Detecció precoç de malalties cardiovasculars en pacients amb Diabetis Mellitus tipus 1

Sergi Grau Corral
Màster Universitari Enginyeria Informàtica – UOC
TFM - Àrea de Intel·ligència Artificial

Nom Consultor: Samir Kanaan Izquierdo
Nom Professor responsable de l'assignatura: Carles Ventura Royo

Data Lliurament: 05/06/2018

El meu agraïment a Cristina Colom i Cristina Baeza, per les dades en que s'ha basat aquest treball.

A Carles Ventura, i sobretot a Samir Kanaan per poder fer el treball en l'àrea de I.A. de la U.O.C., per la seva paciència e indicacions constants per intentar fer-me anar per el bon camí.

A la meva família, pel seu suport en tot el que faig.

FITXA DEL TREBALL FINAL

Títol del treball:	Detecció precoç de malalties cardiovasculars en pacients amb Diabetis Mellitus tipus 1.
Nom de l'autor:	Sergi Grau Corral
Nom del consultor/a:	Samir Kanaan Izquierdo
Nom del PRA:	Carles Ventura Royo
Data de lliurament (mm/aaaa):	06/2018
Titulació o programa:	Màster Universitari Enginyeria Informàtica
Àrea del Treball Final:	Intel·ligència Artificial
Idioma del treball:	Català
Paraules clau	Machine learning, diabetes, cardiovascular disease

Resum del Treball (màxim 250 paraules):

La malaltia cardiovascular es més freqüent i precoç en pacients amb DM1 sent la primera causa de mortalitat. No hi ha informació suficient sobre quins són els factors determinants.

El que es vol aconseguir es un sistema basat en *Machine Learning* que, basat en uns anàlisis previs, pugui recomanar fer un TCMD al pacient per poder confirmar el risc i fer les accions oportunes per reduir-lo. A més de llistar els anàlisis recomanats i en quin ordre.

Abstract (in English, 250 words or less):

Cardiovascular problems are frequent, and happen earlier, in patients with DM1. They are considered as the first cause of death of those patients. There is not enough information on which are the determinant factors. The diagnosis has the extra difficulty that problems happen in young people, and do need a close medical follow up for long time.

Based on AI techniques, as Machine Learning, it has been presented a model to be able to predict the risk of a cardiovascular injury. It can be used to recommend a patient do more exhaustive analysis, as get a Multidetector Computed Tomography (MDCT). The work also shows a list of analysis recommended, and in which order, to help with the diagnosis.

Índex

Llista de figures

Llista de Taules

1 Introducció

1.1 Context i justificació del Treball

La malaltia cardiovascular es més freqüent i precoç en pacients amb Diabetis Mellitus tipus 1 (DM1). El risc de patir una episodi cardiovascular augmenta en la població que te diagnosticada DM1 arribant a ser la seva principal causa de mort. El primer estudi sobre la mortalitat per malaltia cardiovascular en pacients amb DM1 data del 1987 i mostra que a l'edat de 55 anys, la taxa de mortalitat acumulada per malaltia arterial coronaria (*coronary artery disease* –CAD-) es del 35 +/- 5%, molt més alta que la taxa corresponent per a les persones no diabètiques [12].

El risc de mort per causa cardiovascular es per tant més gran en aquesta població que en gent sense diabetis de la mateixa edat. No hi ha informació suficient sobre quins són els factors determinants implicats en la arteriosclerosis precoç en DM1. Això es degut a un diagnòstic tardà i a la necessitat de fer un seguiment mèdic molt prolongat. A demés, una prova com la Tomografia Computada Multi Detectora (TCMD) amb contrast, no esta disponible en tots els centres mèdics, ni tampoc es una prova comú al principi del diagnòstic de la diabetis.

El que es vol aconseguir es un sistema que, basat en una anàlisis previs, pugui recomanar fer un TCMD al pacient per poder confirmar el risc d'una lesió cardíaca. A demés de llistar els anàlisis recomanats i en quin ordre.

Amb un model creat amb el programa adjunt, entrenat amb les dades de l'estudi, es podrien afegir les dades d'un nou pacient per veure si el codi recomana enviar-lo a un centre dotat de TCMD, per confirmar el seu risc de patir una lesió cardíaca.

1.2 Objectius del Treball

De forma resumida, tenim cinc objectius principals;

- **Objectiu_1:** Extreure les característiques més rellevants de la població que forma part de l'estudi clínic. Es farà un estudi i preparació de les dades inicials. Partim d'uns 77 pacients diagnosticats amb DM1, als que se'ls ha realitzat un seguiment de la seva malaltia durant l'estudi clínic, disposant de moltes dades (anàlisis mèdics) per pacient.

- **Objectiu_2:** Entendre les dades de que disposem. Per poder aplicar models de classificació necessitem tenir un conjunt de dades sense

repeticions, sense buits, i amb el major número de mostres possibles. Com que no hi ha més pacients que els que formen part de l'estudi, el que es farà serà intentar que aquestes dades siguin determinants per poder obtenir els millors resultats en els models. Per entendre les dades es faran també reduccions de la dimensionalitat.

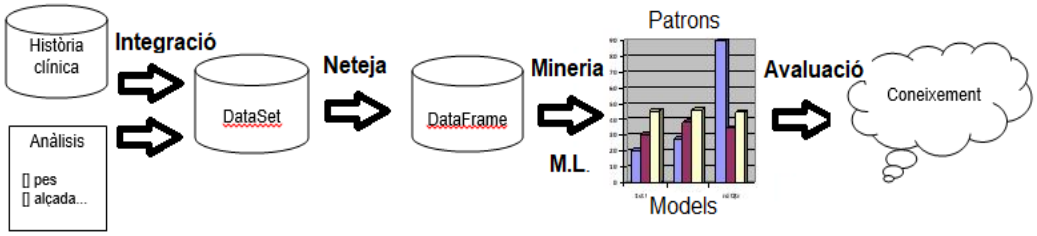
- **Objectiu_3:** Voldrem poder predir si un nou pacient te risc de malaltia cardíaca. Per això es crearan models de predicció basant-se en la variable 'TC2', que ens determina si hi ha lesions cardíaques. En definitiva, s'intentarà implementar diferents models per veure quin pot acomplir millor l'objectiu de predir si un nou pacient de DM1 te risc de lesió cardíaca.

- **Objectiu_4:** Ajudar el metge en el diagnòstic. S'implementaran arbres de decisió que potser podrien guiar el metge en el diagnòstic, a partir de valors de diferents anàlisis mèdics.

- **Objectiu_5:** Determinar les variables/anàlisis mèdics més rellevants. Veient els arbres de decisió, podrem determinar quines són les característiques més utilitzades. Seran aquests els anàlisis recomanats.

1.3 Enfocament i mètode seguit

El següent gràfic mostra, de forma general, el que es vol aconseguir. Es vol donar un punt de vista alternatiu al del propi domini (interpretació estrictament mèdica) o a una visió des de el punt de vista de la estadística tradicional. Així, de les diferents estratègies s'ha optat per utilitzar els algoritmes que formen part de la Intel·ligència Artificial (estudiats en la assignatura del Màster) per obtenir resultats a partir de les dades inicials. En lloc d'utilitzar un programari existent, s'ha utilitzat Python per crear dos codis: un enfocat en la neteja de les dades i un altre centrat en el seu anàlisi i la creació de models.



Il·lustració 1: procés seguit per el treball.

El punt de partida del treball es un DataSet amb dades de pacients amb Diabetis Mellitus de Tipus 1. Aquest va ser generat durant un estudi clínic, a partir de la integració de diferents fonts. Aquest conjunt de dades s'han netejat per poder fer servir després diferents tècniques de Intel·ligència Artificial i *Machine Learning*. Els diferents models obtinguts

han estat avaluats segons el grau de precisió i han permès acomplir els diferents objectius del treball. Es poden utilitzar, per exemple, per predir si un pacient amb DM1 nou tindrà una lesió cardíaca, quina es la importància dels diferents anàlisis i en quin ordre estaran recomanats.

1.4 Planificació del Treball

S'ha pogut disposar de unes dades d'un estudi clínic sobre les que s'han implementat diferents models de Machine Learning.

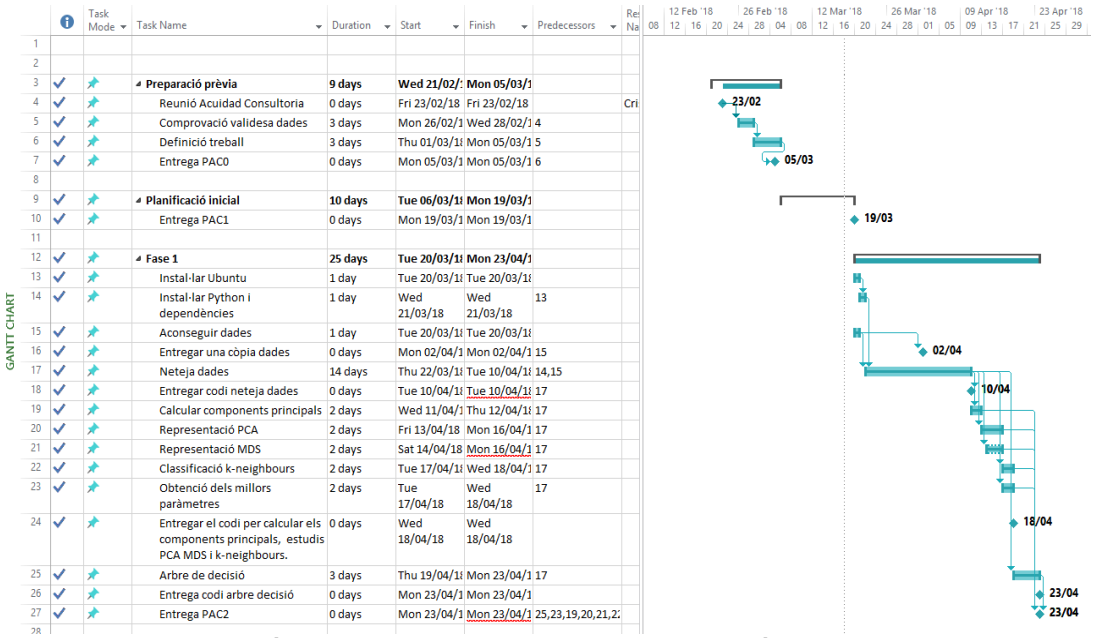
De forma general, el treball ha seguit la següent planificació.

Fase	PAC	Explicació	Fites	Entrega/Data
Definició	PAC0	Definició del treball. Accés a un estudi mèdic.	Definició del treball a fer.	Entrega PAC0 05/03. FET.
Pla	PAC1	Fer la planificació inicial. Contactes amb la investigadora.	Planificació inicial.	Entrega PAC1 el 19/03. FET.
Fase 1	PAC2	Instal·lar PC amb Ubuntu. Instal·lar Phython i dependències. Aconseguir dades mèdiques. Primera neteja (dades redundants). Aprendre l'ús de Pandas i utilitzar les seves estructures per tractar les dades. Calcular components principals. Representacions PCA i MDS per veure com són les dades. Classificació arbre k-neighbours. Afinar el procediment per obtenir els millors resultats. Creació d'un arbre de decisió.	Crear entorn de treball. Endreçar dades de forma automàtica. Calcular components principals, estudis PCA i MDS i classificació k-neighbours. Arbre de decisió.	Entregar una copia de les dades (02/04). FET. Entregar el programari per adaptar les dades (10/04). FET. Entregar el codi per calcular els components principals, estudis PCA MDS i k-nieghbours (18/04). FET. Entregar el codi amb els primers intents d'arbre de decisió (23/04). FET. Entrega PAC2 el 23/04. FET

Fase 2	PAC3	Buscar altres mètodes de classificació i escollir el més adient. Determinar quines variables determinen el risc de malaltia cardíaca. Neteja i publicació dels programes utilitzats. Si hi ha temps, començar amb la redacció de la memòria.	Designació del millor mètode de classificació per les nostres dades. Llistat amb les anàlisis més importants per el nostre objectiu. Programari creat per aquest treball.	Entregar el codi amb arbres de decisió més elaborats i/o qualsevol entrega pendent de la fase 1 (04/05). FET (l'inicial). Llistat de les variables (anàlisis mèdics) més rellevants (07/05). FET. Programari creat, versió 1 (07/05). FET. Programari creat, versió final (21/05). FET. Entrega PAC3 el 21/05. FET.
Memòria	PAC4	Primera redacció. Revisió de la redacció (en català).	Dos esborranys de la memòria i una versió final.	Entrega primer esborrany memòria (25/05). FET. Entrega segon esborrany (01/06). FET. Entrega final amb correccions. Entrega PAC4 el 05/06. FET.
Presentació	PAC5a	Investigar programari per fer la presentació. Preparació de la presentació.	Presentació del treball en versió digital segons els requeriments de la assignatura.	Entrega de la presentació. Entrega PAC5a el 13/06. FET.
Defensa	PAC5b	Defensa del treball	Treball final (codi, memòria i presentació).	Defensa del treball de final de màster. Entrega PAC5b el 25/06.

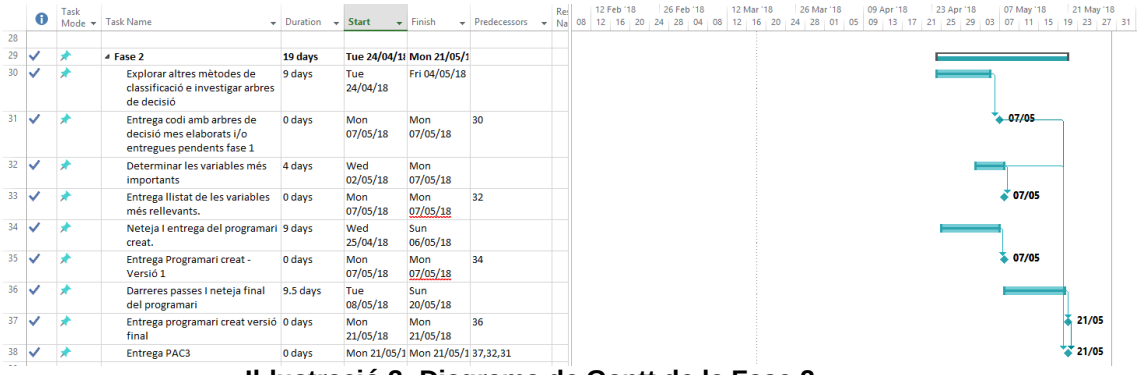
Taula 1: Resum de Fases, PACs i fites.

Juntament amb el treball s'ha entregat un document de Microsoft Project amb la descripció de les tasques amb més detall, així com informes de seguiment i riscos.



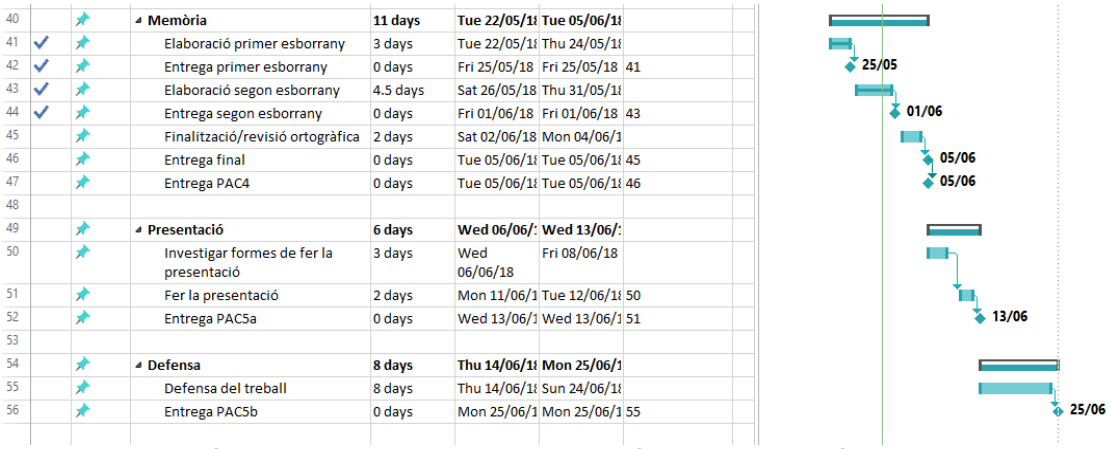
	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1						
2						
3	✓	**Preparació prèvia**	**9 days**	**Wed 21/02/**	**Mon 05/03/1**	
4	✓	Reunió Acuidad Consultoria	0 days	Fri 23/02/18	Fri 23/02/18	
5	✓	Comprovació validesa dades	3 days	Mon 26/02/1	Wed 28/02/1	4
6	✓	Definició treball	3 days	Thu 01/03/18	Mon 05/03/1	5
7	✓	Entrega PAC0	0 days	Mon 05/03/1	Mon 05/03/1	6
8						
9	✓	**Planificació inicial**	**10 days**	**Tue 06/03/18**	**Mon 19/03/1**	
10	✓	Entrega PAC1	0 days	Mon 19/03/1	Mon 19/03/1	
11						
12	✓	**Fase 1**	**25 days**	**Tue 20/03/18**	**Mon 23/04/1**	
13	✓	Instal·lar Ubuntu	1 day	Tue 20/03/18	Tue 20/03/18	
14	✓	Instal·lar Python i dependències	1 day	Wed 21/03/18	Wed 21/03/18	13
15	✓	Aconseguir dades	1 day	Tue 20/03/18	Tue 20/03/18	
16	✓	Entregar una còpia dades	0 days	Mon 02/04/1	Mon 02/04/1	15
17	✓	Neteja dades	14 days	Thu 22/03/18	Tue 10/04/18	14,15
18	✓	Entregar codi neteja dades	0 days	Tue 10/04/18	Tue 10/04/18	17
19	✓	Calcular components principals	2 days	Wed 11/04/1	Thu 12/04/18	17
20	✓	Representació PCA	2 days	Fri 13/04/18	Mon 16/04/1	17
21	✓	Representació MDS	2 days	Sat 14/04/18	Mon 16/04/1	17
22	✓	Classificació k-neighbours	2 days	Tue 17/04/18	Wed 18/04/1	17
23	✓	Obtenció dels millors paràmetres	2 days	Tue 17/04/18	Wed 18/04/18	17
24	✓	Entregar el codi per calcular els components principals, estudis PCA MDS i k-neighbours.	0 days	Wed 18/04/18	Wed 18/04/18	
25	✓	Arbre de decisió	3 days	Thu 19/04/18	Mon 23/04/1	17
26	✓	Entrega codi arbre decisió	0 days	Mon 23/04/1	Mon 23/04/1	
27	✓	Entrega PAC2	0 days	Mon 23/04/1	Mon 23/04/1	25,23,19,20,21,2

Il·lustració 2: Diagrama de Gantt de la Preparació a la Fase 1.



