la predicció dels pensaments d'una persona. De manera general, la mitjana de resultats del coeficient Kappa [7] és el següent:

Dataset	Kappa's average
2A	0,55
2B	0,59

Taula 27. Taula resum dels resultats

Per tant, en la majoria d'execucions s'ha encertat >= 50% de les prediccions, i en alguns casos, s'han obtingut resultats entre el 70 i el 97%. Això últim donaria resposta a una de les problemàtiques que s'ha esmentat al principi d'aquest apartat, que és la correcta interpretació i prediccions dels patrons del pensament. Aquest seria el cas dels següents conjunts de dades:

- A03T: 77%
- A08T: 74%
- **B0403T: 97%**
- B0503T: 70%
- **B0803T: 85%**
- B0903T: 74%

Aquests conjunts de dades serien candidats a ser estudiats amb més profunditat, ja que el motiu del percentatge d'encerts tan elevat podria millorar-se encara més i es podrien aprofitar per intentar aplicar-lo als altres casos. En aquest sentit, i tal com s'ha comentat al principi de les proves, els resultats es podrien millorar més si s'ampliés el ventall de proves amb noves combinacions de paràmetres, com els filtratges de banda, períodes d'entrenament i prova, etc.

Pel que fa a l'aplicació d'algorismes d'aprenentatge computacional, en tots els resultats les sessions hem vist almenys un dels algorismes d'aprenentatge supervisat a excepció de l'arbre de classificació, que en cap cas ha sigut el que ha obtingut els millors resultats.

Per tant, es podria concloure que aquest algorisme és el menys efectiu per fer prediccions de dades com les que hem treballat en aquest projecte. A la banda oposada estaria l'algorisme d'anàlisi discriminant lineal (ADL), que ha sortit en la majoria dels casos amb la millor mètrica de Kappa [7]. Pel que sembla, la seva tècnica per trobar la combinació lineal per poder distingir entre classes, és la millor opció per les dades que s'han treballat.

Finalment, acabarem aquesta secció amb la no aplicació de l'anàlisi de components independents. Pel que hem pogut veure en diversos tests, l'aplicació d'aquesta tècnica no millorava considerablement els resultats, així que s'ha decidit no aplicar-la en les execucions realitzades.

8. Conclusions

A l'apartat anterior ja s'han descrit algunes conclusions de la solució desenvolupada per aquest projecte. Concretament, s'ha fet menció de l'objectiu general, el qual creiem que s'ha assolit de manera satisfactòria, ja que, independentment del coeficient Kappa [7], en tots els resultats el nombre de prediccions correctes és superior al d'incorrectes. Per tant, hem aconseguit implementar una solució a la problemàtica plantejada i ho hem fet amb una aplicació que s'ha desenvolupat segons les millors pràctiques apreses en aquest grau.

A més, s'ha pogut realitzar una bateria de proves que ens ha permès veure uns resultats detallats per a cada escenari. Aquests resultats es poden veure gràficament i amb mètriques i han sigut diferents en cada cas. No obstant a això, no ha sigut objectiu de les proves fer un estudi en profunditat de cada conjunt per aconseguir el millor resultat en cada cas, sinó que hem fet proves amb una sèrie de paràmetres que ens ha donat una visió general del funcionament de l'aplicació.

D'altra banda, cal destacar que s'ha hagut de treballar amb conjunts de dades multi classe, cosa que en certs moments del projecte ha dificultat l'execució d'aquest. No obstant a això, aquesta dificultat ha proporcionat un coneixement addicional que ens servirà de cara al futur. Així mateix, la gestió de les dades com la càrrega, manipulació, normalització, etc. ens ha aportat molt de coneixement i ens ha servit per aplicar conceptes apresos en assignatures anteriors.

En últim lloc, i com veurem en els següents punts, cal destacar la creació de la interfície gràfica **multiplataforma** que s'ha fet per proporcionar una execució ràpida i còmoda per a l'usuari. D'aquesta manera, el personal que hagi de fer servir l'aplicació no necessitaria coneixements de Python per poder realitzar les prediccions dels senyals EEG.

8.1. Assoliment d'objectius

Sobre els objectius específics, a excepció d'algun opcional i personal, tots s'han assolit de manera satisfactòria. A continuació farem unes breus conclusions per a cada tipologia d'objectius.

8.1.1. Objectius funcionals

Com s'ha vist a les seccions anteriors, a excepció de la integració amb tercers, tots els objectius s'han assolit de manera satisfactòria, ja que hem pogut carregar, llegir i interpretar dades que, posteriorment, han sigut tractades per algorismes d'aprenentatge computacional. Pel que fa a la integració amb altres sistemes, encara que aquesta part no estigui realitzada, el model aconseguit permetria aquesta integració de manera senzilla, ja que s'ha aplicat un baix acoblament entre classes de l'aplicació i aquesta entrega un conjunt de dades final amb les prediccions, cosa que seria molt senzill transmetre a altres sistemes, ja que és lleuger i fàcil d'interpretar.

8.1.2. Objectius de projecte

Aquests objectius es donen tots per assolits, ja que a banda de la font fiable de dades utilitzada, s'ha pogut construir una aplicació amb Python que genera una **interfície gràfica** amb les funcionalitats descrites anteriorment. D'altra banda, malgrat que les entregues de l'assignatura es van poder assolir de manera satisfactòria, l'entrega del producte final es va endarrerir dues setmanes. Per tant, l'objectiu relacionat amb l'entrega de lliurables, s'ha satisfet de manera parcial.

8.1.3. Objectius personals

A excepció de l'objectiu relacionat amb la publicació d'aquest projecte en el repositori de projecte de la UOC, tots els altres objectius s'han pogut completar satisfactòriament. Per aquest projecte s'ha hagut d'entendre com funcionen els senyals EEG [2] des del punt de vista físic i hem pogut entregar un producte amb el qual hem aplicat tots els conceptes apresos en assignatures de programació. Finalment, s'ha hagut de treballar amb profunditat amb temes relacionats d'intel·ligència artificial que seran aplicables en altres projectes en el futur.

8.2. Lliçons apreses

Tal com s'ha pogut veure anteriorment, aquest projecte ens ha aportat coneixement addicional que ens ha servit per assolir els objectius. De fet, hem hagut d'aprendre a trobar aquest coneixement i aplicar-lo de manera adequada per A banda, la metodologia progressiva de l'avaluació continuada de l'assignatura ens ha servit per aprendre a desenvolupar el projecte de manera iterativa amb una planificació prèvia. En aquest sentit, l'elaboració de la memòria des de les primeres etapes del projecte, ens ha ajudat a la seva redacció, així que considerem aquesta part com una lliçó significativa.

8.3. Línies de futur

Com a futura línia de treball o continuació del projecte seria la possibilitat d'oferir el servei a terceres parts mitjançant una integració. Aquesta integració es va planificar de manera opcional als objectius i no es va poder implementar. La integració es podria fer mitjançant *sockets* o alguna API que permetés a un altre dispositiu implementar les prediccions fetes per la solució. De fet, amb els resultats que s'obtenen quan es fan les execucions, es podrien recollir les observacions de la matriu final:

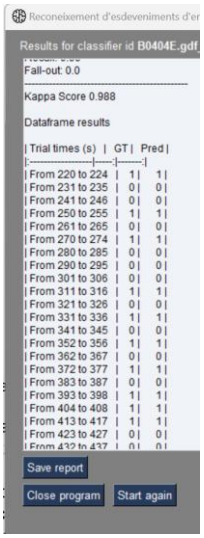


Figura 68. Taula final de prediccions després de l'execució

D'aquesta manera, tindríem una interfície cervell-ordinador que serviria per donar solucions a problemes del camp de la medicina o seguretat.

Així mateix, un altre línia de treball seria la inclusió d'altres tipus de conjunts de dades per poder fer l'eina el més polivalent possible. En aquest projecte s'ha treballat amb dos tipus de conjunts, però existeixen d'altres amb diferent format que també es podrien afegir per extreure les dades i gestionar-les com s'ha fet en aquest cas. Encara més, la integració

amb un lector de dades de senyals EEG [2] completaria la interfície cervell-ordinador que hem esmentat en el paràgraf anterior.

Finalment, el desplegament de la solució en algun servei del núvol també seria una línia interessant. D'aquesta manera, es tindria accés a la plataforma des de qualsevol lloc i es podrien augmentar recursos tècnics com la computació i memòria per treballar amb conjunts de dades més grans que permetessin identificar més classes.

8.4. Seguiment de la planificació

La planificació d'aquest projecte s'ha seguit íntegrament exceptuant la fase d'explotació de dades, que va patir un endarreriment que va provocar que el producte final s'entregués en una altra fase. No obstant a això, podríem concloure que la planificació ha estat acurada, ja que la majoria de tasques s'han assolit a les dates planificades. A més a més, la gestió de riscos i requisits ha resolt futurs problemes, cosa que també ha garantit el compliment de la planificació.

D'altra banda, a cada fase de seguiment del projecte s'ha entregat un prototip funcional de l'aplicació, així que podríem dir que el producte s'ha anat construint de manera iterativa fins a assolir totes les funcionalitats.

Finalment, l'únic canvi que s'ha demanat no va ser relacionat amb l'abast del projecte, sinó en l'entrega del producte final, tal com s'ha comentat anteriorment.

9. Glossari

REE: Reconeixement d'esdeveniments d'encefalogrames.

EEG: Electroencefalograma. Procés no invasiu de lectura de l'activitat elèctrica del cervell.

BCI: Brain Computer Interface. Interfície cervell-ordinador

KNN: K-nearest neighbors: Algoritme de classificació supervisat que es basa en la distància dels veïns més propers.

