Estudi, anàlisi i predicció de l'estat de pacients amb Cirrosi

Cai Selvas Sala

28 de desembre de 2023

Universitat Politècnica de Catalunya
Grau en Intel·ligència Artificial
Introducció a l'Aprenentatge Automàtic

Resum

Aquest és l'informe corresponent a la pràctica individual de l'assignatura d'Introducció a l'Aprenentatge Automàtic del Grau en Intel·ligència Artificial de la Universitat Politècnica de Catalunya (UPC).

En el document s'explicarà com s'ha realitzat el projecte, quines dificultats s'hi han trobat, quins anàlisis s'han realitzat, quins resultats s'han obtingut i quines conclusions se'n poden extreure. A més, s'explica el codi en Python utilitzat durant la pràctica, així com els detalls del model creat (model card).

Índex

1 Introducció

La base de dades utilitzada en aquest treball és el Çirrhosis Patient Survival Prediction Dataset", disponible a través de la Universitat de Califòrnia a Irvine (UCI) [1]. Aquest conjunt de dades conté informació rellevant sobre pacients amb cirrosi hepàtica, incloent 17 característiques clíniques per cada pacient. L'objectiu principal d'aquesta base de dades és permetre la predicció de la supervivència dels pacients amb aquesta condició.

Les característiques clíniques incloses en el conjunt de dades proporcionen una àmplia gamma de dades sobre els pacients, inclòs l'estat de supervivència, que es tracta com a variable objectiu en aquest estudi.

El treball amb aquesta base de dades comporta diverses etapes importants, com ara l'anàlisi estadístic de les variables de manera independent, l'estudi del balanceig de classes, la identificació i gestió de valors perduts (missings) i atípics (outliers), la recodificació de variables si cal, i el particionat del conjunt de dades en subconjunts per a l'entrenament i la validació dels models.

A través d'aquest procés, es buscarà no només desenvolupar habilitats tècniques en manipulació de dades, programació i modelatge estadístic, sinó també cultivar una comprensió més profunda de com l'intel·ligència artificial pot ser aplicada en contextos reals i significatius, com és el cas de la salut i la medicina.

2 Documents i estructura general de l'entrega

En l'entrega d'aquesta pràctica s'hi inclou aquest informe juntament amb un arxiu comprimit `.zip`.

A dins d'aquest arxiu comprimit es poden trobar dues carpetes: assets i src. En el directori assets es troben altres directoris i un arxiu per importar les llibreries necessaries pel codi. En aquests directoris es guarden els arxius `.csv` amb les dades que es guarden (en cas d'especificar-ho així en els paràmetres) en algunes execucions, així com les imatges generades en algunes altres execucions. Per altra banda, en la carpeta src es troba l'arxiu `main.ipynb` amb tot el codi Python utilitzat en aquest projecte. L'estructura d'aquest arxiu Python consisteix principalment en moltes cel·les on es declaren funcions per cada tasca en concret. Cap al final del document Python comença a haver les primeres crides a funcions que criden amb l'ordre correcte a la resta de funcions. Totes les execucions d'experiments passen per la funció `run_experiment()`, ja que és la que s'encarrega de que es cridin ordenadament a totes les funcions i amb els paràmetres pertinents.

3 Anàlisis i preprocessat de dades

3.1 Preprocessat inicial

Una vegada importem el dataset, es pot veure que hi ha bastantes cel·les buides i altres amb el string `'NaNN'`. Per solucionar aquesta inconsistència, s'han reemplaçat tots aquests valors per `pd.NA`.

Per altra banda, s'ha declarat el tipus de cada variable correctament (com a numèriques o com a categòriques) seguint la informació que es proporciona en el metadata file (que es pot trobar en [1]).

A més, com que la variable ID no és res més que l'identificador dels pacients, i no serà necessari pel nostre estudi, s'ha decidit eliminar del dataset per no haver d'eliminar-la manualment a cada procés. És a dir, entrenar un model de predicció tenint en compte l'ID del pacient no té cap sentit i només pot portar a overfitting (si troba patrons entre la variable objectiu i la variable ID). A més, a l'hora de fer gràfics no és una variable que aporti cap informació, ja que és categòrica i amb tantes classes úniques com files hi ha al dataset, de manera no es podrien interpretar els plots de cap manera.

Addicionalment, per una millor comprensió de certes variables, s'ha decidit reanomenar els seus valors, tenint en compte el metadata file, de la següent manera:

- **Variable Status:**
  - **'C'** → 'Alive'.
  - **'CL'** → 'Liver Transplant'.
  - **'D'** → 'Dead'.
- **Variable Edema:**
  - **'N'** → 'NoEdema'.
  - **'S'** → 'EdemaResolved'.
  - **'Y'** → 'EdemaPersistent'.
- **Variable Drug:**
  - **'D-penicillamine'** → 1.
  - **'Placebo'** → 0.
- **Variables Ascites, Hepatomegaly i Spiders:**
  - **'Y'** → 1.
  - **'N'** → 0.

Una vegada realitzats aquests canvis, es pot començar a treballar amb el dataset correctament.

3.2 Anàlisis estadístic de les variables

El primer que s'ha fet per entendre el dataset i poder treballar amb ell és realitzar un anàlisis estadístic de cada una de les variables que el formen. A més, per les variables numèriques podem analitzar la distribució que segueixen mitjançant un histograma, mentre que per les categòriques podem realitzar countplots per veure la distribució entre les seves classes i com de balancejades estan.

3.2.1 Variables numèriques

En les taules 1 i 2 es poden veure estadístiques sobre totes les variables numèriques del dataset (obtingudes mitjançant la comanda `data.describe()` de la llibreria `pandas`).

Statistic	N_Days	Age	Bilirubin	Cholesterol	Albumin	Copper
count	418.0	418.0	418.000000	284.0	418.000000	310.0
mean	1917.782297	18533.351675	3.220813	369.510563	3.497440	97.648387
std	1104.672992	3815.845055	4.407506	231.944545	0.424972	85.61392
min	41.0	9598.0	0.300000	120.0	1.960000	4.0
25%	1092.75	15644.5	0.800000	249.5	3.242500	41.25
50%	1730.0	18628.0	1.400000	309.5	3.530000	73.0
75%	2613.5	21272.5	3.400000	400.0	3.770000	123.0
max	4795.0	28650.0	28.000000	1775.0	4.640000	588.0

Taula 1: Estadístiques de les variables numèriques.

Statistic	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
count	312.000000	312.000000	282.0	407.0	416.000000
mean	1982.655769	122.556346	124.702128	257.02457	10.731731
std	2140.388824	56.699525	65.148639	98.325585	1.022000
min	289.000000	26.350000	33.0	62.0	9.000000
25%	871.500000	80.600000	84.25	188.5	10.000000
50%	1259.000000	114.700000	108.0	251.0	10.600000
75%	1980.000000	151.900000	151.0	318.0	11.100000
max	13862.400000	457.250000	598.0	721.0	18.000000

Taula 2: Estadístiques de les variables numèriques.

Addicionalment, en les figures 1 i 2 es poden veure les histogrames de cada una de les variables numèriques, on es veu la distribució de les seves dades ignorant els valors faltants (missings).

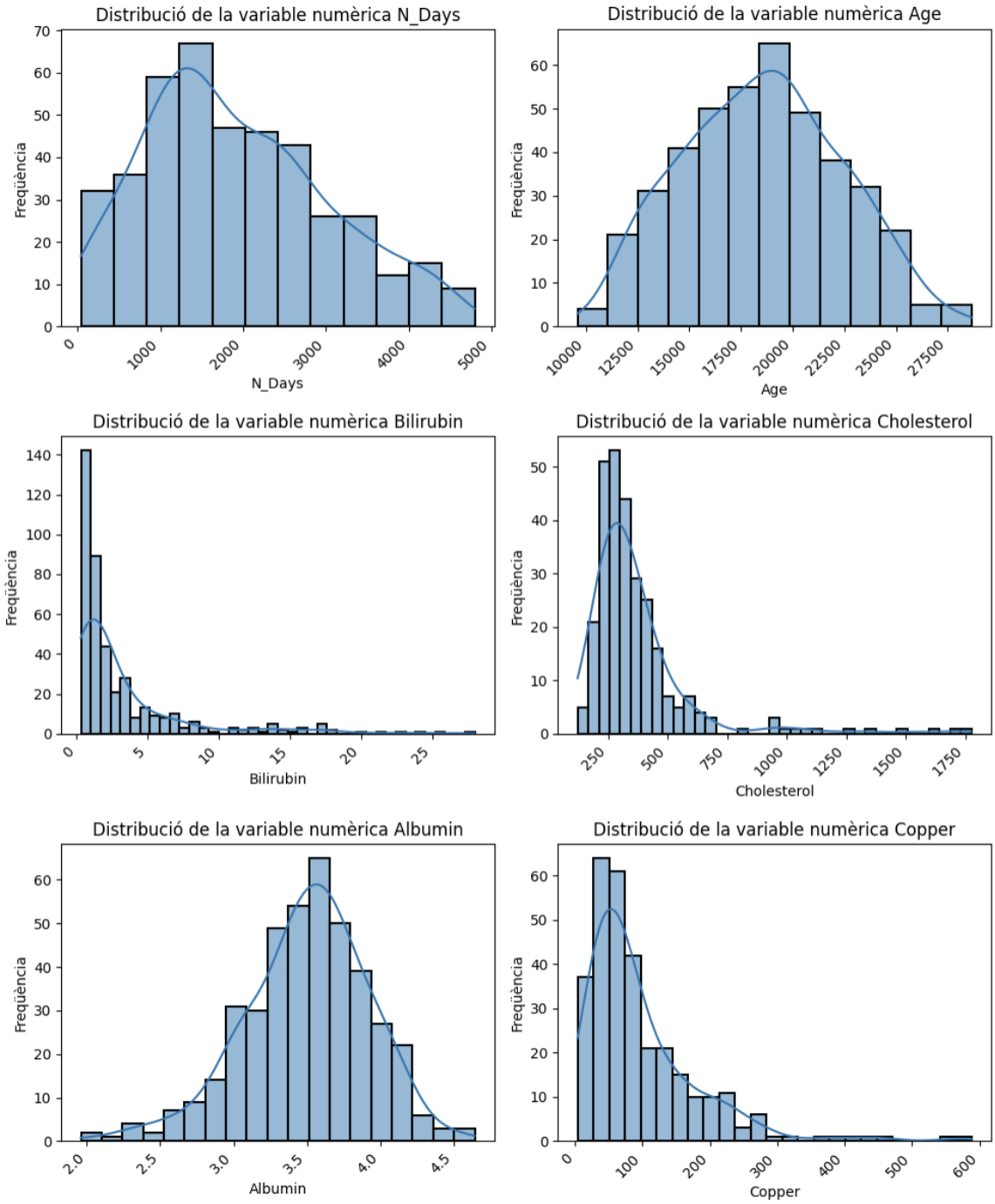


Figura 1: Histogrames de variables numèriques del datset.

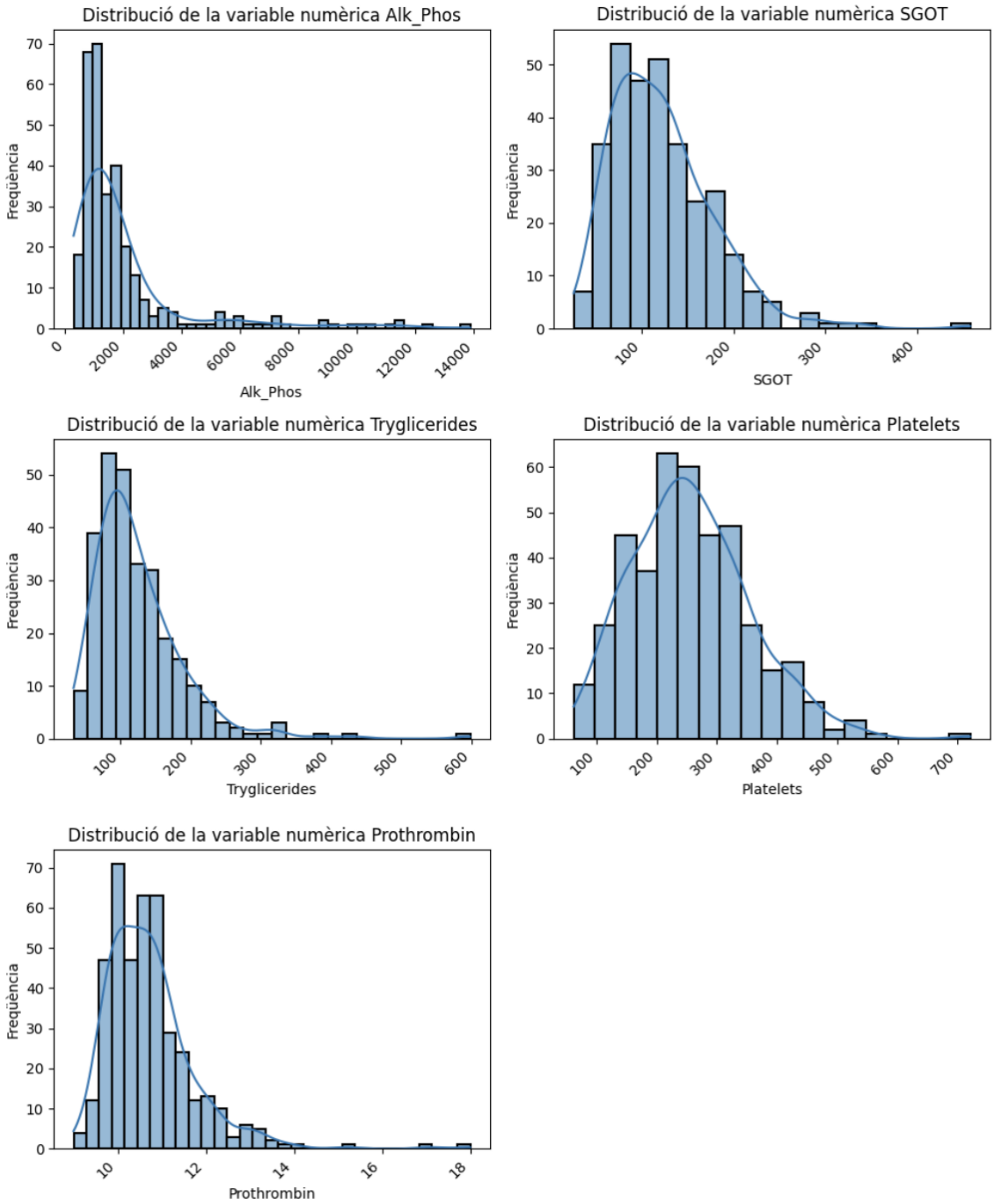


Figura 2: Histogrames de variables numèriques del datset.

