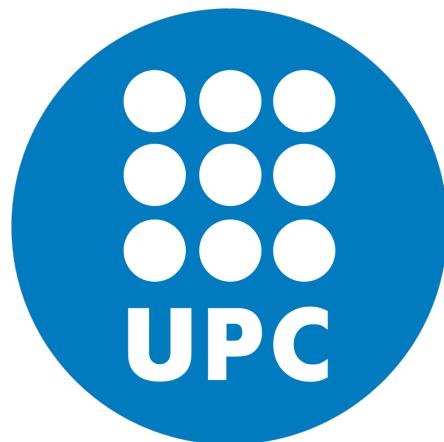


Informe de la Pràctica d'Introducció a l'Aprendentatge Automàtic

Pau Prat Moreno

28 de desembre de 2023



Universitat Politècnica de Catalunya

Grau en Intel·ligència Artificial

IAA 23/24 Q1 Tardor - Laboratori 1

Resum

Aquest document correspon a l'informe de la primera i única pràctica de l'assignatura Introducció a l'Aprenentatge Automàtic del Grau en Intel·ligència Artificial, pertanyent a la Universitat Politècnica de Catalunya (UPC).

Al llarg d'aquest informe, es detallen els passos seguits durant les diferents etapes de la creació del model, des de l'anàlisi inicial de les dades fins a la selecció final del model. S'incorporen taules, gràfics i diversos anàlisis que il·lustren els resultats obtinguts en cada secció diferent. Totes les decisions preses en el procés d'anàlisi, preprocessat de dades, selecció de variables, i construcció de models han estat argumentades de manera explícita, proporcionant una base sólida per a la comprensió i interpretació dels resultats obtinguts.

Al final d'aquest document es presenta la Model Card del model seleccionat. Aquesta inclou informació clau sobre l'objectiu, el context d'ús, detalls tècnics i les mètriques de rendiment del model. També aborda les limitacions, les condicions de prova, i consideracions ètiques i socials rellevants.

Índex

1 Introducció	5
2 Anàlisi i preprocessat de dades	6
2.1 Anàlisi estadístic de les variables de manera independent	6
2.1.1 Classificació de variables	7
2.1.2 Distribució de les variables numèriques	9
2.2 Estudi de balanceig de classes	10
2.2.1 Freqüències per classe de les variables categòriques	10
2.2.2 Anàlisi de la variable objectiu 'Status'	12
2.2.3 Quin mètode aplicar	13
2.3 Anàlisi de dades mancats (Missing Data)	13
2.4 Outliers	15
2.4.1 Criteri per a definir quins valors són outliers	15
2.4.2 Comparació i anàlisi dels features amb i sense outliers	16
2.5 Recodificació de variables	21
2.5.1 Codificació de les variables categòriques	21
2.6 Acabant el Preprocessament	22
3 Preparació de variables	23
3.1 Anàlisi de correlacions entre variables numèriques	23
3.2 Anàlisi de variables categòriques i variable objectiu	26
3.3 Particionat del dataset	30
3.4 Imputació dels missings	31
3.4.1 Imputació per a variables numèriques	31
3.4.2 Imputació per a variables categòriques	32
3.5 Aplicació de 'SMOTENC' pel balanceig de classes	36
3.6 Variables redundants o sorollooses	37
3.7 Normalització de variables	39
3.8 PCA - Anàlisi de Components Principals	40
4 Definició de models	43
4.1 Definició de Mètriques i Motivació	43
4.1.1 Precisió	43
4.1.2 Recall	44
4.1.3 Accuracy	44
4.1.4 F1-score	44
4.1.5 Àrea sota la Corba ROC (AUC-ROC)	44
4.1.6 Mètrica a maximitzar	45
4.2 Primer model - KNN (K-Nearest Neighbors)	46
4.2.1 Simplicitat i Interpretabilitat	46
4.2.2 Hiperparàmetres	46
4.2.3 Volum de Dades	47
4.2.4 Aplicabilitat al Problema	47
4.3 Segon model - Arbre de Decisió	47
4.3.1 Simplicitat i Interpretabilitat	47
4.3.2 Hiperparàmetres	47

4.3.3	Volum de Dades	48
4.3.4	Aplicabilitat al Problema	48
4.3.5	Importància de la Poda	48
4.4	Tercer model - Support Vector Machine (SVM)	48
4.4.1	Simplicitat i Interpretabilitat	48
4.4.2	Hiperparàmetres	48
4.4.3	Volum de Dades	49
4.4.4	Aplicabilitat al Problema	49
4.5	Trobar els hiperparàmetres òptims	49
4.5.1	Procés de Selecció	49
4.5.2	Resultats Obtinguts	50
4.6	Entrenament dels models amb train	51
4.6.1	KNN - Entrenament amb Train	52
4.6.2	Arbre de Decisió - Entrenament amb Train	54
4.6.3	SVM - Entrenament amb Train	59
5	Selecció del model	62
5.1	Resultats en partició de test	62
5.1.1	KNN - Resultats amb Test	62
5.1.2	Arbre de Decisió - Resultats amb Test	65
5.1.3	SVM - Resultats amb Test	67
5.2	Model escollit - SVM	70
5.3	Anàlisi de les limitacions i capacitats del model	72
5.4	Capacitats	72
5.5	Limitacions	72
6	Bonus 1 - Explainable Boosting Machine (EBM)	74
6.1	Definició del Model	74
6.1.1	Motivació	74
6.1.2	Complexitat i Interpretabilitat	74
6.1.3	Hiperparàmetres	74
6.1.4	Volum de Dades	74
6.1.5	Adequació al Nostre Context	75
6.1.6	Selecció d'Hiperparàmetres	75
6.2	Entrenament amb Train	75
6.3	Importància global dels features	77
6.4	Resultats amb Test	78
7	Bonus 2 - Clustering	82
7.1	Identificació de Clusters	82
7.2	Anàlisi dels Clusters	83
7.2.1	PCA	85
7.2.2	Conclusions	87
8	Model Card	88

1 Introducció

En aquest treball ens endinsem en l'estudi de la cirrosi hepàtica, una malaltia crònica del fetge que pot causar un greu deteriorament en la salut de la persona afectada. L'objectiu principal és desenvolupar un model predictiu que, a partir de 17 característiques clíniques, predigui la supervivència dels pacients amb aquesta malaltia.

Per aconseguir això, s'ha utilitzat una base de dades disponible a l'UC Irvine Machine Learning Repository¹, que inclou diverses variables relacionades amb l'estat de salut dels pacients, com ara els nivells de colesterol o bilirrubina del pacient, la presència de símptomes com un edema o una ascitis, i informació personal, com pot ser l'edat del pacient o el gènere.

El treball que s'ha fet per tal d'analitzar profundadament aquest conjunt de dades consta de diverses seccions:

- Anàlisi i preprocessat de dades
- Preparació de variables
- Definició de models
- Selecció de models
- Model Card

El preprocessament de les dades inclourà l'anàlisi estadística de les variables, estudi de balanceig de classes, gestió missings, identificació i tractament d'outliers, així com la necessària recodificació de certs features. També es realitzarà una partició de les dades en conjunts de train/test o train/-validació/test, depenent de les necessitats del model.

A la secció de preparació de variables, es realitzaran tasques com la normalització de variables, anàlisi de correlacions entre variables numèriques, estudi de les variables categòriques, eliminació de variables redundants o sorolloses, i un estudi de la dimensionalitat mitjançant PCA (Anàlisi de Components Principals).

En la definició de models, entrenarem tres tipus de models: KNN, arbre de decisió, i SVM. Es definiran les mètriques a utilitzar, es motivaran els models triats, i s'analitzaran els seus hiperparàmetres. A més, es realitzarà un primer entrenament amb el conjunt de train i s'analitzaran els resultats obtinguts.

Finalment, en la selecció del model, es descriuràn els models triats, es discutiran les seves limitacions i capacitats, i es presentaran els resultats obtinguts en la partició de test en comparació amb el train i la validació.

Així doncs, amb la pauta de quines accions es realitzaran en cada apartat, ja podem començar a analitzar i preprocessar el nostre conjunt de dades.

¹<https://archive.ics.uci.edu/dataset/878/cirrhosis+patient+survival+prediction+dataset-1>

2 Anàlisi i preprocessat de dades

2.1 Anàlisi estadístic de les variables de manera independent

El primer que cal fer és analitzar la base de dades i veure quins valors tenen els diferents features i què representen, per la qual cosa necessitem veure una taula amb les mitjanes, valors mínims i màxims, error estàndard, etc. de cada variable numèrica, ja que d'aquesta manera tindrem una idea global de cada feature de manera independent. Així doncs, si utilitzem el mètode `describe()`, obtenim els següents resultats:

	ID	N_Days	Age	Bilirubin	Albumin	Alk_Phosphatase	SGOT	Prothrombin	Stage
count	418.000000	418.000000	418.000000	418.000000	418.000000	312.000000	312.000000	416.000000	412.000000
mean	209.500000	1917.782297	18533.351675	3.220813	3.497440	1982.655769	122.556346	10.731731	3.024272
std	120.810458	1104.672992	3815.845055	4.407506	0.424972	2140.388824	56.699525	1.022000	0.882042
min	1.000000	41.000000	9598.000000	0.300000	1.960000	289.000000	26.350000	9.000000	1.000000
25%	105.250000	1092.750000	15644.500000	0.800000	3.242500	871.500000	80.600000	10.000000	2.000000
50%	209.500000	1730.000000	18628.000000	1.400000	3.530000	1259.000000	114.700000	10.600000	3.000000
75%	313.750000	2613.500000	21272.500000	3.400000	3.770000	1980.000000	151.900000	11.100000	4.000000
max	418.000000	4795.000000	28650.000000	28.000000	4.640000	13862.400000	457.250000	18.000000	4.000000

Taula 1: Resum estadístic dels features numèrics

D'aquesta taula podem observar diverses coses, com ara que la variable 'ID' sembla ser l'identificador de cada pacient, ja que podem veure com tots els IDs estan presents (418), i sembla que estan distribuïts uniformement, ja que la mitjana és precisament la meitat del valor màxim. També podem veure que 'N_Days' varia significativament (des de 41 fins a 4795 dies), amb una mitjana d'aproximadament 1918 dies. Observem que la desviació estàndard és realment alta (1104.67), per tant això ens indica que aquest feature té una gran variabilitat.

Per altra banda, a primera vista ha sigut sorprendent els valors de 'Age', ja que a priori sembla que són molt alts, però en observar la descripció de cada feature que es troba a la pàgina web, ens podem adonar que està en unitat de dies, per tant posteriorment la transformarem a anys, per tal de poder interpretar els resultats més fàcilment.

Podem veure també com els nivells de bilirrubina varien significativament (de 0.3 a 28), amb una mitjana d'aproximadament 3.22, mentre que els nivells d'albumina mostren menys variabilitat (de 1.96 a 4.64) amb una mitjana de 3.5, la qual cosa podria suggerir un rang més normalitzat en aquesta mesura.

Cal remarcar també la molt alta variabilitat que té la variable Alk_Phosphatase (de 289 a 13862.4), en què la mitjana (1982.66) és molt superior a la mediana (1259), és a dir, ja podem deduir que aquesta variable tindrà una distribució asimètrica.

El feature 'SGOT' també mostra una alta variabilitat, on el valor màxim és realment elevat comparant-lo amb la mitjana, per tant pot ser que posteriorment hi trobem certs outliers.

Ja per acabar, la variabilitat del Prothrombin és relativament baixa en comparació amb altres mesures (amb una mitjana de 10.73), i finalment veiem que el feature Stage varia de 1 a 4, amb una mitjana lleugerament superior a 3, és a dir, deu haver una certa tendència cap a etapes més avançades en el conjunt de dades.

Un cop analitzada detalladament aquesta primera impressió de les variables numèriques, per tal d'estalviar-nos feina en un futur, podem eliminar el feature 'ID' ja que no ens aporta cap informació sobre les dades i és totalment prescindible.

Així doncs, ja podem veure la distribució per a cada variable.

2.1.1 Classificació de variables

Com que la base de dades consta de variables numèriques i categòriques, abans de visualitzar la distribució de les numèriques o la freqüència per classe de les categòriques, necessitem classificar cada variable segons si és numèrica o categòrica.

Per a fer-ho, he decidit mirar quants valors únics té cada variable, i d'aquesta manera poder classificar cadascuna a numèrica o categòrica segons del nombre de valors únics que tingui; si té un nombre elevat de valors únics, la tractarem com a numèrica, mentre que si té pocs valors únics la considerarem una variable categòrica. Per tant, si mirem els resultats podem veure clarament quins features són numèrics i quins són categòrics: Com podem observar, el feature 'Stage' té

Feature	Type	Unique values
Sex	object	2
Status	object	3
Drug	object	3
Ascites	object	3
Hepatomegaly	object	3
Spiders	object	3
Edema	object	3
Stage	float64	4
Prothrombin	float64	48
Bilirubin	float64	98
Tryglicerides	float64	146
Albumin	float64	154
Copper	float64	158
SGOT	float64	179
Cholesterol	float64	201
Platelets	float64	244
Alk_Phosphatase	float64	295
Age	int64	344
N_Days	int64	399

Taula 2: Tipus i nombre de valors únics per cada feature

únicament 4 valors diferents, per la qual cosa l'hem de considerar categòrica. Per altra banda, els features 'Tryglicerides', 'Copper', 'Cholesterol' i 'Platelets' els hem de considerar numèrics, ja que tenen un nombre molt elevat de valors únics. És a dir, aquestes 5 variables (marcades amb un color vermellós a la taula) han de canviar de grup. Així doncs, he creat una funció anomenada `classify_features(df, threshold, target)` que donat un llindar (`threshold`), classifica cada feature com a categòrica o numèrica segons si el seu nombre de valors únics està per sota o per sobre d'aquest llindar. Com podem veure, un bon `threshold` en aquest cas seria el número 10, ja que tots els features que estiguin per sota els considerarem categòrics, i tots els que estiguin per amunt els considerarem numèrics. L'argument `target` es refereix a la variable objectiu (`target='Status'`), ja que aquesta variable no l'incloureïm a cap d'aquests dos conjunts, doncs la tractarem de manera diferent en un futur. Per tant, la funció també comprova que cada feature del dataset 'df' no sigui el `target` feature, per tal de no incloure'l a cap dels dos grups.

D'aquesta manera, en cridar a la següent funció es crea una llista per a les variables numèriques, i una per a les variables categòriques: `numerical_features, categorical_features = classify_features(df, threshold, target)`, que dona com a resultat:

```
numerical_features: ['N_Days', 'Age', 'Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phosphatase', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']
```

```
categorical_features: ['Status', 'Drug', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Stage']
```

Així doncs, un cop ja hem dividit les variables entre categòriques i numèriques, podem procedir a plotejar les distribucions de les numèriques.

2.1.2 Distribució de les variables numèriques

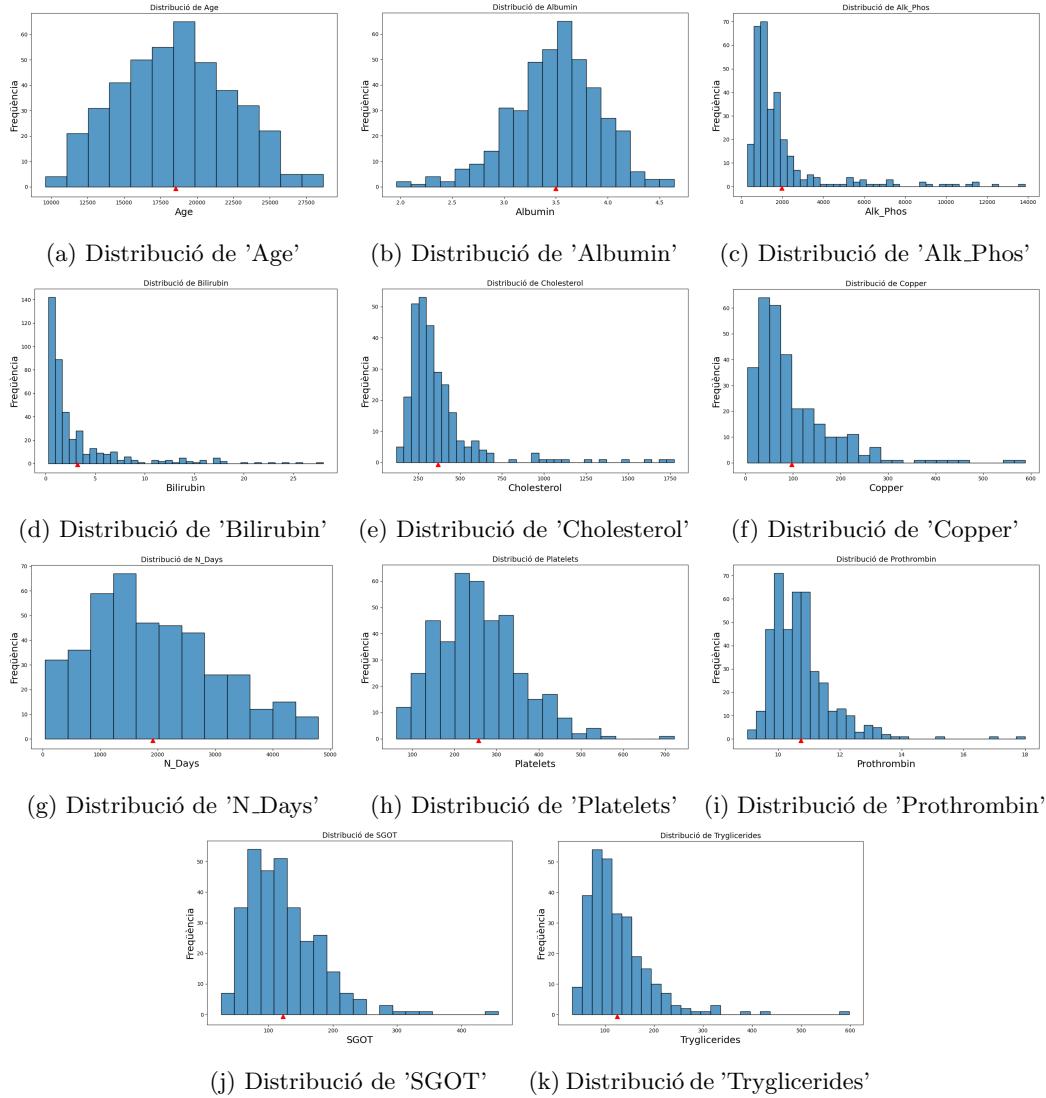


Figura 1: Distribucions de les diferents variables numèriques

D'aquests plots podem destacar la següent informació rellevant:

- Edat: L'histograma indica una distribució normal amb un pic al voltant de 20.000, mostrant que les edats en el conjunt de dades estan centrades al voltant d'aquest valor. Si ho passem a anys seria a prop dels 50 anys, per la qual cosa la majoria dels pacients tenien una edat propera als 50.
- N_Days: L'histograma mostra una inclinació cap a la dreta, indicant que la majoria dels

pacients van tenir un esdeveniment (mort o trasplantament) o van ser analitzats en el temps d'estudi relativament aviat després del seu registre.