SVM: Support vector Machine: Algoritme de classificació supervisat que classifica la informació mitjançant un hiperplà que maximitza la distància entre classes.

MLP: Multi Layer Perceptron: Algortime de classificació supervisada mitjançant xarxes neuronals.

LDA / ADL: Linear discriminant anàlisis / Anàlisi discriminant lineal. Tècnica que mitjançant la combinació lineal troba diferència entre classes.

Decision Tree: Algoritme de classificació supervidada que construeix un arbre de decisió i fa servir els seus nodes i fulles per fer la classificació.

ROC: Receiver operating characteristic: Mesura el rendiment d'un classificador.

Cervell-Ordinador: Interfície que llegueix i interpeta senyals EEG i les transmet a un altre dispositiu per transformar-les en accions.

PCA: Principal component analysis: Tècnica de reducció lineal per extreure els components principals.

ICA: Independent componenet analysis: Tècnica computacional per separar senyals multivariades en subcomponents.

GDF: General data file. Format dse fitxer per emmagatzemar dades de senyals biomèdiques.

MVC: Model vista controlador. Disseny que separa l'aplicació en tres components, Model, Vista i controlador.

EOG: Electrooculograma. Procés no invasiu de lectura de l'activitat elèctrica relacionada amb els ulls.

EMG: Electromiograma: Procés no invasiu de lectura de l'activitat elèctrica dels músculs.

ECG: Electrocardiograma: Procés no invasiu de lectura de l'activitat elèctrica del cor.

CSP: Common spatial pattern: Tècnica que es fa servir en el procés de senyals per extreure característiques mitjançant filtres espacials.

10. Bibliografia

[1] *BCI Competition IV* [en línia] [consulta: 16 de juny 2024]. Disponible a https://www.bbci.de/competition/iv/#dataset1

[2] *Electroencefalograma* [en línia] [consulta: 16 de juny 2024]. Disponible a: https://ca.wikipedia.org/wiki/Electroencefalograma

[3] RODRÍGUEZ, José Ramón. *La gestió de projectes. Conceptes bàsics* [recurs d'aprenentatge]. Barcelona: Universitat Oberta de Catalunya. [consulta: 16 de juny 2024]. Disponible a: https://materials.campus.uoc.edu/daisy/Materials/PID_00247934/pdf/PID_00247934.pdf

[4] RODRÍGUEZ, José Ramón, MARINÉ JOVÉ, Pere. *Planificació del projecte. Conceptes bàsics* [recurs d'aprenentatge]. Barcelona: Universitat Oberta de Catalunya. [consulta: 16 de juny 2024]. Disponible a: https://materials.campus.uoc.edu/daisy/Materials/PID_00247936/pdf/PID_00247936.pdf

[5] *What does epoch mean in EEG?* [en línia] [consulta: 16 de juny 2024]. Disponible a: https://dsp.stackexchange.com/questions/41135/what-does-epoch-mean-in-eeg#:~:text=EEG%20epoching%20is%20a%20procedure,event%20e.g.%20a%20visual%20stimulus

[6] *Receiver operating characteristic* [en línia] [consulta: 16 de juny 2024]. Disponible a: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[7] *Cohen's kappa* [en línia] [consulta: 16 de juny 2024]. Disponible a: https://en.wikipedia.org/wiki/Cohen%27s_kappa

[8] *General Data Format for Biomedical Signals* [en línia] [consulta: 16 de juny 2024]. https://en.wikipedia.org/wiki/General_Data_Format_for_Biomedical_Signals

[9] *Brain-computer interface* [en línia] [consulta: 16 de juny 2024]. https://en.wikipedia.org/wiki/Brain%E2%80%93computer_interface

[10] *10–20 system (EEG)* [en línia] [consulta: 16 de juny 2024]. https://en.wikipedia.org/wiki/10%E2%80%9320_system_(EEG)

[11] *European Data Format (EDF)* [en línia] [consulta: 16 de juny 2024]. https://en.wikipedia.org/wiki/European_Data_Format

[12] BRUNNER, C, LEEB, R, Müller-Putz, G.R, Schlögl, A, Pfurtscheller, G. *BCI Competition 2008 – Graz data set A* [en línia] [consulta: 16 de juny 2024] https://www.bbci.de/competition/iv/desc_2a.pdf

[13] BRUNNER, C, LEEB, R, Müller-Putz, G.R, Schlögl, A, Pfurtscheller, G. *BCI Competition 2008 – Graz data set B* [en línia] [consulta: 16 de juny 2024] https://www.bbci.de/competition/iv/desc_2b.pdf

[14] Electrooculograma [en línia] [consulta: 16 de juny 2024] https://ca.wikipedia.org/wiki/Electrooculograma

[15] *Neuroscience Meets Data Science: Exploring Common Spatial Pattern (CSP) and Its Applications in Healthcare Analytics* [en línia] [consulta: 16 de juny 2024] https://medium.com/geekculture/common-spatial-pattern-and-its-applications-in-the-healthcare-industry-faa4311dab79

[16] *Análisis de componentes principales* [en línia] [consulta: 16 de juny 2024] https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales

[17] *Análisis de componentes independientes* [en línia] [consulta: 16 de juny 2024] https://medium.com/@ab.jannatpour/independent-component-analysis-ica-with-python-code-e7d1dd290241

[18] Electromiografia [en línia] [consulta: 16 de juny 2024] https://es.wikipedia.org/wiki/Electromiograf%C3%ADa

[19] Electrocardiograma [en línia] [consulta: 16 de juny 2024] https://es.wikipedia.org/wiki/Electrocardiograma#:~:text=El%20electrocardiograma%20(ECG%20o%20EKG,en%20forma%20de%20cinta%20continua

11.Annexos

11.1. Annex 1. Manual i repositori

Juntament amb aquesta memòria s'entregarà un document annex anomenat **frangc_REE_manual** que és un manual de posada en marxa de l'aplicació. Aquest document també indica el repositori que conté l'aplicació desenvolupada en aquest projecte i fa una breu descripció de com s'ha d'executar el programari.

De totes maneres, procedirem a descriure el contingut del repositori on tenim tota l'aplicació i documentació relacionada d'aquest projecte.

El repositori en qüestió, es pot trobar en aquest enllaç: frangc_TFG. L'accés al lloc és públic, això que no caldrà cap tipus de permís. Dins del repositori, tindrem els següents directoris:

- **APP:** Conté els binaris de l'aplicació amb les seves dependències
- **DATA:** Conté els conjunts de dades d'entrenament, prova i les *true labels*
- **DOCS:** Conté documentació del TFG, com la memòria, el manual d'instal·lació i els informes de les proves que es van realitzar a la secció 6 d'aquest document
- **PRES:** Conté el vídeo i la presentació del projecte.

11.2. Annex 2. Detall dels orígens de dades

Els orígens de dades tindran les següents característiques:

11.2.1. Característiques comunes

- Freqüència: 250Hz
- Filtratge de banda: 0.5 i 100 Hz
- Filtre Notch: 50Hz
- Entre 600 i 700K observacions per *dataset*
- Artefactes del tipus EOG
- Codis d'esdeveniments i estructura d'execució similar
- Mateix format de fitxer de dades (GDF i Matlab)
- Els fitxers d'entrenament tenen una nomenclatura que acaba en **<nom>.T.gdf**, mentre que els de prova, en **<nom>.E.gdf**

Més informació a https://www.bbci.de/competition/iv/

11.2.2. Característiques dels conjunts 2A [12]

- Imaginació motora multiclasse
- Esdeveniments: 769 (mà esquerrra), 770 (mà dreta), 771 (peu), 772 (llengua)
- 22 canals EEG i 3 EOG
- Conjunt d'entrenament i prova unitari per individu
- 288 tests per sessió

11.2.3. Característiques dels conjunts 2B [13]

- Imaginació motora binària
- Esdeveniments: 769 (mà esquerra), 770 (mà dreta)

- 3 canals EEG i 3 canals EOG
- Múltiples conjunts d'entrenament i prova per individu
- 120 – 160 tests per sessió

11.3. Annex 3. Codi de l'aplicació

11.3.1. controller_run.py

```
"""
This is the controller_run library which is responsible for running the app for all the states
author: frangc
version: 3.1
"""

from src.frangc.edu.gui import gui_preview
from src.frangc.edu.gui import gui_results
from src.frangc.edu.gui import gui_main

"""
Main method, manages all the windows during the session
"""


def main_gui():
    restart = True
    main_gui_ = None
    preview_gui = None
    while restart:
        if main_gui_ is not None and preview_gui is not None:
            del main_gui_
            del preview_gui
        main_gui_ = gui_main.MainGui()
        main_gui_.run_gui()
        result = main_gui_.get_option_selected()
        if result == 1:
            info = main_gui_.get_info()
            preview_gui = gui_preview.Gui(info)
            preview_gui.run_gui()
            classifier_result = preview_gui.get_classifiers_result()
            if classifier_result is not None:
                result_gui = gui_results.GuiResults(classifier_result)
                result_gui.run_gui()
                restart = result_gui.get_back_main()
                del result_gui
            else:
                restart = preview_gui.get_back_main()
        else:
            restart = False


"""
Run method to load main()
"""
if __name__ == "__main__":
    main_gui()
```