En els histogrames de les diferents variables numèriques es pot veure com rarament segueixen una

distribució normal. Això, en certs casos s'ha de tenir en compte en els models per tal de millorar el seu rendiment.

Per altra banda, també es pot veure que hi ha certes variables que tenen valors bastant allunyats de la distribució principal de les dades de la variable (valors atípics o outliers, que es tractaran més endavant).

3.2.2 Variables categòriques

En la taula 3 podem veure altres estadístiques per les variables categòriques del dataset (obtingudes mitjançant la mateixa comanda, però amb el paràmetre `include='category'`).

Statistic	**Status**	**Drug**	**Sex**	**Ascites**	**Hepatomegaly**	**Spiders**	**Edema**	**Stage**
count	418	312	418	312	312	312	418	412.0
unique	3	2	2	2	2	2	3	4.0
top	Alive	1	F	0	1	0	NoEdema	3.0
freq	232	158	374	288	160	222	354	155.0

Taula 3: Categorical data summary of the study.

A més, en les figures 3 i 4 es poden veure els countplots de cada una de les variables categòriques, on es veu la quantitat de mostres que hi ha per cada classe de la variable, evitant els valors faltants (missings).

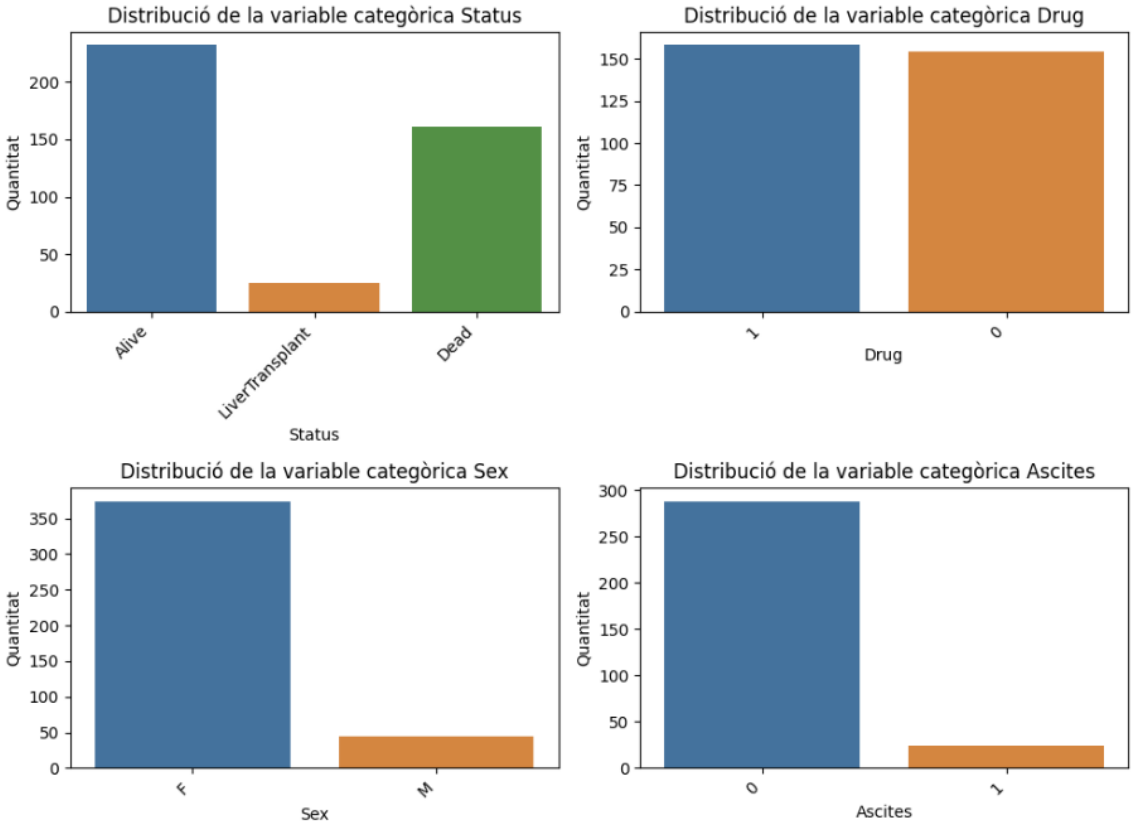


Figura 3: Countplots de variables categòriques del datset.

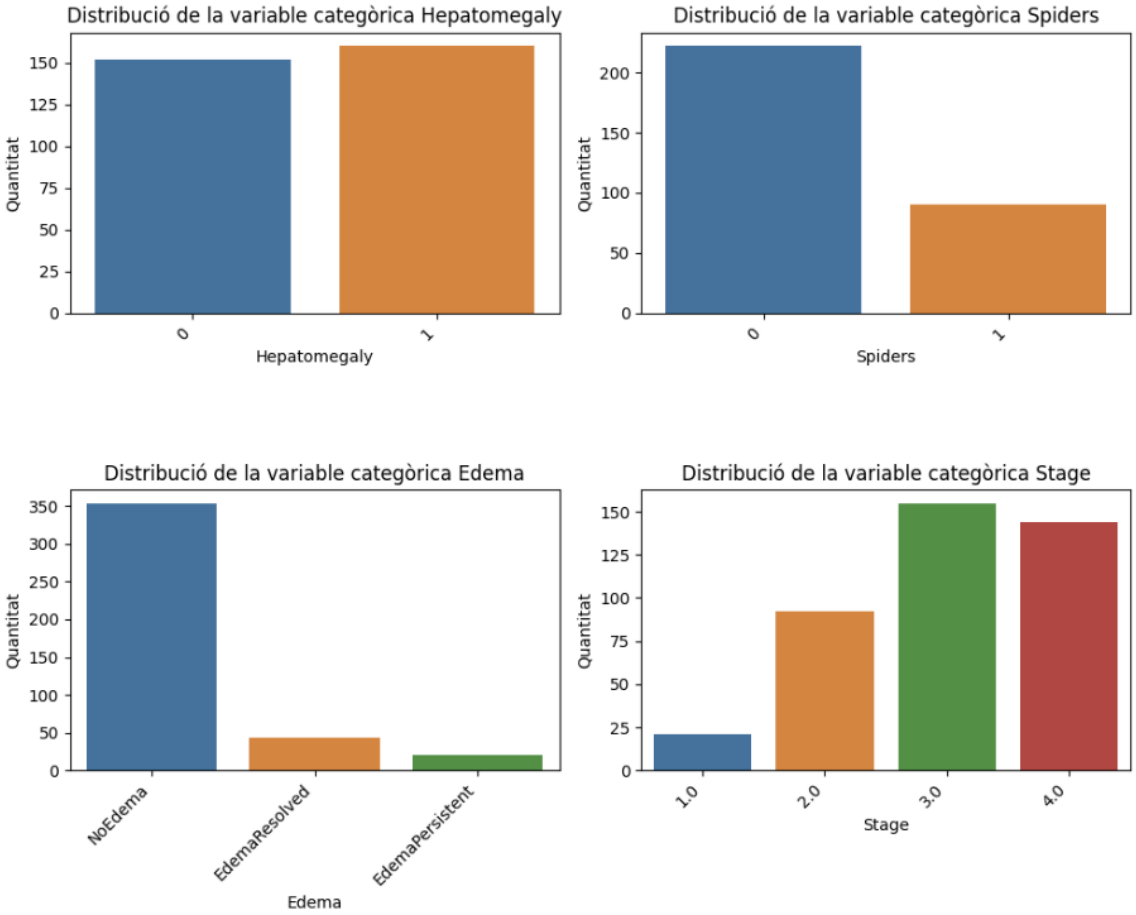


Figura 4: Countplots de variables categòriques del datset.

Es pot veure que les variables *Status* (la variable que determinarem com a *target* més endavant), *Sex*, *Ascites*, *Spiders*, *Edema* i *Stage* pateixen un clar desbalanceig de classes. Per la variable *Status* es contemplarà l'opció de realitzar un balanceig més endavant.

3.3 Outliers

El tractament d'outliers en models de Machine Learning pot ser molt important per millorar el seu rendiment mitjançant l'eliminació de mostres que es consideren atípiques. En la nostra base de dades, tal i com s'ha pogut veure en els histogrames de les variables numèriques (figures 1 i en les taules amb estadístiques de les variables (taules 2) i en les taules 1, 2 i 3), hi ha unes quantes mostres que possiblement siguin outliers. Per exemple, la variable *cholesterol* té un valor màxim de 1775, la qual cosa està exageradament per sobre de 240 (valor a partir del que es considera que una persona té un colesterol molt alt, segons la *Fundación Española del Corazón* [2]).

Per eliminar aquests outliers, s'ha utilitzat l'Interquartile Range (IQR). Inicialment, un factor de multiplicació de 1,5 eliminava 119 files (un 28,47% del dataset), resultant en menys de 300 files. Aquest valor semblava excessiu, així que s'ha optat per crear la funció `compare_iqr_factors()` per tal de visualitzar el percentatge de files eliminades per diversos factors multiplicatius del IQR (com es pot veure en la figura 5).

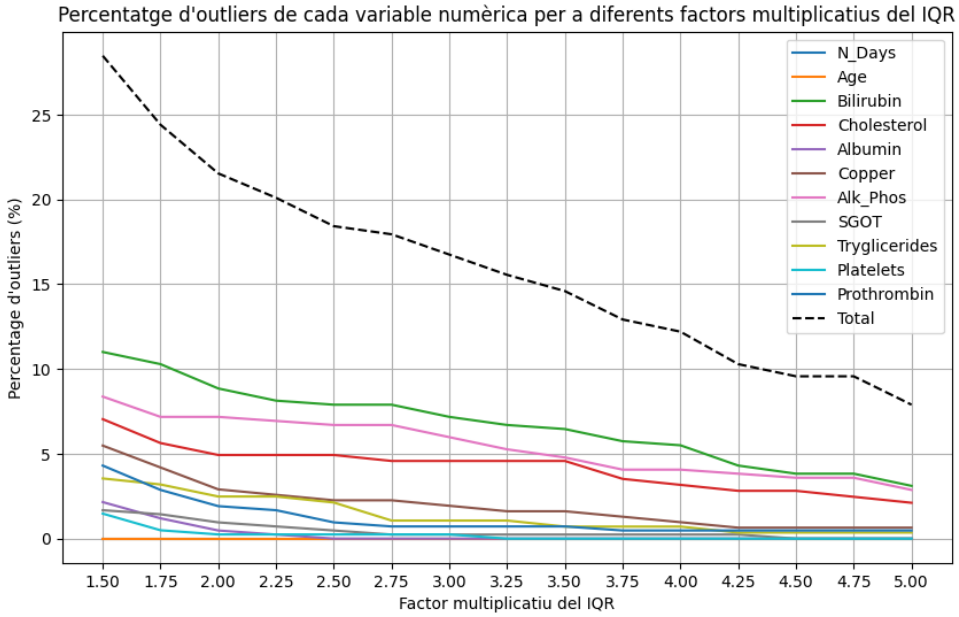
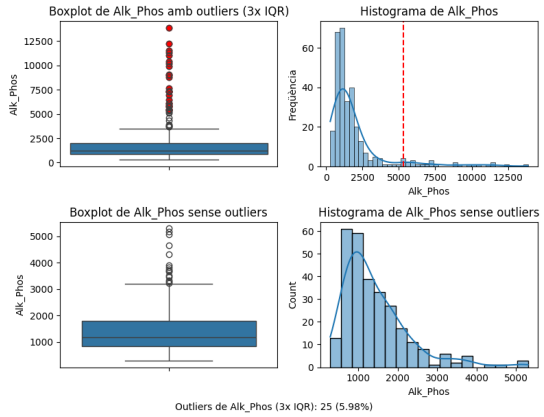
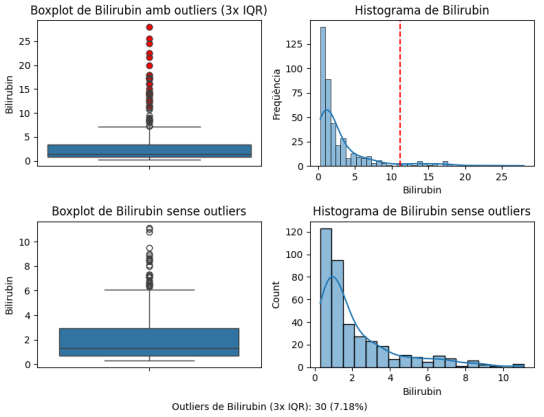


Figura 5: Percentatge de mostres considerades atípiques (outliers) en funció del factor multiplicatiu del IQR.

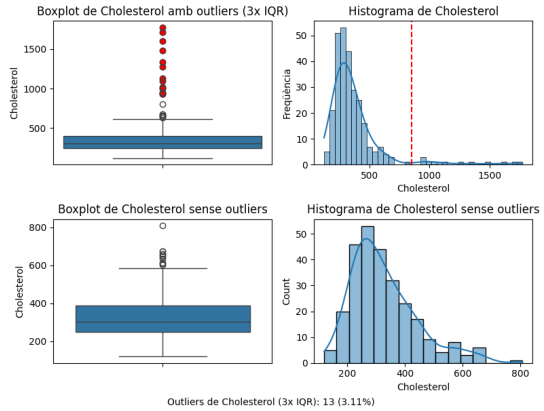
Observant els resultats d'aquesta funció, s'ha decidit escollir un factor de 3, eliminant només 70 mostres (16,75% del dataset). Aquesta selecció es justifica en les figures 6 i 7, on es poden veure els histogrames i boxplots de les variables numèriques amb abans i després de l'eliminació d'outliers. En els historgrames, hi ha línia que indica a partir d'on es considera que una mostra és un outlier tenint en compte el factor de multiplicació 3. Fàcilment es pot apreciar que es consideren outliers només les mostres que se surten de la principal distribució de les dades de cada variable, reafirmant així la nostra selecció del factor de multiplicació del IQR és bona.