Respecte els altres features podem veure que els valors de tots els histogrames estan concentrats a l'esquerra del gràfic, la qual cosa ens indica que la majoria dels pacients tenen uns valors baixos d'aquestes característiques però que també existeixen pacient amb valors molt elevats, per tant, haurem de tractar aquests valors 'extrems' o 'inusuals' en l'apartat de Outliers.

Com podem veure, features com ara 'Age', 'Albumin' i 'Platelets' segueixen una distribució normal, per tant no caldrà que les normalitzem posteriorment. També podem veure que hi ha molts features que semblen tenir outliers, com ara 'Akl.Phos', 'Cholesterol' o 'SGOT', entre d'altres, per tant els haurem d'analitzar en profunditat en l'apartat corresponent.

2.2 Estudi de balanceig de classes

2.2.1 Freqüències per classe de les variables categòriques

Un cop vist les diferents distribucions de les variables numèriques, podem procedir a analitzar les variables categòriques, la qual cosa es pot observar mitjançant els histogrames de freqüències per classe de cada una d'aquestes variables.

Al realitzar els gràfics pertinents, s'obtenen els resultats següents:

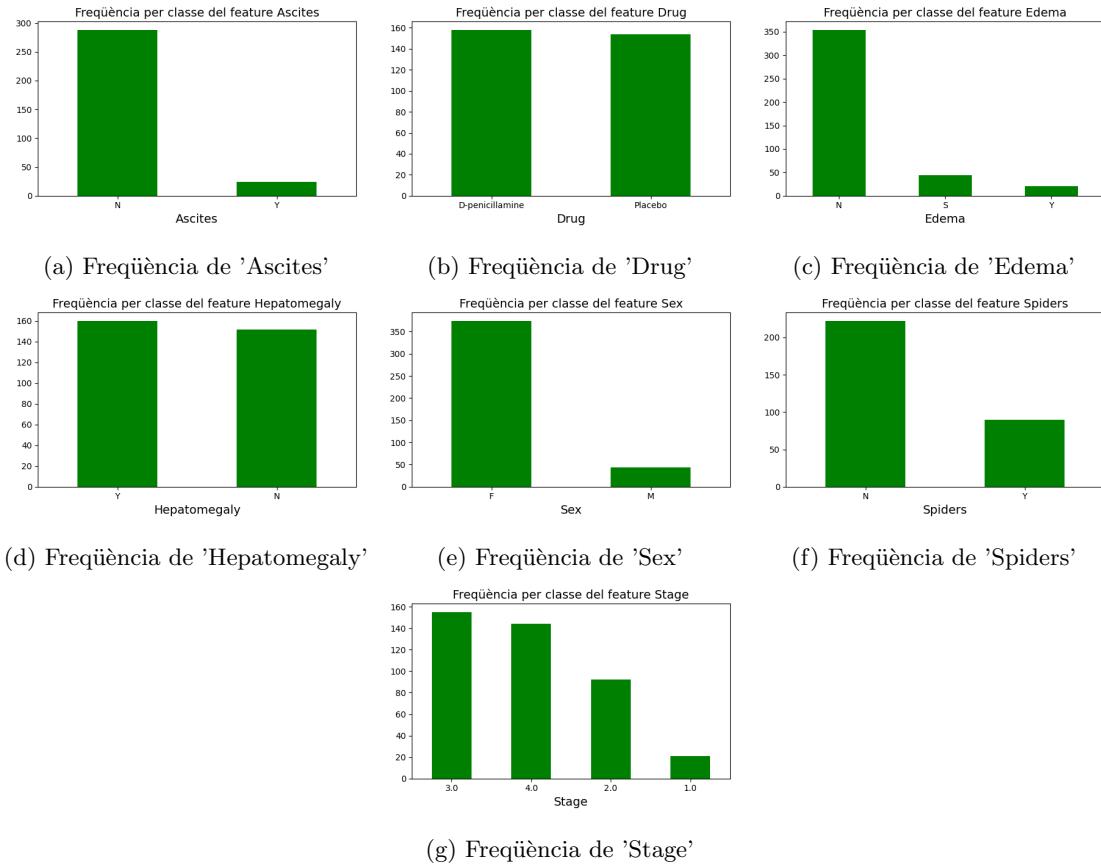


Figura 2: Freqüència per classe de cada variable categòrica

Com es pot apreciar, hi ha features que tenen un gran desbalanceig entre classes, és a dir, es podria dir que excepte 'Drug' i 'Hepatomegaly', tots els altres features tenen classes amb moltes més mostres que altres. Per tant, mitjançant aquests gràfics podem tenir una informació general de cadascuna d'aquestes variables categòriques:

1. Ascites: Existeix un desequilibri significatiu de classes, la categoria 'No' és molt més freqüent que la categoria 'Sí', és a dir, d'entre tots els pacients, són molt pocs els que tenen ascites.
2. Drug: La distribució entre 'D-penicillamine' i 'Placebo' és bastant equilibrada, la qual cosa suggereix que es va voler que la meitat dels pacients prenguessin placebo i l'altra meitat penicil·lamina per tal de veure si hi havia diferències.
3. Edema: Hi torna a haver un desequilibri entre ambdues classes; 'No' és la categoria predominant, mentre que 'Sí' i 'Lleu' són molt menys freqüents. Per tant, la majoria de pacients no tenien presència d'edema.
4. Hepatomegaly: Les classes tenen gairebé el mateix nombre de mostres, sent 'Sí' lleugerament més predominant.

5. Sex: Hi ha moltes més dades de pacients femenins que masculins, el que també podem veure imprimint els valors del feature, en què concretament hi ha 374 dones i 44 homes (ho podem veure a l'executar `df['Sex'].value_counts()`).
6. Spiders: Similar a 'Ascites', hi ha molts més 'No' que 'Sí'.
7. Stage: Hi ha un desequilibri entre les quatre etapes, amb '3.0' i '4.0' sent més comunes que '2.0' i '1.0'.

Com hem vist, hem pogut extreure diverses conclusions a l'hora de veure la freqüència de les diferents categories de cada variable categòrica, i hem vist que mentre hi ha algunes variables que tenen les categories molt balancejades, n'hi ha d'altres que no, però ara el que realment importa és si la variable objectiu, 'Status', està o no balancejada, ja que en el cas que no ho estigui haurem d'aplicar algun mètode per a balancejar les seves classes.

2.2.2 Anàlisi de la variable objectiu 'Status'

La variable objectiu 'Status' consta de tres categories: C, D i CL, amb les següents freqüències respectivament: 232, 161 i 25.

Podem observar ràpidament que hi ha una desigualtat evident en la representació de les classes. Mentre que les categories C i D tenen una presència significativa en el conjunt de dades, la categoria CL està notablement subrepresentada. Aquest desequilibri en les classes pot ser que condueixi a certs problemes durant l'entrenament del model, com ara:

- **Biaix en el Model:** El model pot tendir a predir millor les classes més representades, és a dir, C i D, mentre que la classe minoritària CL pot ser ignorada o mal predita.
- **Avaluació Inexacta:** Les mètriques d'avaluació (accuracy, precision, f1-score...) del model probablement no reflecteixin de manera correcta el rendiment sobre les classes minoritàries.

Així doncs, serà necessari aplicar algun mètode de balanceig de classes. Aquest balanceig pot ajudar a solventar els problemes comentats anteriorment, i millorar així la capacitat del model per aprendre de totes les classes de manera equitativa.

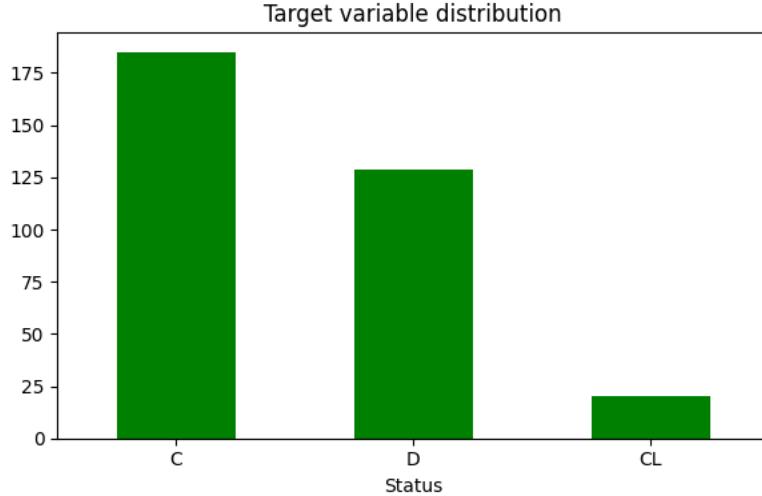


Figura 3: Freqüència per classe de la variable objectiu 'Status'

Gràficament també podem visualitzar aquest desbalanceig tan evident entre categories, per tant, un cop es particioni el dataset, haurem d'aplicar el mètode adient per tal de balancejar les classes de 'Status', ara només falta escollir quin.

2.2.3 Quin mètode aplicar

Per solventar aquest desbalanceig tan evident en la variable target, he decidit que utilitzaré el mètode de oversampling **SMOTENC** (Synthetic Minority Over-sampling Technique for Nominal and Continuous data), que és una extensió de l'algorisme SMOTE original. Aquesta versió està dissenyada per a treballar amb conjunts de dades que tenen un mix de variables numèriques i categòriques, i per tant és ideal per aquest dataset, ja que així podem tractar directament els dos tipus de features (numèriques i categòriques).

L'ús de SMOTENC permetrà generar mostres de les classes minoritàries de manera que s'ajustin tant a les variables numèriques com a les categòriques, lo qual segurament ajudi a millorar el rendiment del model en aquestes classes minoritàries, sense perdre informació important de les variables numèriques o modificar les propietats originals de les característiques categòriques.

Per tant, la implementació de SMOTENC millorarà la precisió del model en la nostra variable objectiu, i també assegurarà una representació més justa i equilibrada de totes les categories de 'Status' en el model final, raó per la qual crec que és una molt bona opció pel balanceig de classes d'aquesta variable.

2.3 Anàlisi de dades mancants (Missing Data)

Per tal de construir un bon model predictiu, l'anàlisi dels missings és un pas molt important que pot influir considerablement en la qualitat i fiabilitat dels resultats. En aquest dataset, la presència de dades mancants pot ser deguda a diverses raons, com ara errors en la recollida de dades, pèrdua

de registres, o la no resposta en algunes columnes dels pacients. Per tant, el tractament adequat d'aquestes dades és essencial per assegurar l'integritat i eficàcia del model.

Així doncs, amb l'objectiu de tenir una visió general de quina quantitat de missing data té cada feature i quin percentatge correspon al mateix, he creat la funció `missing_data(data)`, que crea una taula en què es mostra, per a cada feature, el número total de missing data que té i el percentatge corresponent. Per tant, en aplicar aquesta funció al dataset, obtenim la següent taula:

Feature	Total Missing	Missing in %
Tryglicerides	136	32.54
Cholesterol	134	32.06
Copper	108	25.84
SGOT	106	25.36
Alk_Phosphatase	106	25.36
Drug	105	25.12
Ascites	105	25.12
Hepatomegaly	105	25.12
Spiders	105	25.12
Platelets	11	2.63
Stage	6	1.44
Prothrombin	2	0.48
N_Days	0	0.00
Albumin	0	0.00
Status	0	0.00
Edema	0	0.00
Sex	0	0.00
Age	0	0.00
Bilirubin	0	0.00

Taula 3: Percentatge i nombre de missing data per cada feature

Com podem veure, hi ha fins a 9 features que tenen més d'un 25% de missing data, 3 features que tenen entre menys d'un 3% i els features restants, que són 7, no tenen cap missing. Per tant, hem d'analitzar quina d'aquestes dues opcions és millor prendre per a cada feature:

1. Imputar els missings: Substituir les dades mancants per uns certs valors (estimats o calculats). L'objectiu és conservar totes les observacions substituint les parts incompletes amb valors basats en la resta del dataset.
El principal avantatge d'imputar els missings és la preservació de les dades, ja que es manté el nombre d'observacions, evitant així la pèrdua d'informació. Per contra, això també pot fer que s'introdueixi biaix en l'anàlisi, ja que la imputació pot introduir valors que no reflecteixen la realitat de les dades.
2. Eliminar els missings: Implica descartar les observacions (files) o variables (columnes) del dataset que continguin missings. En aquest cas, com que no es realitzen estimacions, no hi ha risc de biaix introduït per imputacions incorrectes. En contraposició, eliminar els missings pot resultar en una pèrdua important de les dades, en especial si el nombre de missings és gran.

Tenint això en compte, veiem què passaria si eliminéssim totes les observacions que tinguin algun feature amb missing data, és a dir, en el cas que decidíssim eliminar les observacions amb missings ens quedaria el següent dataset:

- **Files sense missings:** 276
- **Percentatge de files sense missings:** 66.03%

Com podem veure, si eliminem totes les observacions que tenen missings, estaríem eliminant 1/3 part del total d'observacions, la qual cosa provocaria que es perdria moltíssima informació, per tant aquesta opció no és viable per a les 9 variables que tenen més d'un 25% de missings (ja que com a mínim s'eliminaria un 25% del dataset). En canvi, si només eliminéssim les observacions que tenen missings en les tres variables amb menys d'un 3% de dades mancants (Platelets, Stage i Prothrombin), no estaríem perdent massa informació, però si imputem els missings tampoc estaríem afegint massa biaix en l'anàlisi, doncs el nombre de missings en aquestes variables és molt baix com per a afectar negativament el model. Per tant, a priori qualsevol d'aquestes dues opcions sembla bona per a aquestes tres variables.

Així doncs, cal decidir com tractar els missings dels features amb més d'un 25% de missings; si imputar, o eliminar directament els features en qüestió.

Com que es tracta d'un estudi clínic, tots els features són importants a l'hora d'analitzar els seus efectes a la variable objectiu (Status) o veure quines correlacions hi ha entre ells. És per aquesta raó que, finalment, la millor opció serà imputar els missings per als features **Tryglicerides, Cholesterol, Copper, SGOT, Alk.Phos, Drug, Ascites, Hepatomegaly**. Com he dit, aquesta decisió es basa en la importància d'aquestes variables en la comprensió de la malaltia hepàtica i la seva relació amb la supervivència dels pacients, ja que si eliminem les variables o les observacions estaríem perdent informació rellevant. Així doncs, la imputació es realitzarà un cop particionat el dataset.

Per a les característiques **Platelets, Stage, i Prothrombin**, amb menys d'un 3% de dades mancants, també es procedirà amb la imputació. Com que els features tenen un percentatge molt baix de missing data, és poc probable que aquesta imputació introduixi un biaix significatiu.

Per tant, un cop particionem el dataset, imputarem tots els missings de cada partició per als diferents features que tenen dades mancants.

2.4 Outliers

2.4.1 Criteri per a definir quins valors són outliers

Per tal de definir, per a cada variable numèrica, què és un outlier i què no, seguirem el criteri que es fa servir en els boxplots per marcar quins valors són outliers. En aquesta secció es detalla com aplicar aquest criteri al dataset i, un cop identificats els outliers, s'haurà de decidir si conservar-los o no.

Els boxplots són una eina gràfica que permet visualitzar la distribució de les dades, la seva centralitat i dispersió, així com detectar els valors atípics. Aquest mètode es basa en el concepte de rang interquartílic (IQR), que és la diferència entre el tercer quartil (Q3) i el primer quartil (Q1).

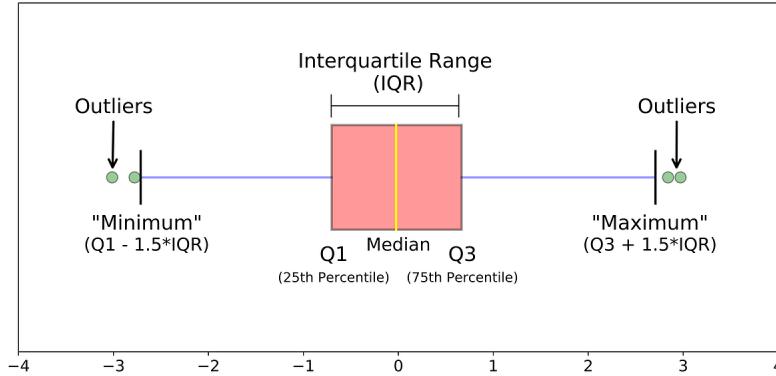


Figura 4: Descripció visual del criteri dels outliers en un boxplot

El primer quartil (Q1) correspon al punt en què el 25% de les dades són menors que aquest valor i el 75% són majors. Inversament, el tercer quartil (Q3) es troba on el 75% de les dades són menors que aquest valor i el 25% són majors. D'aquesta manera, Q1 i Q3 marquen els punts de 25% i 75% en la distribució de les dades, respectivament.