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
28						
29	✓	**Fase 2**	**19 days**	**Tue 24/04/18**	**Mon 21/05/1**	
30	✓	Explorar altres mètodes de classificació e investigar arbres de decisió	9 days	Tue 24/04/18	Fri 04/05/18	
31	✓	Entrega codi amb arbres de decisió mes elaborats i/o entregues pendents fase 1	0 days	Mon 07/05/18	Mon 07/05/18	30
32	✓	Determinar les variables més importants	4 days	Wed 02/05/18	Mon 07/05/18	
33	✓	Entrega llistat de les variables més rellevants.	0 days	Mon 07/05/18	Mon 07/05/18	32
34	✓	Neteja I entrega del programari creat.	9 days	Wed 25/04/18	Sun 06/05/18	
35	✓	Entrega Programari creat - Versió 1	0 days	Mon 07/05/18	Mon 07/05/18	34
36	✓	Darreres passes I neteja final del programari	9.5 days	Tue 08/05/18	Sun 20/05/18	
37	✓	Entrega programari creat versió final	0 days	Mon 21/05/18	Mon 21/05/18	36
38	✓	Entrega PAC3	0 days	Mon 21/05/1	Mon 21/05/1	37,32,31

Il·lustració 3: Diagrama de Gantt de la Fase 2.



40			**Memòria**	**11 days**	**Tue 22/05/18**	**Tue 05/06/18**
41	✓		Elaboració primer esborrany	3 days	Tue 22/05/18	Thu 24/05/18
42	✓		Entrega primer esborrany	0 days	Fri 25/05/18	Fri 25/05/18 41
43	✓		Elaboració segon esborrany	4.5 days	Sat 26/05/18	Thu 31/05/18
44	✓		Entrega segon esborrany	0 days	Fri 01/06/18	Fri 01/06/18 43
45			Finalització/revisió ortogràfica	2 days	Sat 02/06/18	Mon 04/06/1
46			Entrega final	0 days	Tue 05/06/18	Tue 05/06/18 45
47			Entrega PAC4	0 days	Tue 05/06/18	Tue 05/06/18 46
48						
49			**Presentació**	**6 days**	**Wed 06/06/**	**Wed 13/06/**
50			Investigar formes de fer la presentació	3 days	Wed 06/06/18	Fri 08/06/18
51			Fer la presentació	2 days	Mon 11/06/1	Tue 12/06/18 50
52			Entrega PAC5a	0 days	Wed 13/06/1	Wed 13/06/1 51
53						
54			**Defensa**	**8 days**	**Thu 14/06/18**	**Mon 25/06/1**
55			Defensa del treball	8 days	Thu 14/06/18	Sun 24/06/18
56			Entrega PAC5b	0 days	Mon 25/06/1	Mon 25/06/1 55

Il·lustració 4: Diagrama de Gantt de la Memòria, Presentació i Defensa.

1.5 Breu sumari de productes obtinguts

S´han obtingut dos parts de codi, que s'expliquen tot seguit.

La primera part treballa amb les dades. Es capaç de carregar-les en memòria per manipular-les posteriorment. La entrada es pot fer a

través d´un fitxer amb extensió '.sav' o '.csv'. Per a la càrrega i manipulació es fan servir Pandas. En aquesta part es treuen duplicats, o dades no necessàries per l´estudi. També es fan correlacions entre elles per veure si, per exemple, tenim dades duplicades per error amb diferents etiquetes. Aquest primer codi també fa un resum de les característiques de la població que forma part de l´estudi.

A la segona part del codi es fan els estudis per veure quina informació i coneixements podem extreure de les dades. Aquí s'utilitzen els conceptes apressos a la assignatures de Intel·ligència Artificial. Es troben els components principals, es fan representacions PCA, MDS, es fan classificacions k-neighbours, es representen arbres de decisió... El codi funciona sobre diferents grups de dades per veure quina opció es la més rellevant.

Un tercer codi ens dirà quines versions de sistema operatiu, de Python i de les llibreries s´han utilitzat.

Juntament amb el codi s´entrega aquesta memòria i una presentació resum del treball.

1.6 Breu descripció dels altres capítols de la memòria

En el segon capítol trobem una introducció ràpida al entorn de treball. S'ha optat per utilitzar versions lliures de sistema operatiu i programari. També s'han plantejant alternatives al maquinari, en cas que no es pugui de disposar de una maquina física.

Amb aquesta memòria s'han entregat les tres parts de codi.

El primer codi s'encarrega del tractament inicial de les dades. El seu funcionament esta explicat al tercer capítol.

En aquest capítol veurem les característiques de les dades; quantes mostres tenim, quines característiques... També fa una visió ràpida de la població que forma la mostra. Per exemple, la primera informació que podem extreure fàcilment de les dades es que la majoria de pacients te sobrepès i es (o ha sigut) fumador. Finalment es fa la neteja inicial de les dades. S'utilitza la llibreria de Pandas, i la seva facilitat per treballar amb matrius grans de dades. Dintre de la neteja es treuen columnes redundants, duplicats i s'omplen els buits que conté el *Data Set* original.

Seguidament, amb la segona part del codi, s'ha fet un estudi de les dades per veure si podíem aconseguir els objectius d'aquest treball. Això es troba explicat a el quart capítol. Aquest codi trobarà les components principals i farà representacions PCA i MDS. Després dividirà les dades a l'atzar en dos subconjunts (training i test) per alimentar diferents models i veure quins són els millors per la predicció que volem fer. Entre els models

de classificació s'han implementat models k-NN, de regressió logística, SVM.

Finalment s'han fet arbres de decisió (incloent els gràfics) i s'han creat boscos d'arbres de decisió aleatoris, per millorar els resultats.

En el darrer capítol es veuen les conclusions del treball.

En el primer apèndix s'han explicat algunes de les característiques del domini tractat (sistema endocrí) amb algunes definicions breus.

Finalment el tercer codi (només unes línies) ens dona les versions del programari utilitzat (que surten al segon apèndix). Pot haver canvis en algunes llibreries amb dates posteriors a aquet treball que potser no siguin compatibles. Es per això que s'han inclòs les versions utilitzades.

Al tercer apèndix es mostra un llistat d'algunes de les variables de l'estudi, amb una breu explicació. Ni són extenses, ni són totes... són les que hem pogut aconseguir. De totes maneres han ajudat una mica a la comprensió de la informació.

Finalment, al quart apèndix s'ha aprofundit una mica més en algunes de les variables rellevants mostrant els seus marges de valors.

2 Entorn de treball

2.1 Maquinari

Per fer aquest treball s'havia considerat utilitzar una Raspberry Pi 3 B+, amb un cost d'uns 35 eur, o una màquina virtual tipus VirtualBox, Parallels... amb cap cost econòmic. Però al disposar d'un ordinador portàtil dedicat, es va optar per aqueta darrera opció. Les característiques del *Data Set* i els programes fan que no es necessiten grans recursos de memòria o capacitat de computació.

2.2 Sistema operatiu

Per fer corre python, es va escollir el sistema operatiu Ubuntu (distribució de Linux de codi obert). Després de descarregar la darrera imatge del lloc web, fer un CD e instal·lar-ho al disc dur del portàtil, es va optar també per instal·lar Anaconda.

Tal com es descriu al seu lloc web [20], Ubuntu és una paraula africana antiga que es podria traduir com "bondat humana", es podria descriure com "jo sóc el que sóc per el que som tots en el seu conjunt".

La instal·lació d'Ubuntu es molt senzilla; només hi ha que seguir els passos que apareixen per pantalla un cop inserit el CD d'instal·lació [10].

Després es van executar les següents instruccions:

```
>apt-get update
>apt-get upgrade
```

La primera actualitza la llista de paquets disponibles i les seves versions, fent servir els servidors d'origen definits en el 'sources.list'.

Un cop descarregada la llista actualitzada de programari disponible i la versió en la qual es troba, actualitzem els paquets amb la segona instrucció.

2.3 Anaconda.

Anaconda és una mena de gestor d'entorn (en Python i en C) de codi obert, amb una sèrie d'aplicacions i llibreries per la *Data Science* i *Machine Learning*, utilitzant Python. Les anacondes són serps que mengen pràcticament de tot (mamífers, peixos, amfibis i altres rèptils). Moltes implementacions de Linux utilitzen rèptils com a símbol [17][18]. Potser d'aquí ve el nom.

Per instal·lar Anaconda s´ha baixat la versió 'Anaconda3-5.0.1-Linux-x86_64.sh' del lloc web (www.anaconda.com/download/#linux) i després s´han executat les següents instruccions...

- `bash Anaconda3-5.0.1-Linux-x86_64.sh`
- `conda update conda`
- `conda update anaconda`

La segona línia actualitza l'instal·lador, la tercera la distribució. Es pot trobar més informació al seu lloc web relativa a els diferents entorns i com afegir llibreries.

2.4 Python

Python es un llenguatge, creat al 1991, d'alt nivell, per tot tipus d'aplicacions amb l'avantatge que disposa de llibreries especialitzades per carregar i manipular dades, per fer representacions gràfiques, fer càlculs numèrics, treballar amb reconeixement d'imatges... La versió escollida per aquest treball es Python 3.6, ja que la versió 2.x arribarà al '*end of life*', i deixarà de ser suportada (incloent els '*security patches*'), al 2020 [14]. El seu creador holandès, Guido van Rossum, era un gran fan de 'Monty Python's Flying Circus', segurament va agafar d'aquí la idea del nom [26].

2.5 Algunes llibreries

Scikit-learn es un projecte de codi obert que conte moltes llibreries que implementen logaritmes de *Machine Learning*.

NumPy es un dels paquets essencials per a computacions científiques. Conté funcionalitats per fer operacions amb arrays de varies dimensions, àlgebra linial, transformades de Fourier...

Pandas es una altra llibreria de Python per manipular i analitzar dades. La seva estructura principal es el *Data Frame*, una taula semblant a un full de càlcul amb l'avantatge que cada columna pot ser d'un tipus de dada diferent.

3 Tractament de les dades.

3.1 Característiques del *Data Set*

Per l'estudi mèdic, es van fer servir les dades clíniques dels pacients que van participar voluntàriament (diagnosticats entre els anys 1985 i 1994 de DM1). Aquestes van ser completades o corregides (les que presentaven valors dubtosos). Així doncs, es va partir del següent grup de dades:

- Control glucèmic: Hemoglobina glicosilada (HbA_{1C}).

  La glucosa, també anomenat sucre de raïm, es un monosacàrid que utilitzen les cèl·lules com font d'energia i que forma part important en el procés metabòlic. La glucosa es "enganxosa" i s'adhereix a alguns tipus de proteïnes, com la hemoglobina. Els metges utilitzen aquest indicador per obtenir una imatge general dels nivells mitjans de sucre a la sang durant un període de temps llarg (mesos) [28].

- Perfil lipídic: colesterol total, colesterol LDL, colesterol HDL, colesterol VLDL i triglicèrids.

  "L'etimologia de la paraula lípid prové de la forma prefixada del mot grec *lípos*, que significa 'greix' i de la forma sufixada del mot grec ***eĩdos***, que significa 'aparença, figura' " [29].
  El colesterol és una substància cerosa i semblant al greix que es troba en totes les cèl·lules del cos. El fetge produeix colesterol però també es troba en alguns aliments, com la carn i productes lactis.
  Hi ha dos tipus principals de colesterol: colesterol dolent (el LDL -*low density lipoprotein*- i el VLDL –*very low density lipoprotein*-) i colesterol bo (HDL –*high density lipoprotein*-) [30].
  VLDL i LDL són colesterols "dolents" perquè poden contribuir a l'acumulació de placa a les artèries. Aquesta acumulació s'anomena aterosclerosi. La placa que es construeix és una substància adhesiva formada per greixos, colesterol, calci i altres substàncies que es troben en la sang. Amb el temps, la placa s'endureix i redueix les artèries. Això limita el flux de sang rica en oxigen al cos. Pot conduir a malalties coronàries i altres malalties cardíaques.
  Els triglicèrids son un altre tipus de grassa present en la sang.

- IMC.

  L'Índex de Massa Corporal es una xifra que permet avaluar la corpulència d'una persona. Es el seu pes dividit per el quadrat de la seva alçada. Ens permetrà saber si una persona te sobrepès i en quin grau.

- Factor de risc cardiovascular clàssics: hipertensió, tabaquisme, dislipèmia.

  La hipertensió arterial (HTA) és una malaltia crònica consistent en una elevació persistent de la pressió arterial, per sobre de 140/90 mmHG [31].

  Les dislipèmies són alteracions en la quantitat de lípids, triacilglicerols, àcids grassos, colesterol i/o fosfolípids en la sang.

- Complicacions cròniques (retinopatia, nefropatia, polineuropatia, macroangiopatia).

  La retinopatia diabètica és una retinopatia (malaltia que afecta a la retina) conseqüència dels nivells elevats de glucèmia causats per la diabetis [32] [33].

  La nefropatia es refereix al mal, malaltia o patologia del ronyó. La diabetis és una malaltia que impedeix que el cos utilitzi glucosa (sucre) de forma adequada. Si la glucosa es queda a la sang en lloc de metabolitzar, pot provocar toxicitat. El dany que l'excés de glucosa en sang causa a les nefrones es diu nefropatia diabètica. Aquesta patologia es pot diagnosticar gràcies a la microalbuminúria, que és la pèrdua en petites quantitats de proteïnes en l'orina, en concret, l'albúmina [34][35].

  Una polineuropatia és un trastorn neurològic que es produeix quan diversos nervis perifèrics s'afecten al mateix temps. La diabetis pot convertir aquest trastorn en crònic [36].

  La macroangiopatia es una “afectació d'artèries de mitjà i gran calibre com a conseqüència de la diabetis mellitus. Pot afectar el territori coronari, donant lloc a cardiopatia isquèmica; a la circulació cerebral, produint accidents isquèmics o hemorràgics o a la circulació perifèrica (especialment d'extremitats inferiors), el que potencialment es tradueix en claudicació intermitent, úlceres o gangrena” [37].

Després aquestes dades es van integrar amb les dades obtingudes mitjançant una bateria de probes noves durant l’estudi:

- Dades antropomètriques (pes, alçada, perímetre maluc).
- Tensió arterial.
- Analítiques de sang i orina.
- Exploració vascular i neurològica.

Entre aquestes darreres destaca la tomografia computeritzada multi-detectora amb contrast endovenós (en el cas de que no tinguessin contraindicacions). Aquesta tomografia, permet detectar la extensió i severitat d’una malaltia arterial coronaria al epicardi, així com determinat l’estat del teixit adipós. La variable ‘TC2’ marcarà si hi ha lesió arterial

coronaria, en el cas que no es deixi passar mes del 50% de la llum (es a dir, detectarà si hi ha una obstrucció). Aquest es un mètode nou que permet obtenir valors reals. Permet, per exemple, detectar lesions asimptomàtiques no calcificades.

Aquesta serà, per tant, la variable que hem escollit per fer les classificacions dels pacients segons si tenen o no lesió cardíaca. Ens dirà si un pacient, amb certes característiques (determinades per les altres variables) pateix o no una lesió cardíaca. S'utilitzarà més endavant als classificadors d'aquest treball.