11.3.2. main_controller.py

```
"""
This is the main controller class which is responsible for managing all the app
author: frangc
version: 3.1
"""
from src.frangc.edu.model import session_file


class MainController:
    """
    Default constructor
    @data_frames: datasets to be loaded
    """

    def __init__(self, data_frames):
        self.__data_frames = data_frames
        self.__set_session()

    """
    Creates the class that will load the data
    """

    def __set_session(self):
        self.__session = session_file.SessionFile(self.__data_frames)

    """
    Gets the main info of a dataframe
    @:return: the basic info of a dataframe
    """

    def get_dataset_info(self, data_frame):
        return self.__session.get_dataset_info(data_frame)

    """
    Gets the events of a dataframe
    @:return: the events of a dataframe
    """

    def get_dataset_events(self, data_frame):
        return self.__session.get_dataset_events(data_frame)

    """
    Gets the channels of a dataframe
    @:return: the channels of a dataframe
    """

    def get_dataset_channels(self, data_frame):
        return self.__session.get_dataset_channels(data_frame)

    """
    Gets the detailed events of a dataframe
    @:return: the detailed events of a dataframe
    """

    def get_dataset_detailed_events(self, data_frame):
        return self.__session.get_detailed_events(data_frame)

    """
    Creates the instance that will be used to predict the data
    """

    def process_file(self, events, classifiers, low_r, high_r, channels, ica, pca, csp, train_low, train_high,
                     test_low, test_high):
        self.__session.process_file(events, classifiers, low_r, high_r, channels, ica, pca, csp, train_low,
```

```
            train_high,
                                test_low, test_high)

        """
        Run the prediction with the classifiers and gets the result
        @:return: the result of the prediction with extra data
        """

        def get_classifiers_result(self):
            classifiers_result = self.__session.perform_classifiers()
            return classifiers_result

        """
        Gets the sample rate of a dataframe
        @:return: the sample rate of a dataframe
        """

        def get_sample_rate(self, data_frame):
            return self.__session.get_sample_rate(data_frame)

        """
        Gets a plot for a specific dataframe
        @:return: the plot of a specific dataframe
        """

        def get_plot(self, data_frame):
            return self.__session.get_plot(data_frame)
```

11.3.3. gui.py

```
"""
This is the gui super class for the graphic interface
author: frangc
version: 3.1
"""
from abc import abstractmethod
import os


class Gui:
    """
    Default constructor without parameters
    """

    def __init__(self):
        self.window = None
        self.default_color = 'grey'
        self.get_back = None
        self.window_size = None
        self.main_title = None
        self.error_field = None
        self.icon_path = None
        self.__set_current_icon()

    """
    Abstract method to set the parameters of the class
    """

    @abstractmethod
    def __set_parameters(self):
        pass
```

```
    """
    Abstract method to run the graphic interface
    """

    @abstractmethod
    def run_gui(self):
        pass

    """
    Gets a boolean value to check if the user has decided to go back
    @:return: a boolean value
    """

    def get_back_main(self):
        return self.get_back

    def __set_current_icon(self):
        self.icon_path = os.path.abspath(os.path.dirname(os.path.abspath(__file__)) + "/icon.ico")
```

11.3.4. gui_main.py

```
"""
This is the gui_main class which shows the window to choose the datasets
author: frangc
version: 3.1
"""

import PySimpleGUI as sG
import os
from src.frangc.edu.gui import gui


class MainGui(gui.Gui):
    """
    Default constructor without parameters
    """
    def __init__(self):
        super().__init__()
        self.__text_box_train = None
        self.__text_box_test = None
        self.__secondary_title = None
        self.__option_selected = None
        self.__dataframes = None
        self.__set_parameters()

    """
    Run the graphic interface
    """
    def run_gui(self):
        self.__option_selected = 1
        file_test = None
        while True:
            event, values = self.__window.read()
            if event == sG.WIN_CLOSED:
                self.get_back = False
                self.__option_selected = 0
                break
            if event == '-TRAIN-':
                if ".gdf" in os.path.relpath(values["filesTrain"]):
                    files_train = len(list(values["-TRAIN-"].split(";")))
                    text = "Number of files selected " + str(files_train)
                    self.__text_box_train.update(text)
```

```
                    self.__dataframes = list(values["-TRAIN-"].split(";"))
                if event == '-TEST-':
                    if ".gdf" in os.path.relpath(values["filesTest"]):
                        if self.__dataframes is not None:
                            file_test = values["filesTest"]
                            self.__text_box_test.update(os.path.basename(values["filesTest"]))
                        else:
                            self.__text_box_test.update("Select train datasets first")
                if event == "-SUBMIT-":
                    if self.__dataframes is None:
                        self.main_title.update("No training datasets selected")
                    elif file_test is None:
                        self.__secondary_title.update("No test dataset selected")
                    else:
                        self.__dataframes.append(file_test)
                        break
                if event == '-EXIT-':
                    self.__option_selected = -1
                    break
                self.__window.refresh()
            self.__window.close()

        """
        Sets the basic parameters to load the GUI
        """
        def __set_parameters(self):
            self.window_size = (280, 160)
            self.main_title = sG.Text("Select datasets to train", background_color=self.default_color)
            main_title_load = sG.Text("Select GDF file to load", background_color=self.default_color, visible=True)
            self.__text_box_train = sG.In(size=(25, 1), expand_x=True, enable_events=True, key="-TRAIN-",
visible=True,
                                          disabled=True)
            files_browse_train = sG.FilesBrowse(key="filesTrain", enable_events=True, visible=True)
            submit = sG.Button('Next', visible=True, key="-SUBMIT-")
            exit_app = sG.Button('Exit', visible=True, key="-EXIT-")
            self.__secondary_title = sG.Text("Select test dataset", background_color=self.default_color,
visible=True)
            self.__text_box_test = sG.In(size=(25, 1), expand_x=True, enable_events=True, key="-TEST-",
visible=True,
                                         disabled=True)
            files_browse_test = sG.FileBrowse(key="filesTest", enable_events=True, visible=True)
            main_menu = [[self.main_title],
                         [self.__text_box_train, files_browse_train],
                         [self.__secondary_title], [self.__text_box_test, files_browse_test],
                         [exit_app, submit], [main_title_load]]
            layout = [
                [
                    sG.Column(main_menu, key='left', background_color=self.default_color, vertical_alignment='t')
                ]
            ]
            self.__window = sG.Window("REE", layout, size=self.window_size,
                                      background_color=self.default_color, finalize=True, icon=self.icon_path)

        """
        Gets the option selected by the user
        @:return: the selected option by the user
        """
        def get_option_selected(self):
            return self.__option_selected

        """
        Gets the info of a dataframe
        @:return: the info of a dataframe
        """
        def get_info(self):
            return self.__dataframes
```