Per tant, el criteri per decidir quins valors són outliers és utilitzar la distància entre els quartils, en què es consideraran outliers:

- Els valors més petits que $Q1 - 1.5 \times IQR$
- Els valors més grans que $Q3 + 1.5 \times IQR$

2.4.2 Comparació i anàlisi dels features amb i sense outliers

Així doncs, un cop definits a partir de quins límits un valor es considera outlier per a cada variable, podem veure quants outliers té cadascuna:

- N_Days: 0
- Age: 0
- Bilirubin: 46
- Cholesterol: 20
- Albumin: 9
- Copper: 17
- Alk_Phosphatase: 35
- SGOT: 7
- Tryglicerides: 10
- Platelets: 6
- Prothrombin: 18

Per tant, per als features que sí tenen outliers (és a dir, excloem N_Days i Age, ja que no tenen cap outlier següent aquest criteri), volem veure les diferències gràfiques del feature amb i sense outliers, per la qual cosa plotegem, per cada variable, un boxplot junt amb un histograma de la seva distribució amb outliers i a continuació sense outliers:

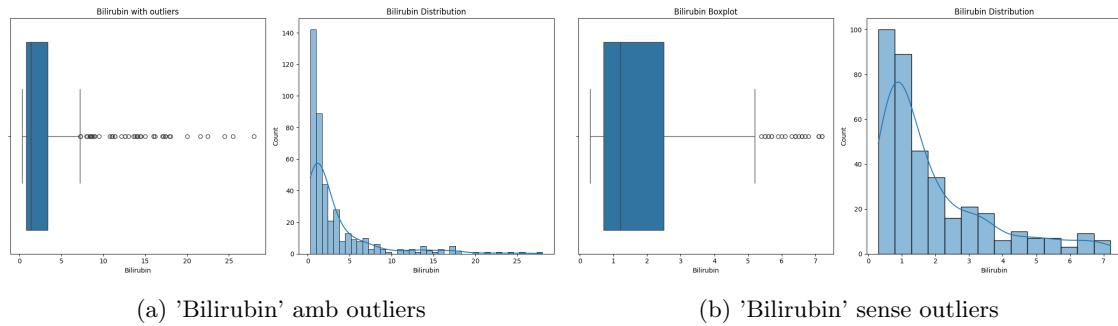


Figura 5: Diferències entre 'Bilirubin' amb i sense outliers

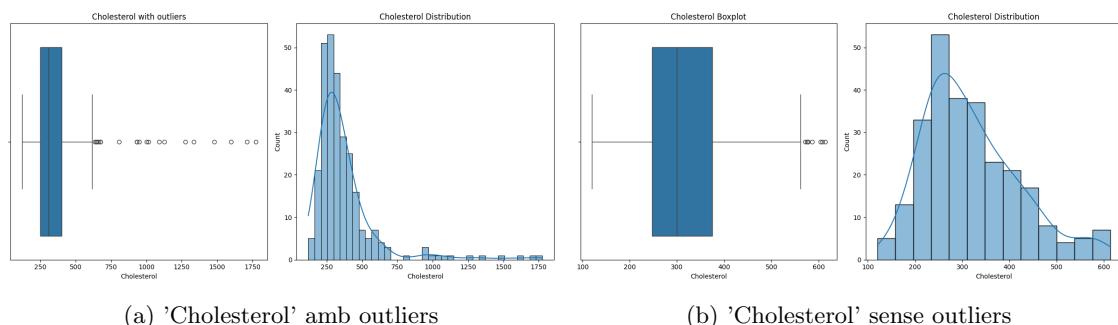


Figura 6: Diferències entre 'Cholesterol' amb i sense outliers

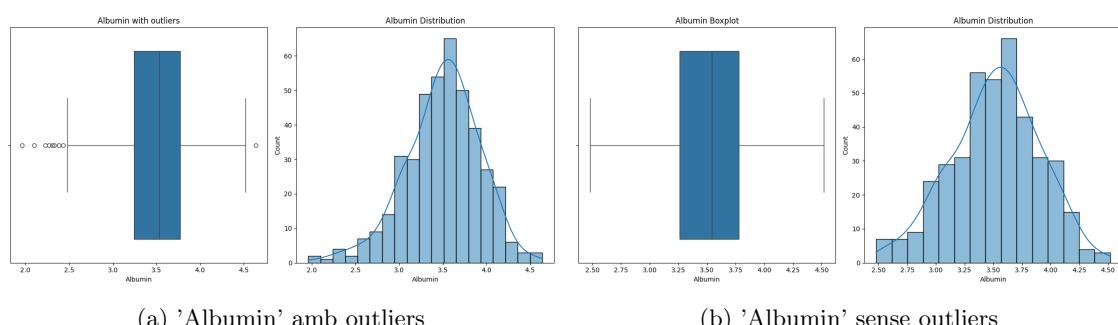


Figura 7: Diferències entre 'Albumin' amb i sense outliers

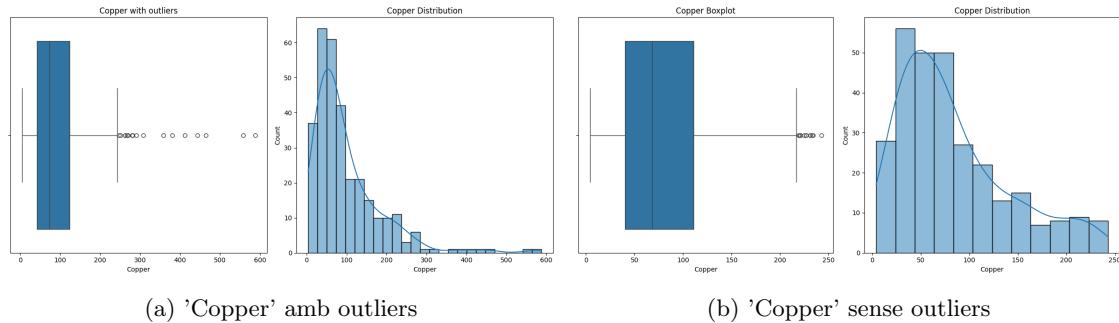


Figura 8: Diferències entre 'Copper' amb i sense outliers

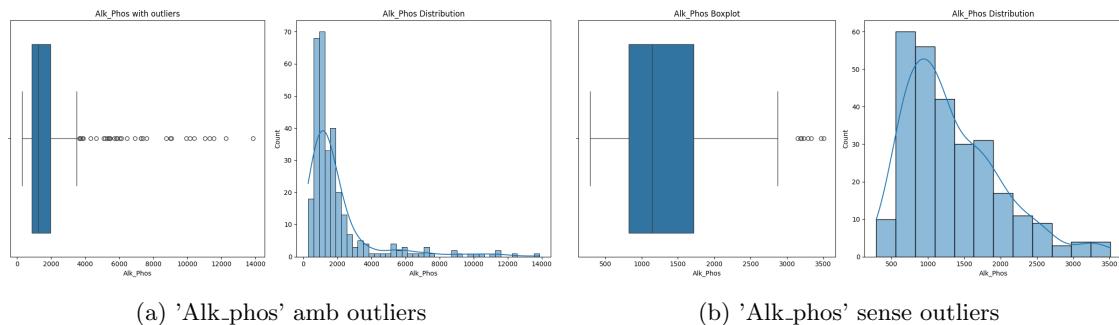


Figura 9: Diferències entre 'Alk_phos' amb i sense outliers

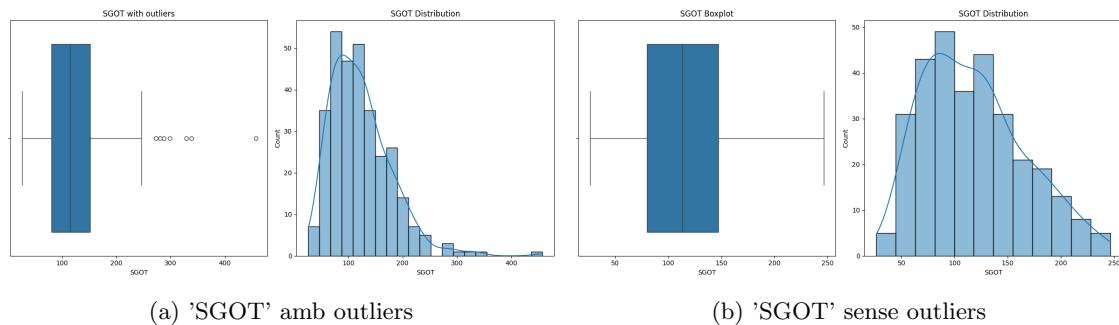


Figura 10: Diferències entre 'SGOT' amb i sense outliers

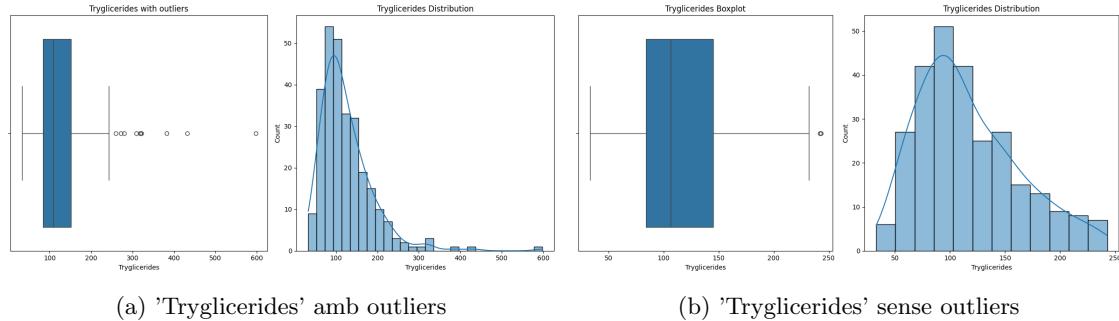


Figura 11: Diferències entre 'Tryglicerides' amb i sense outliers

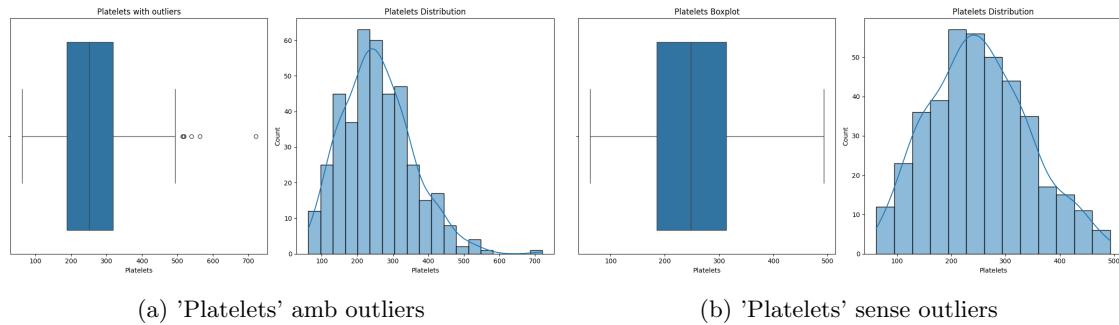


Figura 12: Diferències entre 'Platelets' amb i sense outliers

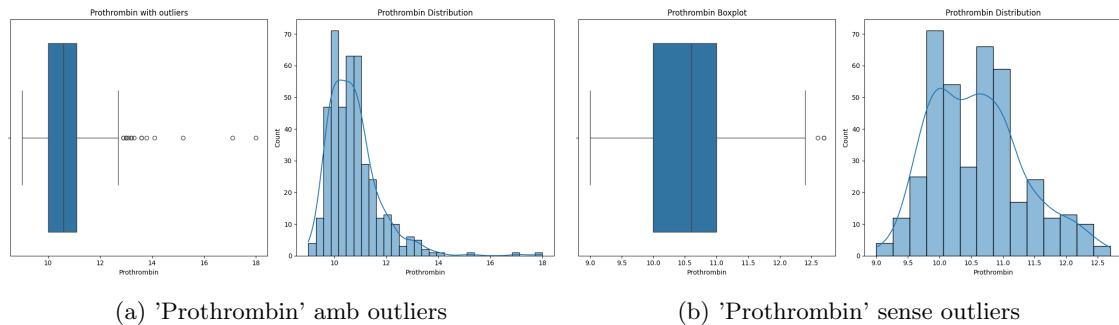


Figura 13: Diferències entre 'Prothrombin' amb i sense outliers

Com podem veure en la majoria dels gràfics, hi ha una clara diferència en la distribució de les variables amb i sense outliers, ja que el fet de no projectar outliers fa que el rang de valors representats en els plots sigui menor, i per tant les distribucions estiguin millor representades.

A més, això es deu a què els outliers poden causar que una distribució sigui asimètrica i afecten a mesures com la mitjana o la mediana. Sense ells, aquestes mesures representen millor la tendència central de les dades, lo que provoca que les distribucions siguin més normalitzades.

Cal remarcar, però, que els valors extremadament alts o baixos en aquests features no són necessàriament errors, sinó que poden ser causats per la pròpia malaltia. És a dir, realment no podem saber si els valors que seguit aquest criteri es marquen com a outliers són condicions mèdiques reals dels pacients o són errors, i per tant, hem de decidir si mantenir-los o eliminar-los del dataset.

Com que els features representen condicions mèdiques dels pacients (observacions), el que hauríem de fer és consultar amb un expert per tal de saber si són rellevants o no, ja que potser podrien ser informació important al representar casos molt extrems en els pacients.

En aquest treball, però, eliminarem tots els outliers per estadística, ja que seguit el criteri explicat anteriorment, en considerar que els valors detectats són outliers, estadísticament els hem d'apartar del dataset.

Malgrat que el fet de mantenir aquests outliers millorés la seva capacitat de detectar casos atípics, si els eliminem estem reduint la variabilitat del model.

Així doncs, finalment he decidit que la millor opció és eliminar els outliers, ja solen al no saber si són dades que poden ser reals o simplement són errors, seguit l'estadística els apartem del nostre dataset. Per tant, al eliminar-los, substituïm el valor per un NA, és a dir, una dada mancant. D'aquesta manera, els nombre de missings del dataset augmentarà i hi haurà variables que, malgrat en l'estudi anterior hagim vist que no tenien cap dada mancant, ara sí que tindran alguns missings. Tot i així, com que hem decidit que imputarem tots els missins per a totes les variables, això no influeix a la nostra decisió ja que aquests 'nous' missings també seran imputats després de particionar el dataset. Per tant, al cap i a la fi, estem substituïnt el valor de cada outlier per a un nou valor que serà imputat en l'apartat d'Imputació de missings (3.4).

D'aquesta manera, un cop eliminats els outliers de cada feature, podem observar com queden ara els valors màxims i mínims, així com la mitjana de totes les variables numèriques:

Feature	Min	Mean	Max
N_Days	41.00	1917.782297	4795.00
Age	9598.00	18533.351675	28650.00
Bilirubin	0.30	1.897849	7.20
Cholesterol	120.00	318.041667	614.00
Albumin	2.48	3.519022	4.52
Copper	4.00	83.105802	243.00
Alk_Phosphatase	289.00	1331.754513	3510.00
SGOT	26.35	117.945016	246.45
Tryglicerides	33.00	116.448529	243.00
Platelets	62.00	252.461347	493.00
Prothrombin	9.00	10.590955	12.70

Figura 14: Valors mínim, màxim i mitjana pels features numèrics sense outliers.

Com podem observar en aquesta taula, ara els valors màxims ja no són tan extrems, i inclús són molt més propers a la mitjana de cada variable. Per tant, ja hem eliminat satisfactoriament tots els outliers del dataset.

2.5 Recodificació de variables

Com hem vist en l'apartat 2.1.1, hem hagut de recodificar certes variables categòriques i passar-les a numèriques, ja que havíem vist que tenien una gran quantitat de valors únics, la qual cosa volia dir que estaven sent codificades erròniament com a categòriques i necessitaven una transformació a numèriques per a poder tractar-les com a tal. Així doncs, amb aquesta feina ja feta, ara falta poder codificar les variables categòriques del dataset per tal que en futurs apartats es puguin fer servir en certs algoritmes que no suporten tractar amb strings, per tant aquest és un pas necessari per tal de poder treballar amb aquests features en un futur.

Abans, però, cal remarcar que en analitzar les variables de manera independent, observant la taula de les variables numèriques (1) em vaig fixar que 'Age' tenia uns valors massa alts, per la qual cosa em pensava que hi havia algun tipus d'error. Per tant, vaig buscar a la pàgina web del dataset i, efectivament, les unitats d'aquest feature són dies, per tant crec que és una bona opció transformar-la a anys, ja que d'aquesta manera els resultats obtinguts seran molt més interpretables, doncs és més fàcil treure conclusions si l'edat d'un pacient són 47 anys, que no si són 17166 dies.

Per tant, he dividit el valor de la variable entre 365.25 (els dies que té un any de mitjana), obtenint així el valor final en anys.

```
df[‘Age’] = df[‘Age’] / 365.25
```

Per tal que aquesta variable no tingui decimals, ja que si només s'executa aquesta sentència apareixen valors amb molts decimals (e.g. 44.520192), també ha sigut necessari transformar la variable a tipus 'integer'.

Ara, si fem un anàlisi estadístic d'aquest feature podem veure que la mitjana són 50.29 anys, amb un error estàndard de 0.47, i els seus valors mínim i màxim són 26 i 78, respectivament. És a dir, el pacient més jove del dataset tenia 26 anys i el més gran en tenia 78.

Així doncs, aquest feature mostra, per cada pacient, el nombre d'anys que té com a un valor enter, fent que posteriorment els resultats que obtingui relacionats amb aquest feature siguin molt més comprensibles.

Ara sí, ja podem procedir a codificar els features categòrics.

2.5.1 Codificació de les variables categòriques

Per a codificar les diferents variables categòriques del dataset, he decidit utilitzar el mètode de codificació anomenat **One Hot Encoder**, el qual serveix per a convertir les diverses variables categòriques en un conjunt de variables binàries, una per cada categoria de la variable original. En aquest mètode, cada categoria es representa amb una columna nova, on el valor és 1 si l'observació pertany a aquella categoria, i 0 si no hi pertany. Aquesta tècnica elimina l'ordre que podria imposar-se amb la codificació numèrica, evitant així que el model interpreti erròniament les dades categòriques com a ordinàries o intervals.