La investigadora va donar dos valors per la variable 'TC2':

- TC2=1 per pacients sense lesions avaluades per a la TCMD, sense estenosis a nivell coronari.
- TC2=2 per pacients que presentaven alguna lesió a nivell coronari, o que presentaven aterosclerosi clínica (infart o accident cerebrovascular, tres pacients inclosos en aquest grup).

Al apèndix 8.3 d'aquest treball tenim algunes definicions més de les variables utilitzades (així com alguns dels seus valors límits al apèndix 8.4).

En *Machine Learning*, cada element individual s'anomena mostra. En el nostre cas, cada pacient es una mostra (*sample*). Les propietats de cada mostra són les característiques (*features*) [4]. Per a nosaltres aquestes variables que representaran l'alçada, el pes,... així com els resultats dels anàlisi mèdics de cada pacient.

Les dades estaran disposades en un array on les files representaran les mostres i les columnes les característiques. Es a dir, utilitzant Pandas, definirem un *Data Frame* on a cada fila tindrem un pacient, i a les columnes els valors de les seves característiques (alçada, pes...).

Ens trobem així amb una matriu de 77 files (pacients) per 137 columnes (característiques). A la matriu no s'observen a primera visa dades errònies, però el que si que tenim són 493 valors que no han estat omplerts.

Dels 77 pacients, nomes tenim 28 amb totes les 137 característiques. Si agaféssim els 77 pacients i traguéssim les columnes on hi ha dades incomplertes, ens quedarien nomes amb 34 files (pacients).

```
Respecte el dataset
===================
(1) Dimensions de la matriu del dataset: (77, 137)
(2) Hi han 493 dades imcomplertes de les 10549 que formen el dataset orignal despres de natejar les columnes.
(3) Dels 77 pacients, nomes hi ha 28 amb totes les dades complertes.
(4) De les 137 variables, nomes 34 columnes tenen tots els valors per a tots els pacients de la mostra.
```

Il·lustració 5: Sortida amb les característiques del DataSet.

3.2 Característiques de la població estudiada.

El programa també analitza una mica les característiques de la població de l'estudi. Dels 77 pacients, aproximadament un 59% són homes. Destaca que al voltant del 67% dels pacients són (o han estat) fumadors. I un 60% té sobrepès. Aquets ja són dos dels factors de risc més coneguts per malalties coronaries.

```
Resum de la poblacio estudiada.
===============================
(1) Numero de pacients: 77
(2) Numero de variables: 137
(3) Distribucio de la mostra segons sexe: Homes-> 46.0 Dones-> 31.0
(4) Edad mitjana en que van ser diagnosticats amb diabetis: 24.8 anys.
(5) Temps de evolucio de la malaltia: entre 18.6 anys i 27.2 anys.
(6) Dosis de insulina diaria (ui/kg/dia): entre 0.28 min i 1.22 max
(7) S'han exposat al tabaq (fumadors actius o ex-fumadors): 50 persones.
(8) Mirant el Index de Massa Corporal (IMC) veiem que tenim...
    Pacients amb sobrepes: 47 (dels quals 14 amb obesitat i 2 amb obesitat morbida).
```

Il·lustració 6: Sortida amb el resum de les característiques de la població.

3.3 Neteja de les dades. Us de Pandas.

Normalment, avants de intentar extreure informació de les dades, cal fer un seguit de tasques per netejar-les. Les dades poden tenir:

- valors incorrectes o impossibles
- valors que no tindrien que estar en l'anàlisi que volem fer
- valors duplicats
- valors absents, o als que falta alguna informació.

Degut a que les dades han format part d'un estudi clínic, sembla poc probable que tinguin valors incorrectes o impossibles ja que han sigut revisades per experts en el domini, així que ens centrem més en les següents tasques.

El primer codi el que farà es carregar les dades d'un arxiu directament a un *Data Frame* de Pandas.

```
import pandas as pd
df=pd.read_csv('dades_originals.csv')
```

Tot i que el codi te una primera preparació de les dades esborrant columnes que no tenen dades mèdiques (columnes marcades amb la etiqueta 'esborrar'), i per tant, no són necessàries per l'estudi, inicialment s'han seguit tres etapes per la neteja del *Data Set*.

3.3.1 Treure columnes redundants.

Primer s'han tret les columnes amb informació redundant. Per exemple, ens hem trobat amb les columnes 'HOME', 'MUJER', 'sexo' (que pot ser home o dona)... així que hem optat per deixar només una de totes aquestes columnes (en aquest cas 'HOME').

Un altre cas són les columnes del IMC. Al ser "l'Índex de massa corporal" (IMC) una xifra que permet avaluar la corpulència d'una persona tot relacionant-ne la seva massa amb la seva talla" [25], es a dir, una relació entre altres dues característiques, l'hem tret del *Data Set*. I hem fet el mateix amb altres columnes que són valors concrets d'aquest índex ('IMCgrup', 'IMCmedioevol', 'IMCmediototal') o la seva interpretació ('sobrepeso', 'obeso').

Aquest índex, tot i que no es molt precís (depèn d'altres factors com la edat de la persona i no té en compte la seva ètnia), serveixen per tindre -de manera aproximada- una visió de quin tipus de població tenim a l'estudi (que es clarament amb sobrepès). Així que s'han refet momentàniament els càlculs quan consideràvem les característiques de la població.

Amb el següent codi s´ha fet el càlcul temporal del IMC.

```
def calcula_IMC(x):
    return round((x.peso/(x.talla*x.talla))*10000,1)
mask=(df.peso.notnull())&(df.talla.notnull())
# comprova que no hi ha ceros avants de fer les operacions
IMC=df[mask].apply(calcula_IMC, axis=1)
```

I amb les següents línies, s´ha interpretat el valor del IMC per a cada cas..

```
def interpretacioIMC (imc):
    if imc <=18: out='magror'
    elif imc>18 and imc<=25: out='corpulència normal'
    elif imc>25 and imc<=30: out='sobrepès'
    elif imc>30 and imc<=40: out='obesitat'
    elif imc>40: out='obesitat mòrbida'
    return out
```

Altres exemples de columnes redundats són les referents a si el pacient fa exercici ('activo', 'EJE', 'Ejerc'), la relació amb el tabac ('fum', 'expotabac', 'Tabaq'), o la relació 'cintura/cadera' (que són de per sí dos columnes diferents).

El codi va allargant la llista 'dades_redundants' amb les etiquetes de les columnes que tindrà que treure. Després las farà caurà del *Data Frame* on tenim la matriu de les dades amb un 'drop'.

```
df=df.drop (labels=dades_redundants, axis=1) # el axis=1 es per les columnes.
```

3.3.2 Duplicats a les dades

El codi mira ràpidament si hi han pacients duplicats. Es a dir, dos pacients amb les mateixes característiques. Per això crea una llista amb els duplicats amb la funció 'dupicated'.

```
df['pacient_duplicat']=df.duplicated(dades)
```

En el nostre cas, després de fer córrer el codi, el *Data Set* no ens mostra pacients duplicats.

La següent tasca es veure si hi ha columnes duplicades, amb els mateixos valors. Durant la integració de les dades es possible que alguna hagi sigut entrada dos cops amb diferents etiquetes, el que dificultarà la seva detecció. Per veure si tenim aquesta situació, el codi crea un fitxer amb la matriu de correlació igual a u entre les variables;

```
(df.corr()==1).to_csv('correlacio_1.csv')
```

La sortida es una matriu quadrada amb les mateixes etiquetes (noms de les variables) tant per columnes com per files. La correlació u només tindria que produir-se quan es creua una variable a les columnes, amb la mateixa variable a les files, mostrant en la matriu una diagonal de uns.

Obrint el fitxer 'correlacio_1.csv' amb el full de càlcul del *Libre Office* veiem que tant 'tevol' i 'tevolmeses' o 'glicomediatotalobj' i 'glicomediatotaligualmenor7' donen aquest valor, així que també els traurem del DataSet.

```
dades_duplicades=['tevol', 'glicomediatotalobj']
```

El codi ens va informant per pantalla de tots els passos fets.

```
Natejant les dades.
====================
(1) Treient les columnes no necesaries i les que conenten informacio redundant:
    ['fechadx', 'FExpl', 'sexo', 'hombres', 'MUJER', 'fum', 'expotabac', 'activo', 'EJE', 'IMC', 'IMCgrup',
    'IMCmedioevol', 'IMCmediototal', 'sobrepes', 'obes', 'cinturacadera', 'SCORE', 'PRE_1', 'PRE_2']
(2) Pacients duplicats: 0
(3) Treient columnes duplicades amb etiquetes lleugerment diferents per designar el mateix,
    (correlacio amb valor u entre variables diferents): ['tevol', 'glicomediatotalobj']
```

Il·lustració 7: Sortida amb els passos per netejar les dades.

3.3.3 Omplint buits

Com el mateix codi ens mostra, de unes 10.500 variables en total del *Data Set*, falten unes 493 que no tenen el seu valor corresponent.

En "faltar" simplement volem dir NA “no disponible” (*not available*) o "no present per qualsevol motiu". Pandas utilitza NaN per els valors buits en un *Data Frame* [16].

En aquest punt s’ha obtat per tres camins diferents:

- Deixar tots els pacients i totes les característiques i omplir els buits.
- Seleccionar només els pacients amb totes les dades plenes (treure files amb NaNs).
- Seleccionar tots els pacients pero treieint les caracteristiques que no tenim omplertes per tots els pacients (treure les columnes amb NaNs).

Per omplir els buits s’han distingit tres casos.

El primer es relatiu al valor de la ‘cadera’. Per omplir aquets NaNs s’ha fet una distició entre homes i dones, agafant el valor mig per a cada cas i omplint el buit amb el corresponent valor. Per això, s’ha creat una columna temporal amb el valor mig segons el sexe del pacient, i després s’ha mogut aquest valor a la columna de ‘cadera’ en el cas que tingués un NaN. Finalment s’ha esborrat la columna temporal. Això s’ha fet amb poques línies de codi.

Calculant el valor mig de maluc per els homes.

```
 # valor cadera per homes
cadera_home=df['cadera'].where(df['HOME']==1)
# suma de valors
cadera_home=cadera_home.sum()
##print ('Sumant els valors de maluc per els homes',cadera_home)
#total homes
##print ("Numero total d'homes:",df['HOME'].sum())
cadera_home_val_mig=cadera_home/df['HOME'].sum()
##print ('Valor mig de maluc per els homes',round(cadera_home_val_mig,1))
```

Calculant el valor mig de maluc per les dones.

```
# suma de valors maluc dones
cadera_dona=df['cadera'].where(df['HOME']==0).sum()
##print ('Sumant els valors de maluc per les dones',cadera_dona)
#total dones
##print ('Numero total de dones:',len(df.index)-df['HOME'].sum())
cadera_dona_val_mig=cadera_dona/(len(df.index)-df['HOME'].sum())
##print ('Valor mig de maluc per les dones',round(cadera_dona_val_mig,1))
```

Omplint els NaNs

```
print ("(1) Omplint els NaN de la columna 'cadera' amb el valor mig
de",round(cadera_home_val_mig,1),"pels homes
i",round(cadera_dona_val_mig,1),"per les dones.")
import numpy as np
```

```
df['temp_cadera_mitja']=np.where(df['HOME']==1,round(cadera_home_val_mig,
1),round(cadera_dona_val_mig,1))
df['cadera'].fillna(df['temp_cadera_mitja'],inplace=True)
df=df.drop (labels='temp_cadera_mitja', axis=1)
```

El segon cas es el d'omplir els NaNs de columnes amb valors numèrics. S'ha fet servir el valor mig a la columna en que es troba el NaN. Per això s'ha utilitzat la següent instrucció:

```
df[columnes_val_amb_NaN]=df[columnes_val_amb_NaN].fillna(df[columnes_va
l_amb_NaN].median())
```

I finalment s'han omplert els NaNs de les columnes amb valors categòrics. En aquest cas s'ha utilitzat el valor més comú en la columna tractada. S'ha utilitzat la següent instrucció per aquest propòsit:

```
df[columnes_cat_amb_NaN]=df[columnes_cat_amb_NaN].apply(lambda
x:x.fillna(x.value_counts().index[0]))
```

Si haguéssim utilitzat ja la llibreria de *scikit-learn*, i les seves estructures de dades, podríem haver utilitzat la funció *imputer* per fer el mateix però Pandas ens permet, d'una forma molt fàcil i neta, treballar amb matrius de dades de grans dimensions.

La següent figura mostra la sortida per pantalla de l'execució del codi.

```
Omplint buits
=============
(1) Omplint els NaN de la columna 'cadera' amb el valor mig de 88.6 pels homes i 70.7 per les dones.
(2) Omplint els NaN de variables numeriques amb el valor mitg de la columna on es troven. Aixo afecte les variables:
['grasa', 'Dosis', 'Ct', 'cLDL', 'cVLDL', 'ApoA1', 'ApoB', 'ApoAII', 'Nefas', 'Lpa', 'HbA1c', 'AST', 'ALT', 'GGT',
'PCR', 'FA', 'creatinina', 'FG', 'TSH', 'Homocisteina', 'urato', 'VITB12', 'albuminuria', 'proteinuria',
'indicealbcreat', 'Diuresis', 'H2', 'H3', 'PON1', 'PAFAHtotal', 'PAFAHHDL', 'PAFAHpHDL', 'incrLagphase',
'LDLmenos', 'masacolHDL', 'masaTgHDL', 'masafosfHDL', 'masacollibreHDL', 'masacolestHDL', 'HDLAI', 'HDLAII',
'HDLE', 'HDLCIII', 'EFLUXHDL', 'IL6', 'IL10', 'TGFb', 'leptina', 'Adiponectina', 'GIMcinternamaxDeI',
'GIMcinternamedioDeI', 'GIMcextmaxDeI', 'GIMcextmedioDeI', 'GIMccdi', 'GIMccmaxDeI', 'GIMfcmedioDeI',
'GIMfcmaxDeI', 'GIMpmaxDeI', 'GIMpmediaDeI', 'HbMediala5a', 'HbA1CMedia', 'glicomediaevolytrasversal',
'mediaevolLDL', 'HDLmedioevol', 'Tg', 'cHDL', 'loggrasepi', 'iEAT']
(3) Omplint els NaN de variables categoriques  amb el valor mes frequent a la columna on es troven. Aixo afecta les
variables:
['RTDp', 'TG150', 'cLDL130', 'cLDL100', 'glico7', 'glico7.5', 'TPO', 'TIROGLOB', 'CELPARIET', 'TRANSGLUT', 'PNP',
'atero1', 'atero2', 'gimp75', 'GIMP75muestra', 'GIMpP75muestra', 'glico5aobj', 'TCHelic', 'GIMfcp75', 'GIMccp75',
'ASYC', 'ASYCsign', 'HbA1CMediaobj', 'glico5a7.5', 'glicomediatotal7.5', 'glicomediatotaligualomenor7', 'SMOMS',
'TC3', 'ALC', 'ATERO1muestra', 'ATERO2muestra', 'iEATP75', 'LC', 'TC2']
```

Il·lustració 8: Omplint els buits al DataSet.

Hem omplert 5 pacients que no es van sotmetre a la TCMD a un valor de dos, indicant lesió. Amb aquest supòsit hem agafat el pitjor cas possible. Caldria veure amb un expert del domini si per els pacients concrets ha sigut una bona opció. De totes maneres, també treballarem sobre el *Data Frame* sense valors buits.

Tenint en compte les indicacions de la tesis doctoral de B.Marlin, publicada al 2008: "*The analysis of missing data processes leads to a theory of missing data in terms of its impact on learning, inference, and prediction. This theory draws a distinction between two fundamental categories of missing data: data that is missing at random and data that is*

not missing at random. When data is missing at random, the missing data process can be ignored and inference can be based on the observed data only" [13].

Al estar els NaNs distribuïts de forma aleatòria podríem prescindir d'ells i treballar només amb les dades que coneixem. De totes maneres, el programa ha treballat en els tres casos anteriors i a creat un arxiu '.csv' per a cada situació. L'altre codi entregat podrà escollir fàcilment amb quines dades treballar.

El codi distingueix entre els tres casos anteriors:

Cas	Definició	DataFrame	Arxiu sortida
1	Matriu original omplint els buits.	df	'df_NaN_filled.csv'
2	Files sense NaNs	df_pacients_amb_totes_les_variables	'df_pacients_amb_totes_variables.csv'
3	Columnes sense NaNs.	df_columnes_sense_NaN	'df_columnes_sense_NaN.csv'

Taula 2: Tres arxius, amb les matrius de les dades, generats per el codi.

Ja que buscarem quines variables ens donen millors resultats, en la següent part del codi ens hem centrat en els dos primers casos. De totes maneres s'ha deixat la sortida del tercer cas. Si un expert en el domini considerés que les columnes tretes no son importants, es podria alimentar fàcilment el segon codi també amb el tercer arxiu.