11.3.5. gui_preview.py

```
"""
This is the gui_preview class which  shows the information of the datasets and the parameters to be modified
from the user in order to perform the prediction
author: frangc
version: 3.1
"""
import PySimpleGUI as sG
import os.path
import numpy as np
import sys
import subprocess
from src.frangc.edu.controller import main_controller
from src.frangc.edu.util import preprocessing_data
from src.frangc.edu.gui import gui


class Gui(gui.Gui):
    """
    Default constructor
    @info: infor of the dataframes to be loaded
    """
    def __init__(self, info):
        super().__init__()
        self.__current_data_frame = None
        self.__data_frames = []
        self.__classifier_results = None
        self.__main_controller = None
        self.__events = None
        self.__channels = None
        self.__combo_info_text = None
        self.__combo_dataframes = None
        self.__combo_info = None
        self.__dataframe_info = None
        self.__list_events = None
        self.__list_classifiers = None
        self.__list_channels = None
        self.__submit_text = None
        self.__multiline_size = None
        self.__sample_rate = None
        self.__low_rate = None
        self.__high_rate = None
        self.__check_pca = None
        self.__check_ica = None
        self.__plot = None
        self.__img = None
        self.__submit = None
        self.__info = info
        self.__set_parameters()

    """
    Run the graphic interface
    """
    def run_gui(self):
        collection_cl = None
        collection_ch = None
        collection_events = None
        check_ica = False
        check_pca = False
        error_options = None
```

```
            error = None
            csp = None
            train_low = None
            train_high = None
            test_low = None
            test_high = None
            self.__pre_load()
            while True:
                event, values = self.__window.read()
                print(event)
                if event == '-BACK-':
                    self.get_back = True
                    break
                if event == sG.WIN_CLOSED:
                    break
                if event == '-LISTBOX_CL-':
                    collection_cl = values[event]
                if event == '-COMBO-':
                    if values['-COMBO-'] == self.__combo_info_text[1]:
                        self.__dataframe_info.update(
                            self.__main_controller.get_dataset_detailed_events(self.__current_data_frame))
                    else:

        self.__dataframe_info.update(self.__main_controller.get_dataset_info(self.__current_data_frame))
                if event == '-COMBO_DATA-':
                    self.__current_data_frame = os.path.basename(values['-COMBO_DATA-'])
                    self.__update_info()
                if event == '-LISTBOX_CH-':
                    collection_ch = values[event]
                if event == '-CHECK_ICA-':
                    if check_ica:
                        self.__check_pca.update(disabled=False)
                    else:
                        self.__check_pca.update(disabled=True)
                    check_ica = values[event]
                if event == '-CHECK_PCA-':
                    if check_pca:
                        self.__check_ica.update(disabled=False)
                    else:
                        self.__check_ica.update(disabled=True)
                    check_pca = values[event]
                if event == '-PLOT_ORIGINAL-':
                    self.open_img(self.__img)
                if event == 'Submit':
                    try:
                        error_options = True
                        if collection_cl and collection_ch is not None:
                            if values['-IN_LR-'] != "" and values['-IN_HR-'] != "":
                                if values['-IN_LR-'].isdigit() and values['-IN_HR-'].isdigit():
                                    if int(values['-IN_LR-']) >= 0 and int(values['-IN_HR-']) >= 0:
                                        self.__low_rate = int(values['-IN_LR-'])
                                        self.__high_rate = int(values['-IN_HR-'])
                                        error_options = False

                            if error_options is False and values['-CSP_IN-'] != "" and values['-CSP_IN-'].isdigit():
                                if int(values['-CSP_IN-']) <= 0:
                                    error_options = True
                                else:
                                    csp = int(values['-CSP_IN-'])
                            if error_options is False and preprocessing_data.check_time_set(float(values['-TRAIN_HIGH-
'']),
                                                                                            float(values['-TRAIN_LOW-'])):
                                train_low = float(values['-TRAIN_LOW-'])
                                train_high = float(values['-TRAIN_HIGH-'])
                            else:
                                error_options = True
                            if error_options is False and preprocessing_data.check_time_set(float(values['-TEST_HIGH-
```

```
        ']),
                                                                    float(values['-TEST_LOW-'])):
                        test_low = float(values['-TEST_LOW-'])
                        test_high = float(values['-TEST_HIGH-'])
                    else:
                        error_options = True
                    if error_options:
                        self.__submit_text.update("Incorrect options detected,\ncheck it before running")
                    else:
                        if error is not None:
                            self.error_field.update(visible=False)
                            error = None
                        self.__submit_text.update("Application running...", font=('Arial', 10, "bold"))
                        self.__window.refresh()
                        self.__main_controller.process_file(collection_events, collection_cl, self.__low_rate,
                                                            self.__high_rate, collection_ch, check_ica, check_pca,
                                                            csp, train_low, train_high, test_low, test_high)
                        self.__classifier_results = self.__main_controller.get_classifiers_result()
                else:
                    self.__submit_text.update("Please, choose at least'\none channel and \none classifier")
            except Exception as exception:
                error = str(exception)
                self.error_field.update(str("Errors found:" + str(exception)), visible=True, font=('Arial', 9))
                self.__submit_text.update("")
            if error is None and error_options is False:
                break
        self.__window.refresh()
    os.remove(self.__img)
    self.__window.close()

"""
Sets the basic parameters to load the GUI
"""
def __set_parameters(self):
    self.window_size = (625, 730)
    self.__multiline_size = (55, 7)
    self.__sample_rate = 0
    self.__low_rate = 7
    self.__high_rate = 13
    self.__combo_info_text = ["Dataset summary", "Events"]
    train_low = 0.75
    train_high = 3
    test_low = 0.5
    test_high = 4
    csp_default = 4
    classifiers_list_data = ["Knn", "Mlp", "Tree", "GaussianNB", "SVM", "LDA"]
    self.main_title = sG.Text("Select GDF file to load", background_color=self.default_color)
    classifiers_list = list(np.loadtxt(classifiers_list_data, delimiter=",", dtype=str))
    self.__combo_dataframes = sG.Combo([], font=('Arial', 10), size=(25, 5), enable_events=True,
                                       visible=True, readonly=True, key='-COMBO_DATA-', expand_x=True)
    back = sG.Button('Back', visible=True, key="-BACK-")
    self.__combo_info = sG.Combo([], font=('Arial', 10), size=(25, 5), enable_events=True,
                                 visible=True, readonly=True, key='-COMBO-')
    events_title = sG.Text("Click to see the events", visible=True, background_color=self.default_color)
    plot_title = sG.Text("Plot preview", visible=True, background_color=self.default_color)
    self.__plot = sG.Image(size=(10, 5), visible=True, background_color=self.default_color)
    self.__plot_original = sG.Button('Zoom', visible=True, key="-PLOT_ORIGINAL-")
    dataset_info = sG.Text("Dataset information ", background_color=self.default_color, visible=True)
    self.__dataframe_info = sG.Multiline(size=self.__multiline_size, key='textbox', disabled=True,
visible=True)
    secondary_title = sG.Text("Events", background_color=self.default_color)
    third_title = sG.Text("Classifiers", background_color=self.default_color)
    self.__list_events = sG.Multiline(size=(25, 3), enable_events=True,
                                      visible=True, key='-LISTBOX_EVENTS-')
    self.__list_classifiers = sG.Listbox(classifiers_list, size=(25, 5), enable_events=True,
                                         select_mode='multiple', expand_y=True,
                                         visible=True, key='-LISTBOX_CL-')
```

```
            channels_text = sG.Text("Channels", visible=True,
                                    background_color=self.default_color)
            self.__list_channels = sG.Listbox([], size=(25, 5), enable_events=True,
                                              select_mode='multiple', expand_y=True,
                                              visible=True, key='-LISTBOX_CH-')
            sample_rate_filter_text = sG.Text("Sample rate filter", visible=True,
                                              background_color=self.default_color)
            low_rate_in = sG.In(self.__low_rate, size=(4, 2), enable_events=True,
                                expand_y=True, visible=True, key='-IN_LR-')
            high_rate_in = sG.In(self.__high_rate, size=(4, 2), enable_events=True,
                                 expand_y=True, visible=True, key='-IN_HR-')
            train_text = sG.Text("Set training interval", visible=True,
                                 background_color=self.default_color)
            train_interval_low = sG.In(train_low, size=(4, 2), enable_events=True,
                                       expand_y=True, visible=True, key='-TRAIN_LOW-')
            train_interval_high = sG.In(train_high, size=(4, 2), enable_events=True,
                                        expand_y=True, visible=True, key='-TRAIN_HIGH-')
            test_text = sG.Text("Set test interval", visible=True,
                                background_color=self.default_color)
            test_interval_low = sG.In(test_low, size=(4, 2), enable_events=True,
                                      expand_y=True, visible=True, key='-TEST_LOW-')
            test_interval_high = sG.In(test_high, size=(4, 2), enable_events=True,
                                       expand_y=True, visible=True, key='-TEST_HIGH-')
            self.__submit_text = sG.Text("Select options and \npress submit", visible=True,
                                         background_color=self.default_color, font=('Arial', 10, "bold"))
            csp_text = sG.Text("Number CSP components", visible=True,
                               background_color=self.default_color)
            csp_in = sG.In(csp_default, size=(4, 2), enable_events=True,
                           expand_y=True, visible=True, key='-CSP_IN-')
            self.__check_ica = sG.Checkbox("ICA", default=False, visible=True, key='-CHECK_ICA-',
                                           background_color=self.default_color, enable_events=True)
            self.__check_pca = sG.Checkbox("PCA", default=False, visible=True, key='-CHECK_PCA-',
                                           background_color=self.default_color, enable_events=True)
            self.error_field = sG.Output(size=(20, 20), visible=False, font=('Arial', 10, "bold"))

            self.__submit = sG.Button('Submit')
            left_menu = [[self.main_title], [self.__combo_dataframes],
                         [events_title],
                         [self.__combo_info], [dataset_info], [self.__dataframe_info], [plot_title], [self.__plot],
                         [self.__plot_original]]
            right_menu = [[secondary_title], [self.__list_events], [third_title], [self.__list_classifiers],
                          [channels_text],[self.__list_channels], [sample_rate_filter_text], [low_rate_in, high_rate_in],
                          [train_text],
                          [train_interval_low, train_interval_high], [test_text], [test_interval_low, test_interval_high],
                          [csp_text], [csp_in], [self.__check_pca, self.__check_ica], [self.__submit_text], [self.__submit,
back],
                          [self.error_field]]
            layout = [
                [
                    sG.Column(left_menu, key='left', background_color=self.default_color, vertical_alignment='t',
                              expand_x=True),
                    sG.Column(right_menu, key='right', background_color=self.default_color, vertical_alignment='t')
                ]
            ]
            self.__window = sG.Window("Reconeixement d'esdeveniments d'encefalogrames - REE", layout,
size=self.window_size,
                                      background_color=self.default_color, finalize=True, icon=self.icon_path)

        """
        Preload the data for the first load
        """
        def __pre_load(self):
            if self.__info is not None:
                self.__update_lists(self.__info)
                self.__update_info()
                self.__update_window()
        """
```

```
        Update the window after loading the data
        """
        def __update_window(self):
            self.__combo_info.update(values=self.__combo_info_text, value=self.__combo_info_text[0])
            self.__list_events.update(self.__events, visible=True)
            self.__list_channels.update(self.__channels, visible=True)

        """
         Update the the dataframes combo list
         @info: infor of the dataframes
         """
        def __update_lists(self, info):
            self.__data_frames = info
            self.main_title.update("File: " + os.path.basename(self.__data_frames[0]))
            self.__main_controller = main_controller.MainController(self.__data_frames)
            self.__current_data_frame = os.path.basename(self.__data_frames[-1])
            self.__combo_dataframes.update(values=self.__data_frames, value=self.__data_frames[-1])

        """
         Update the info after loading the fist time or if the user selects another dataframe to be viewed
         @info: infor of the dataframes
         """
        def __update_info(self):
            self.__dataframe_info.update(self.__main_controller.get_dataset_info(self.__current_data_frame))
            self.__events = self.__main_controller.get_dataset_events(self.__current_data_frame)
            self.__channels = self.__main_controller.get_dataset_channels(self.__current_data_frame)
            self.__sample_rate = self.__main_controller.get_sample_rate(self.__current_data_frame)
            self.__combo_info.update(value=self.__combo_info_text[0])
            self.__img = self.__main_controller.get_plot(self.__current_data_frame)
            self.__plot.update(self.__img, subsample=2)
            state = False
            if self.__current_data_frame == os.path.basename(self.__data_frames[-1]):
                dataset_type = " (Test)"
            else:
                state = True
                dataset_type = " (Train)"
            self.__list_events.update(self.__events)
            self.__list_channels.update(self.__channels, visible=True)
            self.__list_classifiers.update(disabled=state)
            self.__list_channels.update(disabled=state)
            self.__submit.update(disabled=state)
            self.main_title.update("File: " + self.__current_data_frame + dataset_type)

        """
         Get the results of the prediction and extra data
         @return: the results of the prediction
         """
        def get_classifiers_result(self):
            return self.__classifier_results

        """
         Load the plot image to the system's image viewer
         @image_path: the path of the image
         """
        def open_img(self, image_path):
            viewer = {'linux': 'xdg-open', 'win32': 'explorer', 'darwin': 'open'}[sys.platform]
            subprocess.run([viewer, image_path])
```