Per tal d'implementar aquest mètode sense que modifiqui directament el dataset, he decidit crear una funció anomenada **one_hot_encode**, la qual aplica aquest mètode al dataset que se li doni com a argument. També rep com a paràmetres una llista de noms de columnes categòriques (**categorical_features**). Per a cada columna, es creen 'dummies' o variables indicadores, fent servir la funció **pd.get_dummies()** de pandas. Aquestes variables dummies substitueixen la columna categòrica original, resultant en un DataFrame amb les variables categòriques codificades de manera que són aptes per a l'anàlisi amb mètodes estadístics i algorismes d'aprenentatge automàtic. Haver-ho fet en una funció em permetrà reutilitzar el codi fàcilment, ja que si necessito realitzar la mateixa tasca de codificació en diferents datasets (un cop particioni les dades, amb el conjunt de

train o test), la funció permetrà fer-ho sense haver de reescriure el codi una altra vegada.

Sí que és cert que el mètode 'One Hot Encoding' pot augmentar significativament la dimensionalitat del conjunt de dades quan es tracta de variables amb moltes categories, però com hem analitzat anteriorment, només tenim 7 variables categòriques i totes tenen entre 2 i 4 categories, per la qual cosa tampoc s'augmentaran tant les dimensions del dataset que se li apliqui aquesta funció.

Per últim, com que les variables categòriques no són ordinals, 'One Hot Encoding' és una molt bona opció ja que al crear una nova columna per a cada categoria, representant la presència o absència de cada categoria amb 1 o 0, permet que el model tracti cada categoria com independent sense imposar cap ordre, mentre que un altre mètode com 'Label Encoding', que assigna a cada categoria un valor numèric únic (0, 1, 2, ...), pot ser interpret pels models d'aprenentatge automàtic com una relació ordinal, en què els nombres més alts poden ser vistos com 'millors' o 'majors' que els nombres més baixos, per tant en aquest punt la millor opció per a crear la funció ha estat 'One Hot Encoding'.

2.6 Acabant el Preprocessament

Així doncs, un cop ja hem acabat de preprocessar el dataset, podem procedir a veure les correlacions de les diferents variables categòriques i numèriques amb la variable objectiu 'Status', així com les variables sorolloses o redundants per tal de veure quines podem eliminar del model, i un cop fet aquest procés, ja podrem particionar el dataset, imputar missings i balancejar la variable objectiu.

3 Preparació de variables

3.1 Anàlisi de correlacions entre variables numèriques

En aquest punt del treball tractarem de veure si hi ha certs features que estiguin molt correlacionats entre ells, lo qual ens pot indicar que, en el cas que hi existeixin variables altament correlacionades entre sí, aquestes segurament proporcionen informació similar, per lo que pot provocar una certa redundància en el model, i per tant podríem eliminar una de les dues variables per tal de millorar la generalització del model.

Per tant, utilitzant la funció `pairplot` de Seaborn, he generat un gràfic per a cada parell de variables numèriques, lo qual permet visualitzar les relacions i distribucions de cada variable en comparació amb les altres, i tenir una vista general de si hi ha alguna parella de variables amb una alta correlació.

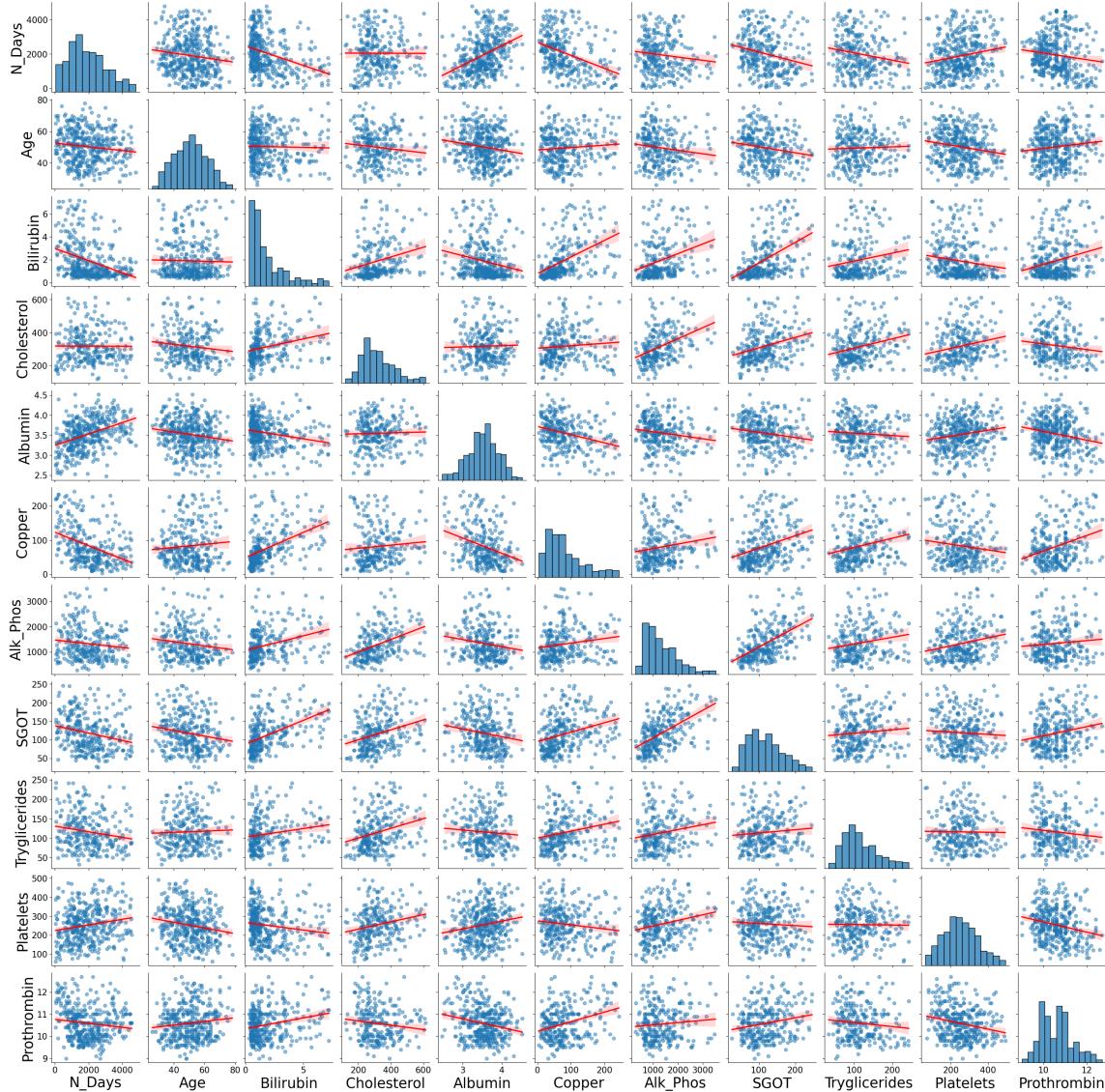


Figura 15: Gràfics de correlacions per cada parell de variables numèriques

A primera vista, podem observar que no hi ha cap gràfic en el que es vegi una alta correlació per parella de variables, ja que en la majoria d'ells els punts estan bastant dispersos i no hi ha un patró clar, tot i que sí que es podria destacar, per exemple, el gràfic SGOT-Alk_Phosphatase, doncs sembla que hi ha una certa tendència (també destacada amb la línia de regressió), tot i que tampoc es poden treure moltes conclusions.

Per tant, per tal de veure estadísticament la correlació per cada parella de features numèrics, podem calcular la matriu de correlació utilitzant el mètode `corr()` i posteriorment la visualitzem mitjançant un mapa de calor, i així veure del cert quins features estan realment correlacionats.

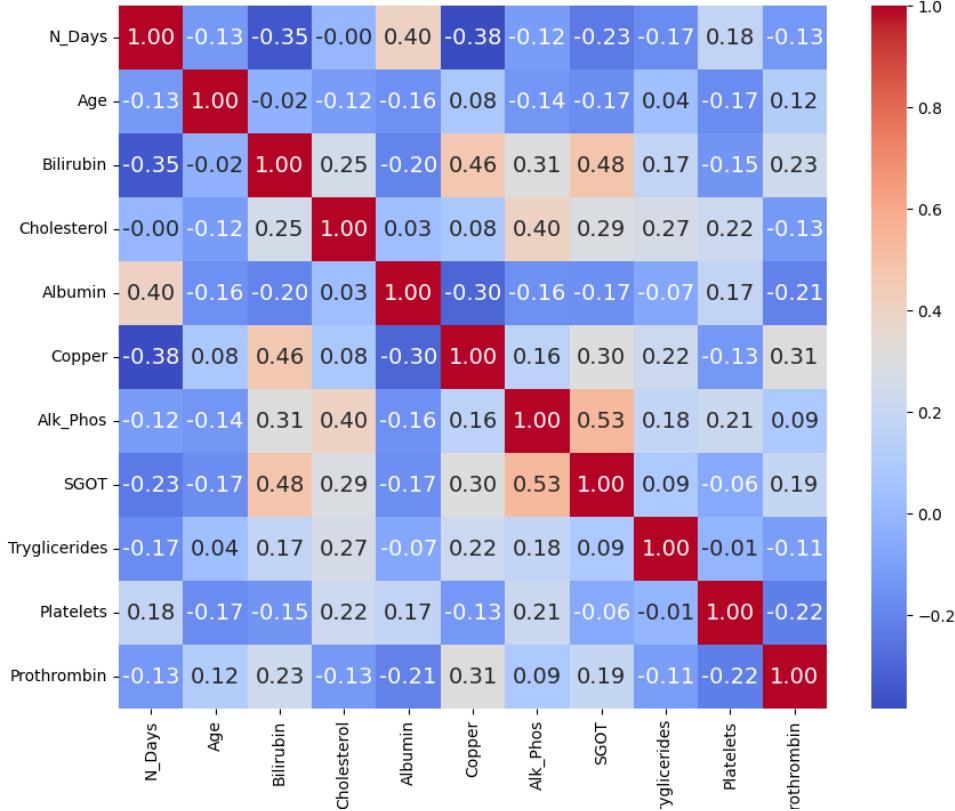


Figura 16: Mapa de calor de la matriu de correlació de les variables numèriques

Com podem veure, aquest mapa utilitza diferents intensitats de color per representar la força i la direcció de les correlacions entre les variables, i podem observar que no hi ha cap parella de variables que tingui un coeficient de correlació de valor absolut més gran o igual que 0.5 excepte 'Alk_Phosphatase' i 'SGOT', tot i que tampoc és un coeficient excessivament alt, per la qual cosa no hi ha cap variable que estigui altament correlacionada amb una altra com per a eliminar una de les dues. Tot i això, podem veure una correlació negativa entre 'Copper' i 'N_Days' amb un valor de -0.38, indicant que a mesura que 'Copper' augmenta, 'N_Days' tendeix a disminuir. També podem trobar una correlació positiva moderada entre 'Albumin' i 'N_Days', amb un coeficient de correlació de 0.40, suggerint que valors alts en 'Albumin' poden estar associats amb valors alts en 'N_Days'. Per últim, algunes variables com 'Tryglicerides' i 'Platelets' mostren una correlació molt baixa (-0.01), lo qual indica que no tenen gairebé cap relació lineal entre elles.

Així doncs, podem concloure que, en aquest punt, no cal que eliminem cap variable numèrica ja que no hi ha cap parella de features que tingui un coeficient de correlació prou alt com per a eliminar una de les dues variables.

3.2 Anàlisi de variables categòriques i variable objectiu

En aquest apartat es tracta d'identificar quines variables categòriques tenen una relació significativa amb la variable 'Status', i així poder descobrir factors importants que influencien en la variable objectiu, ja que això ens permetrà identificar aquelles que són més predictives i, per tant, més valuoses per al model.

Per tant, procedim a visualitzar els gràfics de barres per a cada variable categòrica en relació amb la variable objectiu, i veure com la freqüència de cada categoria varia en funció de la variable objectiu.

Drug

Com podem observar, els pacients tractats amb D-penicil·amina i els que reben placebo tenen distribucions pràcticament idèntiques en quant a la variable objectiu, la qual cosa ens indica que la variable 'Drug' no proporciona informació significativa que diferenciï els resultats de la variable objectiu. Per tant, per tal de millorar la generalització del model i reduir el risc de overfitting (afectant el mínim al rendiment), he decidit treure aquesta variable del dataset, doncs s'ha pogut comprovar amb aquest gràfic que no influeix en la variable objectiu.

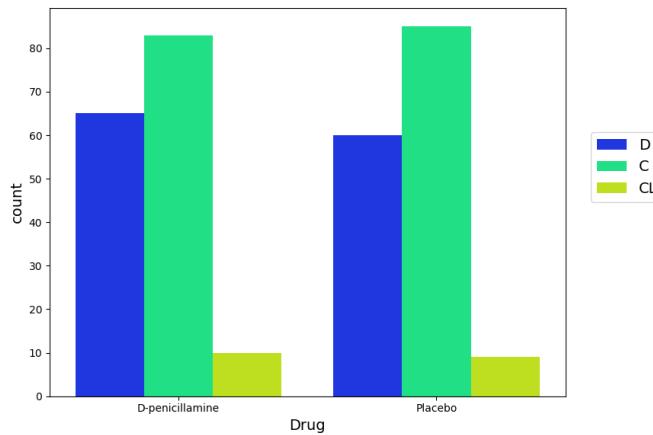


Figura 17: 'Drug' vs 'Status'

Sex

En el cas de la variable **Sex**, és interessant veure que, malgrat el nombre de dones sigui molt més gran que el de homes, una porció gran d'aquests últims pertanyen a la categoria 'D', seguida de 'C'. Per altra banda, per les dones és clarament 'C' la categoria més freqüent, amb una gran diferència respecte 'D'. També cal destacar que en tots dos casos, 'CL' és la classe menys freqüent amb diferència.

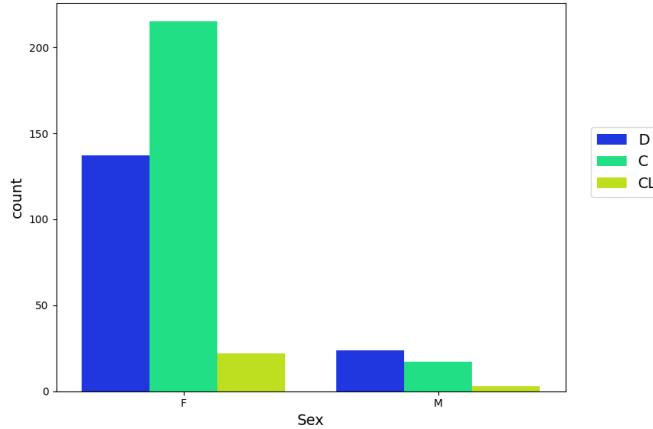


Figura 18: 'Sex' vs 'Status'

Ascites

Com podem veure, 'Ascites' es tracta d'una variable amb les categories molt desbalancejades, ja que hi ha moltes més mostres de pacients que no pateixen aquesta malaltia que d'altres que sí en pateixen. Tot i així, podem veure que en el cas dels pacients que la pateixen, gairebé tots pertanyen a la categoria 'D' de Status, és a dir, la majoria de pacients que pateixen aquesta malaltia estaven morts quan es va fer l'estudi. Per altra banda, en el cas dels pacients que no pateixen 'Ascites' podem veure que és a l'inrevés, en què hi ha menys mostres de pacients amb 'D' a Status, i en canvi el més comú és que pertanyin a la categoria 'C'. Per altra banda, no hi ha cap pacient que pertanyi a la categoria 'CL' de Status que pateixi aquesta malaltia.

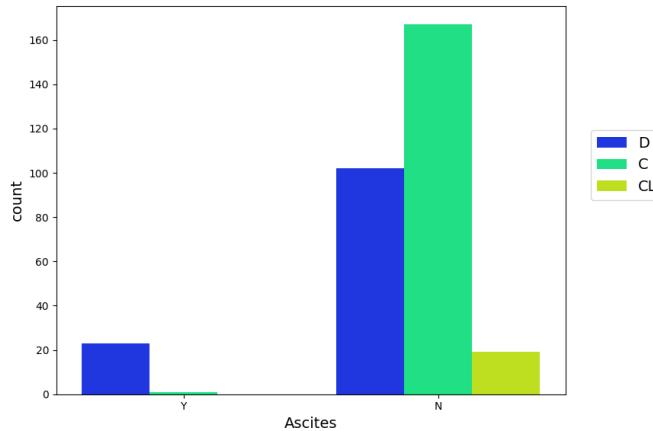


Figura 19: 'Ascites' vs 'Status'

Hepatomegaly

En el cas d'aquesta malaltia podem veure que hi ha una gran diferència entre els pacients que la patien i els que no, ja que observant el gràfic podem veure com hi ha més mostres de pacients amb

hepatomegàlia que tinguin 'D' a **Status** que no pas 'C', mentre que si no patien aquesta malaltia, lo més probable és que pertanyin a la categoria 'C' de la variable objectiu, ja que és amb molta diferència respecte les dues altres classes, la més freqüent en aquest tipus de pacients.

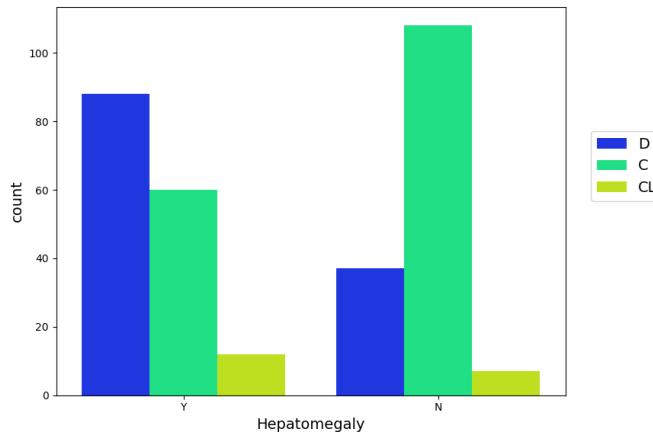


Figura 20: 'Hepatomegaly' vs 'Status'

Spiders

Com es pot observar, proporcionalment hi ha molt més pacients censurats ('C' a **Status**) que no tinguessin aquesta malaltia ('N') en comparació als que sí que la patien ('Y'). També, podem veure que 'D' és la categoria de **Status** més freqüent en els pacients que patien 'Spiders', mentre que 'CL' és la menys freqüent en ambdues.