4 Aplicant tècniques de *Machine Learning*.

En aquest codi es defineix una funció ´programa´ on s' aplicaran els diferents models en el conjunt de dades seleccionat.

```
programa(filename,compo,vei,reco=0)
```

El paràmetre 'filename' es refereix al arxiu en format '.csv' des de on es carregaran les dades (recordem que el codi anterior ens donava tres fitxers). El paràmetre 'compo' conté el número de components principals que volem fer servir. El tercer paràmetre 'vei' conté el número de veïns per el model *k-nearest nieghbors*. Finalment 'reco', per si es vol fer servir nomes el subconjunt de dades recomanades sent: '0' per utilitzar totes les dades, '1' les recomanades per la doctora, '2' les 15 més rellevants del *Data Set* original i '3', les que hem escollit després de córrer el programa varies vagades.

El codi crida la funció 'programa' varies vagades amb diferents paràmetres:

```
print('(1) Amb el cas 1, dades netejades i omplertes')
print('A-utilitzant totes les dades')
programa('df_NaN_filled.csv',44,2,0)
print('B-utilitzant les dades recomanades experta domini')
programa('df_NaN_filled.csv',10,14,1)
print('C-utilitzant les 15 característiques més importants')
programa ('df_NaN_filled.csv',10,8,2)
print('D-utilitzant les dades recomanades per nosaltres')
programa ('df_NaN_filled.csv',17,6,3)

print('(2) Amb el cas 2, nomes els pacients amb totes les dades')
print('A-utilitzant totes les dades')
programa('df_pacients_amb_totes_variables.csv',20,14,0)
print('B-utilitzant les dades recomanades experta domini')
programa('df_pacients_amb_totes_variables.csv',5,2,1)
```

Els paràmetres s'han anat ajustant fent córrer el codi varies vagades i observant els valors més adients. Així, per exemple, coneixem per cada cas el número de components principals necessaris per tenir un 95% de la variància, el número de veïns recomanats per fer la classificació...

Amb les diferents proves s'han vist unes variables que semblen vinculades a la utilitzada per la classe que volem utilitzar per el nostre objectiu. Degut a això, obteníem unes distribucions de punts PCA molt diferenciades i, en algun cas, un arbre amb una precisió del 100% en test. Aquestes variables són: 'TCHelic' (molt semblant a la 'TC2'), 'TC3' (Tc amb lesions <50% vs lesions >50%), 'LC'(lesions > 50% en TC y/o iEAT>P75), 'ALC' (TC amb algun tipus de lesió i/o iEAT>P75), 'atero1',

'atero2' (pacients amb lesions significatives en el TC o GIMcc o GIMfc > percentil75), 'ATERO1muestra', 'ATERO2muestra' (microangiopatia, >P75 i amb lesions en TC >50 per el primer cas), 'ASYC' i 'ASYCsign' (Aterosclerosis i macroangiopatia)

Així que s'ha optat per treure-les del *Data Set* amb les següents línies de codi.

```
objectius=['TCHelic','TC3','LC','ALC','atero1','atero2','ATERO1muestra',
'ATERO2muestra','ASYC','ASYCsign']
df=df.drop(labels=objectius,axis=1)
```

Tot i que aquestes línies de codi es podrien haver 'tallat i enganxat' a la primera part del codi, s'ha decidit deixar-les en aquesta segona per mantenir de forma diferenciada el que es una neteja genèrica de les dades, d'aquesta que es especifica per el nostre objectiu. Ens falten coneixements del domini per poder ratificar si aquesta opció es encertada i quedaria pendent la revisió amb un expert en el domini.

4.1 Característiques més importants

Per reduït el nombre de característiques es pot seleccionar les millors utilitzant la llibreria *SelectKBest* de *Scikit-learn* [6]. Aqueta funció el que fa es mirar la correlació de cada variable amb la classe utilitzada per classificar les mostres. Una manera de millorar aquesta selecció seria omplint la funció només amb les mostres utilitzades per l'entrenament [11].

En aquest cas hem utilitzat la funció 'chi-quadrat' que calcula X^2 entre la característica estudiada i el *target*. Després ens selecciona les primeres 'n_carac' característiques que han obtingut millors valors.

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Sent O_i, les observacions per la classe i, sent E_i les observacions esperades de la classe, si no hi ha relacions entre la característica i el *target* [6].

Podríem pensar que si una característica es independent del *target*, no donarà informació rellevant per la classificació del pacient.

El nostre codi quedarà com segueix.

```
from sklearn.feature_selection import SelectKBest, chi2
# omplint selecció
n_carac=15 # seleccionem les 15
selec = SelectKBest (score_func=chi2, k=n_carac)
selec.fit(dataSet,target)
# utilitzant els índex
```

```
        idxs_selec=selec.get_support(indices=True)
        carac_mes_imp=[]
        for i in idxs_selec:
            carac_mes_imp.extend([etiquetes[i]])
        print ('\nLes',n_carac,'característiques que semblen més importants
són:\n\n',carac_mes_imp)
```

I la sortida quan utilitzem totes les variables (després de omplir els buits) es la següent.

```
Les 15 caracteristiques que semblen mes importants son:

['grasa', 'edad', 'tevolmeses', 'ApoB', 'AST', 'ALT', 'GGT', 'FA', 'urato', 'PON1', 'PAFAHpHDL', 'IL10', 'edaddx', 'iEAT', 'HOME']
```

Il·lustració 9: Variables més rellevants, del *Data Set* original omplint els buits.

Això s'utilitza normalment quan hi ha moltes característiques i es volen provar els models en un grup més reduït, agafant només les més rellevants. Nosaltres el que hem fet es executar el codi en diferents situacions. Primer en el *Data Set* original. Després en subconjunt de només les característiques recomanades per la doctora. També en aquest subconjunt amb les 15 variables més rellevants. I finalment en un subconjunt de característiques que hem recomanat nosaltres. Aquesta darrera recomanació s´ha fet utilitzant les recomanades per la doctora, afegint les que surten en la il·lustració 9 i alguna que hem vist sortir repetidament després de executar el codi varies vagades.

4.1.1 Subconjunts de dades.

Les dades recomanades per la doctora (quan cridem el 'programa' amb el paràmetre 'reco=1') són les següents:

```
recomenat=['cadera','grasa','edad','tevolanys','peso','talla','RTDp','NTDp','PNPp',
'Cisqp','EVCp','AAOOp','DLPp','TtoHTA','TC2']
```

La següent recomanació (quan cridem el 'programa' amb el paràmetre 'reco=2') són les del apartant anterior, les quinze variables més rellevants.

```
recomanat=['grasa', 'edad', 'tevolmeses', 'ApoB', 'AST', 'ALT', 'GGT', 'FA',
'urato', 'PON1', 'PAFAHpHDL', 'IL10', 'edaddx', 'iEAT', 'HOME']
# classe al final
recomanat.extend(['TC2'])
```

Les dades que hem seleccionat nosaltres (quan cridem el 'programa' amb el paràmetre 'reco=3') són les recomanades per la doctora, traient 'tevolanys' i afegint 'tevolmeses'. Després hem afegit algunes de les dades de la il·lustració 9. I finalment algunes dades que hem vist aparèixer en les diferents vegades que hem anat executant el codi (segons la importància per crear arbres de decisió, per exemple).

```
recomanat=['cadera','grasa','edad','peso','talla','RTDp','NTDp','PNPp','Cisqp','EV
Cp','AAOOp','DLPp','TtoHTA']
```

```
# les que surten més a partir de la nostra experiència (algunes apareixen al
llistat de les 15)
recomanat.extend(['HbA1c','Ejerc','Tabaq','gimp75','ApoAII','iEAT','Adiponectina'
,'VITB12','HDLmediototal','TtoDLP','tevolmeses'])
# classe al final
recomanat.extend(['TC2'])
```

En tots els casos afegim la variable 'TC2' al final, ja que la resta de codi contarà amb que aquesta serà la que traurà del conjunt i utilitzarà per a la classificar les variables.

4.2 Components principals, representacions PCA, MDS

Per poder reduir l'espai ens fixaren en els components principals que expliquin gran part de la variància de les dades, per tant escollir els valors propis més grans. Per exemple, en el tercer cas del codi (amb totes les dades netejades i omplertes, però nomes escollint unes columnes específiques), per representar un 95% de la variància necessitarem 17 de les components.

Després de llegir les dades, el codi ficarà la classe al final, i traurà la característica 'TC2' del *Data Set* ('classe'=1 quan no hi ha lesió, 0 quan hi ha lesió).

```
df['classe']=np.where(df['TC2']==1,1,0)
# treiem TC2 del mig del df
df=df.drop(labels='TC2',axis=1)
```

Definirem el 'dataSet', el 'target' (els valors de la classe per pacient), i el grup de pacients.

```
dataSet = df.values
target = df['classe'].values # la classificació
patients = preprocessing.scale(dataSet.astype(float)[:, :len(df.columns)-
2], axis=0) # preprocessat dels pacients, no contenen 'classe' -2
```

Tindrem també un llistat amb les etiquetes de les característiques.

```
features_names=list(df.columns.values)
```

La següent part ens donarà la variància acumulada per a cada component principal:

```
from sklearn.decomposition import PCA
mypca = PCA()
mypca.fit(patients)

print ("\nComponents Principals: \n")
acumvar = [sum(mypca.explained_variance_ratio_[:i+1]) for i in
range(len(mypca.explained_variance_ratio_))]
print(list(zip(range(len(acumvar)), acumvar)))
```

El codi ens mostrarà, per les variables que recomanem, la següent sortida per pantalla.

```
[(0, 0.17973971613818043), (1, 0.30870471930401155), (2, 0.39656647081407936), (3, 0.46417635389040574), (4,
0.52873299426321818), (5, 0.5867909868268657), (6, 0.63885916133916021), (7, 0.68655694877623841), (8,
0.7337014499032708), (9, 0.77326847268318077), (10, 0.80818467646301984), (11, 0.83965720940951327), (12,
0.86536103517666485), (13, 0.89016461045999828), (14, 0.91161670848295617), (15, 0.93034049707997579), (16,
0.94730787474285649), (17, 0.96230046034477634), (18, 0.97422366746180078), (19, 0.98558573155341556), (20,
0.99361939614782691), (21, 0.99932680898383153), (22, 0.99999999999999989)]
```

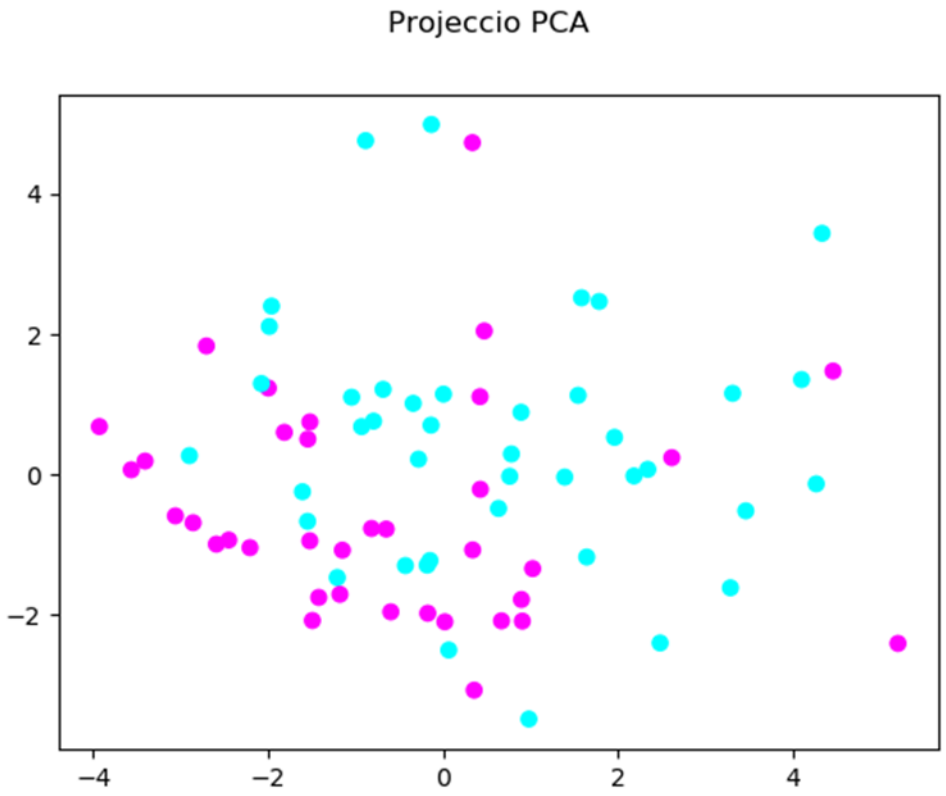
Il·lustració 10: Variància (%) explicada per cada component principal (acumulada).

Amb la representació PCA el que fem es una transformació lineal del espai de variables original, i al mateix temps reduir la dimensionalitat, creant així un nou espai PCA (passant totes les variables inicials a les components principals escollides amb 'compo'). El nou espai PCA continuarà representant la variabilitat de les dades inicial [3].

La següent part de codi ens ajuda a trobar una representació gràfica PCA.

```
from sklearn.decomposition import PCA
mypca = PCA(n_components=compo)  # amb 'compo' tenim 95%
mypca.fit(patients)
values_proj = mypca.transform(patients)

classe=dataSet[:,len(df.columns)-1]

import matplotlib.pyplot as plt
import pylab

# Dibuixar amb els components 0 and 1

fig = plt.figure(1)
fig.suptitle('Projecció PCA')
plt.scatter(values_proj[:,0],values_proj[:,1],marker='o',c=classe,
cmap=pylab.cm.cool)
plt.show()
```

Així obtenim la gràfica (per el cas 1-D) on es mostren les dades en dues dimensions (un punt per cada mostra) amb diferent color segons la classe, aplicant PCA.



Il·lustració 11: Representació PCA amb dues components principals.

A primer cop d'ull veuríem que els punts de color rosa es troben majoritàriament al quadrant inferior esquerra, el que en un principi ens podria corroborar que les mostres es podran classificar correctament.

El nostre codi ens diu com estan distribuïts els casos. Amb el *Data Set* complert, els setanta-set pacients, la següent línia:

```
df.groupby("TC2").size()
```

ens donarà com a sortida:

```
Com estan dividits

 TC2
1.0    36
2.0    41
```

En el gràfic veiem que els trenta-sis que no tenen lesió (TC=1) estan representats de color rosa, mentre que els quaranta-un que tenen lesió (TC2=2) estan representats en color blau.

"La PCA és una tècnica que permet determinar els eixos principals de la distribució del núvol de punts. Trobar els eixos principals del núvol de punts equival a trobar un nou conjunt de variables en les quals les

dades presenten una dispersió màxima, és a dir, aquells eixos en els quals la variabilitat de les dades és més gran.” [2].

La funció *scale* centra les dades en 0 (restant la mitjana) i després les divideix per la desviació estàndard. Així doncs el resultat representat pot ser negatiu (encara que les dades fossin totes positives). Els valors per sota de la mitjana seran negatius en escalar-los, i els que quedaven per sobre seran positius.

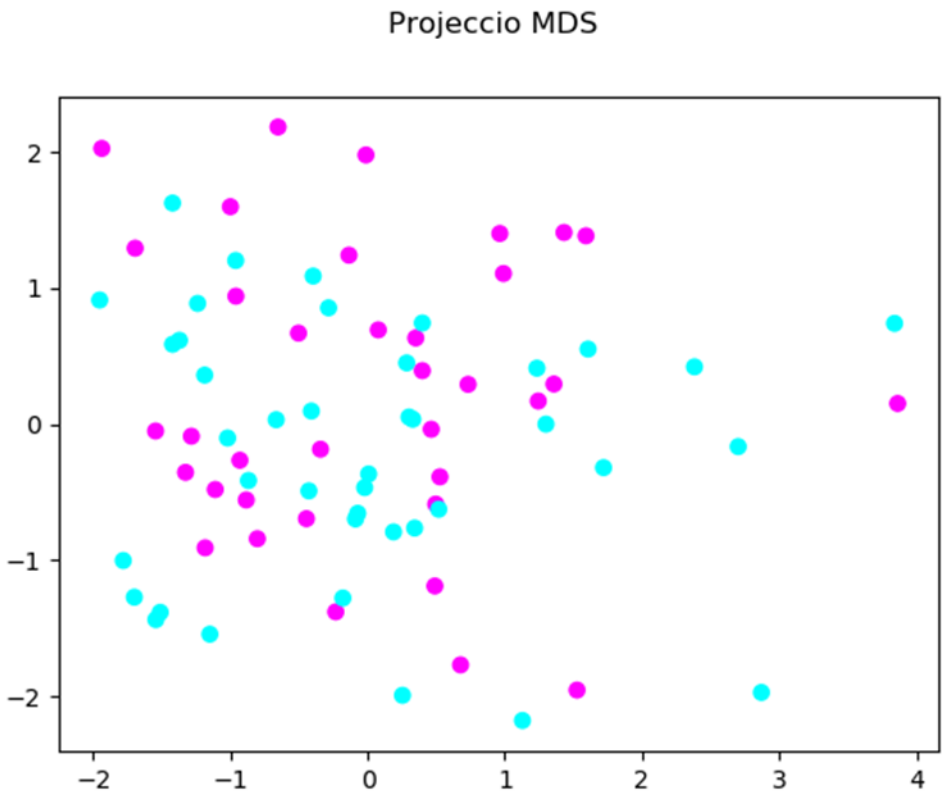
Es podria dir que amb més dades, les gràfiques continuen semblants, però amb major densitat de punts, menys espais blancs entre els punts. El que ajuda lleugerament a fer una distinció millor de les classes.