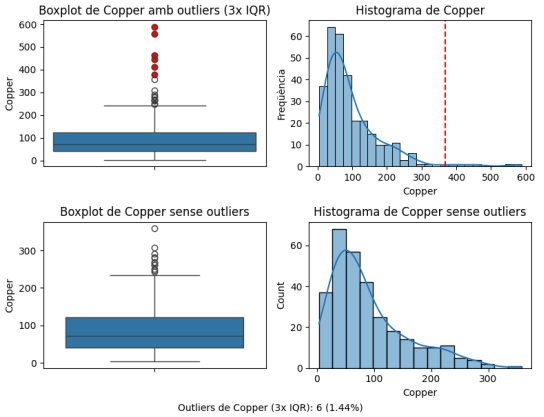
(a) Alk_phos



(b) Bilirubin

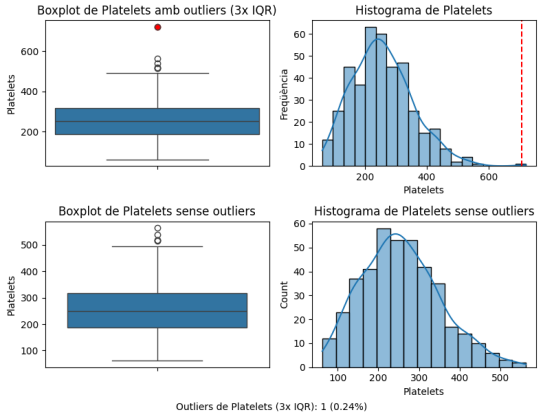


(c) Cholesterol

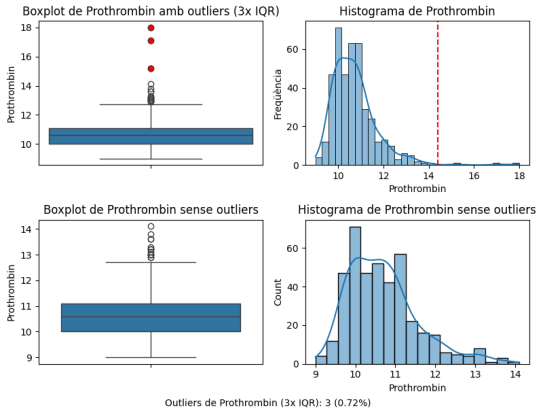


(d) Copper

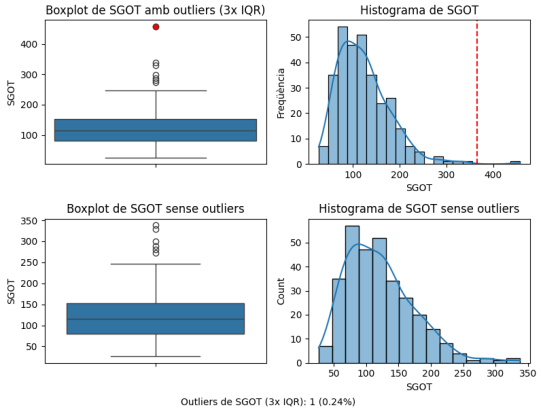
Figura 6: Distribució de dades de cada variable abans i després d'eliminar els outliers. La línia vermella discontínua dels histogramas indica a partir de on es consideren outliers les mostres. Els punts vermells dels boxplots representen outliers.



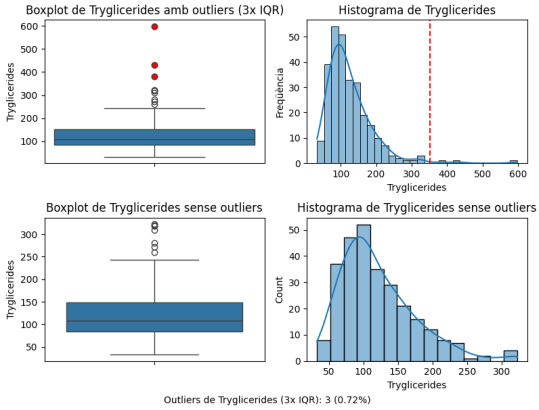
(a) Platelets



(b) Phtothrombin



(c) SGOT



(d) Tryglicerides

Figura 7: Distribució de dades de cada variable abans i després d'eliminar els outliers. La línia vermella discontínua dels histogramas indica a partir de on es consideren outliers les mostres. Els punts vermells dels boxplots representen outliers.

Hem de considerar que no sempre s'obté un benefici en rendiment dels models quan s'eliminen outliers. A més, al tractar-se d'un datset amb dades de pacients malalts, podria ser que els valors que es detecten com a atípics siguin realment importants perquè el model aprengui els patrons de les dades per poder predir la nostra variable objectiu *Status*. És a dir, segurament, totes les mostres del datset que contenen un nivell de colesterol extremadament alt tenen més probabilitat de morir, i eliminant outliers podem perdre part d'aquesta informació en certs casos. Per tant, l'eliminació d'outliers es valorarà per cada model més endavant, però sempre que es faci s'utilitzaran els criteris establerts en aquest estudi inicial de valors atípics (és a dir, s'utilitzarà el factor multiplicatiu del IQR de 3 mitjançant la funció `delete_outliers`).

Per altra banda, cal mencionar que també existeix la possibilitat d'establir els valors atípics com a missings i imputar-los posteriorment. No obstant, com ja es mencionarà més endavant, ja hi ha una gran quantitat de valors faltants (missings) en el dataset, de manera que introduir-ne més

només faria que hi hagués una proporció excessiva de valors imputats (que, evidentment, tenen una qualitat inferior als valors reals proporcionats en la pròpia base de dades).

3.4 Particionat del dataset

Per tal de poder entrenar i avaluar posteriorment els models, s'ha fet una partició del dataset en train i test. Degut a que disposem de poques mostres en la base de dades, s'ha optat per fer una partició de test de només el 15

Aquesta partició de la base de dades es fa en la funció `split_dataset` i cal mencionar que la partició de test no s'utilitzarà en cap moment per entrenar models, ni per calcular mitjanes de cap variable del train, etc. És a dir, en tot moment es respectarà la independència de tots dos conjunts.

Finalment, cal mencionar també que, per tenir una proporció igual de classes de la variable objectiu (*Status*) en tots dos conjunts, s'ha fet el perticionat mitjançant *stratify* (conserva la distribució de classes en tots dos conjunts).

3.5 Missings

Inicialment el dataset conté una gran quantitat de missings. En la taula 4 es pot veure que hi ha un total de 12 variables que contenen valors faltants. Observant el dataset inicial proporcionat, es pot veure que des de l'individu amb $ID = 313$ fins al final del dataset (l'últim individu té $ID = 418$) hi ha múltiples columnes que no tenen ni un únic valor (concretament, 9 columnes).

Variable	**Nº de missings**
Tryglicerides	136
Cholesterol	134
Copper	108
Drug	106
Ascites	106
Hepatomegaly	106
Spiders	106
SGOT	106
Alk_phos	106
Platelets	11
Stage	6
Prothrombin	2

Taula 4: Quantitat de valors faltants per cada variable amb almenys 1 missing.

El dataset té molt poques mostres (418, com hem mencionat) i això pot portar problemes a l'hora de que el model aprengui (mitjançant l'entrenament). Sabent això, s'ha decidit que l'opció d'eliminar totes les files que continguin valors faltants no és viable, ja que es reduiria dràsticament la quantitat

de mostres de la base de dades i no n'hi hauria suficients per tal de que els models trobessin bons patrons en la partició de train que farem més endavant. No obstant, per aquestes files concretes en què hi ha fins a 9 columnes amb valors faltants, s'ha implementat la funció `delete_last_rows()`, que les elimina del datset. Més endavant es faran proves per determinar si és útil eliminar aquestes files o és millor simplement imputar els valors.

Per imputar la resta de valors faltants s'han implementat les funcions `find_best_imputer()` i `impute_data()`. La funció `impute_data()` s'encarrega d'imputar els valors amb mètodes diferents (especificats en els paràmetres que introdueix l'usuari) per les variables numèriques i per les categòriques. Si es crida a la funció amb el paràmetre `numerical_imputer = 'best'`, es cridarà internament a la funció `find_best_imputer()` per trobar el millor imputador numèric. Per les variables categòriques succeeix el mateix, però amb el paràmetre `categorical_imputer = 'best'`.

Per altra banda, la funció `find_best_imputer()` implementa cross validation (sempre amb 5 particions/*folds*) per trobar el millor imputador, tant numèric com categòric. El paràmetre `X_train` és obligatori perquè, encara que s'estiguin intentant imputar els valors de la partició de test (que es menciona més endavant), s'escullen els millors mètodes d'imputació mitjançant només valors del train, per respectar totalment la independència entre les particions d'entrenament i de prova. El cross validation es fa sobre les mostres del train que no contenen missings i consisteix en generar artificialment valors faltants en cada *fold* per posteriorment imputar-los i comprovar la seva eficàcia. Els mètodes d'imputació que es proven són els següents:

- **Variables numèriques:**
  - **KNNImputer:** aquest imputador es prova amb diferents valors de l'hiperparàmetre k (1, 2, 3, 5, 10, 15, 20, 25, 50) i pot ser molt útil degut a que intenta imputar els valors basant-se en la mitjana dels valors dels deus k veïns més propers.
  - **SimpleImputer (mitjana):** aquest imputador és molt senzill. Simplement substitueix els valors faltants per la mitjana de la variable en què es trobi el valor missing. Generalment aquest mètode és pitjor que la resta, però s'ha volgut incloure igualment per si en algun cas proporciona un millor rendiment.
  - **IterativeImputer (Linear Regression):** aquesta tècnica d'imputació és més complexa que les anteriors. L'iterative imputer pretén imputar els valors de cada variable mitjançant combinacions lineals de les altres. En aquest cas, es crea un model de regressió lineal per cada variable amb les altres variables com a predictores (sense tenir en compte els valors missing) i es fa una predicció per cada valor faltant. Al estar utilitzant una regressió lineal, el model rendirà millor si hi ha més relacions lineals entre variables numèriques del dataset.
- **Variables categòriques:** Per les variables categòriques s'ha decidit utilitzar només classificadors, ja que altres mètodes no sempre funcionen per variables categòriques. S'ha de tenir en compte que en el moment d'imputar les dades encara no s'ha codificat el dataset (ja que codificar un dataset amb calors faltants és relativament complex i sovint genera errors). Per tant, les variables categòriques no poden ser utilitzades per algoritmes com ara KNN per tal de calcular les distàncies entre mostres. Per tant, predicció (imputació) de variables categòriques es fa, per cada variable categòrica, entrenant el classificador amb les variables numèriques.

- **KNeighborsClassifier:** aquest classificador obté resultats bastant diferents en funció de l'hiperparàmetre k. Em aquest cas, es proven els mateixos valors de k que en el KNNImputer utilitzat en les variables numèriques. Aquest classificador basa les seves prediccions en la moda (valor més freqüent) dels seus k veïns més propers.
- **DecisionTreeClassifier:** en aquest cas s'imputen els valors mitjançant prediccions d'un arbre de decisió. Aquest arbre de decisió realitzarà, per cada variable categòrica, diverses particions del dataset en conjunts intentant mantenir homogeneïtat en cada un d'ells i posteriorment imputarà el valor faltant mitjançant la classe més freqüent en el conjunt on es situï la mostra a predir. En el nostre cas provem dos models, un amb el criteri d'entropia (entropy) i l'altre amb gini.
- **RandomForestClassifier:** aquest classificador es crea múltiples arbres de decisió i combina les seves prediccions. Generalment, el rendiment d'aquest classificador és més alt que el d'un sol arbre de decisió. En el nostre cas, també es prova tant amb el criteri entropy com amb gini.

Tant els models de Decision Tree com els de Random Forest tenen bastants paràmetres que modifiquen el seu rendiment, especialment controlant si l'arbre fa més o menys overfitting. No obstant, s'ha decidit deixar els paràmetres per defecte per tal de no tenir una quantitat excessiva de models a provar en la funció `find_best_imputer()` i així reduir el cost computacional (i temps d'execució) d'aquesta funció. Més endavant, en el propi model de predicció de la variable objectiu mitjançant l'arbre de decisió, sí que es provaran diferents combinacions de paràmetres.

Pel que fa a l'elecció del millor model d'imputació, s'ha decidit que per les variables numèriques s'utilitzarà la mètrica de R^2 (coeficient de determinació). Un valor alt d'aquest coeficient significa que el model de imputació ha capturat un gran part de la variància dels valors reals, indicant que els valors predits s'assimilen als reals. S'ha escollit aquesta mètrica degut a que no depèn de l'escala de les dades, és robusta a outliers i és fàcilment interpretable.

Per altra banda, per avaluar l'imputador categòric s'ha decidit utilitzar la mètrica de *f1*, ja que considera tant la precisió com el recall. A més, utilitzem la versió *weighted* (ponderada) de la mètrica *f1* per tal de tenir en compte el desbalanceig de les classes i per garantir que no s'obté una millor puntuació simplement pel fet d'estar predint (imputant) la classe majoritària de la variable.

En general, com es mencionarà més endavant, tots els models predictors de la variable target (*Status*) també faran servir la mètrica de *f1-score-weighted* pels mateixos motius.

Finalment, cal mencionar que la imputació de valors faltants sempre es farà per separat en les particions de train i test, per evitar data leakage i mantenir la independència dels dos conjunts.

3.6 Recodificació de variables

Com ja hem mencionat, en els models d'imputació de dades no s'han pogut tenir gaire en compte les variables categòriques degut a que no es poden fer operacions si no estan representades com a valors numèrics. Per altra banda, en el preprocessament inicial ja s'han passat certs valors de variables categòriques a format numèric per tal de poder treballar-hi més fàcilment. No obstant,

ara que ja no hi ha valors faltants en el datset, pot ser molt útil realitzar una codificació de les variabels categòriques perquè els models de predicció de la variable objectiu (*Status*) les puguin tenir en compte.

Hi ha múltiples mètodes de codificació de variables categòriques, com ara One Hot Encoding, Label Encoding, Ordinal Encoding, etc. Inicialment es va plantejar l'us de One Hot Encoding, però aquest codificador augmenta molt la dimensionalitat del dataset, redueix l'eficàcia dels càlculs, no manté la naturalesa dels valors de certes variables i dificulta molt la interpretació de la base de dades (crucial a l'hora de generar gràfics, comprovar que no hi hagi errors de valors en el dataset, etc.). Amb tot això, s'ha decidit descartar aquest encoder.

Pel que fa al Label Encoding i Ordinal Encoding, tots dos són bastant similars en el sentit de que simplement etiqueten cada una de les classes de les variables categòriques amb un número. No obstant, s'ha decidit utilitzar Ordinal Encoding, ja que hi ha variables que tenen un ordre natural en elles (com ara els valors de *Stage*, que segueixen un ordre en funció de la fase en la que es trobi el pacient. És a dir, la fase 1 i la 3 estan "més lluny" que la 3 i la 4). Hi ha altres variables que no tenen un ordre en concret, però la majoria d'aquestes són binàries i no es veuran gaire afectades negativament pel fet de fer Ordinal Encoding. De fet, certs models com ara els arbres de decisió es poden veure beneficiats pel fet de tenir variables binàries (o que no tenen un ordre natural) expressades mitjançant Ordinal Encoding.

La codificació de variables es fa en la funció `encode_variables`. Més endavant es faran proves per veure si és necessari fer encoding en tots els models o només en alguns.

4 Preparació de variables

4.1 Normalització i escalat de variables

En les variables numèriques, s'ha considerat l'ús de dues tècniques diferents per a l'escalat de dades: *Standard Scaler* i *MinMax Scaler*. El *Standard Scaler* transforma les dades de manera que tinguin una mitjana de zero i una desviació estàndard d'una, mentre que el *MinMax Scaler* redimensiona les dades en un rang entre 0 i 1.