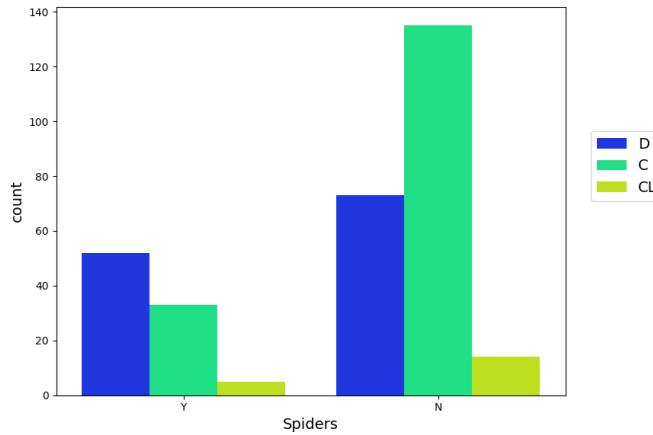


Figura 21: 'Spiders' vs 'Status'

Edema

Aquest feature és molt interessant ja que ens mostra com gairebé tots els pacients que patien 'edema' van morir, en concret un 59.09% dels pacients amb edema lleu ('S') i un 95% dels pacients

amb edema significatiu ('Y') van morir ('D'). Per tant, el grau d'edema sembla que està fortament correlacionat amb la mortalitat dels pacients, amb una taxa molt alta de morts entre els pacients amb edema significatiu. També cal remarcar, però, que es tenen moltes més mostres de pacients que no tenien edema respecte dels que sí que en tenien, i podem veure que en aquest cas la categoria de la variable objectiu més freqüent és 'C', seguida de 'D', sent 'CL' la menys comuna de les tres.

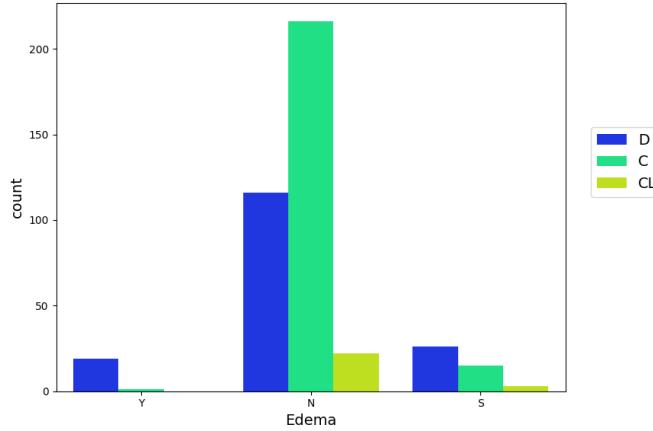


Figura 22: 'Edema' vs 'Status'

Stage

Finalment, la variable 'Stage' veiem que és la que més categories té (4), i podem observar que hi ha certes diferències respecte la variable objectiu en les diferents classes d'aquest feature. Per exemple, gairebé tots els pacients amb '1', '2' o '3' a Stage pertanyen a la classe 'C' de Status, ja que és la més freqüent en tots aquests casos, especialment en els pacients amb '1' a Stage. En contrast, els pacients amb '4' a Stage són més propensos a pertànyer a la categoria 'D' de Status. Si observem el gràfic detingudament, podem observar que el nombre de pacients que pertanyen a 'D' augmenta a mesura que el número de Stage és més gran. Per tant, això vol dir que, a mesura que 'Stage' augmenta, també ho fa el risc de mortalitat del patient.

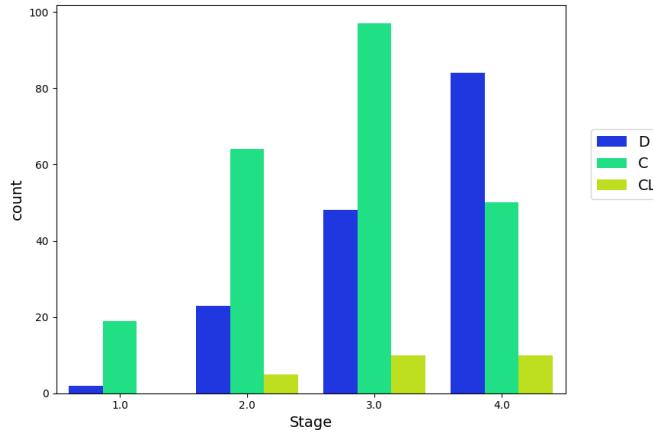


Figura 23: 'Stage' vs 'Status'

Així doncs, l'única variable categòrica que hem eliminat del dataset ha estat **Drug**, ja que com hem vist, tenia molt poca influència en la variable objectiu, doncs les distribucions de les dues categories eren pràcticament les mateixes, la qual cosa ens ha indicat que era una variable que podíem obviar a l'hora de predir 'Status', degut a la seva mínima correlació.

3.3 Particionat del dataset

Un cop ja hem tractat els outliers, hem recodificat les variables necessàries i hem analitzat les diferents variables de manera independent i junt amb la variable objectiu, ja podem partitionar el dataset en dos conjunts: Train i Test.

He decidit no fer una partició de validació ja que, al tenir poques mostres (418), com que el dataset és petit, la creació d'un conjunt de validació podria resultar en que cada conjunt (train, validation, i test) tingués poques dades, la qual cosa afectaria la capacitat del model per aprendre i validar les seves prediccions de manera fiable, ja que probablement el model s'ajustaria massa al conjunt de validació, fent un 'overfitting' en aquest conjunt i provocant que no generalitzés del tot bé.

D'aquesta manera, si particionem només en train i test, ens assegurem que els dos conjunts siguin més representatius de la població total, minimitzant el risc de biaix en el model i optimitzant així l'ús de les nostres dades per a l'entrenament, per tant el model generalitarà millor que no si fessim més particions.

També és important recalcar que, a partir d'aquest punt, no podré usar la partició de test fins l'avaluació del model final. Per tant, no hi podrà haver un 'leakage' de test a train, ja que les dades de test es reservaran per a avaluar la capacitat del model de generalitzar a dades noves, és a dir, dades que no hagi vist mai durant l'entrenament.

La partició del test serà un 20% de les dades (`test_size=0.2`), i el 80%, serà per a l'entrenament. Si ens fixem en la línia de codi corresponent, veiem que s'ha assignat un argument anomenat `stratify: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=33)`

Aquest paràmetre ens assegura que la proporció de les diferents classes en `y` sigui aproximadament la mateixa tant en el conjunt de train com en el de test, la qual cosa pot ser molt útil ja que com hem vist abans, les dades estan bastant desequilibrades.

X es refereix als features que s'utilitzaran per predir la variable objectiu, mentre que y és la pròpia variable objectiu. Així doncs, X és la taula `df` havent eliminat la columna 'Status', i y és únicament la columna 'Status' de la mateixa taula.

Per altra banda, si ens fixem en el paràmetre `random_state=33`, aquest serveix per a que aquesta partició pugui ser reproductible, ja que d'aquesta manera m'asseguro que sempre que s'executi el mateix codi, s'obtindrà la mateixa partició de dades.

Set	Nombre d'Observacions	Nombre de Features
Train	334	17
Test	84	17

Taula 4: Mida de les diferents particions

Si ens fixem en la mida de les particions (Taula 4), veiem que el conjunt de test bastants menys observacions que la partició de train. Concretament, el conjunt de test té 84 files (observacions), en què cada fila representa un pacient diferent, que efectivament correspon al 20% de les 418 files originals. Per altra banda, la partició de train en té 334, que podem deduir que és el 80% de 418. També podem observar que, lògicament, cada partició continua tenint el mateix nombre de features (18).

3.4 Imputació dels missings

3.4.1 Imputació per a variables numèriques

Per a l'imputació de les variables numèriques, he decidit utilitzar el mètode d'imputació basat en KNN (K-Nearest Neighbors).

Després d'haver considerat altres opcions, com la mitja, he conclòs que no era una bona opció, ja que com que hi ha diverses variables que tenen una gran quantitat de missings (més d'un 25%), la imputació per la mitja pot no ser adequada, ja que es pot veure influenciada en gran part pels outliers, la qual cosa provocaria que la imputació no reflecteixi correctament la distribució real de les dades. En canvi, KNN és un mètode no paramètric, que significa que no fa suposicions sobre la distribució de les dades, fet que el fa més flexible i adequat en situacions on la distribució de les característiques no és del tot clara, que és justament el que succeeix en els casos d'aquests features que tenen més d'una quarta part de missing data. Així doncs, per aquests motius el mètode escollit ha estat el KNN, que malgrat pugui ser computacionalment més costós que altres mètodes més senzills, és més precís i representa millor les dades imputades.

Un cop imputats tots els missings per les variables numèriques, podem analitzar l'èxit d'aquesta imputació, basant-nos en diferents característiques estadístiques, com ara la mitjana, l'error estàndard i la mediana abans i després de la imputació per a les variables categòriques.

Feature	Mean (old)	Mean	Std_E (old)	Std_E	Median (old)	Median
N_Days	1916.685629	1916.685629	59.760161	59.760161	1730.00	1730.000
Age	50.485030	50.485030	0.568574	0.568574	51.50	51.500
Bilirubin	1.840541	1.934970	0.093000	0.086215	1.20	1.300
Cholesterol	317.703349	315.912575	7.041542	4.792527	296.00	300.900
Albumin	3.546911	3.543653	0.021346	0.021077	3.56	3.560
Copper	84.782979	87.682635	3.784959	2.845114	69.00	77.000
Alk_Phosphat	1328.345291	1338.410180	44.263549	31.430301	1134.00	1233.600
SGOT	117.722428	118.580485	3.072967	2.320435	110.05	113.947
Tryglicerides	115.342593	115.544910	3.147270	2.137262	106.00	112.200
Platelets	253.106250	252.763473	5.039293	4.851451	249.00	248.500
Prothrombin	10.577918	10.607784	0.040980	0.039995	10.60	10.600

Taula 5: Diferències estadístiques abans (A) i després (D) de la imputació (variables numèriques)

Com podem observar, les variables que no tenien missings, que són *N_Days*, *Albumin*, *Age* i *Bilirubin*, no han patit cap canvi en la mitjana, error estàndard i mediana. Malgrat que això sigui lògic, també és important comprovar-ho ja que així ens assegurem que la imputació no hagi afectat a cap variable que no hauria, és a dir, que no hagi canviat cap valor dels features que inicialment no tenien missings, i efectivament així ha estat.

Ara sí, en els features que hi ha hagut una imputació dels missings, s'ha de destacar les mínimes diferències que han patit gairebé totes les variables, ja que en general s'observa una certa estabilitat en les mitjanes, els errors estàndard i les medianes d'abans i després de la imputació. Això indica que els valors imputats són prou coherents amb les tendències i les distribucions originals de les dades, la qual cosa indica que la imputació ha estat bastante efectiva per a aquests features.

3.4.2 Imputació per a variables categòriques

En el cas de les variables categòriques, es farà servir l'estrategia de la **moda**, que consisteix en substituir els valors mancats per el valor més freqüent en cada variable categòrica.

En primer lloc, la moda assegura que el valor imputat sigui un valor que existeixi dins de la categoria, mantenint així la integritat dels tipus de dades. És a dir, a diferència de les variables numèriques, en què es poden generar valors que no existeixen en realitat, la moda garanteix que el valor imputat sigui un que ja ha estat observat dins del conjunt de dades, ja que pertanyerà a una de les classes del feature.

A més, aquesta estratègia és molt útil sobretot en casos on hi ha categories que predominen significativament sobre les altres, ja que sovint les categories més freqüents tenen una major probabilitat de ser representatives de la mostra general.

Per tant, em sembla una bona opció a implementar en aquest dataset, ja que les variables categòriques presenten una o diverses categories amb una freqüència considerablement més alta que la resta. Per tant, la substitució del missing per la moda representa les tendències del dataset, mantenint així la consistència i representativitat de les dades.

Un cop imputats tots els missings per a les variables categòriques, podem visualitzar si hi ha hagut alguna diferència en la distribució d'aquestes abans i després de la imputació.

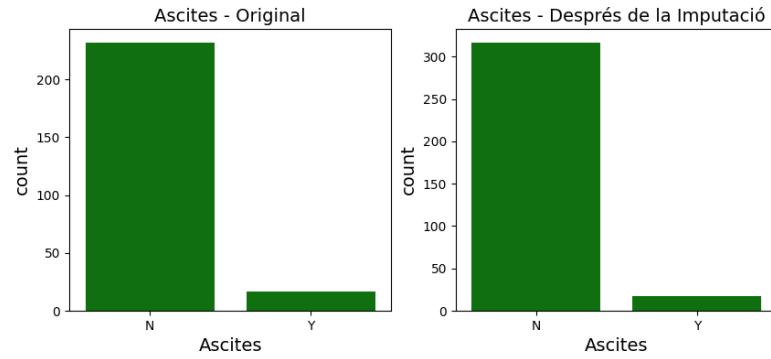


Figura 24: Distribució de 'Ascites' abans i després de imputar amb la moda

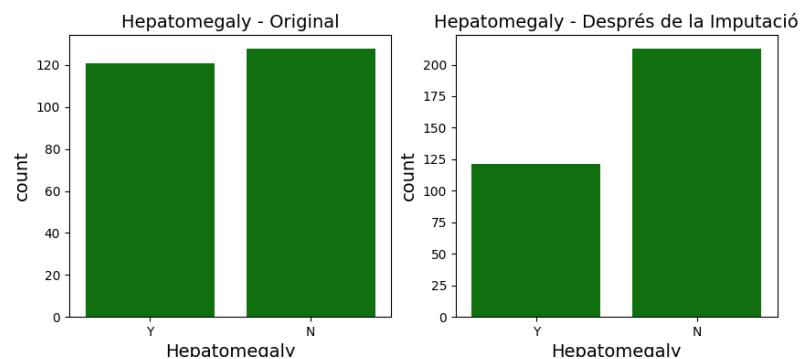


Figura 25: Distribució de 'Hepatomegaly' abans i després de imputar amb la moda

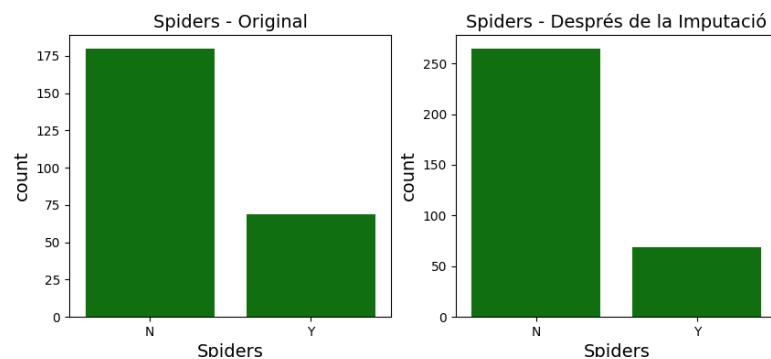


Figura 26: Distribució de 'Spiders' abans i després de imputar amb la moda

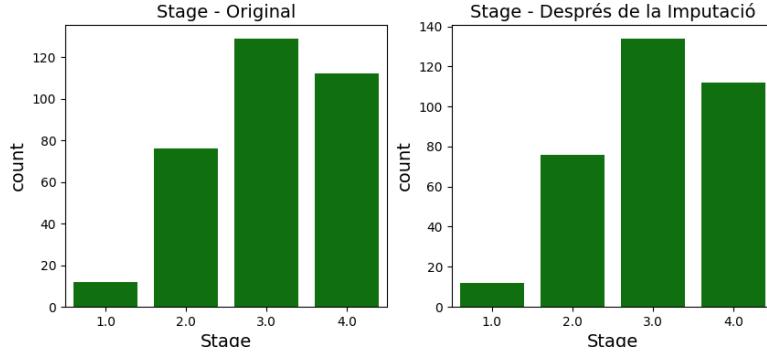


Figura 27: Distribució de 'Stage' abans i després de imputar amb la moda

Com podem veure, els resultats obtinguts en imputar els missings amb la moda mostren que les classes d'alguns feature s'han desbalancejat encara més i no s'ha conservat la proporció original de les categories. Per tant, es pot concloure que aquest mètode finalment no ha estat del tot una bona opció.

Així doncs, he decidit utilitzar el mètode d'**imputació proporcional** per veure si es soluciona aquest problema de mantenir la proporció entre classes. Aquest nou mètode consisteix en imputar els missings d'una variable categòrica basant-se en la distribució de freqüència d'aquesta feature en el conjunt de dades de train.

El principal avantatge és que respecta la distribució original de les categories, evitant el overfitting en una sola categoria que podria ocurrir amb la imputació per la moda.

D'aquesta manera, la imputació proporcional assegura que la representació de cada categoria en el conjunt de dades imputat segueixi sent fidel a la seva representació en el conjunt de dades original. Això és molt important pel model, ja que permet aprengui de manera més precisa les relacions entre els features i la variable objectiu.

Cal remarcar que aquesta funció únicament utilitza la informació del conjunt de train per imputar els missings tant en el conjunt de train com en el conjunt de test.

Pren tres arguments: ‘train_df’, ‘test_df’ i ‘feature’. Per a la característica especificada, calcula la distribució de freqüències dels valors en el conjunt d’entrenament. Després, per a cada missing de la característica del conjunt de train, selecciona un valor a l’atzar de la distribució de freqüències i l’utilitza per omplir el valor que falta.

El mateix procés es repeteix per al conjunt de test, però la distribució de freqüències utilitzada segueix sent la del conjunt d’entrenament. Això és important per evitar el ”data leakage”, ja que no s’ha d’utilitzar la informació del test per influir en el procés de train.