Una altre projecció de la informació es el MDS. “L´escalament Multidimensional -MDS- és una eina que permet projectar les dades en un espai de dimensionalitat reduïda, de manera que la distribució de les dades en l´espai projectat mantingui una similitud amb la distribució de les dades en l’espai original (...) ens permetrà una visualització millor de les dades sense desvirtuar-ne l’estructura original” [2].

Amb el següent codi trobem les representacions MDS.

```
from sklearn.manifold import MDS
myMDS = MDS(n_components = compo)
projection = myMDS.fit_transform(patients)
#fit_transform(X[, y, init])        Fit the data from X, and returns the
embedded coordinates

fig = plt.figure(7)
fig.suptitle('Projeccio MDS')
plt.scatter(projection[:,0],projection[:,1],marker='o',c=classe,
cmap=pylab.cm.cool)
plt.show()
```

La següent gràfica mostra el resultat d’utilitzar MDS (cas 1-D).



Il·lustració 12: Representació MDS de les dades.

Així doncs, MDS (encara que la representació es diferent a la obtinguda amb PCA) dona una visualització de les dades en dues dimensions. Aquesta representació MDS va canviant a mesura que s´executa el codi. Aquí costa més trobar una distinció clara entre classes.

Si tenim, que PCA minimitza les dimensions preservant la covariància entre les dades, mentre MDS la minimitza mantenint la distribució en l'espai, sembla que quan més grans són els valors de covariància, els gràfics resultants seran més semblants. PCA sembla més indicat quan pot haver-hi una relació lineal entre les dades, però quan no es el cas, potser podem fer servir un escalament MDS.

4.3 Aprenentatge supervisat.

En el model d'aprenentatge supervisat voldrem saber la sortida per una entrada disposant de diferents exemples d'entrada dels que coneixem la sortida. El nostre model utilitzarà aquets exemples per aprendre i predir els casos nous. Tindrem pacients amb diferents característiques classificats en si tenen lesió cardíaca o no, dels que el model aprendrà a distingir-los a través de les seves característiques per poder predir així si un nou pacient per el sistema tindrà risc de lesió cardíaca.

En el model d'aprenentatge supervisat distingim dos tipus; els classificadors i els de regressions. I dintre dels classificadors hi ha els anomenats classificadors amb dues classes (o binaris) i els multi-classe.

Diríem que els binaris classifiquen una mostra en relació a la resposta afirmativa o negativa a una pregunta. En el nostre cas busquem classificadors binaris; Un pacient de DM1 nou, amb certes característiques (pes, alçada, sobrepès...), te una lesió cardíaca? La resposta serà binaria (si/no). Un exemple de regressió podria ser si ens hagéssim plantejat intentar predir a quina edat pot aparèixer una lesió, al ser la edat un nombre amb una relació lineal.

4.4 Creació de dos grups de dades; de entrenament i test.

En els següents apartats provarem models d'aprenentatge supervisat. Però avants utilitzem la funció *train_test_split* de *Scikit-learn* per dividir el *Data Set* en dos grups. El primer contindrà els pacients que s'utilitzaran per entrenar el model. Amb ells aprendrà quins pacients, amb determinats valors a les seves característiques, presenten lesions, i quins no. Després utilitzarà el segon grup, el de test, per intentar predir amb les seves característiques si un pacient te lesió o no, podent comprovar després el grau d'encert (al conèixer també des de un principi si tenen lesió). Aquesta divisió es fa de manera aleatòria i amb una relació de tres pacients al grup d'entrenament per un pacient al de test. Això pot fer també que tinguem lleus diferencies en les sortides/gràfics del codi cada cop que s'executa.

La variable per dividir en classes serà la 'TC2' que hem utilitzat per crear la 'Classe' al final de les característiques. La Tomografia Computeritzada es la proba mèdica que ens determinarà si un pacient té lesió cardíaca.

En la funció tindrem X_train, X_test, y_train, y_test. La nomenclatura utilitzada es 'X' per entrada e 'y' per sortida (simulant com seria una funció matemàtica f(x)=y). Les X estan en majúscules al ser bidimensionals (mostres * característiques) –matrius-, mentre que la sortida es uni-dimensional –vectors-[4].

Les línies següents faran aquesta divisió.

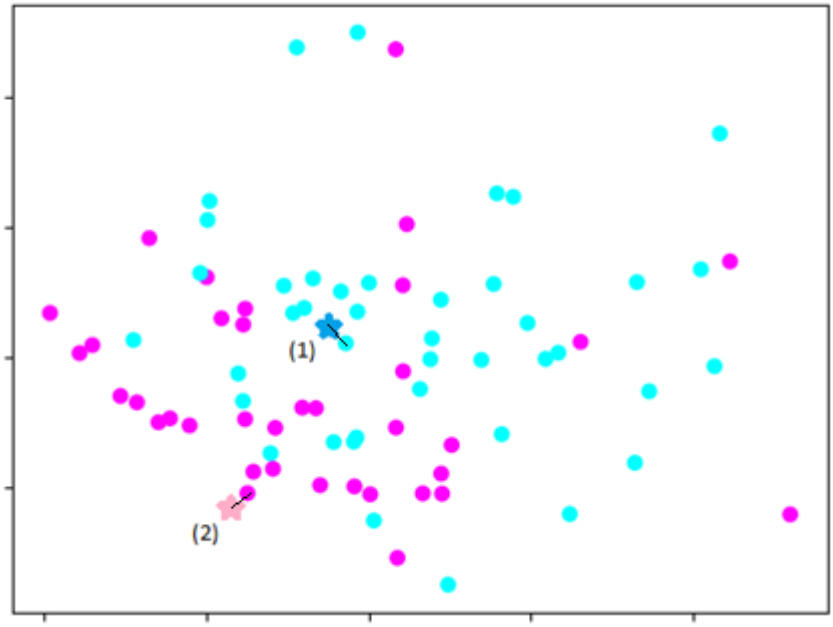
```
from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(dataSet_only, target,
random_state=0)
```

4.5 Classificació kNN.

El primer model utilitzat es el *k-nearest nieghbors* (kNN). Va ser desenvolupat per la predicció estadística i reconeixement de patrons al principi dels 70 [5]. Un cop representats tots els pacients del grup d'entrenament com a punts en un espai, quan afegim el punt que representa un nou pacient, mirarem quin dels punts d'entrenament te més a prop. Llavors assignarem a aquest nou pacient la mateixa classe que te el punt d'entrenament més proper. El mètode es el de *k-nearest nieghbors*

perquè, en lloc d'un únic punt, podem agafar la classe predominant entre els k-veïns, un número específic de veïns.

Per exemple, si en el següent gràfic representem els pacients com a punts de colors, segons si tenen lesió (representats en color blau) o no tenen lesió (representats en rosa), el model pot predir que un nou pacient (estrella amb un 1) tindrà lesió ja que veiem que el veí (o n-veïns) més proper es de color blau. I el mateix passaria amb un altre pacient nou (estrella amb un 2). Degut a les seves característiques diferents estarà representat a un altre lloc, aquest cop a prop de un pacient de color rosa, el model predirà que aquest segon pacient no tindrà lesió.



Il·lustració 13: Buscant el veí més proper per a punts nous.

Si agaféssim un k amb el valor màxim igual al número de mostres de entrenament, segurament la predicció seria sempre la mateixa, la classe predominant al grup d'entrenament [4].

Quan no tenim clases sinó succesions de valors, per exemple si volem predir un valor numèric, també podem fer servir una variant de *k-neighbors* per a regressions.

En *Scikit-learn*, aquest algoritme per classes esta implementant en *KNeighborsClassifier*. Amb el mètode *predict* podrem fer prediccions per a nous pacients.

Per veure si el nostre model es bo, utilitzem el mètode de predicció amb la classe de test. Després podrem comparar els resultats amb la realitat (ja que coneixem des de un principi els valors de la classe –si tenen lesió coronaria o no- per els pacients que estan en el grup de test).

Si un model pot fer prediccions acurades amb dades noves, diem que es pot generalitzar el model del conjunt d'entrenament al de test. Si el model es molt complexa correm el risc de fixar-nos massa en les característiques de les mostres d'entrenament, i potser no pot generalitzar bé a dades noves. Es a dir, podríem tenir bons resultats per entrenament per no per a test o per qualsevol dada nova. En aquesta situació parlem de *overfitting*.

També podríem tenir el cas invers, un model tan simple que no te en compte les variabilitat de les dades (ometent algunes de les seves peculiaritats). Per exemple, si afirméssim que tota persona amb DM1 tindrà una lesió cardíaca . Ens trobem en aquets casos amb *underfitting*.

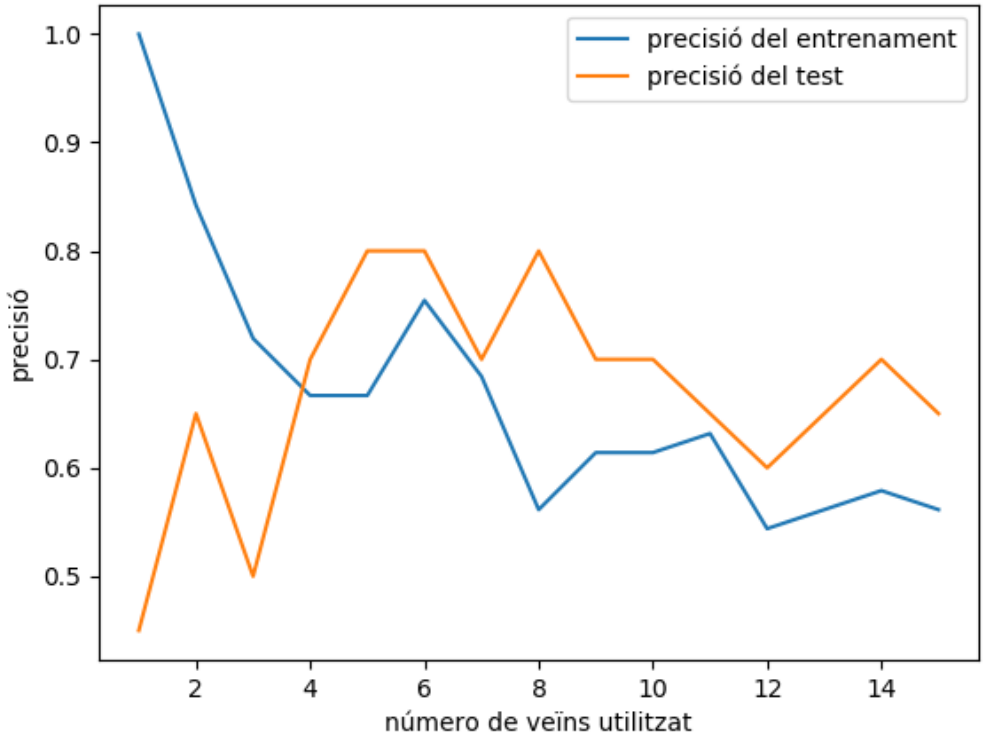
Com moltes coses en l'àmbit de l'enginyeria, la millor opció serà un punt mig.

Quan més mostres tenim, més complexa podrà ser el model sense incorre en *overfitting*. De totes maneres, no ajudarà duplicar o tenir mostres molt similars. Escollir quantes dades agafarem també pot ajudar a ajustar un model complexa [4].

La següent part del codi ens ajuda a veure quin número de veins es el més correcte per optimitzar el grau de precissió.

```
training_accuracy = []
test_accuracy = []
neighbours_settings = range (1, 16)
for n_neighbo in neighbours_settings:
        knn=KNeighborsClassifier(n_neighbors=n_neighbo)
        knn.fit(X_train,y_train)
        training_accuracy.append(knn.score(X_train, y_train))
        test_accuracy.append(knn.score(X_test, y_test))

plt.plot(neighbours_settings, training_accuracy, label = "precisió del
entrenament")
plt.plot(neighbours_settings, test_accuracy, label = "precisió del test")
plt.ylabel("precisió")
plt.xlabel("numero de veïns utilitzat")
plt.legend()
plt.show()
```

La sortida (per el cas 1-D) serà un gràfic com el següent:



Il·lustració 14: Selecció del número de veïns més adient.

Podem observar com amb un únic veí, la predicció d´entrenament es perfecta, però cau a mesura que afegim veïns indicant que el model es massa complexa [4]. Però quan ens anem apropant a 14 veïns, el model comença a ser molt simple. Sembla així que un punt entremig, en aquets cas, amb 6 veïns obtenim els millors resultats de entrenament i test. De fet es el que dona els valors de predicció més encertats per a entrenament i test.

Una recomanació, quan tenim un sistema de classificació binari, seria agafar un valor de K senar, per intentar evitar que es produeixi el cas de tindre un empat entre classes (per exemple, si k=4 podríem tenir dos veïns d'una classe i els altres dos veïns de l'altre)

La següent part del codi (amb el número de veïns òptim)

```
veins=vei
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=veins)

knn.fit(X_train,y_train)

y_pred=knn.predict(X_test)

print ("\nClasificador KNeighbors")
print ("======================")
```

```
print ("Precisió de Test vs real: {:.2f} %".format(knn.score(X_test,
y_test)*100), " utilitzant {} veïns".format(veins))
```

Ens donarà una sortida com la següent.

```
Clasificador KNeighbors
=======================
Precisió de Test vs real: 80.00 %  utilitzant 6 veïns
```

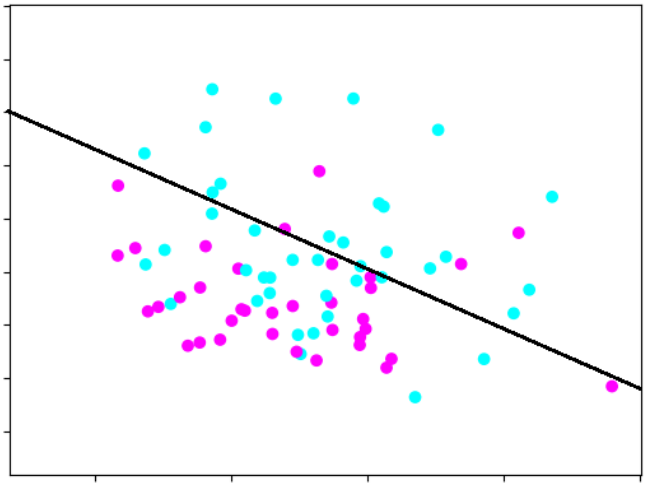
Il·lustració 15: Precisió del classificador.

Es a dir, si ens vingués un pacient nou, amb les característiques escollides i fent servir aquest model, tindríem un 80% d'encert en determinar si te una lesió cardíaca, utilitzant k-NN amb 6 veïns.

4.6 Altres mètodes

4.6.1 Regressió logística

Un model lineal de classificació farà la separació entre els punts del *Data Set* mitjançant línies.



Il·lustració 16: model lineal de classificador.

La regressió logística s'utilitza per predir el resultat d'una classe en funció de les variables independents.

Amb aquest codi…

```
tradeoff = [100, 50, 25, 1, 0.1, 0.001, 0.0001, 0.00001, 0.000001]
for c in tradeoff:
        regres = LogisticRegression(C=c).fit(X_train, y_train)
        print ("valor de C:", c, "Precisio entrenament:
{:.2f}".format(regres.score(X_train, y_train)*100), "Precisio test:
{:.2f}".format(regres.score(X_test, y_test)*100))

print ("*L1 regulation")
#tradeoff = [100, 50, 25, 1, 0.1, 0.001, 0.0001, 0.00001, 0.000001]
```

```
        for c in tradeoff:
                regres = LogisticRegression(C=c, penalty="l1").fit(X_train,
y_train)
                print ("valor de C:", c, "Precisio entrenament:
{:.2f}".format(regres.score(X_train, y_train)*100), "Precisio test:
{:.2f}".format(regres.score(X_test, y_test)*100))
```

Obrindriem la seguent sortida:

```
Regressió Logística
===================
*Regulació L2
valor de C: 100 Precisió entrenament: 82.46 Precisió test: 45.00
valor de C: 50 Precisió entrenament: 82.46 Precisió test: 45.00
valor de C: 25 Precisió entrenament: 84.21 Precisió test: 45.00
valor de C: 1 Precisió entrenament: 85.96 Precisió test: 50.00
valor de C: 0.1 Precisió entrenament: 80.70 Precisió test: 55.00
valor de C: 0.001 Precisió entrenament: 70.18 Precisió test: 65.00
valor de C: 0.0001 Precisió entrenament: 63.16 Precisió test: 55.00
valor de C: 1e-05 Precisió entrenament: 64.91 Precisió test: 50.00
valor de C: 1e-06 Precisió entrenament: 57.89 Precisió test: 55.00
Regulació L1
valor de C: 100 Precisió entrenament: 82.46 Precisió test: 50.00
valor de C: 50 Precisió entrenament: 82.46 Precisió test: 50.00
valor de C: 25 Precisió entrenament: 82.46 Precisió test: 50.00
valor de C: 1 Precisió entrenament: 82.46 Precisió test: 50.00
valor de C: 0.1 Precisió entrenament: 71.93 Precisió test: 70.00
valor de C: 0.001 Precisió entrenament: 50.88 Precisió test: 60.00
valor de C: 0.0001 Precisió entrenament: 50.88 Precisió test: 60.00
valor de C: 1e-05 Precisió entrenament: 50.88 Precisió test: 60.00
valor de C: 1e-06 Precisió entrenament: 50.88 Precisió test: 60.00
```

Il·lustració 17: Sortida valors de precisió per regressió logística.