11.3.6. gui_results.py

```
"""
This is the gui_results class that shows the results of the predictions
```

```
        the prediction
        author: frangc
        version: 3.1
        """

        import PySimpleGUI as sG
        from src.frangc.edu.util import printing_data
        from src.frangc.edu.gui import gui


        class GuiResults(gui.Gui):
            """
            Default constructor
            @classifier_results: the results of the prediction session and another metrics
            """
            def __init__(self, classifier_results):
                super().__init__()
                self.__current_dataset = None
                self.__current_process_id = None
                self.__classifier_results = classifier_results
                self.__dataframe_title = None
                self.__window_resize = None
                self.__window_resize_no_list = None
                self.__main_title_next = None
                self.__main_info = None
                self.__save_as_final = None
                self.__current_cl = None
                self.error_field = None
                self.save_field = None
                self.__set_parameters()

            """
            Run the graphic interface
            """
            def run_gui(self):
                resized = False
                while True:
                    saved_path = None
                    error = None
                    self.get_back = False
                    event, values = self.__window.read()
                    if event == "Start again":
                        self.get_back = True
                        break
                    if event == "-COMBO_RESULTS-":
                        self.__current_cl = values[event]
                        if resized is False:
                            self.__window.set_min_size(size=self.__window_resize)
                            resized = True
                        self.__save_as_final.update(visible=True)
                        self.__main_info.update(printing_data.print_confusion_matrix(
                            self.__classifier_results[values[event]][1], values[event], False) + self.__dataframe_title +
                                        self.__classifier_results[values[event]][2].to_markdown(index=False),
                                        visible=True)
                        self.__current_dataset = self.__classifier_results[values[event]][2]
                        self.main_title.update("Results for classifier id")
                        self.__current_process_id = self.__classifier_results[values[event]][0]
                        self.__main_title_next.update(self.__current_process_id, visible=True)
                        self.__window.set_min_size(size=self.__window_resize)
                        self.__window.set_title("Reconeixement d'esdeveniments d'encefalogrames - REE")
                    if event == "Close program" or event == sG.WIN_CLOSED:
                        break
                    try:
                        self.error_field.update(visible=False)
                        self.save_field.update(visible=False)
                        if event == "-SAVE_AS_FINAL-":
                            saved_path = values[event]
```

```
                printing_data.pdf_output(self.__classifier_results[self.__current_cl], self.__current_cl,
                                         self.__current_process_id, values[event])
            except Exception as exception:
                error = str(exception)
                self.error_field.update(str(exception), visible=True)
            if saved_path is not None and error is None:
                self.save_field.update("Report saved in:\n" + values[event], visible=True)
            self.__window.refresh()
        self.__window.close()

    """
    Sets the basic parameters to load the GUI
    """
    def __set_parameters(self):
        self.window_size = (250, 100)
        self.__window_resize = (475, 625)
        self.__window_resize_no_list = (475, 600)
        self.__dataframe_title = "\nDataframe results\n\n"
        self.__default_color = 'grey'
        self.__current_cl = list(self.__classifier_results.keys())[0]
        self.main_title = sG.Text("Results for classifier id", background_color=self.__default_color, pad=(0, 0))
        self.__current_process_id = self.__classifier_results[self.__current_cl][0]
        self.__main_title_next = sG.Text(self.__current_process_id, background_color=self.__default_color,
                                         font=('Arial', 10, "bold"), pad=(0, 0))
        close = sG.Button('Close program')
        start_again = sG.Button('Start again')
        self.__main_info = sG.Multiline(size=(90, 30),
                                        font=('Arial', 9), key='textbox', disabled=True, visible=False)
        list_results = sG.Combo([], "Select classifier", font=('Arial', 10),
                                size=(25, 5), enable_events=True,
                                visible=False, readonly=True, key='-COMBO_RESULTS-')
        self.__save_as_final = sG.SaveAs("Save report", file_types=(('PDF', '.pdf'),),
                                         key='-SAVE_AS_FINAL-', visible=False, enable_events=True)

        self.error_field = sG.Multiline(size=(700, 10), visible=False, font=('Arial', 10, "bold"))
        self.save_field = sG.Text(size=(700, 10), visible=False, background_color=self.__default_color,
                                  font=('Arial', 10, "bold"))
        main_menu = [[self.main_title, self.__main_title_next], [list_results], [self.__main_info],
                     [self.__save_as_final], [close, start_again], [self.save_field, self.error_field]]
        layout = [
            [
                sG.Column(main_menu, key='left', background_color=self.__default_color)
            ]
        ]

        self.__window = sG.Window("Reconeixement d'esdeveniments d'encefalogrames - REE", layout,
                                  size=self.window_size,
                                  background_color=self.__default_color, finalize=True, icon=self.icon_path)
        if len(self.__classifier_results) > 1:
            self.main_title.update("Classifier list")
            self.__main_title_next.update(visible=False)
            list_results.update(values=list(self.__classifier_results.keys()), value="Select classifier")
            list_results.update(visible=True)
            self.__window.set_title("REE")
        else:
            self.__save_as_final.update(visible=True)
            self.__main_info.update(
                printing_data.print_confusion_matrix(self.__classifier_results[self.__current_cl][1],
                                                     self.__current_cl, False) +
                self.__dataframe_title + self.__classifier_results[self.__current_cl][2].to_markdown(index=False),
                visible=True)
            self.window_size = self.__window_resize
            self.__window.set_min_size(size=self.__window_resize_no_list)
            self.__current_dataset = self.__classifier_results[self.__current_cl][2]
```

11.3.7. session_file.py

```
"""
This is session_file class which loads the minimum information to be shown by gui_preview class
author: frangc
version: 3.1
"""

import mne
import os
from datetime import datetime
from src.frangc.edu.model import processed_file


class SessionFile:
    """
    Default constructor
    @data_frames_file_path: the path of the dataframes to be loaded
    """
    def __init__(self, data_frames_file_path):
        self.__data_frames = {}
        self.__classifier_results = None
        self.__processed_file = None
        self.__bci_type = None
        self.__load_data(data_frames_file_path)
        self.__set_id()

    """
    Loads all the dataframe
    @data_frames_file_path: the path of the dataframes to be loaded
    """
    def __load_data(self, data_frames_file_path):
        dataset_type = "Train"
        for data_frame in data_frames_file_path:
            if self.__bci_type is None:
                self.__bci_type = os.path.basename(data_frame)[0]
            elif self.__bci_type != os.path.basename(data_frame)[0]:
                raise Exception("Error - different bci types")
            if data_frame == data_frames_file_path[-1]:
                dataset_type = "Test"
            data_tmp = mne.io.read_raw_gdf(data_frame, verbose=False, preload=True)
            self.__data_frames[os.path.basename(data_frame)] = \
                [dataset_type, data_tmp]

    """
    Sets the date method

    def __set_date(self):
        self.__session_date = datetime.now()"""

    """
    Sets an id for this class
    """
    def __set_id(self, ):
        self.__session_id = 'S_' + datetime.now().strftime("%d%m%Y%H%M%S")

    """
    Gets the session ID
    @return: the session ID
    """
    def get_id(self):
        return self.__session_id

    """
    Return a dataframe given its name
    @data_frame: the name of the dataframe
```

```
        @return: the raw dataframe
        """
        def get_dataset(self, data_frame):
            return self.__data_frames[data_frame][1]

        """
        Return a dataframe's info given its name
        @data_frame: the name of the dataframe
        @return: the info of the dataframe
        """
        def get_dataset_info(self, data_frame):
            return self.__data_frames[data_frame][1].info

        """
        Return a dataframe's sample rate given its name
        @data_frame: the name of the dataframe
        @return: the sample rate of the dataframe dataframe
        """
        def get_sample_rate(self, data_frame):
            return self.__data_frames[data_frame][1].info["sfreq"]

        """
        Return a dataframe's events given its name
        @data_frame: the name of the dataframe
        @return: events of the dataframe
        """
        def get_dataset_events(self, data_frame):
            frame = self.__data_frames[data_frame][1]
            return list(mne.events_from_annotations(frame, verbose=False)[1])

        """
        Return a dataframe's detailed events given its name
        @data_frame: the name of the dataframe
        @return: detailed events of the dataframe
        """
        def get_detailed_events(self, data_frame):
            events_detailed = ""
            for ev in self.get_dataset(data_frame).annotations:
                event_id = ev["description"]
                start = ev["onset"]
                end = ev["onset"] + ev["duration"]
                events_detailed += "Event " + event_id + " goes from " + str(round(start, 10)) + " to " + str(
                    round(end, 10)) + "\n"
            return events_detailed

        """
        Return a dataframe's channels given its name
        @data_frame: the name of the dataframe
        @return: channels of the dataframe
        """
        def get_dataset_channels(self, data_frame):
            frame = self.__data_frames[data_frame][1]
            return list(frame.ch_names)

        """
        Return a pot of dataframe given its name
        @data_frame: the name of the dataframe
        @return: the plot of the dataframe
        """
        def get_plot(self, data_frame):
            file_name = 'temp_figure.png'
            self.__data_frames[data_frame][1].plot(show=False, show_scrollbars=False,
    show_scalebars=False).savefig(
                file_name)
            return file_name

        """
```

```
        Return a pot of dataframe given its name
        Creates the processed_file object that will prepare the data for the prediction
        @events: events of the dataframe
        @classifiers: classifiers to perform the prediction
        @low_r: low rate of the dataframe
        @high_r: high rate of the dataframe
        @channels: channels of the dataframe
        @ica: boolean value to perform ICA
        @pca: boolean value to perform PCA
        @csp: number of CSP components
        @train_low: start of the training data
        @train_high: end of the training data
        @test_low: start of the testing data
        @test_high: end of the testing data
        """
        def process_file(self, events, classifiers, low_r, high_r, channels, ica, pca, csp, train_low, train_high,
                         test_low, test_high):
            self.__processed_file = processed_file.ProcessedFile(self.__data_frames, self.__session_id,
    self.__bci_type,
                                                                 events, classifiers,
                                                                 low_r, high_r, channels, ica, pca, csp, train_low,
                                                                 train_high, test_low, test_high)

        """
        Perform the prediction
        @return: the prediction result and extra data
        """
        def perform_classifiers(self):
            self.__classifier_results = self.__processed_file.predict_data()
            return self.__classifier_results
```