Per tant, si imputem els missings amb aquest mètode, obtenim els següents resultats:

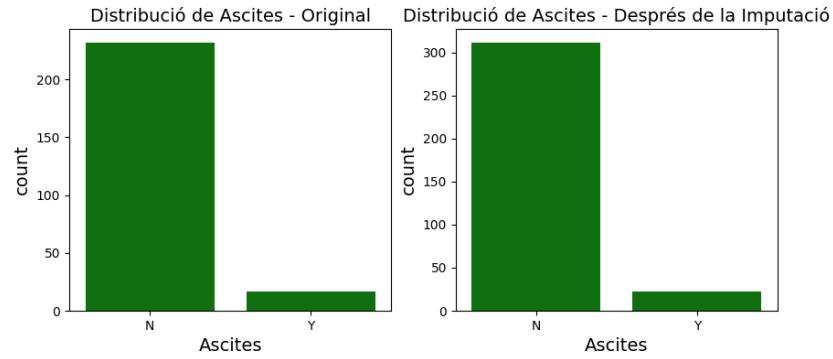


Figura 28: Distribució de 'Ascites' abans i després de la imputació proporcional

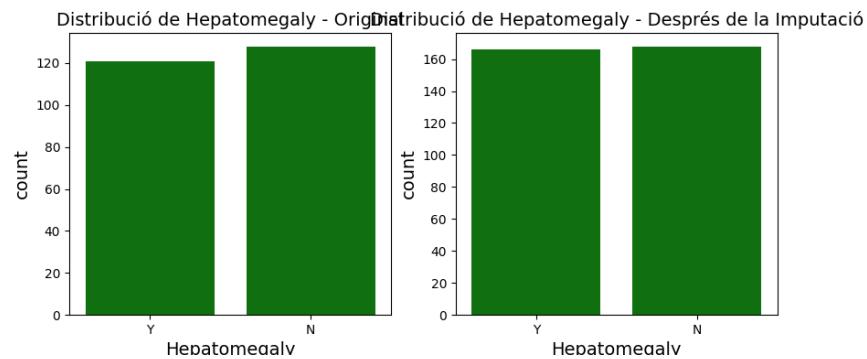


Figura 29: Distribució de 'Hepatomegaly' abans i després de la imputació proporcional

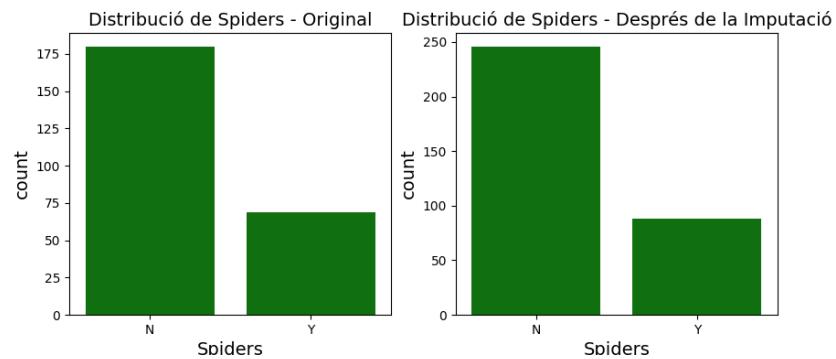


Figura 30: Distribució de 'Spiders' abans i després de la imputació proporcional

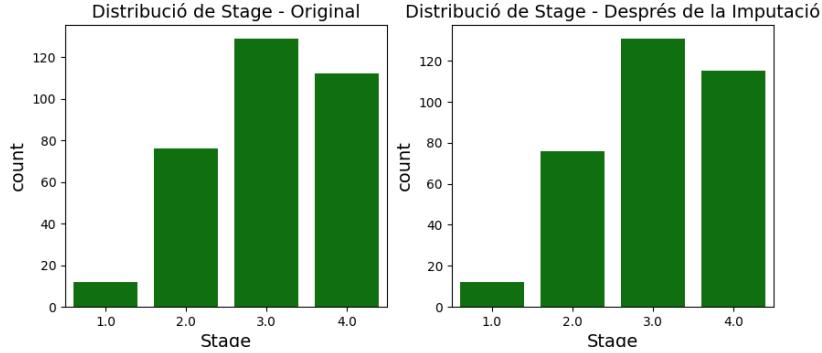


Figura 31: Distribució de 'Stage' abans i després de la imputació proporcional

Com podem veure, ara sí que s'han respectat les distribucions originals i, per tant, ja hem imputat tots els missings de manera efectiva.

Per tal de comprovar que la imputació ha funcionat del tot, cal comprovar que les dues particions (train i test) ja no tenen cap missing, i si ho comprovem ens fixem que, en efecte, ja no hi ha cap dada mancant en cap d'aquests conjunts:

```
Total number of missing values in X_train:0
Total number of missing values in X_test:0
```

3.5 Aplicació de 'SMOTENC' pel balanceig de classes

Com hem explicat en l'apartat sobre el mètode per balancejar les classes de la variable objectiu (apartat 2.2.3), la decisió ha estat SMOTENC. És important remarcar que això s'aplica només a la partició de train, per tal d'evitar que el model dongui prioritat a les classes majoritàries ja que són les que tenen més mostres per aprendre, lo qual pot provocar un biaix en el model, on les classes minoritàries són mal classificades o directament ignorades. Per tant, el balanceig de classes en la partició del train ajuda a evitar aquest biaix.

Per altra banda, en la partició de test, el conjunt de dades representa una simulació de dades reals que l'algoritme mai ha vist abans, per la qual cosa és important que aquest conjunt de test mantingui la seva integritat original, i així avaluar el model en condicions realistes, ja que com he explicat anteriorment aquesta partició no pot estar influenciada per cap modificació.

Per tal d'aplicar-lo correctament, s'han realitzat els següents passos:

1. Creem una llista booleana que indica quines característiques en `X_train` són categòriques.
2. Instanciem `SMOTENC`, especificant les característiques categòriques a través de la llista booleana creada. També es fixa una `random_state` per a què sigui reproductible (42).
3. Finalment, apliquem `SMOTENC` per oversamplejar les classes minoritàries en `X_train` i `y_train`, balancejant així la distribució de la variable objectiu.

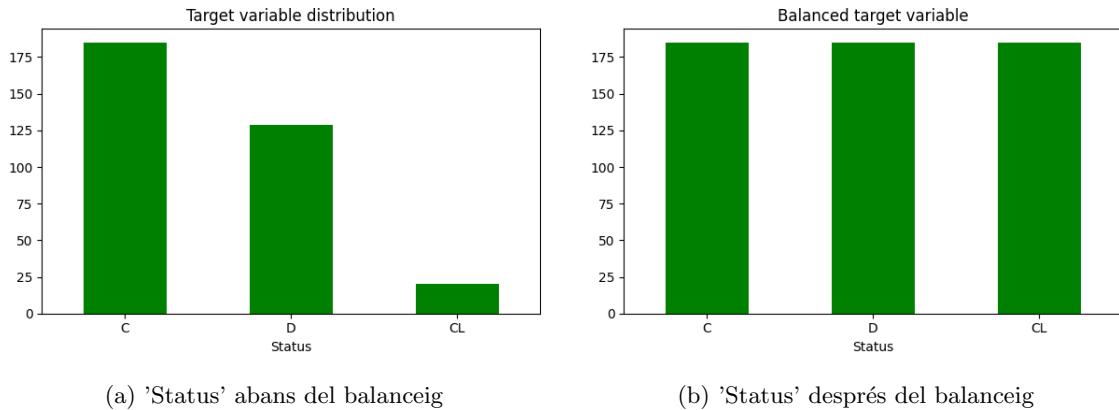


Figura 32: Comparació de la freqüència per classes de la variable 'Status' abans i després del balanceig

Un cop aplicat aquest mètode, podem veure clarament que ara sí la variable objectiu 'Status' ja està equilibrada, això ho podem veure tan gràficament (gràfic 32) com imprimint el comptador de valors de `y_train`, que dona com a resultat:

Status	Count
D	185
C	185
CL	185

Taula 6: Valors per classe del feature 'Status'

Per tant, ja tenim la variable objectiu 'Status' balancejada, tenint el mateix nombre de valors per cada classe del feature.

Aquest canvi en la distribució de la variable objectiu és molt important per a l'entrenament del model, ja que permet que aquest aprengui amb la mateixa probabilitat de cada classe, per tant, d'aquesta manera evita que s'esbiaixi cap a la classe majoritària que existia abans del balanceig. A més a més, això també contribueix a que el model generalitzi millor, ja que s'evita fer un overfitting a les classes més freqüents. Així doncs, tant la precisió com la recall seran millors quan s'avaluï el model en les particions de validació i test, doncs es tractaran les classes amb la mateixa importància, permetent així fer prediccions més fiables i equilibrades.

Per tant, el balanceig de la variable 'Status' mitjançant SMOTENC ha estat un pas important per tal que el model sigui més robust i fiable.

3.6 Variables redundants o sorolloses

Ara, un cop ja hem balancejat les classes i s'han creat noves mostres de pacients, vull identificar si en aquest hi ha alguna variable que sigui redundant o sorollosa, és a dir, que no contribueixi de manera significativa a la capacitat predictiva del model o la seva variació no estigui significativament associada amb la variable objectiu. D'aquesta manera el model seria encara més precís i no es sobreajustaria a dades no pertinents o sorolloses.

Per tant, per tal d'identificar aquestes variables dins del nostre dataset, s'ha utilitzat la prova de chi-quadrat de contingència, que examina si hi ha una relació significativa entre cada variable independent categòrica i la variable objectiu 'Status'. En el cas que el **p-value** resultant de la prova superi el nivell de significació establert (0.05), s'assumeix que la variable en qüestió no té una associació significativa amb la variable objectiu i, per tant, s'afegeix a una llista anomenada **redundant_features**.

En aquest anàlisi, s'han identificat les següents variables com a possibles redundants, junt amb el p-value corresponent:

- 'N_Days' (0.40165)
- 'Cholesterol' (0.13723)
- 'Copper' (0.08080)
- 'Alk_Phosphat' (0.30488)
- 'Triglycerides' (0.06618)
- 'Platelets' (0.09787).

Això ens indica que la contribució d'aquestes variables a la predició de 'Status' no és massa notable des del punt de vista estadístic.

Per tant, per tal de 'filtrar' encara més i veure quins són els features més sorollosos/redundants, s'ha aplicat la mesura de **V Cramer**, que està basat en l'estadístic chi-quadrat, i és una mesura de la força de l'associació entre dues variables categòriques. Aquest valor varia entre 0 i 1, on 0 indica cap associació i 1 indica una associació perfecta.

Per tant, per tal de quantificar la força de l'associació entre aquestes variables i la variable objectiu, he creat la funció **cramers_v(x,y)** que imprimeix les variables que tenen un valor de Cramer inferior a 0.15, el qual és un límit bastant raonable per considerar que l'associació entre la variable independent i la variable objectiu és suficientment feble per a ser considerada no significativa o redundant en el model.

Així doncs, en aplicar aquesta funció a les diverses variables identificades de moment com a 'possibles redundants/sorollosos', s'ha obtingut que per a 'N_Days' i 'Alk_Phosphat', els valors de V de Cramer han estat 0.09037 i 0.13808 respectivament, els quals indiquen una associació bastant débil entre aquests features i 'Status'. Per tant, aquesta evidència estadística reforça la decisió de considerar aquestes variables com a redundants.

Com a resultat, he decidit treure els features 'N_Days' i 'Alk_Phosphat' del model, ja que d'aquesta manera es simplifica en certa part el propi model sense empitjorar massa la seva capacitat predictiva. Per tant, amb aquest canvi també busco incrementar la capacitat de generalització del propi model.

És important remarcar que no s'ha eliminat cap variable categòrica completa del model ja que la presència de categories individuals dins d'una variable categòrica que no mostren una associació significativa no necessàriament invalida la contribució de la resta de categories de la variable, és a dir, en calcular el p-value ha sortit que **Stage_1** podia ser redundant, ja que el seu valor era major que 0.05, però com les altres categories de **Stage** no han estat detectades com a tal, he decidit no modificar ni eliminar el feature. Com que ha succeït el mateix per a totes les categòriques, ja que no hi ha hagut cap que apareguessin totes les seves classes com a sorollosos, no se n'ha eliminat cap.

Així doncs, les variables identificades com a redundants ('N_Days', 'Alk_Phosphat') han estat eliminades dels conjunts de train i test, així com de la llista de variables numèriques.

3.7 Normalització de variables

Per a normalitzar les diferents variables del dataset, he decidit escalar-les utilitzar el mètode de MinMax, el qual transforma els feature ajustant-los dins d'un rang entre 0 i 1. La fórmula per aquesta transformació és:

$$X_{\text{norm}} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

on X és el valor original, $\min(X)$ és el valor mínim de la característica, i $\max(X)$ és el valor màxim.

Aquest mètode conserva la distribució de les variables, ja que només les redimensiona de manera que siguin comparables, però no altera la pròpia distribució de les dades.

Per tal d'implementar-lo correctament, s'ha utilitzat la classe `MinMaxScaler` de la biblioteca 'sklearn.preprocessing' per normalitzar les variables numèriques. Aquest procés ha consistit en els següents passos:

1. Creació d'una instància de `MinMaxScaler`.
2. Aplicació del mètode `fit_transform` a les característiques numèriques del conjunt d'entrenament (`X_train`). Aquest pas calcula el mínim i màxim de les dades i després aplica la transformació per escalar les dades dins del rang de 0 a 1.
3. El mètode `transform` s'aplica a les variables numèriques del conjunt de prova (`X_test`). En aquest cas només s'utilitza `transform` (i no `fit_transform`), ja que el scaler ja ha estat ajustat amb els valors de `X_train`. Això ens assegura que la transformació aplicada a `X_test` està basada en els mateixos paràmetres de normalització utilitzats per `X_train`, mantenint així la consistència entre les dades d'entrenament i de prova.

Per tant, un cop hem aplicat aquest mètode, obtenim un nou dataset anomenat `X_train_normalized`, que només consta dels features numèrics normalitzats, i que si en un futur el volem utilitzar, simplement l'haurem de concatenar amb les columnes categòriques de `X_train`, obtenint així el mateix dataset que `X_train` però amb les variables numèriques ja normalitzades.

Així doncs, si ara provem de visualitzar la distribució de cada variable numèrica, podem veure que, efectivament, totes estan en el rang [0,1]:

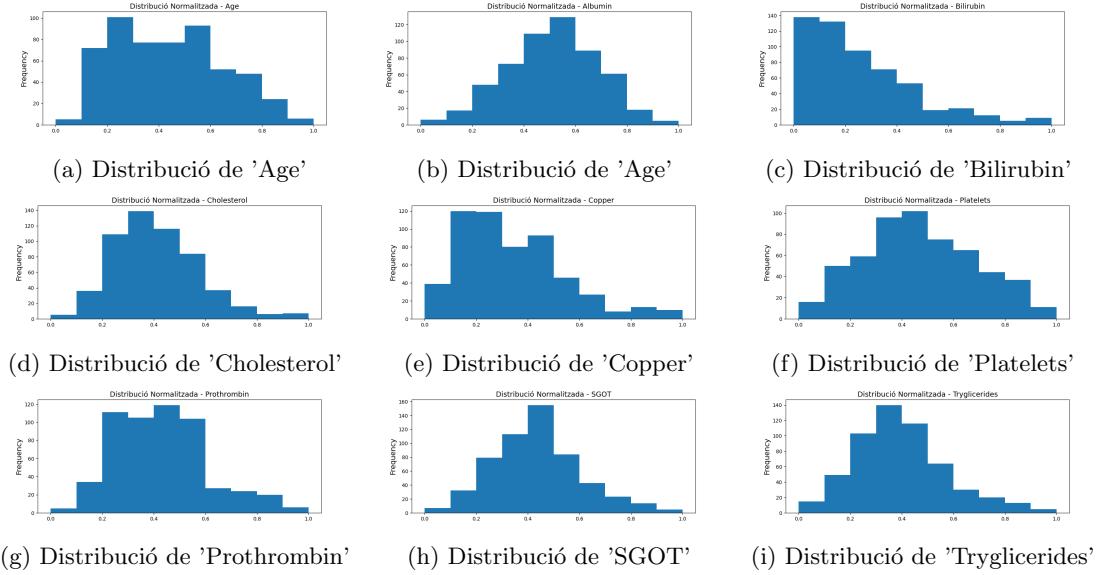


Figura 33: Distribució escalada de cada variable numèrica

3.8 PCA - Anàlisi de Components Principals

L'anàlisi de components principals és una tècnica utilitzada per reduir la dimensionalitat d'un dataset intentant conservar la major quantitat possible de la variància original. Aquesta reducció es realitza identificant els "components principals", que es tracta d'unes noves variables no correlacionades que representen la major part de la informació de les dades originals.

Com que PCA es basa en la matriu de covariància de les dades o en la descomposició de la matriu de correlació, si les variables estan en diferents escales, aquelles amb una variància més gran i unitats de mesura més grans dominarien el primer component principal. Per tant, el primer que he hagut de fer ha estat estandarditzar els conjunts de train i test, creant així dos nous conjunts anomenats `X_train_std` i `X_test_std`.