Tenint en compte la nostra base de dades i l'ús que farem de les varianbles numèriques (princpalment per trobar distàncies entre mostres, etc.), s'ha decidit que, en general, el *MinMax Scaler* és més útil o té més sentit per a la nostra aplicació. Aquesta decisió es basa en la manera en què el *MinMax Scaler* preserva la relació entre les variables originals (mantenint les mateixes distàncies relatives), sent particularment beneficiós quan les característiques estan en diferents escales.

La normalització o l'escalat pot millorar significativament el rendiment dels nostres models (KNN, arbre de decisió i SVM), ja que es basen en les distàncies entre les punts o en la definició de marges entre les classes. Per tant, si s'ha de realitzar algun escalat es farà *MinMax*. Arà bé, més endavant es valorarà per cada model si és millor realitzar ecalat o no.

Cal mencionar que escalat de dades es realitza sempre per separat en train i test, per tal de no tenir fuga de dades d'un conjunt a l'altre. A més, es fa abans d'imputar els valors faltants, ja que així els models d'imputació es poden beneficiar també dels avantatges de l'escalat de dades.

4.2 Anàlisi de correlacions entre variables numèriques

Per poder analitzar la correlació entre variables numèriques del dataset d'entrenament, s'ha fet una matriu de correlacions. No obstant, com ja s'ha mencionat, és possible que no s'utilitzi exactament la mateixa base de dades per entrenar tots els models. Per exemple, pot ser que en algun model es faci escalat de dades i en algun altre no, en algun s'eliminin outliers i en un altre no, etc..

Donada la gran quantitat de datasets d'entrenament que podem tenir en funció de les modificacions que hi fem, per realitzar la matriu de correlacions s'ha decidir fixar una llavor (*seed*) per poder replicar els experiments i s'ha tingut en compte el dataset d'entrenament reultant a l'aplicar els següents canvis:

- Llavor (random_state) establerta arbitràriament al valor 42 en tots els processos que tinguin aleatorietat.
- Eliminació d'oultliers (amb factor multiplicatiu del IQR de 3).
- Sense eliminar les files amb 9 missings.
- Sense codificació de variables categòriques (això no hauria d'influir en les correlacions de variables numèriques, però s'especifica per si de cas).

- Escalat de variables numèriques mitjançant *MinMax*.
- Missings imputats mitjançant els imputadors amb millors resultats amb la llavor escollida (mitjançant la funció `find_best_imputer()`) i determinats mitjançant la mètrica de R^2 (per variables numèriques) i *f1-score-weighted* (per variables categòriques). Amb tot això, per a variables numèriques ha resultat ser millor l'IterativeImputer (amb Lineal Regression com a estimador), mentre que per a variables categòriques ha sigut el RandomForestClassifier amb el criteri gini.
- Sense realitzar cap mètode de balanceig de dades (explicats més endavant).

Tenint en compte totes aquestes consideracions, la matriu de correlacions entre variables numèriques resultant és la que es pot veure en la figura 8.

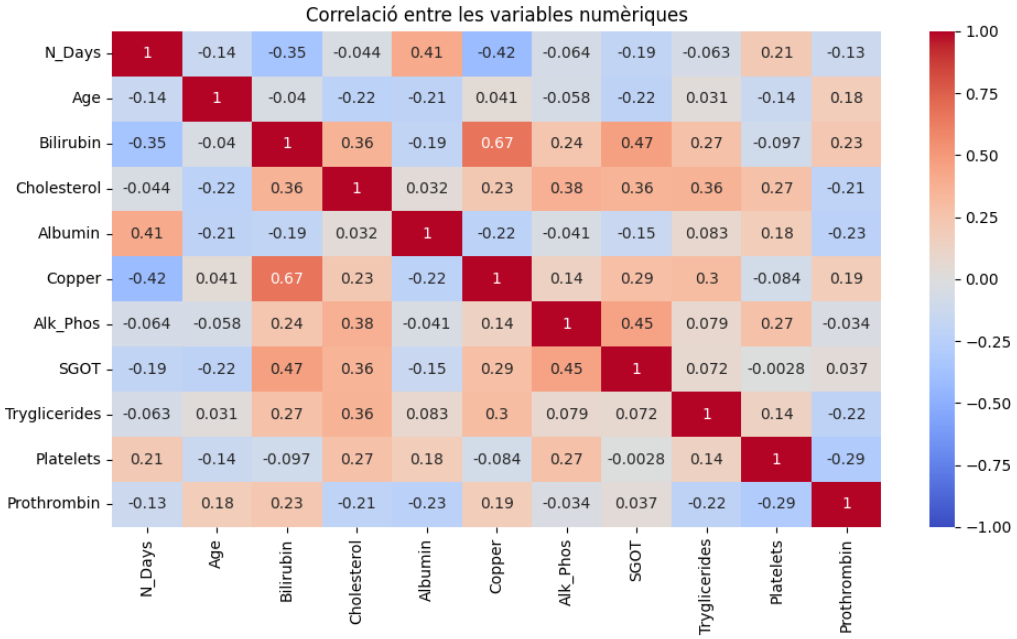


Figura 8: Matriu de correlacions entre totes les parelles de variables numèriques.

Cal mencionar que la matriu de correlacions que es pot observar a la figura 8 ens diu el coeficient de correlació per cada parella de variables, basant-se només en relacions lineals entre elles. Per tant, és possible que algunes de les variables numèriques tinguin clares relacions no lineals, però això no es pot apreciar en aquesta matriu.

La correlació entre *Bilirubin* i *Copper* és de 0,67, la qual cosa indica una forta correlació positiva. Això significa que a mesura que el valor de *Bilirubin* augmenta, el valor de *Copper* també tendeix a augmentar.

Bilirubin i *SGOT* tenen una correlació positiva de 0,47, també bastant forta; i *SGOT* i *Alk_Phos* de 0,45.

Copper i *N_Days* destaquen pel que fa a les correlacions negatives, amb un coeficient de $-0,42$.

Finalment, la variable *Cholesterol* té fins a 3 coeficients de correlació amb valors de 0,38, 0,36 i 0,36 amb les variables *Alk_Phos*, *SGOT* i *Tryglicerides*, respectivament.

Una vegada analitzades les correlacions més fortes, podem veure que les variables *Bilirubin*, *Copper*, *SGOT* i *Alk_Phos* són les que es troben en més correlacions més fortes entre variables.

És important destacar que una correlació no implica causalitat. Això significa que, encara que dues variables estiguin correlacionades, no es pot assegurar que el valor d'una sigui causant del valor de l'altre. Per tant, no podem dir que el valor d'aquestes tres variables depengui dels altres, però en el nostre dataset d'entrenament es dona la casualitat (o potser causalitat) de que hi ha una alta correlació entre aquestes variables i altres variables del dataset.

Sabent tot això, si s'hagués d'eliminar alguna variable, les millors opcions per variables numèriques serien alguna de les 4 que hem mencionat, ja que part de la informació que aporten ja està explicada per altres variables. No obstant, s'ha considerat que, per la tasca de predicció de la variable objectiu i tenint en compte les dimensions del dataset, no cal eliminar cap variable numèrica per motiu de la seva correlació amb altres variables numèriques.

4.3 Anàlisi de variables categòriques i variable objectiu

Com que l'objectiu d'aquest projecte és acabar predint el valor de la variable *Status*, és molt important veure com influeix el valor de la variable objectiu en cada variable categòrica.

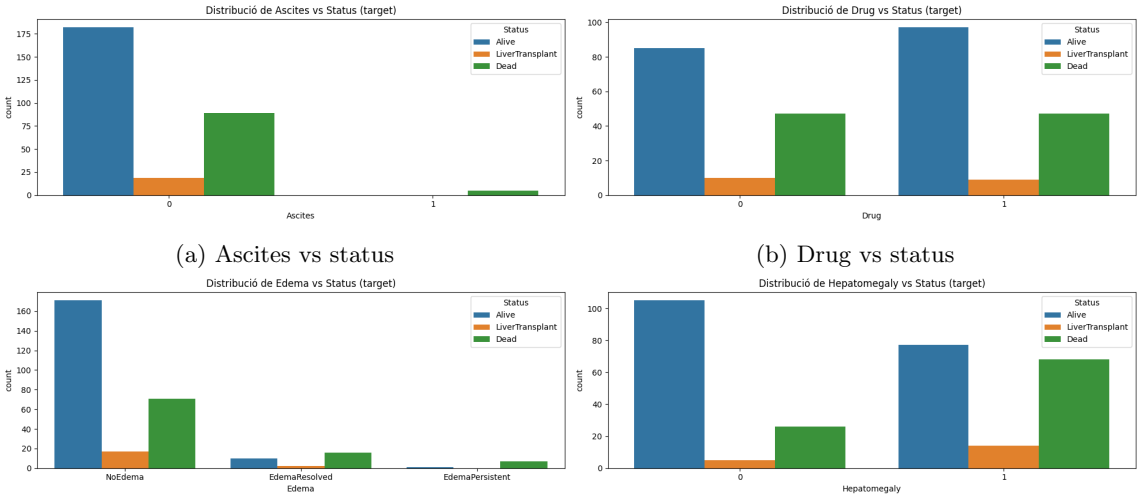
Abans de res, cal mencionar que aquestes relacions han estat estudiades amb la mateixa base de dades d'entrenament que en l'apartat anterior. És a dir, aplicant exactament les mateixes modificacions al dataset inicial per tal de tenir el mateix conjunt de train. En funció del model que s'utilitzi posteriorment, aquest conjunt d'entrenament pot variar lleugerament.

En les figures 9 i 10 es poden veure aquestes relacions. Si entre els les diferents classes de la variable categòrica hi ha una distribució similar dels valors de la variable *Status*, això ens indica que aquella variable categòrica no té gaire influència en la variable objectiu. Aquest és el cas de variables com *Drug*.

Per altra banda, es pot veure que no hi ha cap variable amb una gran influència cap a la variable objectiu. A més, es destaca el desbalanceig de la variable objectiu, que tractarem a continuació.

Amb tot el que s'ha vist, podem dir que si fos necessari eliminar una variable categòrica de l'entrenament, es podria eliminar la variable *Drug*, ja que sembla ser la que menys influeix en la variable objectiu. No obstant, s'ha considerat que no és necessari eliminar-la degut a les característiques del nostre dataset.

És interessant observar que, si ens parem a pensar en la conclusió d'aquest anàlisi, s'ha mencionat que la variable *Drug* (que originalment tenia valors 'D-penicillamine' i 'Placebo', i al preprocessing inicial s'han reemplaçat per 1 i 0, respectivament) no té pràcticament influència en la variable *Status* (que indica si un pacient ha sobreviscut o no a la malaltia). Per tant, això ens està indicant que el tractament que es va utilitzar per realitzar aquest estudi sobre la cirrosis hepàtica no era efectiu. És a dir, els pacients que se'ls proporcionava el tractament (*Drug* = 'D-penicillamine', ara modificat a *Drug* = 1) tenien les mateixes possibilitats de morir/sobreviure que els pacients que se'ls proporcionava un simple placebo.



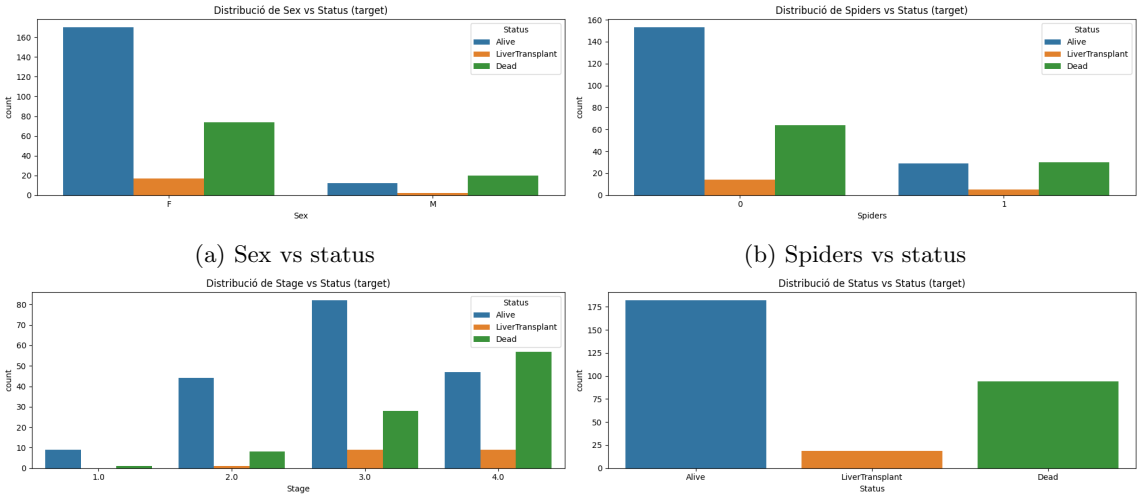
(a) Ascites vs status

(b) Drug vs status

(c) Edema vs status

(d) Hepatomegaly vs status

Figura 9: Relació entre les variables categòriques i la variable objectiu (*Status*)



(a) Sex vs status

(b) Spiders vs status

(c) Stage vs status

(d) Status vs status

Figura 10: Relació entre les variables categòriques i la variable objectiu (*Status*)

4.4 Balanceig de classes de la variable objectiu

Com ja hem vist en la figura 10d, la variable objectiu (*Status*) està clarament desbalancejada. Hi ha moltes mostres amb valor 'Alive', unes quantes amb valor 'Dead' i molt poques amb valor 'LiverTransplant'.

Això pot causar que els nostres models es dediquin a predir la classe majoritària i tinguin molt mal rendiment predient valors de la classe minoritària. Per poder solucionar-ho, s'han implementat diversos mètodes de balanceig de dades en la funció `balance_target_classes()` per aplicar al conjunt d'entrenament. En concret, s'ha implementat Oversampling, Undersampling, SMOTE i SMOTEENN.

Degut a la poca quantiat de dades del nostre dataset, ja podem saber que no té cap mena de sentit intentar aplicar Undersample, ja que ens quedaríem amb molt poques dades en el conjunt d'entrenament.

El mètode de Oversampling pot ser útil degut a la seva senzillesa. Simplement replica mostres del datset fins que les classes estan equilibrades. No obstant, s'ha d'anar amb compte amb aquest mètode perquè pot provocar overfitting.

El mètodes de SMOTE consisteix en generar mostres sintètiques basant-se en combinacions lineals dels veïns més propers. Aquest mètode és computacionalment més car, però redueix el risc de sobreajustament (overfitting) en comparació al Oversample.

Finalment, el mètode SMOTEENN es basa en un SMOTE, però posteriorment elimina aquelles mostres que considera de pitjor qualitat. Aconsegueix un menor balanceig de classes però assegura

una millor qualitat de les mostres generades sintèticament.