De la sortida anterior veiem que el millor model ens donaria casi un 72% de precisió amb les dades d'entrenament, i un 70% amb les de test, després de fer la regulació (penalty='l1') i de ajustar el valor de **c** a 0.1

4.6.2 Suport Vector Machines (SVM)

SVM es un altre model de aprenentatge supervisat.

Quantes més característiques tenim, més dimensions tindrem i més complicat serà trobar aquestes fronteres de separació entre classes. El model SVM aprendrà, durant la fase d'entrenament, com de importants són cadascun de els punts per definir aquestes fronteres, creant un *subset* amb els que més ajuden. Aquets seran els punts de suport o *suport vectors* [4].

Al tindre una nova mostra que es representarà amb un punt, el mecanisme de predicció mirarà la importància i distancia dels vectors de suport respecte a aquest nou punt i així determinarà la seva classe.

Tenim per tant, en l'algoritme d'implementació de SVM, dos paràmetres que podem variar per veure com afecten en la predicció dels nostres models. Podrem definir el que es considera com 'proper a un vector' (variant l'ample de la funció gaussiana) i la importància que te un punt (variant el paràmetre c).

Amb el següent codi...

```
tradeoff = [100, 50, 25, 1, 0.1, 0.001, 0.0001, 0.00001, 0.000001]
for c in tradeoff:
        regres = LinearSVC(C=c).fit(X_train, y_train)
        print ("valor de C:", c, "Precisio entrenament:
{:.2f}".format(regres.score(X_train, y_train)*100), "precisio test:
{:.2f}".format(regres.score(X_test, y_test)*100))
```

Obtenim el següent resultat (cas 1-D):

```
SVC
===
valor de C: 100 Precisió entrenament: 56.14 precisió test: 60.00
valor de C: 50 Precisió entrenament: 50.88 precisió test: 60.00
valor de C: 25 Precisió entrenament: 50.88 precisió test: 50.00
valor de C: 1 Precisió entrenament: 54.39 precisió test: 70.00
valor de C: 0.1 Precisió entrenament: 57.89 precisió test: 60.00
valor de C: 0.001 Precisió entrenament: 71.93 precisió test: 80.00
valor de C: 0.0001 Precisió entrenament: 70.18 precisió test: 65.00
valor de C: 1e-05 Precisió entrenament: 63.16 precisió test: 55.00
valor de C: 1e-06 Precisió entrenament: 63.16 precisió test: 50.00
```

Il·lustració 18: Sortida valors de precisió per SVC.

Per a C=0.001 trobarem un compromís amb una precisió del 80% amb les dades d'entrenament i un 75% en test. A

4.7 Arbres de decisió.

4.7.1 Un arbre

Els arbres de decisió també són models de classificació. Es basen en si es compleix o no una condició. Es construeixen amb les dades i amb l'aprenentatge supervisat. Es tracta aquí de començar fent preguntes –anomenades 'tests'- fins arribar al objectiu de la forma més ràpida i eficient. Per exemple; utilitzant l'arbre obtingut amb totes les dades del *Data Set*, si fem el test " El pacient te més de 53.5 anys?" i després el segon test "cLDL es més gran de 3.18?"... si les respostes han sigut afirmatives, segurament el pacient te lesió cardíaca.

"En la representació en forma d'arbre tenim que l'arrel correspon a l'estat inicial, els nodes de l'arbre són estats, els arcs corresponen a les accions i les fulles de l'arbre corresponen als estats terminals dels camins. D'aquesta manera quan resseguim les branques de l'arbre des del node inicial a una fulla estem repassant un camí des de l'estat inicial a un de terminal" [47].

4.7.2 Rellevància de les variables per crear l'arbre.

Per les dades escollides, la rellevància de les característiques ens diu com han sigut utilitzades per l'arbre de decisions. Els valors estan entre 0 i 1 sent aquest darrer el més important per dir que una variable defineix perfectament l'objectiu. Tenir característiques amb valor 0 no vol dir que no aportin informació. Potser les seves característiques estan englobades en altres característiques. Només diu que l'arbre no la agafat. Hi ha que tindre en compte que no hi ha una relació simple entre les característiques i la classe final [4].

4.7.3 Implementació de l'arbre.

Amb la primera versió del codi...

```
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier(max_depth=None, random_state=0)
tree.fit(X_train, y_train)

print ("\nArbre de decisio")
print ("================")

print ("Precisio entrenament: {:.2f}".format(tree.score(X_train,
y_train)*100), "Precisio test: {:.2f}\n".format(tree.score(X_test, y_test)*100))
```

Es va veure que per tots els casos, teníem *overfitting*. Es a dir uns models molt complexes adaptats per encabir totes les dades del grup de entrenament.

```
Arbre de decisió
================
Precisió entrenament: 100.00 precisió test: 45.00
```

Il·lustració 19: Sortida arbre de decisió (amb *overfitting*).

A la sortida es veu que la precisió de l'arbre era del 100% per el grup de dades d'entrenament.

Hi ha diferents mecanismes per intentar solucionar això. El que es diu fer una 'poda' del arbre de decisió. Un es parar el creixement de l'arbre (*pre-pruning*) avants d'hora, per prevenir així aquest efecte. Dintre d'aquest mecanisme trobem les opcions de limitar la màxima profunditat de l'arbre, en número màxim de fulles, el número mínim de punts que ha de tenir un node... [40][41]. Una altra es deixar que el arbre creixi fins el final, i després treure els nodes que contenen poca informació (*post-*

pruning o *pruning*). Per aquesta poda recorrerien l'arbre de manera ascendent (de les fulles a l'arrel).

La implementació amb Scikit-Learn només permet el primer tipus de poda. Després de diverses proves s'ha optat per limitar la profunditat a quatre nivells i el numero mínim de mostres per fulla a cinc.

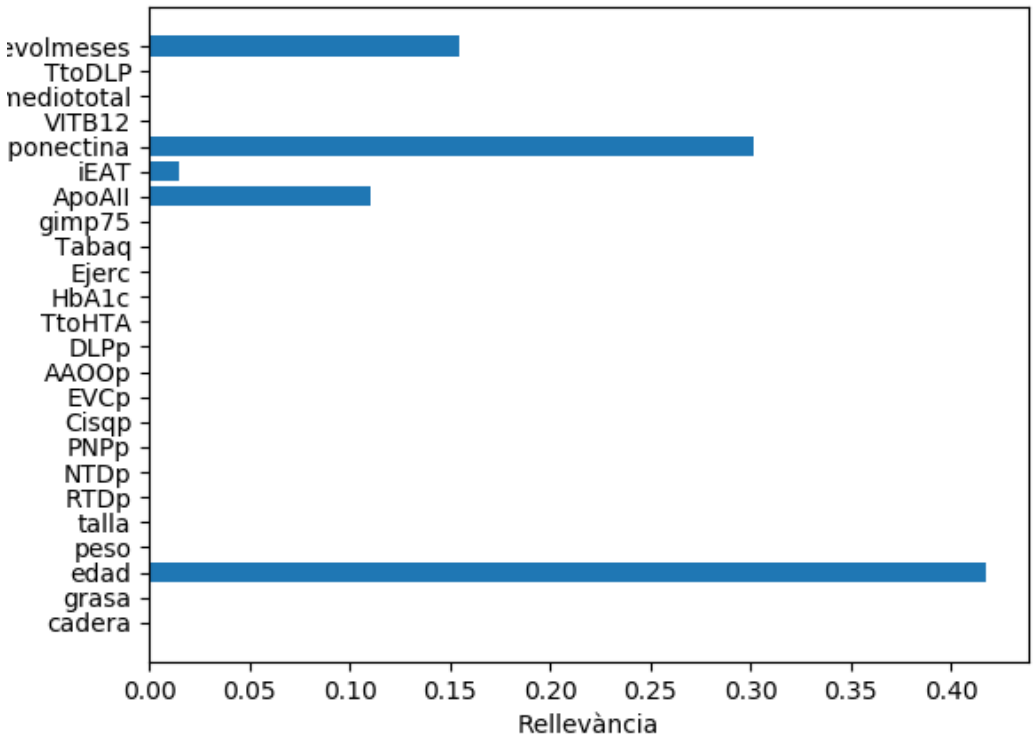
```
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier(max_depth=4, min_samples_leaf=5,
random_state=None)
tree.fit(X_train, y_train)

print ("\nArbre de decisió")
print ("================")

print ("Precisió entrenament: {:.2f}%".format(tree.score(X_train,
y_train)*100), "precisió test: {:.2f}%\n".format(tree.score(X_test, y_test)*100))

print ("Importància de les variables:
\n{}".format(tree.feature_importances_))

n_features = len(df.columns)-1
import numpy as np

def plot_feature_relevancy(model):
    n_features = df.shape[1]
    plt.barh(range(n_features), model.feature_importances_,
align='center')
    plt.yticks(np.arange(n_features), features_names)
    plt.xlabel("Rellevància")
    plt.ylabel("Variable")
    plt.show()
plot_feature_relevancy(tree)
```

Tenim la següent sortida on se´ns mostra la importància de les variables per generar l´arbre.

```
Arbre de decisió
================
Precisió entrenament: 85.96% precisió test: 55.00%

Importància de les variables:
[ 0.          0.          0.41779963  0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.11090235  0.0151813   0.30175225
  0.          0.          0.          0.15436447]
n_variables= 23
```

Il·lustració 20: Sortida precisió e importància de les variables per el arbre.

Que es poden representar de forma gràfica de la següent manera.



Il·lustració 21: Valors de la il·lustració 17 representats en un gràfic.

I el següent codi

```
from sklearn.tree import export_graphviz
export_graphviz(tree, out_file='arbre.dot', class_names=["lesió",
"no_lesió"], feature_names=features_names_only, impurity=False, filled=True)

import graphviz
with open('arbre.dot') as f:
        dot_graph=f.read()
graphviz.Source(dot_graph)
```

Ens mostrarà una implementació del arbre. També podrem veure l'arbre utilitzat amb la següent instrucció:

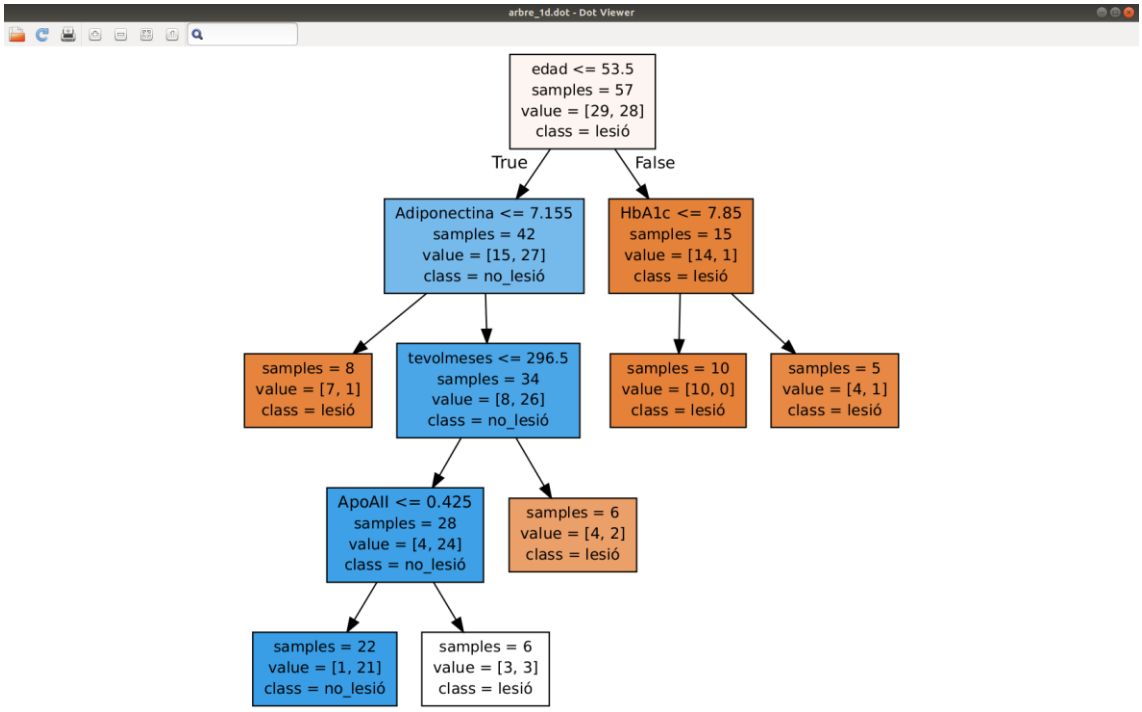
- xdot arbre.dot

Si no es te la llibreria gràfica 'graphviz' instal·lada, es pot instal·lar amb:

- conda install graphviz

I l'aplicació 'xdot' amb:

- sudo apt install xdot



Il·lustració 22: Arbre de decisions.

Un consell en els arbres, seria seguir el camí que classifica el major número de mostres.

4.7.4 Un bosc

El 'bosc aleatori' (random forest) es una combinació d'arbres de decisió. Cada arbre es una mica diferent dels altres, de forma aleatòria. Així, es pot fer el valor mig del resultat minimitzant el major inconvenient dels arbres de decisió: el *overfitting*. El bosc es crea a partir de varis arbres de decisió.

Al codi, en el paràmetre 'n_estimators' decidim el nombre de arbres que utilitzarem per fer el valor mig.

Cada un funcionarà sobre un conjunt de dades diferent; *bootstrap sample*. Aquest es farà traient algunes mostres a l'atzar del *Data Set* original i duplicant d'altres (tenint sempre el mateix número de mostres que en l'original). Seguidament s'aplica un arbre de decisió per cada *bootstrap*, amb la diferencia que també es selecciona un subconjunt de característiques en cada node del arbre. Aquets dos mecanismes (seleccions aleatòria de mostres i de característiques) fan que els arbres del bosc siguin diferents [4].

Amb 'max_features' podríem variar el nombre de característiques si no volguéssim el valor per defecte. Amb valors petits, tindrem arbres més profunds, i amb valors més grans (fins al número total de característiques) tindrem arbres més semblants [42].

Per evitar que el bosc mostri *ovefitting* tenim també paràmetres que podem ajustar. Si fos el cas, la millor solució seria afegir mes mostres. Sinó podríem fer els següents ajustos als paràmetres del model:

- n_estimators: en general, quan més arbres millor. Com més petit serà el model, més semblant a un únic arbre.
- max_features: Com més petits, menys probabilitats de *overfitting*, però massa petits pot fer que el nostre model no aprengui.
- max_depth: redueix la complexitat dels models. Es recomana començar amb valors petits i anar pujant.
- min_samples_leaf: la branca deixarà de dividir una vegada que les fulles tinguin aquest nombre de mostres cadascuna.

En aquest treball, després de diverses proves, s'han optat per els següents paràmetres. S'ha agafat el número màxim de característiques per arbre igual a l'arrel quadrada del número total de característiques: *max_features=sqrt(n_features)*. També s'ha agafat una profunditat màxima de tres per arbre i un número mínim de cinc mostres per fulla. El bosc estarà format per tres-cents arbres.

Per fer les prediccions amb '*random forest*', s'utilitza un mecanisme de votació. Cada arbre te una valor de predicció amb una classe al final. Després es fa el valor mig de les prediccions i s'agafarà la classe resultant amb més probabilitat [4].