Seguidament, he aplicat el PCA al conjunt de train estandarditzat, per tal de veure si es pot simplificar el dataset reduint variables però sense perdre massa explicabilitat. Per tant, si visualitzem la proporció de variància explicada per cada component principal i la variància acumulada, obtenim el següents gràfic:

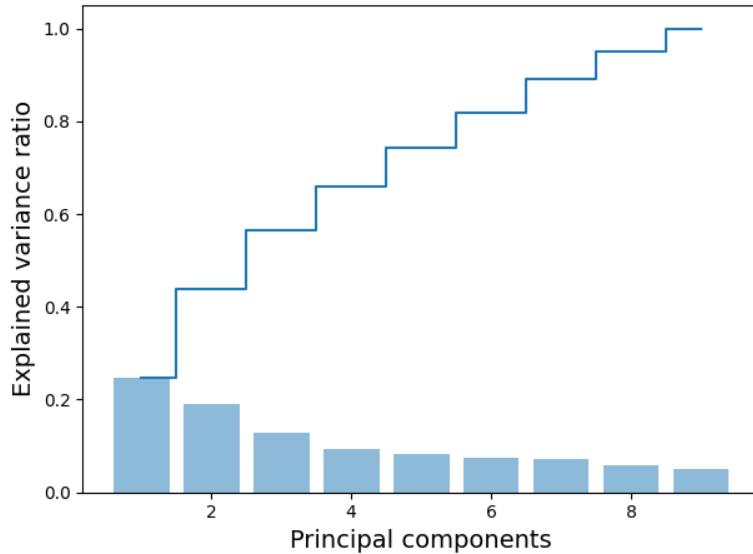


Figura 34: Variància explicada pels components principals i variància acumulada

Això permet determinar el nombre de components a retenir per a una representació simplificada de les dades. He considerat que per tal de no perdre explicabilitat, hauré de mantenir com a mínim el 80% de la variància explicada, per la qual cosa si observem el gràfic (34), fixant-nos en la inèrcia acumulada podem veure que els 5 primers components no arriben a explicar un 80% de la variància explicada, per la qual cosa hauríem d'agafar com a mínim els 6 primers components principals. Com que són massa components els que hauríem de retenir i perdrem una part de la informació, no surt a compte reduir la dimensionalitat, ja que això implicaria la pèrdua d'explicabilitat del problema, doncs cada component principal és una combinació lineal de totes les variables originals, i per tant, la interpretació directa de les variables es perd.

A continuació, he volgut plotejar el conjunt de dades reduït a només dues components principals, i visualitzar aquestes dades transformades. Aquesta representació gràfica de les dues primeres components principals serveix per a obtenir una visió de la distribució de les dades i si hi ha o no una separació visible entre classes.

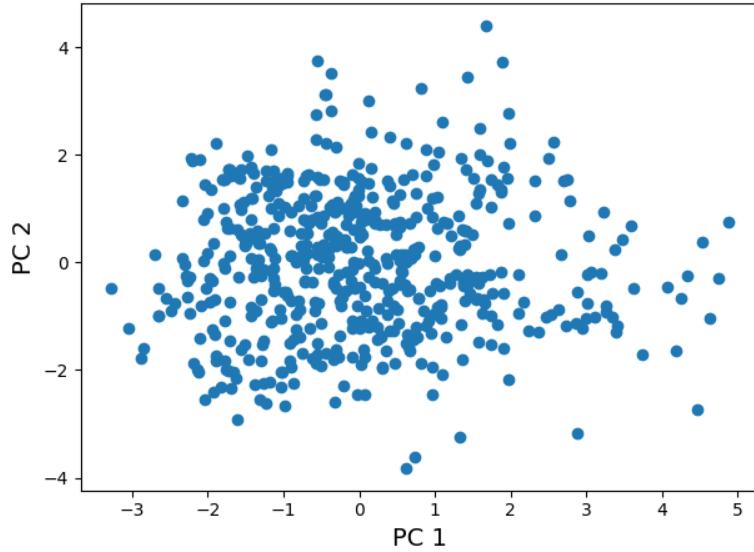


Figura 35: Dispersió de les dades en els dos primers components principals

Observant el gràfic obtingut, podem veure que hi ha una distribució bastant homogènia dels punts al voltant del centre del plot, sense cap agrupació distintiva o separació clara entre els possibles grups. Això clarament indica, tal i com havíem previst, que els dos components principals no són suficients per a fer una separació clara dels dades en grups diferenciats, per tant aquests dos components no proporcionen una bona representació de les dades originals.

A més, aquesta distribució dels punts en el gràfic també ens indica que qualsevol model de classificació que s'entreni utilitzant només aquestes dues dimensions podria tenir certes dificultats a l'hora de diferenciar entre categories de la variable objectiu, ja que no hi ha una separació visible que es pugui utilitzar per a una classificació clara.

Així doncs, després d'aquest anàlisi, ha quedat bastant clar que és important mantenir la totalitat de les dades per a la integritat del model predictiu. Per tant, he decidit no utilitzar la tècnica de reducció de dimensions PCA per assegurar-me que la interpretació del model sigui precisa i les seves prediccions siguin fiables.

4 Definició de models

4.1 Definició de Mètriques i Motivació

Per avaluar els models que desenvoluparé posteriorment (KNN, arbre de decisió i SVM) he decidit considerar un conjunt de mètriques específiques per avaluar el rendiment dels models. Aquestes mètriques són la precisió, el **recall**, l'**accuracy**, el **f1-score** i l'anàlisi de la **corba ROC**. La selecció d'aquestes mètriques s'ha basat en la necessitat d'una avaluació completa i equilibrada que tingui en compte diferents aspectes del rendiment del model. És important primer entendre què vol dir cada cosa per entendre la fórmula de les diferents mètriques que s'expliquen a continuació:

- **True Positives (TP)**: El model ha predit correctament la classe positiva, és a dir, són els positius reals que el model ha identificat correctament com a positius.
- **True Negatives (TN)**: El model ha predit correctament la classe negativa, és a dir, són els negatius reals que el model ha identificat correctament com a negatius.
- **Falsos Positius (FP)**: El model ha predit incorrectament que una instància negativa era positiva (ha classificat erròniament una instància negativa com a positiva).
- **Falsos Negatius (FN)**: El model no ha estat capaç d'identificar una instància positiva real com a positiva (ha classificat erròniament una instància positiva com a negativa).

Aquestes definicions són la base per calcular les diferents mètriques de rendiment que s'utilitzen per avaluar i comparar els models de classificació. Per tant, les mètriques que he considerat adients per a avaluar els 3 models diferents han estat les següents.

4.1.1 Precisió

He escollit la precisió com a mètrica principal degut a la seva simplicitat i efectivitat a l'hora de proporcionar una visió general del rendiment del model. Aquesta mètrica mesura la proporció de prediccions correctes (tant positius com negatius) en relació al total de casos, és a dir, quantes de les meves prediccions com a veritables (T) són realment veritables. Per tant, serveix per entendre la capacitat del model de no classificar com a positius els casos que són realment negatius. És especialment important en un entorn mèdic, on un fals positiu pot tenir conseqüències significatives, per tant en el cas d'aquest estudi mèdic és una mètrica que hem de tenir molt en compte.

$$\text{Precisió} = \frac{\text{True Positives}}{\text{True Positives} + \text{True Negatives}} \quad (1)$$

Per tant, si obtenim una Precisió alta, això indica que d'entre totes les instàncies que el model ha predit com a positives (pertanyents a la classe minoritària, en aquest cas), una gran proporció d'elles són realment positives. Això significa que quan el model diu que un cas és positiu, és molt probable que realment ho sigui.

Per altra banda, si la Precisió és baixa significa que moltes de les instàncies que el model ha predit com a positives en realitat no ho són. Així doncs, el fet d'utilitzar aquesta mètrica m'ofereix una manera ràpida i clara de determinar la capacitat general del model per a fer prediccions correctes.

4.1.2 Recall

El recall mesura quantes de les instàncies reals veritables (T) el model està predient com a veritables. Per tant, aquesta és una mètrica important per assegurar que el model identifica correctament els casos positius, reduint el risc de falsos negatius, els quals poden ser perillosos en aplicacions mèdiques.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (2)$$

Per tant, si obtenim un Recall alt, significa que el model és capaç de detectar una gran proporció dels casos de la classe minoritària. És a dir, entre tots els casos reals d'aquesta condició en el conjunt de dades, el model n'identifica correctament molts. En canvi, si el Recall és baix, això vol dir que el model està perdent molts d'aquests casos més estranys (classes menys freqüents).

4.1.3 Accuracy

L'accuracy mesura quantes prediccions estan correctes en comparació amb el total, per la qual cosa és una mètrica que proporciona una visió general ràpida del rendiment del model, però pot ser enganyosa en el cas de les variables que tenen les categories molt desequilibrades, en què una classe és molt més freqüent que l'altra. Per tant, és possible que obtinguem un valor massa alt en aquesta mètrica en el cas d'algunes variables, doncs com hem vist anteriorment tenim certs features que estan molt desbalancejats. Per aquesta raó, com que el dataset consta de features en què hi ha un clar desequilibri de classe, utilitzem altres mètriques com el recall, la precisió, el f1-score, i l'anàlisi de la corba ROC (ROC-AUC) per avaluar el rendiment.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Positives} + \text{Negatives}} \quad (3)$$

4.1.4 F1-score

El f1-score és una mesura que combina la precisió i el recall en una sola mètrica, oferint un equilibri entre ambdues. És útil quan volem un balanç entre la identificació de casos positius (recall) i la precisió en la classificació d'aquests casos. A més, aquesta mètrica és útil, a diferència de l'accuracy, en datasets que consten de variables amb un desequilibri entre classes.

$$\text{F1-score} = 2 \times \frac{\text{Precisió} \times \text{Recall}}{\text{Precisió} + \text{Recall}} \quad (4)$$

Aquesta fórmula calcula la mitjana harmònica de la precisió i el recall. La mitjana harmònica és una manera d'agrupar dues taxes en una sola mesura que dóna més pes als valors baixos, el que significa que si qualsevol de les dues mètriques (precisió o recall) és baixa, el f1-score també serà baix. Per tant, ens servirà molt per avaluar el model ja que no només importa detectar la majoria dels casos positius (alt recall), sinó també mantenir un alt nivell de confiança en les prediccions positives (alta precisió).

4.1.5 Àrea sota la Corba ROC (AUC-ROC)

Finalment, utilitzaré la corba ROC i l'àrea sota la corba (ROC-AUC) com a mètriques per a comprovar el rendiment de cada model. La corba ROC proporciona una visió completa de com el rendiment del model varia en diferents llindars de classificació, oferint una visió més global.

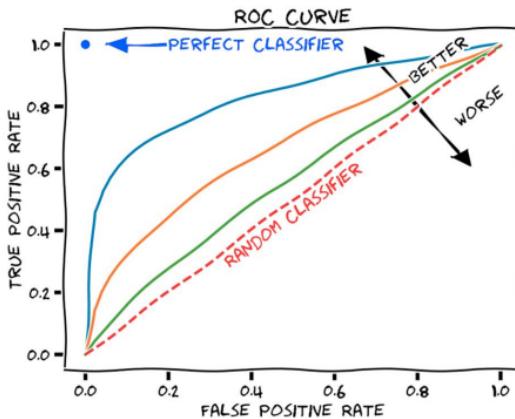


Figura 36: Dibuix d'una Corba ROC

L'AUC (Àrea Sota la Corba) dóna una mesura quantitativa de la capacitat del model per distingir entre les classes, per tant, un valor més alt d'AUC-ROC indica que el model té una millor capacitat general per discriminar entre pacients amb diferents pronòstics.

4.1.6 Mètrica a maximitzar

En el context de l'estudi, en què es busca predir la supervivència de pacients amb cirrosi hepàtica, la mètrica que maximitzem ha de ser seleccionada amb molta cura, donat l'alt impacte que les decisions basades en les prediccions del model poden tenir en la vida real dels pacients. Considerant la gravetat del problema mèdic en qüestió, és realment important minimitzar tant els falsos positius com els falsos negatius per les següents raons:

-Minimització dels Falsos Positius: Un fals positiu en aquest context significa que el model prediu erròniament que un pacient amb cirrosi hepàtica sobreviurà quan, de fet, el pronòstic real és menys favorable. Aquesta situació pot portar a decisions clíniques errònies, com la disminució de l'atenció mèdica intensiva o el retard en la preparació per a intervencions més agressives. Per exemple, un pacient que necessita urgentment un trasplantament de fetge podria no rebre'l a temps a causa d'una percepció equivocada de la seva supervivència. Aquesta circumstància no només afecta negativament la salut del pacient, sinó que també pot impactar emocionalment a la família i els cuidadors, que podrien no estar preparats per afrontar la progressió de la malaltia.

-Minimització dels Falsos Negatius: D'altra banda, els falsos negatius en aquest estudi són igualment molt greus. Això succeeix quan el model no identifica un pacient que està en risc de no sobreviure. Aquest error pot resultar en una falta de tractaments vitals per a sobreviure o en no proporcionar cures adequades a temps. Per exemple, si un pacient amb un alt risc de mortalitat és erròniament classificat com a baix risc, aquest pot ser que no rebi l'atenció mèdica que realment necessita, lo qual podria accelerar el seu declivi o inclús provocar la seva mort si no s'aconsegueix tractar a temps, doncs aquesta situació podria deixar a la família i els cuidadors sense la preparació i el suport necessaris per gestionar les necessitats del pacient en les últimes etapes de la malaltia.

Per tant, per aquestes raons són de vital importància tant la Precisió com el Recall. Una Precisió

alta assegura que les prediccions positives del model són realment casos que requereixen atenció especial, mentre que un alt Recall garanteix que el model identifica la majoria dels pacients que estan en risc. La combinació d'ambdues mètriques en un equilibri adequat és clau per tal de proporcionar prediccions fiables en un context mèdic tan delicat com és la supervivència en pacients amb cirrosi hepàtica.

Així doncs, com que hi ha una gran necessitat de balancejar la Precisió i Recall dels models, el **F1-score** sembla que és la mètrica ideal que haurem de maximitzar, ja que al ser la mitjana harmònica de la Precisió i el Recall (com hem vist anteriorment), com que aquesta mètrica proporciona un equilibri entre les dues mètriques, això garanteix que el model no es decanti excessivament cap a la identificació de casos positius o la precisió de les seves prediccions. Això és especialment pertinent en el context mèdic, en què cada decisió pot tenir conseqüències significatives. Per tant, abans de trobar els hiperparàmetres òptims dels models, necessitem analitzar cada model per separat i definir quines són les característiques desitjables respecte al problema per cadascun del tres models diferents.

4.2 Primer model - KNN (K-Nearest Neighbors)

4.2.1 Simplicitat i Interpretabilitat

El KNN és un model simple i molt interpretable, el qual funciona sobre el principi de semblança de característiques, on la predició per a un punt de dades nou es basa en la proximitat a els punts de dades més propers en l'espai de característiques, és a dir, es determina segons les etiquetes dels seus veïns més propers en l'espai de característiques. Això a banda de simplificar el procés d'entrenament, també permet una bona interpretació dels resultats, ja que facilita l'anàlisi de com les diferents característiques influeixen en les prediccions del model.

4.2.2 Hiperparàmetres

Un dels principals hiperparàmetres en el KNN és el nombre de veïns (K). Aquest hiperparàmetre permet un ajust flexible de la complexitat del model. Un valor més baix de K pot fer que el model sigui més sensible al soroll dels dades (overfitting), mentre que un valor més alt pot proporcionar una frontera de decisió més suau. Però no només tindrem aquest hiperparàmetre, sinó que haurem de considerar els tres següents:

- **n_neighbors (Nombre de Veïns):** Com acabo de comentar, és el nombre k , que determina el nombre de veïns més propers que s'utilitzaran per a la predició. Un nombre menor de veïns pot fer que el model sigui més sensible a les anomalies dins del conjunt de dades, lo que pot portar a un possible overfitting, per tant el model podria adaptar-se massa als detalls del conjunt de train i no generalitzaria bé a dades noves. Per altra banda, un nombre major de k pot augmentar la generalització, fent que el model sigui més robust davant de variacions en les dades, però aquesta aproximació pot ser menys precisa en capturar les particularitats específiques dels dades, el que pot resultar en un underfitting.
- **weights (Pesos):** Aquest hiperparàmetre determina si tots els veïns contribueixen igualment a la predició ('uniform') o si els veïns més propers tenen una influència major ('distance'). 'Uniform' tracta totes les instàncies dins del veïnat amb igual importància, mentre que 'distance' dóna més pes als veïns més propers.

- **metric (Mètrica de Distància):** Aquest hiperparàmetre es tracta de la mètrica utilitzada per a calcular la distància entre els punts, les quals soLEN ser la distància euclidiana, de Manhattan, i de Minkowski. La selecció de la mètrica de distància adequada és realment important, ja que diferents mètriques poden tenir un impacte significatiu en la manera com el model interpreta la proximitat i, per tant, en el seu rendiment.

4.2.3 Volum de Dades

El KNN és adequat per conjunts de dades de mida mitjana. Aquest model pot ser menys efectiu en conjunts de dades molt grans a causa de la necessitat de calcular la distància de cada punt de dades nou a tots els altres punts de dades, però com que el conjunt de train consta de poques mesures, no serà un factor a tenir en compte per a l'eficiència d'aquest model.

4.2.4 Aplicabilitat al Problema

Considerant que l'objectiu és predir la supervivència de pacients amb cirrosi hepàtica, el KNN pot ser particularment útil per identificar patrons locals en l'espai de característiques. La capacitat del KNN per captar similituds en petits grups de dades podria ser clau per identificar subgrups de pacients amb característiques clíniques similars.

4.3 Segon model - Arbre de Decisió

4.3.1 Simplicitat i Interpretabilitat

Els arbres de decisió, com el KNN, també destaquen per la seva simplicitat i interpretabilitat. Aquests models mapegen les decisions i les seves possibles conseqüències, presentant-les en una estructura d'arbre que és fàcil d'entendre i visualitzar. Cada node de l'arbre representa una característica del conjunt de dades, cada branca una decisió, i cada fulla una predicció final. Per tant, aquesta estructura clara permet comprendre com el model arriba a les seves conclusions, fent-lo una eina útil per a l'anàlisi exploratòria de dades i la presa de decisions.

4.3.2 Hiperparàmetres

Max Depth: Es tracta de la profunditat màxima de l'arbre. Una profunditat més gran pot permetre al model captar més detalls, però també pot conduir a un overfitting. Ajustar aquest hiperparàmetre pot ajudar a trobar un equilibri entre biaix i variància, a més d'assegurar que el model generalitzi bé amb dades que no ha vist mai.