Tots aquests mètodes es provaran més endavant per cada model i es determinarà quin és el que proporciona millors resultats per a cada un. Cal mencionar que tant SMOTE com SMOTEENN requereixen que les dades hagin estat codificades, ja que no poden tractar valors que no siguin numèrics. Així doncs, quan el dataset no ha estat codificat, no es podrà aplicar cap d'aquests dos mètodes.

4.5 Eliminació de variables

Pel que fa a la eliminació de possibles variables, ja s'ha mencionat en apartats anteriors que no s'ha considerat necessari en aquest projecte. El dataset té una mida relativament petita i no hi ha variables extremadament redundants o sorolloses. Per tant, eliminar variables no és necessari.

En cas que es volgués eliminar alguna variable numèrica, ja s'ha mencionat que les millors opcions serien *Bilirubin*, *Copper*, *SGOT* o *Alk_Phos*. Per altra banda, en les variables categòriques s'optaria per eliminar *Drug*.

4.6 Estudi de dimensionalitat (ACP)

En aquest apartat s'ha realitzat un estudi sobre la dimensionalitat de la base de dades d'entrenament mitjançant l'Anàlisi de Components Principals (ACP o PCA).

En la figura 11 es pot veure el percentatge de variància explicada en funció del nombre de components principals. Es pot veure que amb els 7 primers components principals ja s'explica un 80% de la variància. Per tant, en cas que es desitgés fer una reducció de la dimensionalitat, amb tan sols 7 components principals (en comptes de les 11 variables numèriques que hi ha ara) es podria explicar un 80% de la variància, que generalment es considera suficient per no perdre informació rellevant i fins i tot eliminar soroll o redundància.

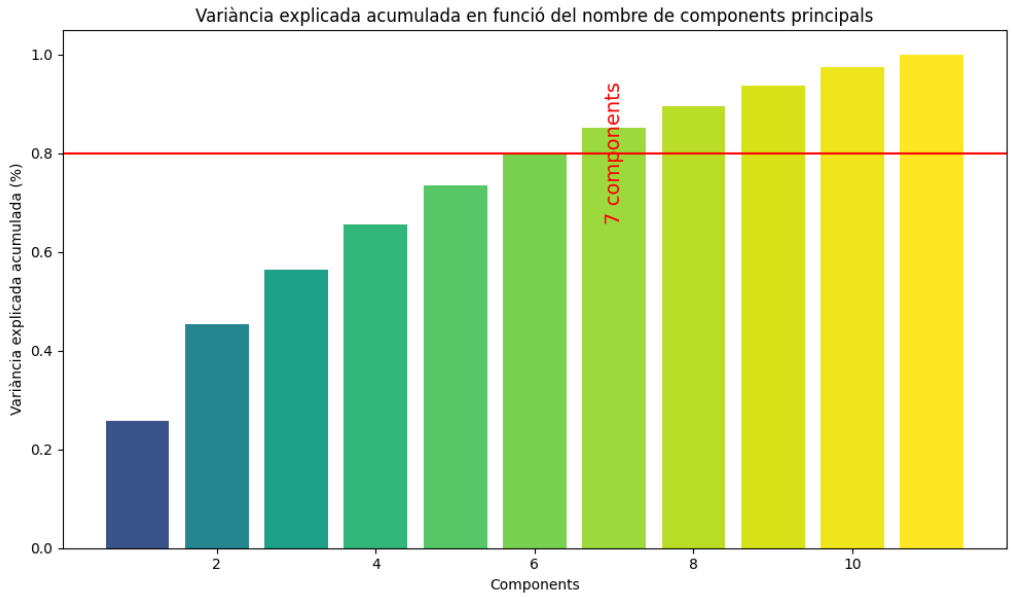


Figura 11: Percentatge de variància explicada en funció del nombre de components principals.

En la figura 12 es poden veure les projeccions de les variables numèriques en els dos primers eixos principals. El primer eix principal explica un 25,77% de la variància i el segon un 19,64% de la variància, de manera que tenim aproximadament un 45% de la variància explicada. Amb això, les conclusions que extraiem d'aquí s'han de tractar amb molt de compte, ja que no seran conclusions extretes de tota la informació del dataset, sinó d'aproximadament la meitat. Es pot apreciar, com ja hem mencionat en l'anàlisi de correlacions entre variables numèriques, que les variables *Bilirubin*, *Copper* i *SGOT* estan bastant relacionades, i en aquest cas veiem que es projecten bastant sobre el primer eix principal.

Per tant, el primer eix principal es caracteritza princpalment per els valors positius de *Bilirubin*, *Copper*, *SGOT* i lleugerament *Cholesterol*; i els valors negatius (relació inversa) lleugerament amb *N_Days*.

Pel que fa al segon eix principal, veiem que hi té projectades positivament *Age* i *Prothrombin*; i negativament sobretot *Platelets*, però també lleugerament *Cholesterol* i *Albumin*. Amb la informació del segon eix, es podria dir que el valor de l'edat té una relació inversa amb el de *Platelets*; és a dir, la gent més gran tendeix a tenir un valor de *Platelets* més petit.

Resumint, es podria dir que el primer eix principal conté sobretot característiques sobre les dades mesurades del pacient, mentre que el segon component ve determinat especialment per l'edat i les plaquetes (platelets) del pacient.

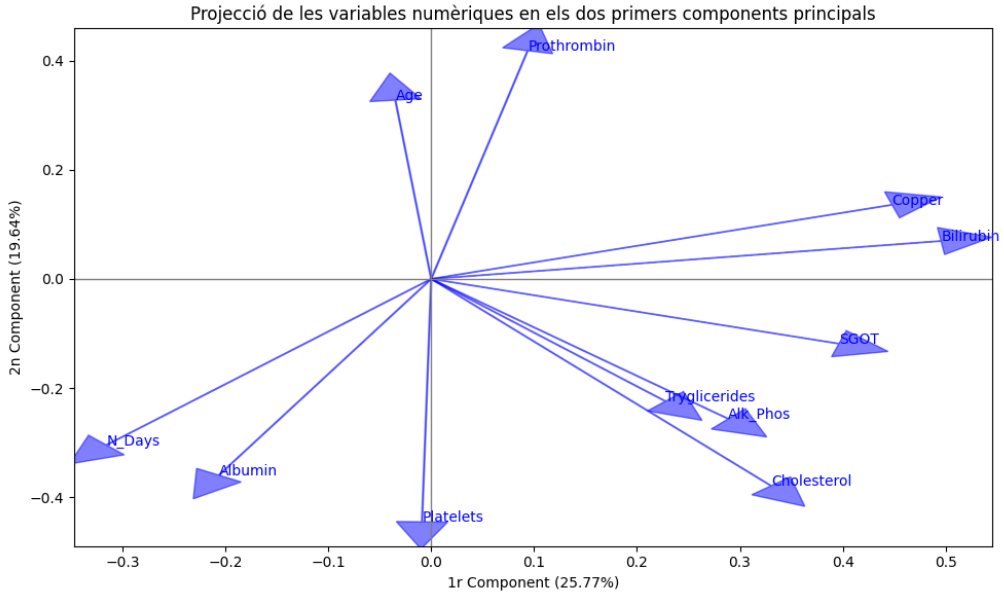


Figura 12: Projecció de les variables numèriques en els dos primers components principals.

Finalment, en la figura 13 es pot veure la projecció dels centroides de les classes de les variables categòriques en els dos primers eixos principals. S'hi poden veure les classes de la variable target (*Status*) clarament diferenciades. A més, hi ha curiositats com que el sexe masculí (M) es troba més aprop de 'Dead' que el femení (F), indicant-nos que els homes tendeixen més a morir (*Status* = 'Dead') que les dones. També podem veure com el fet de tenir *Spiders* o tenir *Hepatomegaly* "apropa" al pacient cap a 'Dead' (és a dir, en mitjana, és més probable que mori). Finalment, es pot mencionar la clara tendència de *Stage* a apropar-se a 'Dead' a mesura que augmenta el seu valor (com és lògic, en etapes més avançades de la malaltia és menys probable sobreviure).

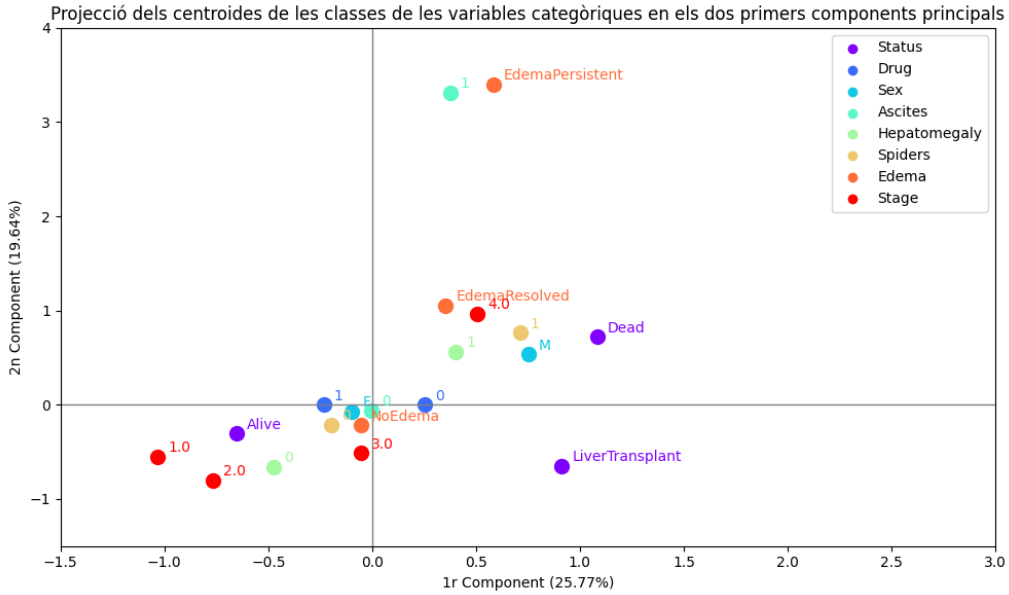


Figura 13: Projecció dels centroides de les classes de les variables categòriques en els dos primers components.

Ajuntant la informació de les variables categòriques i de les numèriques (figures 12 i 13) podem dir que les variables numèriques que tenen més relació positiva amb la 'Dead' són *Copper* i *Bilirubin*, mentre que *Albumin* i *N_Days* tenen una relació positiva amb 'Alive'. És a dir, tenen més probabilitats de sobreviure els pacients que tenen valors baixos de *Copper* i *Bilirubin* i valors alts de *Albumin* i *N_Days*.

Com ja s'ha mencionat, aquestes conclusions no poden interpretar-se al peu de la lletra, ja que estan fetes sobre un 45% de la variància explicada (i no pas un 100%). No obstant, poden donar-nos unes nocions sobre les relacions entre variables, tant numèriques com categòriques com amb *Status*.

En cas de que es desitgés fer una reducció de dimensionalitat per treballar amb components, les avantatges serien que hi hauria menys dimensions i seria més fàcil gestionar les dades. No obstant, s'ha de considerar que, com acabem de veure, es perd molta interpretabilitat i cada component passa a ser una barreja de conceptes i valors difícilment interpretable.

5 Definició de models

Per la predicció de la variable objectiu (*Status*) s'han realitzat 3 tipus diferents de models: un K-Nearest Neighbors (KNN), un Arbre de Decisió (Decision Tree) i un Support Vector Machine (SVM).

En primer lloc, cal mencionar que tots aquests models treballen amb dades numèriques, de manera que requereixen que les dades categòriques hagin estat codificades. No obstant, en la implementació que s'ha fet, es contempla el cas en què les variables categòriques no han estat codificades i llavors el model prediu només amb les variables categòriques. Això ens permet poder fer proves amb i sense codificar les dades, ja que en certs models i datasets potser s'obté millor rendiment obviant les variables numèriques en la predicció.

Per cada un dels models, es definiran uns paràmetres generals fixes (els que generalment venen per defecte en els predictors de `scikit-learn`) i es provaran totes les possibles combinacions de modificacions al dataset (codificar o no, eliminar o no outliers, escalar variables o no, balancejar classes amb un mètode o amb un altre, etc.) mitjançant la funció `find_best_dataset()`. En cada una d'aquestes combinacions s'entrena el model i es fa la predicció, obtenint els resultats. El dataset (combinació de modificacions al dataset inicial) que aconsegueixi millors resultats (predient en el test) s'establirà com el "millor dataset pels models d'aquell tipus".

Una vegada determinat el millor dataset per cada tipus de model, es provaran moltes combinacions de paràmetres en les funcions `X_prediction()` (on `X` serà `knn`, `dt` o `svm`, depenent del model) mitjançant cross validation en la partició de train. Els que donin millors resultats de mitjana en la validació s'utilitzaran per predir en el test i es determinarà com el "model definitiu d'aquell tipus" (ja que s'haurà entrenat amb un dataset específic per al model en concret i s'hauran trobat els paràmetres que rendeixen millor).

Evidentment, les proves es podrien fer directament sempre al test (no a la partició de validació) i també provar una major quantitat de paràmetres i combinacions de datasets dels que es provaran. No obstant, es considera que una de les parts fonamentals d'aquest projecte consisteix en saber filtrar què pot funcionar i què no, en comptes de provar-ho tot a la força bruta. És a dir, encara que els paràmetres que s'escullin en el validation no acabin sent els millors pel test (ja que poden estar donant mètriques més altes en el validation degut a overfitting), no és una solució eficient (ni en termes de computació, ni de temps, ni de escalabilitat, etc.) provar tots aquests paràmetres en el test directament. A més, les particions de validació estan fetes precisament per ajudar a escollir paràmetres.

Una vegada mencionat això, quan ja es tinguin els 3 "models definitius", es compararan els resultats per escollir el millor dels 3 models i procedir al seu anàlisi més exhaustiu, model card, etc.

Totes aquestes proves o execucions es realitzaran amb la llavor (seed o random_state) amb valor 42 arbitràriament per una millor reproductibilitat. És evident que s'obtindrien millors resultats realitzant l'experiment amb múltiples llavors i escollint els paràmetres que més es repetissin en les diferents execucions (per així reduir el factor aleatori), però cada una d'aquestes execucions té una durada aproximada de 20 minuts, de manera que no és viable perdre-hi tant de temps. No obstant, si s'haguessin d'implementar aquests models en un cas real, seria molt important afegir

més execucions amb llavors diferents per poder tenir resultats més robustos al factor aleatori.

A continuació es detallen els diferents tipus de models, els paràmetres provats, el conjunt de modificacions del dataset adient per cada un, les mètriques utilitzades, etc.

5.1 K-Nearest Neighbors (KNN)

5.1.1 Motivació

El nostre dataset conté dades relacionades amb pacients amb cirrosi hepàtica, on l'objectiu principal és predir l'estat del pacient (variable *Status*). L'algorisme *K-Nearest Neighbors* (KNN) es presenta com una elecció adequada per aquesta tasca per diverses raons.