Amb el codi següent

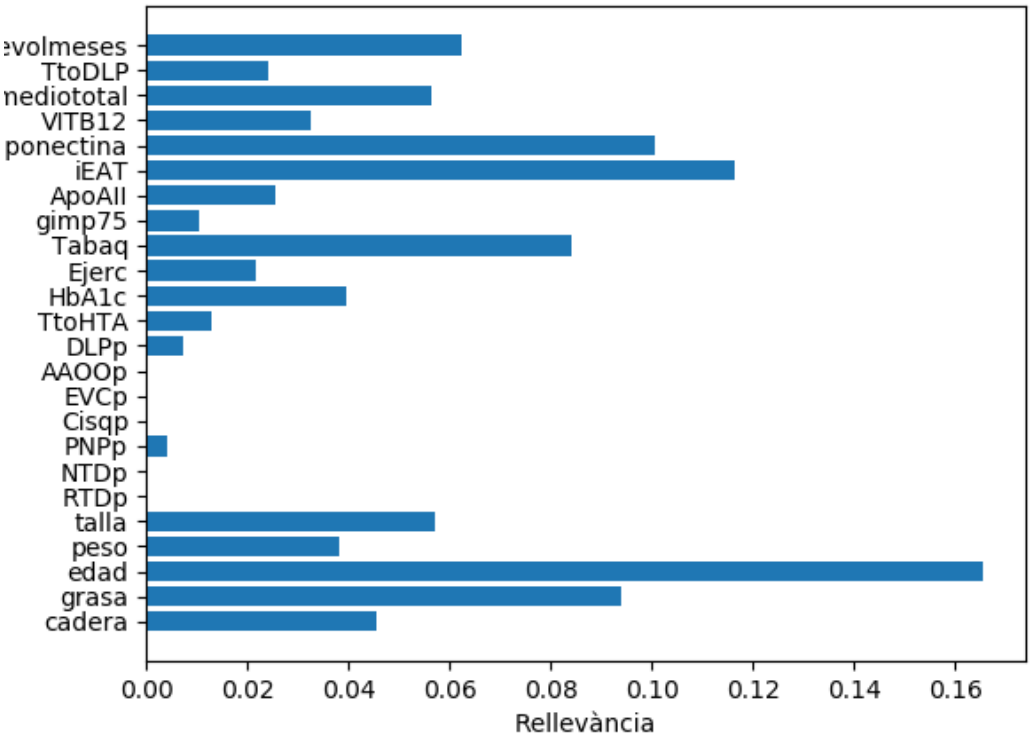
```
        from sklearn.ensemble import RandomForestClassifier
        forest= RandomForestClassifier (n_estimators=300,
max_features='auto', max_depth=3, min_samples_leaf=5, random_state=0)
        forest.fit(X_train, y_train)

        print ("\nBosc aleatori")
        print ("=============")

        print ('Precisió entrenament: {:.2f}%'.format(forest.score(X_train,
y_train)*100), 'precisió test: {:.2f}%'.format(forest.score(X_test, y_test)*100))

        print ("Importància de les variables:
\n{}".format(forest.feature_importances_))

        plot_feature_relevancy(forest)
```

Obtenim els valors de precisió per entrenament i test, així com la importància de les variables per generar el bosc.

```
Precisio entrenament: 100.00% Precisio test: 65.00%
Importancia de les variables:
[ 0.06001271  0.06684469  0.11594477  0.06032188  0.07085039  0.0067489
  0.00560731  0.01358675  0.00109642  0.000663    0.          0.00556658
  0.01034468  0.07268857  0.02198281  0.05921425  0.01237519  0.03992139
  0.10423241  0.07952884  0.05461206  0.05680724  0.01579463  0.06525454]
```

Il·lustració 23: Sortida amb la precisió e importància de les variables per el bosc.

Les dades anteriors, de forma gràfica quedarien com segueix...



Il·lustració 24: Representació gràfica dels valors de la il·lustració 20.

El següent quadre resumeix tots els models que s-han implementat:

Cas	dataset	Conjunt de dades	array	Compo fins 95%	KNN precisió / veïns	Millor cas: %entrenament / %test			
						RL	SVC	Arbre	Bosc
1	Matriu original omplint els buits. Classes 36/41	A	(77, 127)	44	70% / 2	80.9% /71.4% L2, c=1e-05	85.9% / 70% c=1	89.4% / 50%	98.2% / 60%
		B	(77,15)	10	75% / 14	71.9% / 70% L1 c=0.1	73.6% / 70% C=0.001	82.4% / 60%	84.2% / 60%
		C	(77,16)	10	60% / 8	70% / 65% L2 c=0.1	71.9% / 60% c=50	84.2% / 70%	100% / 55%
		D	(77, 25)	17	80% / 6	71.9% / 70% L1 c=0.1	71.9% / 80% c=0.001	85.9% / 55%	85.9% / 65%
2	Files sense NaNs Classes 13/15	A	(28, 127)	20	71.4 % / 14	80.9% / 71.4% L2 c=1e-05	95.2% / 71.43% c=0.1	85.7% / 71.4%	95.2% / 71.4 %
		B	(28/15)	5	85.7% / 2	80.9% / 71.4% L1 c=0.1	76.1% / 85.7% c=1	80.95% / 57.1%	76.1% / 57.1%

Taula 3: Taula resum dels 20 models.

El codi funciona sobre tots els conjunt de dades i a la taula 3 podem observar els casos en que obtenim millors precisions.

Recordem que el cas 1 es per totes les dades (omplint els buits) i que el cas 2 es només per els pacients que no tenen dades amb NaN. Els conjunts de dades A, B, C i D fan referència als subconjunts de dades recomanades.

Podem observar que amb els boscos, mantenint pràcticament la mateixa precisió per l'entrenament que amb els arbres, mantenim o millorem la precisió per al conjunt de les dades de test, sobretot quan disposem de mes mostres i/o característiques.

Tot i que aconseguim bons resultats per SVC en els casos 1D i 2B, hem optat per la senzilla e intuïtiva implementació de KNN per 1D. Aquesta ja ens dona una bona classificació per a nous pacients de forma consistent en diferents execucions del codi.

Veiem també que la recomanació de la metgessa sembla més encertada quan tenim més pacients amb dades complertes.

La majoria dels exemples descrits en aquest escrit són per el cas 1-D, en el que hem agafat el *Data Set* original, i només ens hem fixat en un grup de variables, les variables recomanades per la doctora de l'estudi, a les que hem afegit les que hem vist que sortien més sovint fent els arbres i boscos de decisió així com les recomanades per el mateix codi (apartat 4.1 d'aquest treball).

Caldria també valorar amb un expert del domini si aquestes decisions han sigut les encertades o s´hi s'hagués volgut, per exemple, utilitzar un altre sub-conjunt de dades.

El treball, com mostra la taula 3, ha fet la anàlisis en els sub-conjunts mencionats, però també en el conjunt total de dades.

Cal recordar que al anar canviant els grups d'entrenament i test (tenen un component aleatori cada cop que s'executa el codi) ens podríem trobar petites variacions en els resultats. No s'han adjuntat totes les sortides del codi (gràfics, text...) per a tots els casos per no fer aquest document massa extens.

5 Conclusions

De forma general s'han assolit els objectius i s'ha seguit la planificació durant tot el treball. Alguns problemes han sobrevingut per la desconeixença del domini i la dificultat de entendre les dades representades. La informació de l'apèndix 8.3, obtinguda posteriorment, continua sent una mica escassa.

Breu anàlisis dels resultats;

- Amb la manipulació inicial de les dades veiem ja d'entrada que la població del estudi mostra clarament dos factors tradicionals de risc: tabaquisme i sobrepès (afectant especialment als homes).

- Amb el grup reduït de variables podem predir el risc de lesió cardíaca en un pacient nou amb un encert del 80% utilitzant el model k-NN.

- Factors, variables importants:

  La edat sembla determinar també les probabilitats de patir lesions. Altres anàlisis a tenir en compte són els valors de 'iEAT' (grassa per superfície corporal), 'HbA1c' (Hemoglobina glicosilada, que com hem dit, serveix per mesurar els nivells de sucre a la sang), 'telvomeses' (temps de evolució, en mesos, de la diabetis en el pacient) i la talla.

  Veiem que les variables més utilitzades en els arbres (il·lustracions 20 i 21) són: 'edad', 'iEAT', 'adiponectina' (un estudi del 2004 la relaciona, com un fet innovador, amb malalties cardíaques en pacients amb diabetis tipus 2 [43]) , 'HbA1c', 'talla'...

  En generar el bosc veiem (il·lustracions 23 i 24) que torna a sortir la 'edad' com un factor important. També ens trobem amb 'iEAT'. Juntament amb la 'adiponectina'. Aquestes son les tres variables amb major índex d'utilització.

  En el *KBest*, per tot el primer *Data Frame*, (il·lustració 9) observem 'telvomeses', 'ApoB' ("l'apolipoproteïna B pot ser un marcador eficaç en la prevenció de les malalties cardiovasculars" [44][45]), i el sexe masculí com variables importants.

  Així que, de forma resumida, podríem enumerar alguns dels factors que podrien influir en aquest risc de patri lesió: la edat, el sexe, l'hemoglobina glicosada, la grassa per superfície corporal, la adiponectina i la talla.

- Hem obtingut arbres que es podrien utilitzar per seguir els anàlisis i determinar la possibilitat de que un pacient tingui lesions.

De totes maneres faltaria una revisió des de el punt de vista mèdic per veure si els resultats són vàlids o, segurament, necessitarien d'algun ajust. El no tindre coneixements mèdics dificulta el tenir que treballar amb un munt de característiques per pacient de les que no coneixem els seus valors límits ni de fet, en la majoria de cassos, el que representen.

Per exemple, en la introducció del quart capítol hem tret unes dades que semblen molt relacionades amb la 'TC2'. Després d'això veiem que les gràfiques de PCA i MDS mostren més dificultat en classificar les mostres, així com un encert més dolent en els models escollits. Ens faltaria una opinió mèdica per ratificar si el nostre supòsit (que han utilitzat 'TC2' per determinar aquets valors) es cert o no. Si no fos el cas, voldria dir que els nostres models serien encara millors ja que les podríem tornar a incloure (i ja hem vist que milloren les prediccions).

En un futur intentaria contactar amb algú que pogués fer una revisió des de el punt de vista del domini, així com intentar aconseguir més mostres.

El número de pacients d'aquest estudi es força petit comparant amb altres casos en el que es poden aplicar models de *Machine Learning*. Potser es podria intentar aconseguir dades d'algunes de les associacions de diabetis que les cedeixen per treballs d'investigació o universitaris.

Algunes de les associacions, o organitzacions que es podrien contactar són:

- European Association for the Study of Diabetes (https://www.easd.org/)
- International Diabetes Federation (https://www.idf.org/)
- UC Irvine Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Diabetes)

També esmentar que s'han tingut que refer algunes parts del codi al descobrir la llibreria de Pandas a mig camí. El llibre de Mckinney, referenciat a la bibliografia al número catorze, ha sigut de gran ajut.

6 Glossari

Del domini treballat:

DM1: Diabetis Mellitus Tipus 1.

TCMD: Tomografia computeritzada multi detectora.

IMC: Índex de massa corporal.

HbA1c: Hemoglobina glicosilada.

De la metodologia utilitzada:

IA: Intel·ligència Artificial. John McCarthy va encunyar aquest nom per primer cop al 1956. Te moltes definicions, la de Marvin Minsky es de les més senzilles, es la “realització de sistemes informàtics amb un comportament que en l’ésser humà qualificaríem d’intel·ligent” [1].

ML: Machine Learning.

Model: es una especificació matemàtica, o probabilística, de la relació que existent entre diferents variables [9].

PCA: Principal Component Analysis. Anàlisi de Components Principals.

MDS: Multidimensional Scaling. Escalat Multi Dimensional.

SVM: Suport Vector Machines. Màquina de Vector de Suport.

k-NN: k-nearest neighbors algorithm. Classificador del veí més proper.

Pandas: és una biblioteca de codi obert, amb llicència BSD, que proporciona estructures de dades d'alt rendiment i fàcil d'usar així com eines d'anàlisi de dades per al llenguatge de programació Python. Es van crear al 2010. Les estructures principals són els DataFrame, una estructura de dades tabular amb columnes, amb etiquetes per les files i les columnes, i les Series. Aquestes són arrays d’objectes amb una dimensió [14].

NumPy: abreviació de Numerical Python, es un paquet per a càlculs informàtics científics amb Python. Els ndarray, numpyarrays, són molt eficients per guardar i manipular dates. Llibreries escrites en altres llenguatges (com a C o Fortran) poden accedir a aquestes dades directament.

Matplot: són unes llibreries per Python que permeten fer gràfics i diferents tipus de visualitzacions de dades en 2 dimensions.

7 Bibliografia

1. Barceló, M. (2005). *La intel.ligencia artificial.* Barcelona: UOC.
2. Benítez , R., Escudero , G., & Kanaan, S. *Intel.ligència artificial avançada.* FUOC PID_00174125.
3. Benitez Iglesias, R., & UOC. *Principal Component Analysis*. Recollit de https://player.vimeo.com/video/109796748
4. C.Muller, A., & Guido, S. (2016). *Introduction to Machine Learning with Python. A guide for Data Scientists.* Sebastapol: O´Reilly Media, Inc.
5. Chapmann, J. (2017). *Machine Learning Algorithms.* Leipzig: Amazon Distribution.
6. *Chi-Squared For Feature Selection*. (30 / 12 / 2017). Recollit de https://chrisalbon.com/machine_learning/feature_selection/chi-squared_for_feature_selection/
7. Colom Comi, C. (02 / 11 / 2016). *Aterosclerosis silente en la diabetes tipo 1 prevalencia y perfil de riesgo. ISBN: 9788449068287* . Recollit de https://tesisenred.net/handle/10803/399829
8. *Features Selection*. (2018). Recollit de http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
9. Grus, J. (2017). *Data Science from Scratch.* Sebastopol: O´Reilly Media, Inc.
10. *Instal·lació Ubuntu.* (2018). https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop#0.
11. JanniK, E. (19 / 02 / 2017). *Feature Selection*. Recollit de https://www.wildcardconsulting.dk/useful-information/never-do-this-mistake-when-using-feature-selection/
12. Krolewski AS, Kosinski EJ, Warram JH, Leland OS, Busick EJ, Asmal AC, . . . Kahn CR. (1 / Apr / 1987). *Magnitude and determinants of coronary artery disease in juvenile-onset, insulin-dependent diabetes mellitus.* Recollit de 59(8):750-5.: https://www.ncbi.nlm.nih.gov/pubmed/3825934
13. Marlin, B. M. (2008). *Missing Data Problems in Machine Learning*. Recollit de https://tspace.library.utoronto.ca/bitstream/1807/11232/1/Marlin_Benjamin_M_200806_PhD_thesis.pdf
14. Mckinney, W. (2017). *Python for Data Analysis - 2nd edition.* Sebastapol: O´Reilly Media, Inc.
15. Nadeau, K. (2018). *Insulin resistance in adolescents with type 1 diabetes and its relationship to cardiovascular function.* Recollit de https://www.ncbi.nlm.nih.gov/pubmed/19915016
16. *Pandas-Na*. (2018). Recollit de https://pandas.pydata.org/pandas-docs/stable/missing_data.html
17. *QueComenLasAnacondas*. (2018). Recollit de https://www.quecome.com/que-comen-las-anacondas/
18. *SuseCamaleon*. (2018). Recollit de https://slashdot.org/story/00/03/01/0754201/suse-name-the-mascot-contest-is-over

19. Tagliaferri, L. (2018). Recollit de https://www.digitalocean.com/community/tutorials/how-to-install-the-anaconda-python-distribution-on-ubuntu-16-04
20. *Ubuntu_about*. (n.d.). Retrieved from Ubuntu: https://www.ubuntu.com/about
21. *Wikipedia-Arterioesclerosi*. (2018). Recollit de https://ca.wikipedia.org/wiki/Arterioesclerosi
22. *Wikipedia-DM1*. (2018). Recollit de https://ca.wikipedia.org/wiki/Diabetis_mellitus_tipus_1#cite_note-2
23. *Wikipedia-Ecocardiografia*. (2018). Recollit de https://ca.wikipedia.org/wiki/Ecocardiografia
24. *Wikipedia-Endocrinologia*. (2018). Recollit de https://ca.wikipedia.org/wiki/Endocrinologia
25. *Wikipedia-IMC*. (2018). Recollit de https://ca.wikipedia.org/wiki/%C3%8Dndex_de_massa_corporal
26. *Wikipedia-Python*. (2018). Recollit de https://en.wikipedia.org/wiki/Python_(programming_language)
27. *La Vanguardia-online*. (2018). Recollit de http://www.lavanguardia.com/ciencia/cuerpo-humano/20180302/441189311258/cinco-tipos-diabetes-diagnostico.html
28. *Wikipedia-Hemoglobina*. (2018). Recollit de *https://ca.wikipedia.org/wiki/Hemoglobina_glicosilada*
29. *Wikipedia-Lípds*. (2018). Recollit de *https://ca.wikipedia.org/wiki/L%C3%ADpid*
30. *Medlineplus-Cholesterol*. (2018). Recollit de *https://medlineplus.gov/spanish/ldlthebadcholesterol.html*
31. *Wikipedia-Hipertensió*. (2018). Recollit de *https://ca.wikipedia.org/wiki/Hipertensi%C3%B3_arterial*
32. *Wikipedia-Retinopatia*. (2018). Recollit de *https://ca.wikipedia.org/wiki/Retinopatia*
33. *Wikipedia-RetinopatiaDiabètica*. (2018). Recollit de *https://ca.wikipedia.org/wiki/Retinopatia_diab%C3%A8tica*
34. *Wikipedia-Nefropatia*. (2018). Recollit de *https://es.wikipedia.org/wiki/Nefropat%C3%ADa*
35. *Wikipedia-NefropatiaDiabètica*. (2018). Recollit de *https://es.wikipedia.org/wiki/Nefropat%C3%ADa_diab%C3%A9tica*
36. *Wikipedia-Polineuropatia*. (2018). Recollit de *https://ca.wikipedia.org/wiki/Polineuropatia*
37. *Cun-MacroangiopatiaDiabètica*. (2018). Recollit de *https://www.cun.es/diccionario-medico/terminos/macroangiopatia-diabetica*
38. *Medlineplus-ChoresteolTest*. (2018). Recollit de *https://medlineplus.gov/spanish/labtests/cholesteroltest.html*
39. *Choresterolmenu-Levels*. (2018). Recollit de *https://www.cholesterolmenu.com/cholesterol-levels-chart/*
40. *ScikitLearn-DecisionTree. (2018). Recollit de http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier*
41. *ScikitLearn-DecisionTree. (2018). Recollit de http://scikit-learn.org/stable/modules/tree.html#tree-classification*

42. *ScikitLearn-RandomForest. (2018). Recollit de http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html*
43. *Estudi del* Dr. A. Pérez Servei d'Endocrinologia i Nutrició*. (2004). Recollit de http://www.sld.cu/galerias/pdf/sitios/diabetes/adiponectina-un_nuevo_nexo_entre_obesidad,_resistencia_a_la_insulina_y_enfermedad_cardiovascular.pdf*
44. *RiesgoAlipoproteinas.Ningu singa l'article. (2018). Recollit de https://www.tuotromedico.com/temas/riesgo_apolipoproteinas.htm*
45. *APOb. (2018). Recollit de https://www.infobioquimica.com/wrapper/CDInterpretacion/te/bc/089.htm*
46. *Wikipedia-DM2. (2018). Recollit de https://ca.wikipedia.org/wiki/Diabetis_mellitus_tipus_2*
47. Vicenç Torra i Reventós. Resolució de problemes i cerca*.* FUOC • PID_00163090.