11.3.8. processed_file.py

```
"""
This is the processed_file class which prepare the data in order to construct the dataframes which will be
sent to the
classifier class
author: frangc
version: 3.1
"""

import numpy as np
import mne
from datetime import datetime
from src.frangc.edu.model import classifier
from src.frangc.edu.util import preprocessing_data


class ProcessedFile:
    """
    Default constructor
    @data: dataframes loaded
    @id_file: ID of the session
    @bci_type: type of bci, A or B
    @events: events of the dataframe
    @classifiers: classifiers to perform the prediction
    @low_r: low rate of the dataframe
    @high_r: high rate of the dataframe
    @channels: channels of the dataframe
    @ica: boolean value to perform ICA
    @pca: boolean value to perform PCA
    @csp: number of CSP components
```

```
        @train_low: start of the training data
        @train_high: end of the training data
        @test_low: start of the testing data
        @test_high: end of the testing data
        """
        def __init__(self, data, id_file, bci_type, events, classifiers, low_r, high_r, channels, ica, pca, csp,
    train_low,
                     train_high, test_low, test_high):
            self.__data = data
            self.__events = events
            self.__classifier = None
            self.__session_id = id_file
            self.__processed_dataframe_test = None
            self.__processed_dataframe = None
            self.__bci_type = bci_type
            self.__channels = channels
            self.__ica = ica
            self.__pca = pca
            self.__csp = csp
            self.__train_low = train_low
            self.__train_high = train_high
            self.__test_low = test_low
            self.__test_high = test_high
            self.__set_process_id()
            self.__low_r = low_r
            self.__high_r = high_r
            self.__create_multi_dataframe()
            self.__classifiers = classifiers

        """
        Sets an ID for this class
        """
        def __set_process_id(self):
            self.__processed_id = str(list(self.__data.keys())[-1]) + '_' +
    datetime.now().strftime("%d%m%Y%H%M%S")

        """
        Sets the train and tests dataframe for the future prediction
        """
        def __create_multi_dataframe(self):
            csp = mne.decoding.CSP(n_components=self.__csp, reg=None, log=True, norm_trace=False)
            data_key = str(list(self.__data.keys())[-1])
            annotations_test = self.__data[data_key][1].annotations
            self.__vector_test = []

            for key in self.__data:
                data_raw = self.__data[key][1]
                data_frame_pre = self.__prepare_dataframe(data_raw)
                if key != data_key:
                    if self.__processed_dataframe is None:
                        self.__processed_dataframe = self.__create_dataframe(data_frame_pre, csp, "train")
                    else:
                        train_data, train_labels = self.__create_dataframe(data_frame_pre, csp, "train")
                        self.__processed_dataframe[0] = np.concatenate((self.__processed_dataframe[0], train_data),
    axis=0)
                        self.__processed_dataframe[1] = np.concatenate((self.__processed_dataframe[1], train_labels),
                                                        axis=0)

                else:
                    self.__vector_test.append(self.__get_trials_descriptions(annotations_test))
                    self.__vector_test.append(self.__create_dataframe(data_frame_pre, csp, "test"))
                    self.__vector_test.append(preprocessing_data.get_dataframe_mat(data_key))

        """
        Extract the data for the creation of a dataframe given options
        @data_frame: dataframe to extract the data
        @csp: number of csp components
```

```
        @data_type: type of data, could be train or test
        @return: train data with labels or just test data
        """
        def __create_dataframe(self, data_frame, csp, data_type):

            train_data = None
            train_labels = None
            test_data = None
            event_id = {'769': 10, '770': 11}

            if data_type == "train":
                if self.__bci_type == 'A':
                    event_id = {'769': 7, '770': 8, '771': 9, '772': 10}
                train_data, train_labels = self.__extract_data_from_epoch(data_frame, self.__train_low,
self.__train_high, event_id,
                                                          data_type)
                train_data = self.__csp_data(csp, train_data, train_labels)
                pass
            else:
                event_id = {'783': 11}
                test_data = self.__extract_data_from_epoch(data_frame, self.__test_low, self.__test_high, event_id,
data_type)
                test_data = self.__csp_data(csp, test_data, None)
            if test_data is None:
                return [train_data, train_labels]
            else:
                return test_data

        """
        Runs epoch method to extract the data and events
        @data_frame: dataframe to extract the data
        @start_event: start time of the event
        @end_event: end time of the event
        @event_id: the id of the event
        @data_type: type of data, could be train or test
        @return: the epochs and labels or just the epochs
        """
        def __extract_data_from_epoch(self, data_frame, start_event, end_event, event_id, data_type):
            data = data_frame.copy()
            events, _ = mne.events_from_annotations(data, event_id=event_id)
            epochs = mne.Epochs(
                data,
                events,
                tmin=start_event,
                tmax=end_event,
                proj=True,
                baseline=None,
                preload=True,
            )
            if data_type == "train":
                sub_class = 10
                if self.__bci_type == 'A':
                    sub_class = 6
                labels = np.array(epochs.events[:, -1] - sub_class)
                return [epochs, labels]
            else:
                return epochs

        """
        Performs the CSP technique
        @csp: CSP object
        @data: data to fit and transform (train) or only fit (test)
        @labels: the grountruth of the train data
        """
        def __csp_data(self, csp, data, labels):

            clean_data = preprocessing_data.check_nans(data.get_data())
```

```
            if labels is not None:
                return csp.fit_transform(clean_data, labels)
            else:
                return csp.transform(clean_data)

        """
        Get the trials description for test dataset
        @annotations_test: the annotations of the test dataset
        @return: a vector with the trials description
        """
        def __get_trials_descriptions(self, annotations_test):
            vector_tmp = []
            for ann in annotations_test:
                event_id = ann["description"]
                start_event = int(ann["onset"])
                end_event = int((ann["onset"] + ann["duration"]))
                if event_id == "783":
                    vector_tmp.append("From " + str(
                        round(start_event, 10)) + " to " + str(
                        round(end_event, 10)))
            return vector_tmp

        """
        Prepares the dataframe for extracting the data
        @data_frame: the dataframe to be processed
        @return: a new dataframe
        """
        def __prepare_dataframe(self, data_frame):
            data_raw = None
            if isinstance(self.__high_r, (int, float, complex)) and isinstance(self.__low_r, (int, float, complex)):
                if int(self.__high_r) != 0 and int(self.__low_r) != 0:
                    data_raw = data_frame.copy().filter(l_freq=self.__low_r, h_freq=self.__high_r, verbose=True)
            elif self.__high_r is None or self.__low_r is None:
                data_raw = data_frame.copy().filter(l_freq=self.__low_r, h_freq=self.__high_r, verbose=True)

            return data_raw.copy().pick_channels(self.__channels, verbose=False)

        """
        Performs the prediction of the dest data
        @return: the prediction result and extra data
        """
        def predict_data(self):
            self.__classifier = classifier.Classifier(self.__processed_dataframe, self.__vector_test,
                                                      self.__classifiers, self.__processed_id, self.__ica, self.__pca)
            self.__classifier.train_predict()
            classifier_results = self.__classifier.get_classifier_results()

            return classifier_results
```