En primer lloc, KNN és un algorisme basat en instàncies, el que significa que fa prediccions basant-se en la proximitat i similitud de les mostres en l'espai de característiques. Això és particularment útil en el nostre cas, on les característiques dels pacients com l'edat, sexe, indicadors bioquímics i la resposta al tractament poden influir directament en el seu estat de salut. La capacitat de KNN per capturar aquestes relacions espacials i fer prediccions sense la necessitat de que hi hagi patrons explícits el fa un bon candidat per conjunts de dades amb relacions complexes i no lineals entre les característiques i la variable objectiu.

A més, la naturalesa intuïtiva i la facilitat d'interpretació de KNN són avantatges significatius quan es tracta de dades mèdiques. La possibilitat de explicar les prediccions en termes de "pacients semblants"(*Nearest Neighbors*) pot ser molt valuosa en l'àmbit mèdic, on la comprensió i la confiança en el model són també de gran importància.

5.1.2 Mètriques

Per l'avaluació del model, s'ha utilitzat la mètrica de *f1-score-weighted*, tal i com s'ha dit que es faria en tots els models (inclosos els de imputació explicats en apartats anteriors). Aquesta mètrica és calcula fent la mitjana harmònica entre la *precision* i el *recall*, penalitzant els valors extrems (calen valors bons tant de *precision* com de *recall* per tal de tenir un bon *f1*). La mitjana de valors de totes les prediccions es fa ponderada (*weighted*), de manera que a cada classe se li dona una importància proporcional a les seves aparicions. D'aquesta manera, es puntua millor que s'aconsegueixi classificar correctament la classe majoritària (on s'haurà de fer més prediccions).

Si en un altre experiment es desitgés predir donar molta importància la classe minoritària (en aquest cas "LiverTransplant", que no és tant rellevant en aquest estudi com "Alive" o "Dead") es podrien utilitzar altres mètriques que ho tinguessin més en consideració.

5.1.3 Hiperparàmetres

L'únic hiperparàmetre que s'ha modificat és la k (`n_neighbors` en la implementació de `scikit-learn`), que indica el nombre de veïns més propers que es consideren a l'hora de fer la predicció. Els valors que s'han provat són 1, 2, 3, 5, 10, 15, 20 ,25 i 50.

5.1.4 Millor conjunt de dades

Per trobar el conjunt de dades que proporciona un millor rendiment a aquest model, s'han fixat els hiperparàmetres a $k = 3$ i s'ha executat la funció `find_best_dataset()`. Després que es provin totes les combinacions possibles de modificacions al dataset inicial (un total de 64 combinacions), la que ha obtingut millors resultats de *f1-score-weighted* en el test és la següent:

- No eliminar outliers.
- Eliminar les files amb més de 9 NaNs.
- Escalar les variables numèriques amb *MinMax*.
- Imputar variables numèriques amb KNN Imputer (amb $k = 15$) i les categòriques amb Random Forest Classifier (amb el criteri 'gini'). Els resultats de la imputació (calculats tal i com s'ha explicat en l'apartat 3.5) han estat de 0,3808 de mitjana entre la puntuació R^2 de les numèriques i *f1-score-weighted* de les categòriques.
- No codificar les variables categòriques.
- No balancejar les classes de la variable objectiu.

Amb aquestes modificacions i els paràmetres fixes que s'han mencionat, s'ha aconseguit un resultat de 0,7838 de *f1-score-weighted* en el conjunt de prova (test), que es pot considerar un resultat bastant bo. La partició de train conté 277 mostres i la de test 50. A partir d'aquí s'ha considerat que aquests són els millors conjunts de dades per els models KNN, i el següent pas serà trobar els millors paràmetres per el model en aquest dataset.

5.1.5 Millors hiperparàmetres pel dataset escollit

Una vegada determinat el millor dataset per els models KNN, cal provar quins són els millors paràmetres. Això es fa mitjançant la funció `run_experiment()` que cridarà a `knn_prediction()`. En aquesta segona funció s'implementa cross validation amb 5 *folds* per cada possible combinació de paràmetres. Una vegada s'hagin provat tots els paràmetres, la combinació de paràmetres que tingui un millor *f1-score-weighted* de mitjana en la partició de validació (és a dir, el que obtingui una millor puntuació en la validació creuada) es determinarà com la millor per el model KNN i s'executarà la predicció en el test.

Una vagada s'ha fet la l'execució, s'ha obtingut que el millor paràmetre és $k = 25$, amb una puntuació en la partició de validació de 0,6287.

5.1.6 Resultats

Amb el dataset escollit i els millors hiperparàmetres triats, la predicció en el conjunt ha obtingut les mètriques que es poden veure en la taula 5

Metric	**Value**
accuracy	0.76
f1-micro	0.76
f1-macro	0.5228251507321274
f1-weighted	0.736537677002583
precision-micro	0.76
precision-macro	0.5075757575757576
precision-weighted	0.7145454545454545
recall-micro	0.76
recall-macro	0.539072039072039
recall-weighted	0.76

Taula 5: Mètriques obtingudes en l'execució de KNN en la partició de test amb el millor dataset pel model i els millors paràmetres ($k = 25$).

Si ens fixem en la mètrica que hem escollit (*f1-score-weighted*), podem veure que s'ha obtingut un valor de 0,7365, que està bastant bé. Aquest valor és més alt del que s'ha obtingut en la partició de validació. Això segurament succeeix per un factor aleatori.

Pel que fa a les prediccions, en la figura 14 es pot veure la matriu de confusió d'aquest model en les prediccions del conjunt de test. Es pot apreciar que la majoria de prediccions són bones. No obstant, hi ha hagut 5 mostres 'Alive' que s'han predit com a 'Dead' i 4 mostres de 'Dead' que s'han predit com 'Alive', de manera que el model no és perfecte i en alguns casos s'equivoca. Pel que fa a la classe 'LiverTransplant', hi ha molt poques mostres (només 3) i el model no ha sigut capaç de predir-les correctament, ja que les ha predit com a 'Alive'. Tot i això, cal considerar que realment els pacients que ha tingut un transplant de fetge ('LiverTransplant') han acabat sobrevivint... de manera que aquest error en la predicció és menys greu que els de les altres classes.

Per poder extreure més conclusions sobre les prediccions de 'LiverTransplant' seria útil conèixer en quines condicions una persona pot acabar rebent un transplant de fetge. És a dir, el pacient ha de complir unes condicions de salut per tal de que es faci el transplant o simplement cal que s'aconsegueixi un donador? Això podria ajudar a determinar millor si és millor que els valors que nos s'han predit correctament de 'LiverTransplant' es prediguin com a 'Alive' o com a 'Dead'.

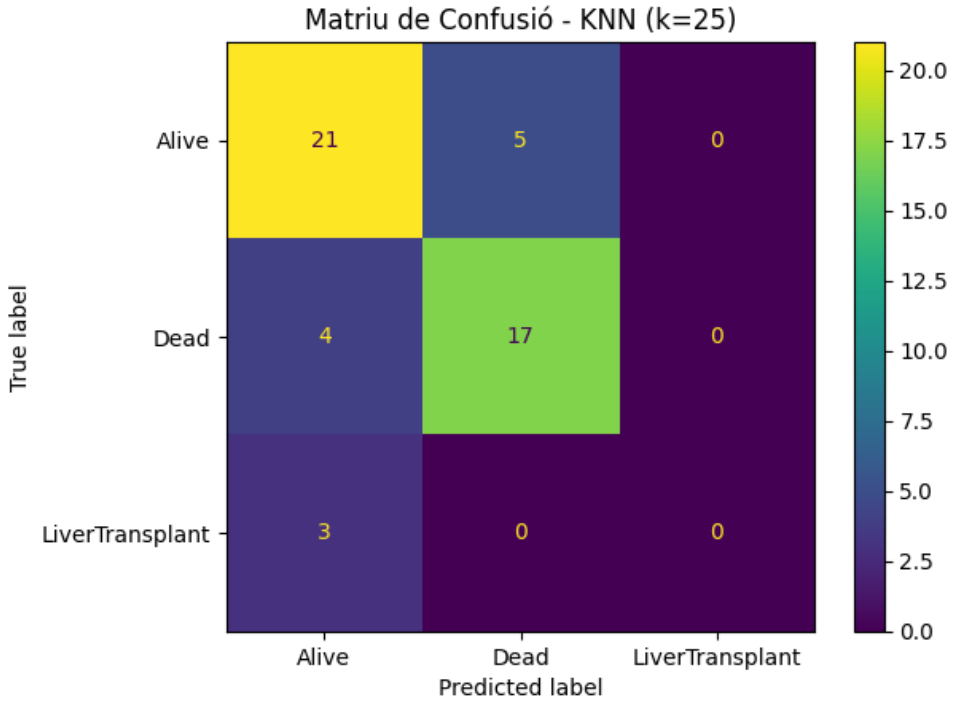


Figura 14: Matriu de confusió de les prediccions del KNN (amb paràmetres $k = 25$) en el conjunt de test.

Tot i el desbalanceig de classes, el model és aproximadament igual de bo predint la classe 'Alive' (amb una proporció de $\frac{21}{26} \approx 0.8077$ prediccions correctes en aquesta classe) que la classe 'Dead' (amb una proporció de $\frac{17}{21} \approx 0.8095$ prediccions correctes en aquesta classe).

5.2 Arbre de Decisió

5.2.1 Motivació

El nostre dataset presenta dades complexes i diverses sobre pacients amb cirrosi hepàtica, on la tasca principal és predir l'estat del pacient (*Status*). L'ús d'un arbre de decisió per a aquesta finalitat pot ser útil degut a que són models no lineals que poden capturar interaccions complexes entre les variables. Això és particularment útil en el nostre cas, ja que la condició dels pacients pot estar influenciada per una combinació de factors com ara l'edat, el sexe, els indicadors bioquímics, etc. . Un arbre de decisió pot dividir l'espai de les dades en subconjunts basats en aquestes característiques, facilitant la comprensió de com aquests factors interactuen i afecten el *Status* del pacient.

5.2.2 Mètriques

De la mateixa manera que s'ha fet per al KNN, en l'avaluació del model s'ha utilitzat la mètrica de *f1-score-weighted*. Aquesta mètrica és calcula fent la mitjana harmònica entre la *precision* i el *recall*, penalitzant els valors extrems (calen valors bons tant de *precision* com de *recall* per tal de tenir un bon *f1*). La mitjana de valors de totes les prediccions es fa ponderada (*weighted*), de manera que a cada classe se li dona una importància proporcional a les seves aparicions. D'aquesta manera, es puntua millor que s'aconsegueixi classificar correctament la classe majoritària (on s'haurà de fer més prediccions).

Si en un altre experiment es desitgés predir donar molta importància la classe minoritària (en aquest cas "LiverTransplant", que no és tant rellevant en aquest estudi com "Alive" o "Dead") es podrien utilitzar altres mètriques que ho tinguessin més en consideració.

5.2.3 Hiperparàmetres

Pel que fa als hiperparàmetres de l'arbre de decisió, s'han modificat el criteri de decisió, la profunditat màxima, el nombre mínim de mostres per cada fulla i el nombre mínim de mostres per dividir un conjunt de l'arbre.

Els valors que es provaran per cada un dels paràmetres són els següents:

- **Criteri**: 'entropy' i 'gini'.
- **Max_depth**: None, 2, 3, 5, 10, 20, 50.
- **Min_samples_leaf**: 1, 2, 3, 5, 10, 20 ,50.
- **Min_samples_split**: 2, 3, 5, 10, 20, 50.

Totes les possibles combinacions d'aquests paràmetres es provaran mitjançant Grid Search i cross validation per trobar els que millor rendiment aporten al model.

5.2.4 Millor conjunt de dades

Per trobar el conjunt de dades que proporciona un millor rendiment a aquest model, s'han fixat els següents hiperparàmetres:

- **Criteri**: 'entropy'.
- **Max_depth**: None.
- **Min_samples_leaf**: 1.

- **Min_samples_split**: 2.

Amb aquests paràmetres fixats, s'ha executat la funció `find_best_dataset()`. Després que es provin totes les combinacions possibles de modificacions al dataset inicial (un total de 64 combinacions), la que ha obtingut millors resultats de *f1-score-weighted* en el test és la següent:

- No eliminar outliers.
- Eliminar les files amb més de 9 NaNs.
- No escalar les variables numèriques.
- Imputar variables numèriques amb Iterative Imputer (amb Lineal Regression com a estimador) i les categòriques amb Random Forest Classifier (amb el criteri 'gini'). Els resultats de la imputació (calculats tal i com s'ha explicat en l'apartat 3.5) han estat de 0,3581 de mitjana entre la puntuació R^2 de les numèriques i *f1-score-weighted* de les categòriques.
- Codificar les variables categòriques amb *Ordinal Encoding*.
- Balancejar les classes de la variabe objectiu mitjançant *Oversampling*.

Amb aquestes modificacions i els paràmetres fixes que s'han mencionat, s'ha aconseguit un resultat de 0,7849 de *f1-score-weighted* en el conjunt de prova, que es pot considerar un resultat bastant bo. La partició de train conté 441 mostres i la de test 50. A partir d'aquí s'ha considerat que aquests són els millors conjunts de dades per els models d'Arbre de Decisió, i el següent pas serà trobar els millors paràmetres per el model en aquest dataset.

5.2.5 Millors hiperparàmetres pel dataset escollit

Una vegada determinat el millor dataset per els models d'Arbre de Decisió, cal provar quins són els millors paràmetres. Això es fa mitjançant la funció `run_experiment()` que cridarà a `dt_prediction()`. En aquesta segona funció s'implementa cross validation amb 5 *folds* per cada possible combinació de paràmetres. Una vegada s'hagin provat tots els paràmetres, la combinació de paràmetres que tingui un millor *f1-score-weighted* de mitjana en la partició de validació (és a dir, el que obtingui una millor puntuació en la validació creuada) es determinarà com la millor per el model d'Arbre de Decisió i s'executarà la predicció en el test.

Una vagada s'ha fet la l'execució, s'ha obtingut que la millor combinació de paràmetres és la següent:

- **Criteri**: 'gini'.
- **Max_depth**: None.
- **Min_samples_leaf**: 1.
- **Min_samples_split**: 2.

Amb aquests paràmetres s'ha obtingut una puntuació en la partició de validació de 0,8319.

5.2.6 Resultats

Amb el dataset escollit i els millors hiperparàmetres triats, la predicció en el conjunt ha obtingut les mètriques que es poden veure en la taula 6

Metric	**Value**
accuracy	0.7
f1-micro	0.7
f1-macro	0.5724135182528296
f1-weighted	0.7096724055475849
precision-micro	0.7
precision-macro	0.5666666666666667
precision-weighted	0.722
recall-micro	0.7
recall-macro	0.5897435897435898
recall-weighted	0.7

Taula 6: Mètriques de les prediccions de l'Arbre de Decisió (amb paràmetres *criterion = 'gini', max_depth = None, min_samples_leaf = 1* i *min_samples_split = 2*) en el conjunt de test.