8 Annexos

8.1 El sistema endocrí.

"El sistema endocrí està compost per diverses glàndules, a diferents parts del cos, que secreten hormones directament al torrent sanguini. Les hormones tenen diferents funcions i maneres d'actuar; una hormona pot tenir diversos efectes depenent dels òrgans diana, i alhora, un òrgan pot ser diana de diverses hormones" [24]. A diferencia del sistema nerviós que utilitza impulsos elèctrics, el sistema endocrí utilitza hormones per regular el funcionament de diferents òrgans.

Un trastorn hormonal pot causar malalties greus; insuficiència suprarenal, osteoporosis, diabetis –insípida, mellitus- …

"La diabetis és un trastorn general del metabolisme que es manifesta per una elevació anormal dels nivells de glucosa a la sang. Aquesta anomalia pot ser deguda a una producció insuficient d'insulina o bé a un mal aprofitament d'aquesta hormona per part de l'organisme [22].

"Segons la classificació actual, la diabetis de tipus 1, que es desenvolupa generalment en la infància, es produeix quan el cos no genera suficient insulina per regular la glucosa, a causa que el sistema immune ataca per error les cèl·lules del pàncrees que produeixen l'hormona i les destrueixen" [27].

La diabetis tipus 2 (antigament anomenada diabetis de l'adult o diabetis no-insulinodependent) és una malaltia caracteritzada pels alts nivells de glucosa en sang en un context de resistència a la insulina i deficiència relativa d'insulina. Tot i que inicialment sovint es controla només incrementant l'exercici i amb una modificació de la dieta, quan la malaltia progressa els medicaments són necessaris [46].

Hi ha estudis, però, que indiquen que poden haver no dos, sinó cinc tipus de diabetis [27]. Aquest treball es centra en els pacients amb Diabetis Mellitus tipus 1.

La grassa epicàrdica, o teixit adipós epicàrdic, es una forma de grassa visceral que s'ubica sobre el miocardi (solcs auriculo ventricular e interventricular) i al voltant de les arteries coronaries. En condicions normals actuen com a un buffer que protegeix el cor de la lipotoxicitat i a demès proveeix dels lípids necessaris per la obtenció de energia mitjançant la beta-oxidació. En condicions patològiques es produeix una disfunció que porta a la pèrdua del efecte cardio protector, es crea una expansió, i es torna hipotòxic, sent capaç de alliberar molècules pro-inflamatòries i pro-aterogeniques [7].

"L'arteriosclerosi, popularment denominada *enduriment de les artèries*, és una malaltia crònica i progressiva produïda pel dipòsit de greixos a la paret interna de les artèries que en provoca l'estenosi, és a dir, la disminució de la secció o llum interior" [21]. L'arteriosclerosi es responsable de esdeveniments cardiovasculars mortals en pacients diagnosticats amb diabetis [15]. Factors com augment de la tensió arterial, el colesterol total i el colesterol LDL, presencia de marcadors inflamatoris i resistència a la insulina també influeixen [7].

Tot i que actualment hi han tècniques per detectar aterosclerosis precoç, no hi ha un consens en com fer el cribratge per estudiar els pacients DM1 asimptomàtics.

La presencia de calci en les arteries coronaries també es un marcador de aterosclerosis que prediu malalties cardiovasculars en les persones (sense tenir que ser diabètics). Es defineix el índex de calcificació coronaria o score càlcic (CAC).

La tomografia computeritzada multi detectora (TCMD) permet la detecció de estenosis luminal (produïdes per plaques –calcificades o no-) en les arteries coronaries. Això permet una predicció de malalties coronaries. A l'estudi mèdic es va fer servir un TCMD de 256 talls (Philips iCT256 de Philips Health Care, Amsterdam, Països Baixos) amb un contrast endovenós [7].

La ecocardiografia "també conegut com a ultrasò cardíac, fa servir tècniques estàndards d'ultrasò per produir imatges en llesques de dues dimensions del cor" [23] es una proba senzilla però nomes en un pla, la TC amb ressonància magnètica aporta més informació.

Es poden fer també estudis per a la detecció de estenosis i/o rigidesa arterial amb una tonometria arterial perifèrica així com medicions del flux sanguini mitjançant pletismografia venosa d'extremitats.

8.2 Versions utilitzades durant el treball.

L'entorn de treball te les següents versions de OS, Python i llibreries...

```
Versions:
=========

Python version: 3.6.3 |Anaconda custom (64-bit)| (default, Oct 13 2017, 12:02:49)
[GCC 7.2.0]
IPython version: 6.1.0
NUMPY version: 1.13.3
MATPLOTLIB version: 2.2.2
PANDAS version: 0.22.0
SKLEARN version: 0.19.1
graphviz version: 0.5.2
```

Amb una mica més de detall:

```
INSTALLED VERSIONS
------------------
commit: None
python: 3.6.3.final.0
python-bits: 64
OS: Linux
OS-release: 4.13.0-41-generic
machine: x86_64
processor: x86_64
byteorder: little
LC_ALL: None
LANG: en_US.UTF-8
LOCALE: en_US.UTF-8

pandas: 0.22.0
pytest: 3.2.1
pip: 10.0.1
setuptools: 36.5.0.post20170921
Cython: 0.26.1
numpy: 1.13.3
scipy: 0.19.1
pyarrow: None
xarray: None
IPython: 6.1.0
sphinx: 1.6.3
patsy: 0.4.1
dateutil: 2.6.1
pytz: 2017.2
blosc: None
bottleneck: 1.2.1
tables: 3.4.2
numexpr: 2.6.2
feather: None
matplotlib: 2.2.2
openpyxl: 2.4.8
xlrd: 1.1.0
xlwt: 1.3.0
xlsxwriter: 1.0.2
lxml: 4.1.0
bs4: 4.6.0
html5lib: 0.999999999
sqlalchemy: 1.1.13
pymysql: None
psycopg2: None
jinja2: 2.9.6
s3fs: None
fastparquet: None
pandas_gbq: None
pandas_datareader: None
```

8.3 Algunes definicions de les variables utilitzades.

Algunes de les variables que formen part de l'estudi, de les que si tenim alguna informació, són:

cadera	Perímetre maluc
grasa	Grassa epicardica
edad	Edat 1a visita
fechadx	Data diagnòstic
FExpl	Data exploració
tevolaños	Anys evolució
tevolmeses	Temps evolució
peso	Pes (Kg)
talla	Talla (cm)
IMC	IMC (kg/m2)
sobrepes	Sobrepès
obes	Obès
CIsqp	Cardiopatia isquèmica
EVCp	Malaltia vascular cerebral
AAOOp	Anys evolució
TtoHTA	Tractament per HTA
TtoDLP	Tractament hipolipemiant
Tabaq	Tabaquisme
Ejerc	Activitat física
TipoTTDM	Tipus tto de diabetis actual
Dosis	Dosis de insulina(U/Kg/dia)
AAS	Qualsevol tto antiagregant
cLDL	LDLc
cLDL130	LDL <130
cLDL100	LDL<100
cVLDL	VLDLc
ApoA1	ApoA-I
ApoB	ApoB
ApoAII	ApoA-II
Nefas	NEFA
Lpa	Lp(a)
HbA1c	Valor Hba1c AS actual
creatinina	Creatinina en UI
FG	Filtrat glomerular
Homocisteina	Homocisteïna en UI
urato	Urats en UI
VITB12	Vitamina B12
TPO	ac TPO
TIROGLOB	ac anti tiroglobulina
CELPARIET	ac anti cèl·lula parietal
TRANSGLUT	ac anti transglutaminasa

albuminuria	Albuminúria 24h
proteinuria	Proteïnúria 24h
indicealbcreat	Índex alb/creatinina
Diuresis	Diüresis 24h
masacolHDL	% massa col HDL
masaTgHDL	% massa Trig HDL
masafosfHDL	% massa phos HDL
masacollibreHDL	% massa col libre HDL
masacolestHDL	% massa col ester HDL
HDLAI	% massa apoA1 HDL
HDLAII	% massa apoAII HDL
HDLE	% massa ApoE-HDL
HDLCIII	% massa ApoCIII-HDL
EFLUXHDL	% Eflux HDL
IL6	IL6 (pg/ml)
IL10	IL10 (pg/ml)
TGFb	TGFb (ng/ml)
leptina	Leptin (ng/ml)
Adiponectina	Adiponectin (ug/ml)
Pulsos	Absència de polsos perifèrics
PNP	Signés de polineuropatia
SCORE	Score càlcic segons índex de Agatston (TC multidetector)
atero1	Pacients con lesiones significatives el el TC o GIMcc o GIM fc>percentil 75
atero2	Atero2
gimp75	GIM carotidi o femoral>P75 de mostra de referència
GIMcinternamaxDel	Grossor de íntima media carotídia interna màxima, mitja costat Dret i Esqu
GIMcinternamedioDel	Grossor de íntima mitja carotídia interna , mitjana del costat Dret i Esquerre
GIMcextmaxDel	Grossor de íntima mitja carotídia externa, màxim del costat Dret i Esquerre
GIMcextmedioDel	Grossor de íntima mitja carotídia mitja, màxim del costat Dret i Esquerre
GIMccdi	mitja entre costat dret i esquerre
GIMP75muestra	GIMcarotide o femoral máx i/o mitja > P75 de mostra
GIMpP75muestra	GIMpopliteo max i/omedio >P75 mostra
HbMedia1a5a	HbA1C mitja 5 1º años
HbA1CMedia	HbA1C mitja total (de cada pacient des de el 1er any)
activo	Actiu?
EJE	Exercici regularment?
micromacro	Presencia de complicacions micro o macro
TCHelic	TC helicoïdal presencia de lesiones estenoticas
GIMfcp75	GIMfemoral>P75

GIMccp75	GIMcarotideo >P75
ASYC	aterosclerosis clínica o subclinica (lesions en TC, GIM f o cc > P75 o macroangiopatia)
ASYCsign	macroangiopatia, >P75, lesions TC>50
sexo	Sexe
cintura	Perímetre cintura cm
Tg	Tg (mmol/L)
cHDL	HDLc
TC3	TC amb lessions <50% vs lesions > 50%
iEAT	Grassa/superfí corpo
iEATP75	Pac amb iEAT>p75
LC	Lesiones >50% en TC y/o iEAT>P75
TC2	TC con algun tipus de lesió vs cap lesió
ALC	TC con algun tipus de lesio y/o iEAT>P75
expotabac	NUNCA/ activo o exfum
ATERO1muestra	Macroangiopatia, >mostra, lesions TC>50
ATERO2muestra	Macroangiopatia, >P75, alguna lesio TC
fum	Ha sigut fumador algun cop?

8.4 Alguns rangs de valors de les variables utilitzades.

8.4.1 Colesterol

Tipus	Nivells	Categoria
Nivell de colesterol total	Menys de 200mg/dL	Desitjable
	Entre 200 i 239 mg/dL	Limit superior del rang normal
	240 mg/dL o més	Alt
Colesterol LDL (colesterol *dolent*)	Menys de 100 mg/dL	Valor òptim
	Entre 100 i 129 mg/dL	Per sobre del òptim
	Entre 130 i 159 mg/dL	Límit superior rang normal
	Entre 160 i 189 mg/dL	Alt
	190 mg/dL o més	Molt alt
Colesterol VLDL (colesterol dolent)	Menys de 29 mg/dL	Òptim
	Més gran de 30 mg/dL	Alt
Colesterol HDL (colesterol *bo*)	60 mg/dL o més	Protegeix el cor
	Entre 40 i 59 mg/dL	Quant mes alt millor
	Menys de 40 mg/dL	Factor de risc per el cor
Triglicèrids	Menys de 149 mg/dL	Desitjable
	De 150 a 199 mg/dL	Limit inferior al rang alt
	De 200 a 499 mg/dL	Alt
	Mes de 500 mg/dL	Molt alt

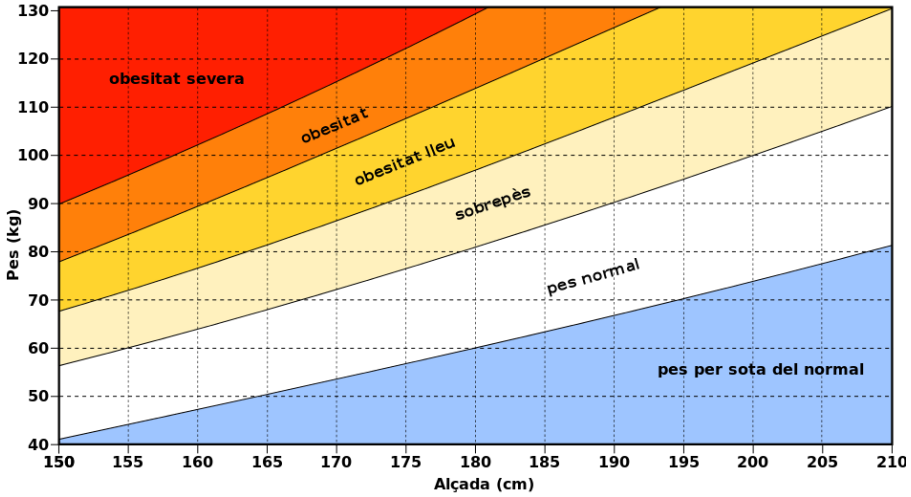
Il·lustració 25: nivells de colesterol [38][39].

8.4.2 IMC

En el treball es van seguir els criteris de la "Societat Española para el Estudio de la obesidad" per considerar que un pacient té sobrepès si presentava un IMC entre 25 i 29.9 kg/m^2 i té obesitat si presenta un IMC ≥ 30 kg/m^2.

IMC (kg * m^2)	Interpretació
Menys de 18	Magror
Entre 18 i 25	Corpulència normal
Entre 25 i 30	Sobrepès
Entre 30 i 40	Obesitat
Més de 40	Obesitat mòrbida

Il·lustració 26: Interpretació IMC [25].



Il·lustració 27: Taula IMC [25].

8.4.3 Síndrome metabòlic

A l'estudi es van considerar els criteris de la "International Diabetes Federation" (te en compte les diferencies ètniques). Es considera que hi ha el síndrome metabòlic quan hi ha presencia de obesitat central (perímetre abdominal ≥ 94 cm per els homes i ≥ 80cm per les dones) juntament amb dos més dels criteris següents:

- Triglicèrids ≥ 150 mg/dL o tractament específic per els mateixos
- Colesterol HDL < 40 mg/dL en homes i < 50mg/dL en dones
- Pressió arterial sistòlica ≥ 130 mmHg i/o diastòlica ≥ 85 mmHg o en tractament antihipertensiu.
- Glucèmia venosa basal ≥ 100 mg/dL o diabetis prèviament diagnosticada.

8.4.4 Sedentarisme

Es va considerar al pacient com sedentari si no realitzava exercici aeròbic més de dos dies per setmana o be caminava menys de 60 minuts al dia.

8.4.5 Tabaquisme

El criteri per marcar un pacient com a fumador es si consumia tabac diàriament.

8.4.6 Hipertensió Arterial

Es va definir hipertensió arterial quan la pressió arterial sistòlica ≥ 140 mmHg i/o la pressió diastòlica ≥ 90 mmHg o la toma d'un tractament antihipertensiu.