11.3.9. classifier.py

```
"""
This is the classifier class which preprocess the data and run the training and predictions
the prediction
author: frangc
version: 3.1
"""

import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from src.frangc.edu.util import preprocessing_data
from sklearn.metrics import confusion_matrix
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import roc_curve, auc


class Classifier:
    """
    Default constructor
    @data: train dataset
    @data_test: test dataset
    @classifier: classifiers type to be performed
    @processed_id: ID of the processed session
    @ica: boolean value to perform ICA or not
    @pca: boolean value to perform pca or not
    """
    def __init__(self, data, data_test, classifiers, processed_id, ica, pca):
        self.__data = data
        self.__data_test = data_test
        self.__ica = ica
        self.__pca = pca
        self.__roc_curve = None
        self.__multi_class = False
        self.__processed_id = processed_id
        self.__classifiers = classifiers
        self.__classifier_result = {}
        self.__trials_results = {}

    """
    Trains, predict and save the results
    """
    def train_predict(self):

        X_train = self.__data[0]
        y_train = self.__data[1]
        X_test = self.__data_test[1]
        y_test = self.__data_test[2]

        X_train, X_test = preprocessing_data.normalize_data(X_train, X_test)

        if self.__pca or self.__ica:
            X_train, X_test = preprocessing_data.component_analysis(X_train, X_test, self.__pca, self.__ica)


        if len(np.unique(y_test)) > 2:
            self.__multi_class = True
        for cl in self.__classifiers:
            model_trained = self.__perform_training(X_train, y_train, cl)
            predict = model_trained.predict(X_test)

            if self.__multi_class:
                self.__set_multiclass_roc_curve(y_test, X_train, y_train, X_test, cl)
            else:
                self.__set_binary_roc_curve(y_test, model_trained.predict_proba(X_test)[:, 1])
            self.__trials_results[cl] = predict
            self.__save_results(y_test, cl)

    """
    Performs training
    @X_train: dataset to be trained
    @Y_train: groundtruth of the train data
    @cl: classifier's type to perform the training
    """
    def __perform_training(self, X_train, y_train, cl):
```

```
        neighbors = 100
        max_iter = 3000
        model_trained = None

        if cl == "Knn":
            model_trained = KNeighborsClassifier(n_neighbors=neighbors).fit(X_train, y_train)
        elif cl == "Mlp":
            model_trained = MLPClassifier(max_iter=max_iter).fit(X_train, y_train)
        elif cl == "Tree":
            model_trained = DecisionTreeClassifier().fit(X_train, y_train)
        elif cl == "GaussianNB":
            model_trained = GaussianNB().fit(X_train, y_train)
        elif cl == "LDA":
            model_trained = LinearDiscriminantAnalysis().fit(X_train, y_train)
        elif cl == "SVM":
            model_trained = svm.SVC(probability=True).fit(X_train, y_train)

        return model_trained

    """
    Creates a ROC curve for binary classification
    @y_test: groundtruth of the test data
    @y_pred_proba: probabilities of the test data
    """
    def __set_binary_roc_curve(self, y_test, y_pred_proba):
        self.__roc_curve = []
        fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
        self.__roc_curve.append(auc(fpr, tpr))
        self.__roc_curve.append([fpr, tpr, thresholds])

    """
    Creates a ROC curve for multiclass classification
    @X_train: dataset to be trained
    @Y_train: groundtruth of the train data
    @X_test: dataset to be predicted
    @y_test: groundtruth of the test data
    @cl: classifier's type to perform the training
    """
    def __set_multiclass_roc_curve(self, y_test, X_train, y_train, X_test, cl):

        self.__roc_curve = []

        n_classes = np.unique(y_train)
        fpr = [0] * len(n_classes)
        tpr = [0] * len(n_classes)
        thresholds = [0] * len(n_classes)
        for event_class in range(len(n_classes)):
            vector_temp = []
            train = np.hstack((X_train, y_train.reshape(y_train.size, 1)))
            test = np.hstack((X_test, y_test.reshape(y_test.size, 1)))

            np.place(train[:,-1], train[:,-1] != event_class + 1, 0)
            np.place(train[:, -1], train[:, -1] == event_class + 1, 1)
            np.place(test[:,-1], test[:,-1] != event_class + 1, 0)
            np.place(test[:, -1], test[:, -1] == event_class + 1, 1)


            X_train_temp = train[:, :-1]
            y_train_temp = train[:, -1]
            X_test_temp = test[:, :-1]
            y_test_temp = test[:, -1]

            second_model_trained = self.__perform_training(X_train_temp, y_train_temp, cl)
            y_pred_proba = second_model_trained.predict_proba(X_test_temp)[:, 1]
            fpr[event_class], tpr[event_class], thresholds[event_class] = roc_curve(y_test_temp, y_pred_proba)
            vector_temp.append(auc(fpr[event_class], tpr[event_class]))
            vector_temp.append([fpr[event_class], tpr[event_class], thresholds[event_class]])
```

```
                self.__roc_curve.append(vector_temp)

        """
        Saves the results of the prediction
        @y_test: groundtruth of the test data
        @cl: classifier's type of the prediction
        """
        def __save_results(self, y_test, cl):

            if self.__multi_class is False:
                cm = self.__get_binary_confusion_results(self.__trials_results[cl], y_test)
            else:
                cm = self.__get_multiclass_confusion_results(self.__trials_results[cl], y_test)

            final_dataframe = self.__create_final_dataset(self.__data_test[0], y_test, self.__trials_results[cl])
            self.__classifier_result[cl] = [self.__processed_id, cm, final_dataframe, self.__roc_curve]

        """
        Creates a final dataset give the results of the prediction
        @X_test: dataset to be predicted
        @y_test: groundtruth of the test data
        @final_prediction: the groundtruth predicted by the classifier
        """
        def __create_final_dataset(self, X_test, y_test, final_prediction):

            df_temp = pd.concat([pd.DataFrame.from_dict(X_test), pd.DataFrame(y_test),
    pd.DataFrame(final_prediction)],
                                axis=1)
            df_temp.columns = ['Trial times (s)', 'GT', 'Pred']
            return df_temp

        """
        Creates a confusion matrix for binary classification
        @predict: the groundtruth predicted by the classifier
        @ground_truth: the groundtruth of the test_dataset
        """
        def __get_binary_confusion_results(self, predict, ground_truth):

            tp = np.sum(predict * ground_truth).astype(int)
            tn = np.sum((1 - predict) * (1 - ground_truth)).astype(int)
            fp = np.sum(predict * (1 - ground_truth)).astype(int)
            fn = np.sum((1 - predict) * ground_truth).astype(int)

            return [tp, tn, fp, fn, None]

        """
        Creates a confusion matrix for multiclass classification
        @predict: the groundtruth predicted by the classifier
        @ground_truth: the groundtruth of the test_dataset
        """
        def __get_multiclass_confusion_results(self, predict, ground_truth):

            classes = np.unique(ground_truth)
            classes_result = {}

            cm = confusion_matrix(ground_truth, predict, labels=classes)

            for i in range(len(cm)):
                current_class = i + 1
                # TP
                classes_result[current_class] = np.zeros(len(classes))
                classes_result[current_class][0] += cm[i][i]
                for a in range(len(cm)):
                    if a + 1 != current_class:
                        # FN
                        classes_result[current_class][1] += cm[i][a]
                        # FP
```

```
                    classes_result[current_class][2] += cm[a][i]
                    # TN
                    for z in range(len(cm)):
                        if z + 1 != current_class:
                            classes_result[current_class][3] += cm[a][z]

        return [classes_result, cm]

    """
    Gets the classifiers result
    @return: return the dictionary with the classifiers results
    """
    def get_classifier_results(self):
        return self.__classifier_result
```

11.3.10. preprocessing_data.py

```
"""
Library that store methods to be used by some classes of the model
author: frangc
version: 3.1
"""
import numpy as np
import scipy.io
import pandas as pd
import pathlib
import os
from sklearn.decomposition import FastICA

"""
Check if the nan values is not > 50% of the entire data an change them by the average of the column
@return: the data without nan values
"""


def check_nans(data):
    raise_except = False

    if isinstance(data, np.ndarray):
        nan_values = np.count_nonzero(np.isnan(data))
        if nan_values == 0:
            if nan_values / data.size > 0.5:
                raise_except = True
            else:
                col_means = np.nanmean(data, axis=0)
                nan_positions = np.where(np.isnan(data))
                data[nan_positions] = np.take(col_means, nan_positions[1])
    else:
        nan_values = data.isna().sum().sum()
        if nan_values > 0:
            if nan_values > 0 and nan_values / (len(data) * len(data.columns)) > 0.25:
                raise_except = True
            else:
                the_mean = data.mean()
                data.fillna(the_mean, inplace=True)

    if raise_except:
        raise Exception("Error - Too much NaN values")
    else:
        return data
```

```
    """
    Normalize the data with standardization
    @return: the data standardized
    """


    def normalize_data(train_data, test_data):
        train_means = np.mean(train_data, 0)
        train_std = np.std(train_data, 0)
        train_data_norm = np.divide((train_data - train_means), train_std)
        if test_data is None:
            return train_data_norm
        else:
            test_data_norm = np.divide((test_data - train_means), train_std)
        return train_data_norm, test_data_norm


    """
    Applies PCA or ICA to the data
    @X_trainN: Train normalized data
    @X_testN: Test normalized data
    @pca: boolean value to determine whether to apply PCA
    @ica: boolean value to determine whether to apply PCA
    @return: the data with PCA or ICA applied
    """


    def component_analysis(X_trainN, X_testN, pca, ica):
        X_train_final = None
        X_test_final = None
        train_means = np.mean(X_trainN, 0)
        test_means = np.mean(X_testN, 0)
        X_trainN = np.subtract(X_trainN, train_means)
        X_testN = np.subtract(X_testN, test_means)
        numSamples = X_trainN.shape[0]
        theCovariance = np.dot(X_trainN.T, X_trainN) / numSamples

        [eigen_values, eigen_vectors] = np.linalg.eigh(theCovariance)

        sort = np.argsort(eigen_values, axis=0)[::-1]
        eigen_values, eigen_vectors = [eigen_values[sort], eigen_vectors[:, sort]]
        variance_summary = np.cumsum(np.divide(eigen_values, np.sum(eigen_values)))
        num_vectors = np.argmax(variance_summary > .95)
        if pca:
            eigen_vectors = eigen_vectors[:, :num_vectors + 1]
            projected_transpose = np.transpose(eigen_vectors)
            original_transpose = np.transpose(X_trainN)
            result_mult = np.matmul(projected_transpose, original_transpose)
            X_train_final = np.transpose(result_mult)
            result_mult = np.matmul(projected_transpose, np.transpose(X_testN))
            X_test_final = np.transpose(result_mult)
        elif ica:
            ICA = FastICA(n_components=num_vectors)
            X_train_final = ICA.fit_transform(X_trainN)
            X_test_final = ICA.transform(X_testN)

        return X_train_final, X_test_final


    """
    Loads the true labels (groundtruth) of the test dataset
    @original_file: name of the file to load
    """


    def get_dataframe_mat(original_file):
        mat_file = original_file.replace('gdf', 'mat')
```

```
    current_path = os.path.dirname(os.path.abspath(__file__))
    final_path = str(pathlib.Path(current_path).parents[3])
    mat = scipy.io.loadmat(os.path.abspath(final_path + "/true_labels/" + mat_file))

    data_frame = pd.DataFrame(mat['classlabel'])

    if data_frame[0].nunique() == 2:
        data_frame.loc[data_frame[0] == 1, 0] = 0
        data_frame.loc[data_frame[0] == 2, 0] = 1

    data_frame.columns = ["GT"]

    return data_frame.to_numpy().reshape(-1)


"""
Checks that the low rate set by the user is no more than the high rate
@return: a true or false value
"""


def check_time_set(high, low):
    result = False
    if high != "" and low != "":
        if 0 <= float(low) < float(high) and float(high) >= 0:
            result = True
    return result
```