Si ens fixem en la mètrica que hem escollit (*f1-score-weighted*), podem veure que s'ha obtingut un valor de 0,7097, que és un valor considerablement bo. A més, en aquest cas el valor sí que es menor a l'obtingut en la validació creuada, de manera que podem dir que segurament el nostre model té un lleuger overfitting en les dades del train. Tot i això, a l'hora de generalitzar els resultats en el test, el nostre model és capaç de rendir suficientment bé.

Pel que fa a les prediccions, en la figura 15 es pot veure la matriu de confusió d'aquest model en les prediccions del conjunt de test. Es pot veure que la majoria de prediccions són bones, però hi ha hagut 6 mostres 'Alive' que no s'han predit correctament, 7 de 'Dead' i 2 de 'LiverTransplant'. Per les classes, 'Alive' i 'Dead', el model és lleugerament pitjor que el KNN (i per això ha obtingut un *f1-score-weighted* menor), però per 'LiverTransplant' ha aconseguir fer almenys una predicció correcta.

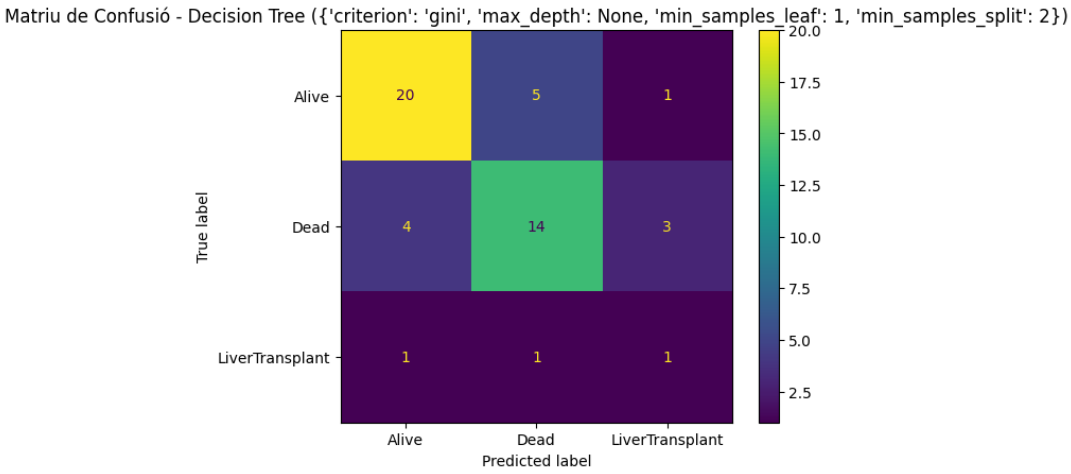


Figura 15: Matriu de confusió de les prediccions de l'Arbre de Decisió (amb paràmetres *criterion = 'gini', max_depth = None, min_samples_leaf = 1* i *min_samples_split = 2*) en el conjunt de test.

En aquest model, la classe 'Alive' s'ha predit correctament en una proporció de $\frac{20}{26} \approx 0,7692$ i en la classe 'Dead' $\frac{14}{21} \approx 0,6667$. Aquests resultats són pitjors que pel KNN i en aquest cas el model és millor predint la classe majoritària ('Alive'). Com ja hem dit, la part positiva d'aquest model és que ha aconseguit predir correctament una de les tres mostres de 'LiverTransplant'; però, com que hi ha molt poques mostres, s'ha de tenir menys en compte aquesta classe.

5.3 Support Vector Machine (SVM)

5.3.1 Motivació

Les SVM ofereixen una gran flexibilitat mitjançant l'ús de diferents funcions de kernel. Això ens permet explorar diferents maneres de modelar les relacions no lineals entre les variables i la variable objectiu "Status". Per exemple, un kernel polinòmic o de base radial (RBF) pot capturar relacions complexes que podrien ser crucials per la nostra predicció.

Un altre avantatge important de les SVM és la seva capacitat de controlar l'equilibri entre l'error de classificació i la complexitat del model a través del paràmetre de regularització C. Això és especialment útil quan es treballa amb dades mèdiques, on el balanç entre sensibilitat i especificitat és fonamental.

Finalment, la robustesa de les SVM davant dades amb soroll o outliers les fa una opció atractiva per a datasets complexos i desordenats com pot ser el nostre.

5.3.2 Mètriques

De la mateixa manera que s'ha fet per al KNN i per a l'Arbre de Decisió, en l'avaluació del model s'ha utilitzat la mètrica de *f1-score-weighted*. Aquesta mètrica és calcula fent la mitjana harmònica entre la *precision* i el *recall*, penalitzant els valors extrems (calen valors bons tant de *precision* com de *recall* per tal de tenir un bon *f1*). La mitjana de valors de totes les prediccions es fa ponderada (*weighted*), de manera que a cada classe se li dona una importància proporcional a les seves aparicions. D'aquesta manera, es puntua millor que s'aconsegueixi classificar correctament la classe majoritària (on s'haurà de fer més prediccions).

Si en un altre experiment es desitgés predir donar molta importància la classe minoritària (en aquest cas "LiverTransplant", que no és tant rellevant en aquest estudi com "Alive" o "Dead") es podrien utilitzar altres mètriques que ho tinguessin més en consideració.

5.3.3 Hiperparàmetres

Pel que fa als hiperparàmetres de la SVM, s'han modificat el kernel, el paràmetre C i el paràmetre gamma.

Els valors que es provaran per cada un dels paràmetres són els següents:

- **Kernel**: 'linear', 'poly', 'sigmoid', 'rbf'.
- **C**: 0.1, 0.5, 1, 2, 3, 5.
- **Gamma**: 'scale', 'auto'.

El significat de cada paràmetre és el següent:

- **Kernel**: aquest paràmetre representa la funció matemàtica que s'utilitzarà per transformar les dades a un espai de major dimensionalitat i així poder trobar relacions no lineals per separar les dades per un hiperplà.
  - 'linear': aquest kernel no fa cap transformació, sinó que manté les dades en el seu espai original. És simple i computacionalment més barat.
  - 'poly': aquest kernel transforma les dades a un espai de major dimensió on les combinacions polinòmiques dels atributs originals esdevenen les noves dimensions. Permet modelar interaccions més complexes entre variables que el kernel lineal. El seu comportament depèn d'un paràmetre 'degree' que controla el grau polinòmic. En el nostre cas, s'ha deixat sempre el grau per defecte, que és 3.
  - 'sigmoid': aquest kernel utilitza una funció sigmoide (semblant a la funció d'activació en xarxes neuronals). És capaç de transformar l'espai de dades de maneres no lineals i no polinòmiques.

  - 'rbf': aquest és un dels kernels més populars en les SVM. Transforma les dades a un espai on la distància radial (basada en la distància euclidiana) des d'un punt central determina el valor de cada nova dimensió. És molt efectiu per capturar relacions complexes en les dades i depèn del paràmetre gamma, que controla l'amplitud de la 'zona d'influència' de cada un dels punts de suport.

- **C**: aquest paràmetre controla l'equilibri entre la maximització del marge i la minimització de l'error en la classificació. Un valor més alt penalitza més els errors i disminueix el marge (pot portar a overfitting); mentre que un valor més baix permet més errors i augmenta el marge (pot portar a underfitting).
- **Gamma**: aquest paràmetre és per el kernel 'rbf' i controla si es té en compte una àrea més o menys gran per cada punt de suport a l'hora de determinar el límit de decisió.
  - 'scale': amb aquest paràmetre, la gamma s'adapta automàticament a l'escala de les dades. Ajuda a prevenir que el model sigui massa sensible a les variacions el les dades quan hi ha moltes dimensions o quan les característiques tenen variàncies molt diferents.
  - 'auto': aquest enfoc no té en compte la variància de les dades i pot no ser tant òptim som 'scale' en certs casos.

Totes les possibles combinacions d'aquests paràmetres es provaran mitjançant Grid Search i cross validation per trobar els que millor rendiment aporten al model.

5.3.4 Millor conjunt de dades

Per trobar el conjunt de dades que proporciona un millor rendiment a aquest model, s'han fixat els següents hiperparàmetres:

- **Kernel**: 'rbf'.
- **C**: 1.
- **Gamma**: 'scale'.

Amb aquests paràmetres fixats, s'ha executat la funció `find_best_dataset()`. Després que es provin totes les combinacions possibles de modificacions al dataset inicial (un total de 64 combinacions), la que ha obtingut millors resultats de *f1-score-weighted* en el test és la següent:

- No eliminar outliers.
- Eliminar les files amb més de 9 NaNs.
- Escalar les variables numèriques amb *MinMax*.
- Imputar variables numèriques amb KNN Imputer (amb $k = 15$) i les categòriques amb Random Forest Classifier (amb el criteri 'gini'). Els resultats de la imputació (calculats tal i com s'ha explicat en l'apartat 3.5) han estat de 0,3808 de mitjana entre la puntuació R^2 de les numèriques i *f1-score-weighted* de les categòriques.

- Codificar les variables categòriques amb *Ordinal Encoding*.
- Balancejar les classes de la variabe objectiu mitjançant *SMOTE*.

Amb aquestes modificacions i els paràmetres fixes que s'han mencionat, s'ha aconseguit un resultat de 0,8201 de *f1-score-weighted* en el conjunt de prova, que es pot considerar un resultat bastant bo. La partició de train conté 441 mostres i la de test 50. A partir d'aquí s'ha considerat que aquests són els millors conjunts de dades per els models d'Arbre de Decisió, i el següent pas serà trobar els millors paràmetres per el model en aquest dataset.

5.3.5 Millors hiperparàmetres pel dataset escollit

Una vegada determinat el millor dataset per els models de SVM, cal provar quins són els millors paràmetres. Això es fa mitjançant la funció `run_experiment()` que cridarà a `svm_prediction()`. En aquesta segona funció s'implementa cross validation amb 5 *folds* per cada possible combinació de paràmetres. Una vegada s'hagin provat tots els paràmetres, la combinació de paràmetres que tingui un millor *f1-score-weighted* de mitjana en la partició de validació (és a dir, el que obtingui una millor puntuació en la validació creuada) es determinarà com la millor per el model de SVM i s'executarà la predicció en el test.

Una vagada s'ha fet la l'execució, s'ha obtingut que la millor combinació de paràmetres és la següent:

- **Kernel**: 'linear'.
- **C**: 100.
- **Gamma**: 'scale'.

Amb aquests paràmetres s'ha obtingut una puntuació en la partició de validació de 0,7463. Cal mencionar que el paràmetre gamma no és rellevant en un kernel linear, així que no s'ha de considerar que afecti al model.

5.3.6 Resultats

Amb el dataset escollit i els millors hiperparàmetres triats, la predicció en el conjunt ha obtingut les mètriques que es poden veure en la taula 7

Metric	**Value**
accuracy	0.8
f1-micro	0.8000000000000002
f1-macro	0.7362502681006221
f1-weighted	0.8139203970953212
precision-micro	0.8
precision-macro	0.7143939393939394
precision-weighted	0.8601363636363637
recall-micro	0.8
recall-macro	0.8626373626373626
recall-weighted	0.8

Taula 7: Mètriques de les prediccions del Support Vector Machine (amb paràmetres *C = 5, gamma = 'scale'* i *kernel = 'linear')* en el conjunt de test.

Si ens fixem en la mètrica que hem escollit (*f1-score-weighted*), podem veure que s'ha obtingut un valor de 0,8139, que és un valor molt bo, inclús lleugerament més alt del que s'ha vist en el cross validation. Això pot ser degut a factors aleatoris.

Pel que fa a les prediccions, en la figura 16 es pot veure la matriu de confusió d'aquest model en les prediccions del conjunt de test. La majoria de prediccions són bones, però hi ha hagut 7 mostres 'Alive' que no s'han predit correctament, 3 de 'Dead'. En aquest model, la classe 'LiverTransplant' s'ha predit bé el 100% de les vegades (tot i només haver-hi 3 mostres), a més d'haver valors de 'Alive' i de 'Dead' que s'han predit erròniament com a 'LiverTransplant'. Observant això, podem dir que aquest model té una major tendència a predir la classe 'LiverTransplant' en comparació als dos models previs.

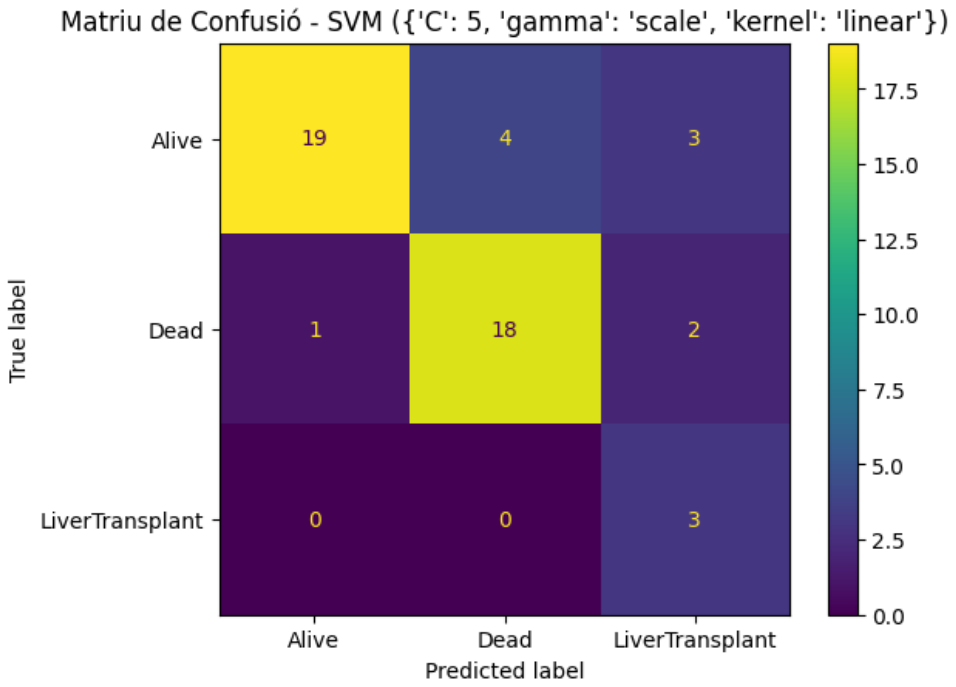


Figura 16: Matriu de confusió de les prediccions del Support Vector Machine (amb paràmetres *C = 5, gamma = 'scale'* i *kernel = 'linear')* en el conjunt de test.

En aquest model, la classe 'Alive' s'ha predit correctament en una proporció de $\frac{19}{26} \approx 0,7308$ i en la classe 'Dead' $\frac{18}{21} \approx 0,8571$. Aquests resultats són més semblants al KNN que no pas a l'Arbre de Decisió. no obstant la sorpresa, com ja s'ha dit, és que s'han predit correctament tots els valors de 'LiverTransplant'.

Es pot dir que, en aquest model SVM, no hi ha pràcticament tendència a predir arbitràriament la classe majoritària perquè la majoria de falsos positius es troben quan es predeia 'LiverTransplant' o 'Dead' (classes minoritàries en comparació amb 'Alive'), a més de que són les classes on hi ha més percentatge de prediccions correctes.

6 Selecció del model

Una vegada d'han descrit i comentat els tres models per separat, cal ajuntar els resultats i escollir un dels models.

Model	Cross Validation	Test
KNN	0.6287	0.7365
DT	0.8319	07097
SVM	0.7463	0.8139

Taula 8: Mètrica *f1-score-weighted* dels 3 models en cross validation i en test.

En la taula 8 es poden veure els resultats de *f1-score-weighted* per els 3 models, tant en la validació com en el test. El model amb millor puntuació en el test és clarament la SVM, així que sembla que serà el model escollit per descriure més detalladament.

Com a observacions, es pot veure que en els tres datasets escollits pels models no s'eliminen outliers. És a dir, la eliminació d'outliers redueix el rendiment en aquests models. A més, en tots s'eliminen les mostres amb 9 NaNs. Això segurament és així degut a que la imputació de valors no és destacablement bona, de manera que és més efectiu eliminar les files amb molts missings i posteriorment aplicar algun mètode de balanceig (en aquests casos *Oversampling* i *SMOTE*) a partir de dades de major qualitat.

Per altres modificacions en el dataset, com ara l'escalat de variables o la codificació, només s'apliquen a dos dels tres models, de manera que no es pot dir de forma general que aplicar o no aquestes modificacions sigui millor pel rendiment general dels models predictius. No obstant, en la imputació de valors tots els models han escollit al Random Forest Classifier (amb mètode 'gini') com el millor mètode per variables categòriques, de manera que es pot dir que és millor de forma general que els altres mètodes provats.

Amb tots els resultats observats, s'ha decidit que el model escollit per fer la Model Card i una descripció més detallada serà la SVM, degut a la seva puntuació en el test i a la seva poca tendència de predir la classe majoritària per sobre de les altres.

6.1 Descripció del model triat

En aquest model, s'utilitza un kernel lineal. Això significa que el model busca una frontera de decisió lineal en l'espai de característiques original. En altres paraules, intenta separar les classes utilitzant una línia, pla o hiperpla, depenent del nombre de característiques d'entrada.

El kernel lineal és particularment útil quan es treballa amb un gran nombre de variables degut a que tenen una menor tendència al sobreajustament (overfitting) en comparació als kernels no lineals. Això pot ser un dels motius pels que el model ha aconseguit resultats tan bons en el test.

El paràmetre C (en aquest model model, C = 5) controla el compromís entre classificar correctament tots els punts d'entrenament i tenir una frontera de decisió el més simple possible.

Un valor més alt de C significa que el model penalitza més els errors de classificació durant l'entrenament, conduint a una frontera de decisió que intenta classificar més correctament tots els punts d'entrenament, sovint a costa de generar overfitting. Per altra banda, un valor més baix de C dona més importància a la simplicitat de la frontera de decisió, la qual cosa pot millorar la generalització a costa de cometre més errors en el conjunt d'entrenament.

Finalment, el paràmetre gamma ja s'ha mencionat que no té influència en aquest cas, ja que només afecta a kernels no lineals.

En resum, aquest model SVM busca una frontera de decisió lineal que equilibra la classificació correcta dels punts d'entrenament amb la simplicitat de la frontera de decisió. El valor de C = 5 és més propens a overfitting que no pas el valor per defecte que proporciona `scikit-learn` (C = 1), motiu pel qual segurament ha obtingut millors resultats en la validació). No obstant, aquest overfitting es compensa pel fet de fer servir un kernel lineal, que generalment té menys tendència al sobreajustament.

6.2 Anàlisi de les limitacions i capacitats del model

Tal i com s'ha vist en la taula 7 i en la figura 16, aquest model té una bona *accuracy* global (de 0,8) que indica que, en general, el model és capaç de classificar correctament un 80% de les instàncies. Això es reafirma amb un score *f1-score-weighted* de 0,8139 en el conjunt de test, el qual indica una bona harmonia entre la *precision* i el *recall* del model.

No obstant això, l'anàlisi detallada de la matriu de confusió mostra algunes limitacions del model:

- **Confusions entre classes**: Observem que hi ha una mica de confusió entre la classe 'Alive' i 'Dead', amb 4 instàncies 'Alive' que s'han classificat com a 'Dead' i 1 instància 'Dead' classificada com a 'Alive'. Això pot ser indicatiu d'una frontera de decisió que no és totalment òptima entre aquestes dues classes. A més, també hi ha 3 instàncies de 'Alive' i 2 de 'Dead' classificades com a 'LiverTransplant', indicant que hi ha una lleugera tendència a classificar 'LiverTransplant' en casos on no s'hauria de fer.
- **Variància en els scores de validació i test**: La puntuació de la validació creuada (0,746) és lleugerament inferior al *f1-score-weighted* obtinguts en el conjunt de test, suggerint que el model podria estar beneficiant-se d'alguna sort d'overfitting específic del conjunt de test o que el procés de validació creuada no captura completament la variabilitat del conjunt de dades. Per tant, en certs escenaris, potser el model no generalitzaria prou bé.

En resum, tot i que model té molt bones mètriques en general, té desperfectes a l'hora de realitzar la predicció, confonent algunes prediccions, especialment tendint a tenir més falsos positius en 'LiverTransplant' i 'Dead'. A més, és possible que hagi estat relativament afavorit pel factor aleatori.

6.3 Resultats

Els resultats del model SVM escollit ja s'han mencionat anteriorment. No obstant, per una major rigorositat de l'anàlisi, s'han realitzat 5 execucions amb exactament les mateixes condicions, datasets i paràmetres que anteriorment, però modificant la llavor aleatòria (concretament, s'han utilitzat les llavors: 29, 34, 85, 32, 5. La mitjana de *f1-score-weighted* en aquests 5 experiments ha estat de 0,6398, considerablement més baixa que la que ha sortit en l'experiment anterior (amb la llavor 42). Per tant, aquest model no és tant bo com semblava, ja que el seu resultat varia bastant en funció de la llavor utilitzada.

Com ja s'ha dit anteriorment, els paràmetres dels models s'haurien d'escollir fent execucions amb múltiples llavors i amb la mitjana dels resultats. No obstant, en aquest estudi no s'ha fet degut a que s'ha considerat que l'enfoc més important es troba en l'anàlisi i no tant en el propi rendiment dels models. A més, amb una sola llavor la reproductibilitat és més fàcil, s'estalvien temps d'execució, fàcil comparació amb altres models i simplificació.

En el següent apartat d'aquest document es realitza una Model Card sobre el model escollit. Una vegada s'ha fet la reflexió de que realment el model escollit no és tant bo com semblava, per aquesta model card es tindran en consideració només els resultats amb la llavor 42, ja que fer una model card d'un model que no rendeix correctament és molt difícil (ja que s'estaria intentant justificar un model que no rendeix suficientment bé). Es vol reiterar que el propòsit d'aquest pràctica es centra en fer un bon anàlisi, i no pas un bon model. Per tant, es farà una model card justificant el rendiment del model com si, de forma general, fos igual de bo que l'obtingut amb la llavor 42. Evidentment, si aquest model s'hagués aplicar en la realitat i el propòsit d'aquest projecte no fos adquirir coneixement en Machine Learning i anàlisi de dades, es farien modificacions en el model i es faria la Model Card tenint en compte molts més resultats.

7 Model Card per Support Vector Machine (SVM)

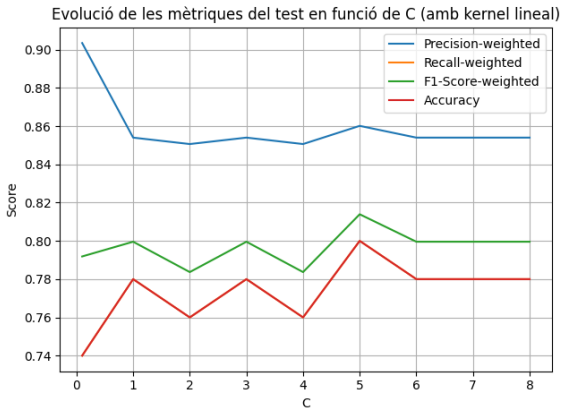
7.1 Informació General

- **Nom del model**: SVM per a la predicció de l'estatus de pacients amb Cirrosi Hepàtica.
- **Tipus de model**: Support Vector Machine (SVM).
- **Font**: model general extret de la llibreria scikit-learn de Python.
- **Desenvolupadors**: Cai Selvas Sala, estudiant del Grau en Intel·ligència Artificial de la Universitat Politècnica de Catalunya.
- **Data de creació**: 28/12/2023
- **Versió del model**: 1.0
- **Descripció del Model**: El SVM és un model de classificació utilitzat per a predir l'estatus ('Alive', 'Dead', 'LiverTransplant') de pacients amb Cirrosi Hepàtica. Utilitza un kernel lineal per a separar les classes i ha estat entrenat amb un conjunt de dades que inclouen variables clíniques i demogràfiques.

7.2 Hiperparàmetres

Han estat determinats mitjançant una validació creuada amb *K-Fold* ($k = 5$).

- **Kernel**: s'ha utilitzat un kernel lineal.
- **C**: s'ha utilitzat el valor $C = 5$. No es recomana, però es pot modificar si es desitja.



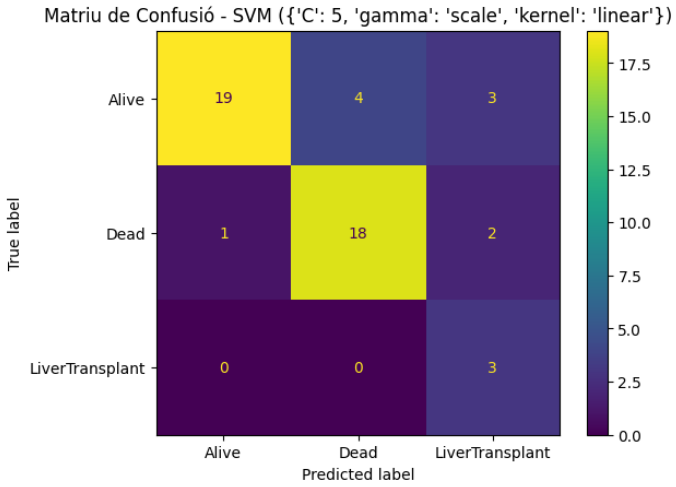
Evolucició de les mètriques en el test en funció del paràmetre C. *Precision-weighted* i *recall-weighted* es solapen en la gràfica, van junts.

7.3 Dades

- **Font de les dades**: Estudi de Mayo Clinic sobre Cirrosis [1].
- **Període de recopilació**: 1974 - 1948.
- **Número de mostres inicials**: 418 pacients.
- **Preprocessament**:
  - **Particionat en train i test**: 85% train i 15% test (amb el mètode *stratify* per mantenir la distribució de classes).
  - **Tractament d'outliers**: no s'han eliminat outliers.
  - **Tractament de missings**:
    - **Eliminació de mostres**: no s'han considerat les mostres amb 9 valors faltants per garantir una millor qualitat de les dades.
    - **Imputació de missings**: KNN Imputer ($k = 15$) per variables numèriques. Random Forest Classifier (amb criteri 'gini') per categoòriques.
  - **Codificació de variables**: variables categòriques codificades mitjançant *Ordinal Encoding*.
  - **Escalat de variables**: variables numèriques escalades a través de *MinMax Scaler*.
  - **Balanceig de classes**: mètode *SMOTE* per a balancejar les classes de la variable objectiu (*Status*). Amb això, el dataset d'entrenament consta de 441 mostres (pacients).
  - **Característiques incloses**: Edat, Sexe, Resultats de laboratori (Presència o quantitat de Ascites, Hepatomegaly, Spiders, Edema, Cholesterol, Albumin, Copper, Alk_Phos, SGOT, Tryglicerides, Platelets, Prothrombin), estat de la malaltia (Stage), dies del pacient en el tractament, administració del tractament o placebo, estat final del pacient.

7.4 Validació

- **Mida del conjunt de test**: 50 mostres (pacients).
- **Rendiment**:
  - **Rendiment mitjà en validació creuada**: F1-Score Weighted = 0,7463.
  - **Rendiment en la validació (test)**:
    - F1-Score Weighted = 0,8139.
    - Accuracy = 0,8.
    - Precision Weighted = 0.8601.
    - Recall Weighted = 0.8.
  - Matriu de confusió:



El rendiment pot variar segons la naturalesa del conjunt de dades de validació/test. Les prediccions de la classe 'LiverTransplant' només han pogut ser provades amb les 3 mostres del conjunt de test, de manera que és difícil que es mantinguin els resultats que es mostren en la matriu de confusió. Generalment, el model tendeix a predir falsos positius en 'LiverTransplant' i 'Dead'.

7.5 Advertències i Recomanacions

Degut a desbalangeigs en atributs com ara *Sex*, pot haver biaixos de sexe.

El model està dissenyat per a ser utilitzat com a eina de suport a la decisió clínica. En cap cas s'hauria d'utilitzar com a única font per a prendre decisions clíniques sensibles. Pot servir per fer una estimació de l'estat final d'un pacient amb Cirrosi per tal de prendre mesures sobre com guiar el seu tractament, però sempre sota la responsabilitat i aprovació d'un metge o professional especialitzat.

7.6 Limitacions

- Per funcionar correcatment amb dades externes, aquestes s'han de preprocessar de la mateixa manera en què el model ha estat entrenat.
- EL model no és totalment precís, en certs casos pot predir una classe que no és correcta, tal i com es veu en la matriu de confusió.
- El model pot tenir lleuger overfitting, de maner que els resultats siguin pitjors en un escenari real.

8 Bonus 2: Anàlisi no supervisat de les dades

9 Conclusions

9.1 Valoració de l'aprenentatge adquirit

10 Referències

[1] E. Dickson; P. Grambsch; T. Fleming; L. Fisher; A. Langworthy. Cirrhosis Patient Survival Prediction. UCI Machine Learning Repository, 2023. DOI: https://doi.org/10.24432/C5R02G.

[2] Fundación Española del Corazón. Colesterol y riesgo cardiovascular. `https://fundaciondelcorazon.com/prevencion/riesgo-cardiovascular/colesterol.html`, 2023. Accés: 27/12/2023.