11.3.11. printing_data.py

```
"""
Library that store printing methods to be used by the gui_results class
author: frangc
version: 3.1
"""

import numpy as np
import fpdf
import matplotlib.pyplot as plt
import seaborn as sns
import os

"""
Prints confusion matrix
@array_results: array with all the results
@cl: classifier performed for the prediction
@report: boolean value to check if the user has asked for a report
@return: string with the confusion matrix printed
"""


def print_confusion_matrix(array_results, cl, report):
    """Prints the results of each classified"""

    result = 'Classifier: ' + cl + '\n'
    result += '-----------------------------------------------\n'
    result += 'Confusion Matrix\n'

    if len(array_results) > 2:
        confusion_matrix = get_cm(array_results)
        result += str(confusion_matrix[0]) + '\n'
        result += str(confusion_matrix[1]) + '\n'
```

```
            result += print_binary_class_results(confusion_matrix)
        else:
            result += str(array_results[1]) + '\n'
            result += '\n' + 'Kappa Score: ' + str(
                kappa_score_manually(array_results[1])) + '\n'

            if report is False:
                result += print_multiclass_results(array_results, None)

        return result


    """
    Prints results for binary classification
    @confusion_matrix: confusion matrix for binary classification
    @return: metrics and kappa's score
    """


    def print_binary_class_results(confusion_matrix):
        tp = confusion_matrix[0][0]
        fp = confusion_matrix[0][1]
        fn = confusion_matrix[1][0]
        tn = confusion_matrix[1][1]

        accuracy = str(round((tp + tn) / (tp + tn + fp + fn), 2)) + '\n'
        result = get_results(tp, fn, fp, tn, accuracy)
        result += 'Kappa Score ' + str(kappa_score_manually(np.array(confusion_matrix))) + '\n'
        return result


    """
    Prints results for multiclass classification
    @array_results: results of the multiclass classification
    @return: metrics and kappa's score
    """


    def print_multiclass_results(array_results, class_id):
        if class_id is not None:
            result = '\nResults for class ' + str(class_id) + '\n'
            result += get_results(array_results[0], array_results[1], array_results[2],
                                  array_results[3], None)
        else:
            result = ""
            for key in array_results[0].keys():
                result += '\nResults for class ' + str(key) + '\n'
                result += get_results(array_results[0][key][0], array_results[0][key][1], array_results[0][key][2],
                                      array_results[0][key][3], None)
        return result


    """
    Calculates metrics for the results
    @tp: true positives
    @fn: false negatives
    @fp: false positives
    @tn: true negatives
    @accuracy: accuracy value
    @return: a string containing the metrix
    """


    def get_results(tp, fn, fp, tn, accuracy):
        result_tmp = ""
        result_tmp += 'Right predictions: ' + str(tp + tn) + '\n'
        result_tmp += 'Wrong predictions: ' + str(fp + fn) + '\n\n'
```

```
        if accuracy is None:
            accuracy = str(round((tp + tn) / (tp + tn + fp + fn), 2)) + '\n'
        result_tmp += 'Accuracy: ' + accuracy
        result_tmp += 'Precision: ' + str(round(tp / (tp + fp), 2)) + '\n'
        result_tmp += 'Recall: ' + str(round(tp / (tp + fn), 2)) + '\n'
        result_tmp += 'Fall-out: ' + str(round(fp / (tn + fp), 2)) + '\n'
        result_tmp += '----------------------------------------------\n'

        return result_tmp


    """
    Calculates the cohen's kappa value'
    @cm: confusion matrix
    @return: cohen's kappa value'
    """


    def kappa_score_manually(cm):
        n = np.sum(cm)
        sum0 = np.sum(cm, axis=0)
        sum1 = np.sum(cm, axis=1)
        expected = np.outer(sum0, sum1) / n

        n_classes = cm.shape[0]
        identity = np.identity(n_classes)
        p_o = np.sum((identity * cm) / n)
        p_e = np.sum((identity * expected) / n)
        return round((p_o - p_e) / (1 - p_e), 3)


    """
    Creates a PDF document with all the metrics and plot
    @data: array with different results
    @title: title of the document
    @path: path of the document
    """


    def pdf_output(data, cl, title, path):
        multi_class = False
        confusion_matrix = None
        pdf = fpdf.FPDF()
        pdf.add_page()
        pdf.set_font('Arial', 'B', 10)
        pdf.multi_cell(0, 5, "Dataset evaluated: " + title + '\n' + "Classifier: " + cl)
        pdf.set_font('Arial', '', 10)

        if isinstance(data[1][0], dict):
            multi_class = True
            draw_confusion_matrix(data[1][1], list(data[1][0].keys()), cl)
        else:
            confusion_matrix = get_cm(data[1])
            draw_confusion_matrix(confusion_matrix, ['0', '1'], cl)

        pdf.image('./cm_' + str(cl) + '.png', 50, 20, 100, 100)
        os.remove('./cm_' + str(cl) + '.png')
        pdf.multi_cell(100, 100, "")
        if multi_class:
            y = 210
            pdf.cell(100, 30, 'Kappa Score: ' + str(kappa_score_manually(np.array(data[1][1]))), ln=1)
            pdf.set_font('Arial', 'B', 10)
            pdf.cell(100, 5, "Metrics and ROC Curve", ln=2)
            pdf.set_font('Arial', '', 10)
            for key in data[1][0].keys():
                pdf.multi_cell(100, 5, print_multiclass_results(data[1][0][key], key))
                pdf.cell(100, 5, "ROC score " + str(round(data[3][key - 1][0], 2)))
```

```
                draw_roc_curve([data[3][key - 1][0], data[3][key - 1][1]], key)
                pdf.image('./' + str(key) + '.png', 50, y, 100)
                os.remove('./' + str(key) + '.png')
                if y == 60:
                    y = 200
                    pdf.multi_cell(100, 75, "")
                else:
                    y = 60
                    pdf.add_page()
        else:
            pdf.multi_cell(100, 5, print_binary_class_results(confusion_matrix))
            pdf.multi_cell(10, 10, "")
            pdf.set_font('Arial', 'B', 10)
            pdf.cell(100, 5, "ROC Curve", ln=1)
            pdf.set_font('Arial', '', 10)
            pdf.cell(100, 5, "ROC score " + str(round(data[3][0], 2)), ln=2)
            draw_roc_curve([data[3][0], data[3][1]], cl)
            pdf.image('./' + str(cl) + '.png', 50, 175, 100, 100)
            os.remove('./' + str(cl) + '.png')
            pdf.add_page()

        pdf.set_font('Arial', 'B', 10)
        pdf.cell(100, 20, "Final Dataset ", ln=1)
        pdf.set_font('Arial', '', 10)
        pdf.multi_cell(100, 5, data[2].to_string(index=False))

        pdf.output(path)


    """
    Draws a roc curve for the pdf document
    @roc_curve: array with roc score and other results
    @file_name: name of the file with the plot to be saved
    """


    def draw_roc_curve(roc_curve, file_name):
        plt.figure()
        plt.plot(roc_curve[1][0], roc_curve[1][1], label='ROC curve (area = %0.2f)' % roc_curve[0])
        plt.plot([0, 1], [0, 1], 'k--', label='No Skill')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('ROC Curve')
        plt.legend()
        plt.savefig(str(file_name) + '.png')


    """
    Draws a confusion matrix for the pdf document
    @cm: confusion matrix
    @classes: classes of the data
    @file_name: name of the file with the plot to be saved
    """


    def draw_confusion_matrix(cm, classes, file_name):
        fig = plt.figure()
        confusion_plot = fig.add_subplot(111)
        sns.heatmap(cm, annot=True, fmt='g', ax=confusion_plot)

        confusion_plot.set_xlabel('Predicted labels')
        confusion_plot.set_ylabel('True labels')
        confusion_plot.set_title('Confusion Matrix')
        confusion_plot.xaxis.set_ticklabels(classes)
        confusion_plot.yaxis.set_ticklabels(classes)
```

```
        confusion_plot.figure.savefig('cm_' + str(file_name) + '.png')


    """
    Gets a confusion matrix given an array
    @return: confusion matrix
    """


    def get_cm(data):
        return [data[0], data[2]], [data[3], data[1]]
```