Criterion: Determina com l'arbre decideix on dividir les dades en cada node. Les opcions comunes són "gini" per a la impuresa de Gini i "entropy" per a la guany d'informació. 'Gini' mesura la impuresa de Gini, una mesura de la freqüència amb què un element aleatori seria incorrectament etiquetat si es etiquetés aleatoriament segons la distribució d'etiquetes en el conjunt. 'Entropy', per altra banda, utilitza el concepte de guany d'informació, basat en la teoria de la informació, per seleccionar les divisions que maximitzin la reducció de l'entropia.

Min Samples Split: Especifica el nombre mínim de mostres necessàries per dividir un node. Valors més grans prevenen l'aprenentatge de soroll i overfitting, ja que requereixen que un node tingui un nombre suficient de mostres abans de permetre una divisió més. Això pot ajudar a evitar que l'arbre aprengui el soroll present en el conjunt d'entrenament, però si el valor és massa alt, pot conduir a un underfitting.

4.3.3 Volum de Dades

Els arbres de decisió poden ser efectius en una àmplia gamma de volums de dades. Són efectius en conjunts de dades petits a mitjans, on poden captar patrons complexos sense requerir una gran quantitat de dades. Tot i això, en conjunts de dades molt grans poden ser menys eficients (sobretot en quant a temps de càlcul), per la qual cosa seria necessari un bon ajustament dels hiperparàmetres com ara ‘max_depth’ i ‘min_samples_split’. Com hem comentat abans, al ser un dataset amb poques mostres no ens preocupa en absolut si és un algoritme que no és eficient amb grans quantitats de dades, ja que no és el cas.

4.3.4 Aplicabilitat al Problema

En el context de la predicció de la supervivència de pacients amb cirrosi hepàtica, l’arbre de decisió ofereix una manera intuitiva d’analitzar com diverses característiques clíniques influeixen en els resultats dels pacients. La capacitat d’aquest model per desglossar el procés de decisió en una sèrie de preguntes simples basades en les dades el pot fer realment útil per tal d’identificar factors de risc i patrons en les dades mèdiques.

4.3.5 Importància de la Poda

També cal destacar la importància de la poda de l’arbre, que és una tècnica per a prevenir el overfitting. Això implica la eliminació de parts de l’arbre que no aporten poder predictiu realment significatiu. Per tant, amb un ús adequat de la poda, podem assegurar que l’arbre de decisió manté la seva generalitzabilitat, evitant al mateix temps una complexitat d’aquesta innecessària.

4.4 Tercer model - Support Vector Machine (SVM)

4.4.1 Simplicitat i Interpretabilitat

Tot i que les Màniques de Vectors de Suport (SVM) poden ser conceptualment més complexes que els models anteriors, oferixen un enfocament potent i flexible per a la classificació. Els SVM treballen construint un hiperplà que millor separa les classes de dades en un espai de característiques transformades, optimitzant el marge entre aquestes classes. Això pot ser visualitzat en problemes de baixa dimensió, encara que en espais de característiques d’alta dimensió, la interpretació es torna menys directa.

4.4.2 Hiperparàmetres

C (Regularització): Serveix com a terme de regularització en els SVM. Un valor més alt de ‘C’ indica menys regularització, permetent que el model tingui més llibertat per ajustar-se als dades i per tant captar més detalls, augmentant així el risc d’overfitting. Un valor més baix de ‘C’ fa que el model sigui més simple amb una major regularització, reduint el risc d’overfitting però augmentant el risc d’underfitting.

Kernel: Determina el tipus de transformació espacial que s’aplica a les dades abans de trobar el hiperplà de separació. Les opcions comuns inclouen ‘linear’, ‘poly’, ‘rbf’ i ‘sigmoid’, per tant haurem de trobar l’òptim ja que això afecta directament a la capacitat del model per captar la complexitat i les interaccions entre les característiques en les dades.

Degree: Utilitzat només quan ‘kernel’ està establert com a ‘poly’, aquest paràmetre determina el grau del polinomi en la transformació del kernel. Un grau més alt permet models més complexos,

però també pot augmentar el risc d'overfitting. Per tant, si en trobar els hiperparàmetres òptims veiem que 'kernel' no és 'poly', no haurem de definir 'Degree'.

Gamma (en Kernels no Lineals): Defineix l'abast d'influència d'una única mostra d'entrenament. Valors més alts donen lloc a fronteres de decisió més ajustades, que poden captar millor les complexitats de les dades, però amb un augment del risc d'overfitting. Un valor més baix resulta en fronteres més suaus, que poden ser millors per a generalitzar en dades que el model no ha vist.

4.4.3 Volum de Dades

Les SVM són eficients en la gestió de dades d'alta dimensió i es comporten bé en conjunts de dades de mida mitjana a gran. No obstant això, el temps d'entrenament pot ser considerable en conjunts de dades molt grans, especialment amb kernels no lineals i valors alts de 'gamma'.

4.4.4 Aplicabilitat al Problema

En el context de la predicció de la supervivència en pacients amb cirrosi hepàtica, les SVM ofereixen un enfocament robust per manejar espais de característiques complexes i interaccions no lineals. La capacitat de les SVM per modelar fronteres de decisió complexes les fa adequades per a aquest tipus de problemes mèdics, on les relacions entre les característiques poden ser no lineals i multidimensionals.

4.5 Trobar els hiperparàmetres òptims

Per a determinar els millors hiperparàmetres pels models de KNN, Arbre de Decisió i SVM, he utilitzat la tècnica de GridSearchCV de la llibreria `sklearn`. Aquesta tècnica permet una cerca exhaustiva a través de diferents combinacions d'hiperparàmetres per trobar aquells que maximitzen una mètrica específica, en aquest cas, `f1-score`.

4.5.1 Procés de Selecció

He definit una sèrie de valors possibles per als hiperparàmetres més rellevants de cada model i he executat GridSearchCV 10 vegades per assegurar-me que s'obté una bona estimació dels millors hiperparàmetres. He creat tres variables (`best_knn_params`, `best_tree_params`, `best_svm_params`) que s'inicialitzen a `None` i es fan servir per emmagatzemar els millors hiperparàmetres de cada model, junt amb els seus f1-scores associats. D'aquesta manera, en les 10 iteracions del bucle ens quedem amb els hiperparàmetres que donen un major f1-score per al model corresponent. Per tal de fer-ho correctament i no fer un leakage del test al train, aquests hiperparàmetres s'escullen fent una Cross Validation de 10 folds en el conjunt d'entrenament, i obtenint d'aquí els f1-scores corresponents. He decidit fer 10 folds ja que, a diferència de fer-ho amb 5, cada model s'entrena en un subconjunt lleugerament més gran de dades, la qual cosa proporciona una millor estimació de la capacitat del model de generalitzar a noves dades. Amb menys folds, com ara 5, cada model tindria menys dades per aprendre, la qual cosa augmentaria el risc de biaix. Com que tenim poques mostres d'entrenament, 10 folds proporcionen una bona balança entre la quantitat de dades per a l'entrenament i per a la validació en cada iteració. Això permet que cada model s'entreni i es validi en una varietat més àmplia de subconjunts de dades, proporcionant una estimació més precisa i generalitzable del rendiment del model. També, donada la mida relativament petita del dataset, el cost computacional addicional de fer 10 folds enllloc de 5 no és massa significatiu, per lo que surt a compte fer-ne 10 folds per al procés de validació creuada.

Així doncs, un cop entés com funciona el codi per tal de trobar els hiperparàmetres òptims, a continuació es mostren quins valors es proven per cada hiperparàmetre de cada model diferent:

KNN

- `n_neighbors`: 3, 5, 7, 9, 11
- `weights`: 'uniform', 'distance'
- `metric`: 'euclidean', 'manhattan', 'minkowski'

Arbre de Decisió

- `max_depth`: None, 3, 5, 7, 10
- `criterion`: 'gini', 'entropy'
- `min_samples_split`: 2, 5, 10

SVM

- `C`: 0.1, 1, 10
- `kernel`: 'linear', 'poly', 'rbf', 'sigmoid'
- `degree`: 2, 3, 4
- `gamma`: 'scale', 'auto'

4.5.2 Resultats Obtinguts

Després d'executar GridSearchCV, els hiperparàmetres obtinguts que donen un major valor de f1-score per a cada model han estat els següents:

Model	Millors Hiperparàmetres
KNN	{ <code>metric</code> : 'euclidean', <code>n_neighbors</code> : 3, <code>weights</code> : 'distance'}
Arbre de Decisió	{ <code>criterion</code> : 'gini', <code>max_depth</code> : 10, <code>min_samples_split</code> : 5}
SVM	{ <code>C</code> : 1, <code>degree</code> : 2, <code>gamma</code> : 'scale', <code>kernel</code> : 'rbf'}

Taula 7: Hiperparàmetres òptims per a cada model

Els hiperparàmetres seleccionats reflecteixen un equilibri entre la capacitat del model per aprendre de les dades i la seva habilitat per generalitzar a dades noves.

En el cas del KNN, la distància euclidiana indica que el model està calculant la proximitat entre els punts en un espai euclidià, que és el mètode més comú i natural de mesurar distàncies en l'espai. Això suggereix que les característiques són considerades en un espai geomètric simple, on les distàncies lineals són rellevants. El nombre de veïns (3) és un valor realment baix, el qual implica que el model depèn molt dels veïns més propers, que pot ser beneficiós si les relacions locals són més significatives que les tendències generals, tot i que també és cert que això pot augmentar el risc d'overfitting, ja que el model pot ser massa sensible a les anomalies en les dades. Per últim, veiem que pesos ha sortit 'Distance', que significa que en fer les prediccions, els veïns més propers tenen més influència que els veïns més llunyans. Per tant podem deduir que les mostres més similars

tendeixen a tenir la mateixa etiqueta.

En el cas del Decision Tree, el criteri 'Gini' és utilitzat per a mesurar la impuresa dels nodes, és a dir, és efectiu per a dividir les mostres de manera que les classes estiguin el més separades possible dins de cada node. Això suggereix una eficàcia en la segregació de les classes en el conjunt de train. La profunditat màxima (10) indica que segurament pugui captar complexitats en les dades, però també s'arrisca a aprendre detalls específics que podrien no generalitzar bé, tot i que com que el dataset no és massa gran, aquest risc pot ser moderat. Finalment, Min Samples Split (5) indica que un node ha de tenir almenys 5 mostres abans de considerar una divisió, lo que pot ajudar a prevenir un creixement excessiu de l'arbre (overfitting), equilibrant la necessitat de captar detalls amb la necessitat de mantenir una certa generalització.

Per últim, en el model SVM hem obtingut un valor $C = 1$, que indica un equilibri entre la regularització i la llibertat del model per ajustar-se a les dades. No és ni massa restrictiu (evitant underfitting) ni massa lliure (evitant overfitting). El kernel 'rbf' (Radial Basis Function) és potent per capturar relacions no lineals en les dades, lo que indica que el model pot modelar interaccions complexes entre les característiques. Degree (2): Com que el 'degree' és rellevant només per a kernels polinòmics i hem obtingut que l'òptim és 'rbf', no haurem d'especificar-lo a l'hora de crear el model. Per finalitzar, Gamma = 'Scale' ajusta el paràmetre 'gamma' de manera automàtica en funció de la mida del conjunt de dades i el nombre de característiques, proporcionant una manera equilibrada d'affectar la influència de cada mostra individual.

Així doncs, l'ús d'aquesta metodologia assegura que els models estan ben ajustats al dataset, maximitzant la seva eficàcia en la tasca de predicción. Aquests resultats seran fonamentals per a la posterior avaluació dels models i la seva aplicació en la predicción de la supervivència de pacients amb cirrosi hepàtica, que és l'objectiu del treball.

4.6 Entrenament dels models amb train

Com que ja hem trobat els millors hiperparàmetres fent validació creuada, tot i que ara entrenem cada model en el conjunt d'entrenament, no tindria massa sentit retocar els hiperparàmetres, ja que ja tenim escollits els òptims (trobats en l'apartat anterior). És a dir, com que no tenim partició de validació, si ara fem cross validation no obtindrem que en canviar algun hiperparàmetre s'obté un valor major de **f1-score**. Per aquesta raó, en aquest punt del treball simplement podem veure com funciona cada model en el conjunt d'entrenament, fent cross validation, i posteriorment veure el rendiment real amb dades que no ha vist mai, que es tracta del conjunt de Test. En el cas que ens doni resultats no massa bons, la qual cosa evidentment pot succeir, no podrem modificar res del conjunt d'entrenament, ja que si ho féssim estaríem fent un leakage de Test a Train, la qual cosa provocaria que el model s'especialitzés amb les dades de Test, que justament existeixen per a simular un conjunt de dades mai vistes pel model (per tant seria com fer 'trampes'). Així doncs, primer visualitzarem els resultats obtinguts a l'hora de fer validació creuada pels tres models, sense tocar cap hiperparàmetre (ja que actualment ja disposem dels òptims), els analitzarem i en la següent secció (5) veurem el rendiment real de cada model amb la partició de Test, i escollirem el millor.

Per a avaluar els diferents models, he decidit fer servir una funció personalitzada anomenada `evaluate_model`, la qual utilitza `cross_val_score` de `sklearn` amb una validació creuada de 10 folds per calcular el **f1-score** (torno a utilitzar 10 folds per les mateixes raons que abans), que

com hem vist anteriorment aquesta tècnica divideix el conjunt de dades d'entrenament en 10 parts (folds), utilitzant cada part com a conjunt de validació una vegada, mentre que les altres 9 parts s'utilitzen per a l'entrenament, proporcionant així una visió completa del rendiment del model en les dades de train. A continuació es calcula la mitjana de les diverses mètriques que hem considerat importants per avaluar cada model, és a dir, la precisió (`precision_macro`), el recall (`recall_macro`), l'accuracy i l'`f1-score` (`f1_macro`). D'aquesta manera, aquestes mètriques es calculen tant per a l'entrenament com per a la validació, permetent-nos comparar directament el rendiment del model en aquests dos conjunts, i veure si hi ha un overfitting o underfitting (tot i que el més probable sigui que tots facin overfitting).

A més, en la funció faig servir `cross_val_predict` per a generar prediccions per cada mostra en les dades d'entrenament, utilitzant la mateixa tècnica de validació creuada. D'aquesta manera s'obté una estimació més realista de com el model actuarà en dades no vistes. Per a imprimir els resultats, es genera un informe de classificació utilitzant `classification_report` de `sklearn`, el qual inclou mesures com la precisió, recall i F1-score per a cada classe.

Finalment, es fa la matriu de confusió utilitzant una altra funció personalitzada `plot_confusion_matrix`, que ens permetrà veure els valors predictius i reals de cada classe en aquesta partició de train, i d'aquesta manera posteriorment podrem veure les diferències amb les matrius de confusió obtingudes a l'avaluar els models en el conjunt de test, en què segurament els resultats siguin pitjor.

Així doncs, un cop ja sabem con entrenarem cada model en el conjunt de train i què tindrem en compte a l'hora d'avaluar-los, podem procedir a entrenar cada model per separat i veure els resultats al fer validació creuada.

4.6.1 KNN - Entrenament amb Train

Per aquest primer model, començó creant una instància del classificador KNN utilitzant `KNeighborsClassifier` de `sklearn`. He configurat el classificador amb els hiperparàmetres òptims que hem trobat anteriorment en avaluar el f1-score del model al llarg de 10 iteracions (fent Cross Validation), que són: `n_neighbors=3, weights='distance', metric='euclidean'`.

Així doncs, seguidament faig servir el mètode `.fit` per entrenar el model KNN amb `X_train` i `y_train`. Això fa que el model aprengui com les característiques de `X_train` es relacionen amb les etiquetes de `y_train`.

Un cop evaluem el model amb la comanda `evaluate_model(knn, X_train, y_train)`, obtenim en primer lloc el següent classification report:

F1-score for the model: 0.7716

	Precision	Recall	F1-score
C	0.79	0.64	0.71
CL	0.75	0.98	0.85
D	0.82	0.71	0.76
Accuracy	0.78		
Macro Avg	0.79	0.78	0.77
Weighted Avg	0.79	0.78	0.77

Taula 8: Classification Report

Com podem observar, el model ha aconseguit un **F1-score general de 0.7716**. Aquest valor

indica un equilibri raonable entre precisió i recall, suggerint que el model és bastant efectiu en general per a la predicción de les classes.

Per la classe 'C', una Precisió de 0.79 indica que un alt percentatge de les instàncies predites com a classe C són correctes. El Recall (0.64) és més baix en comparació amb la precisió, lo qual suggerí que el model està perdent un nombre significatiu d'instàncies reals de classe C. Per altra banda el F1-score (0.71) reflecteix un rendiment bastant bo, encara que hi ha espai per a millora, especialment en augmentar el recall.

Per a la classe 'CL', la precisió (0.75) és lleugerament inferior a la de la classe C, però encara indica una bona capacitat de classificació. En aquest cas el recall és molt alt (0.98), que significa que el model ha identificat correctament gairebé totes les instàncies de la classe CL. A més, té el F1-score més alt entre totes les classes (0.85), indicant un rendiment excel·lent en la predicción de la classe CL.

Per últim, la classe 'D' consta d'una precisió de 0.82, sent la més alta entre les tres classes, la qual cosa indica una alta proporció de prediccions correctes per a la classe D. En aquesta categoria també podem veure que el recall és millor que en la classe C però més baix que en la classe CL, i finalment té també un f1-score robust (0.76), reflectint un bon equilibri entre precisió i recall per a la classe D.

Que l'Accuracy sigui de 0.78 vol dir que el 78% de totes les prediccions eren correctes, un resultat bastant bo per al conjunt de train. Per altra banda, com que en aquest conjunt la variable 'Status' està balancejada, sempre tindrem el mateix resultat en el 'Macro Avg' i 'Weighted Avg', que podem veure que en aquest cas és de 0.77, el qual és un rendiment general bastant bo.

Així doncs, els resultats indiquen que el model KNN funciona bé amb el conjunt de dades d'entrenament, mostrant una bona capacitat per a classificar les mostres en les diferents classes. No obstant això, hi ha oportunitats per a millora, particularment en augmentar el recall per a algunes classes.

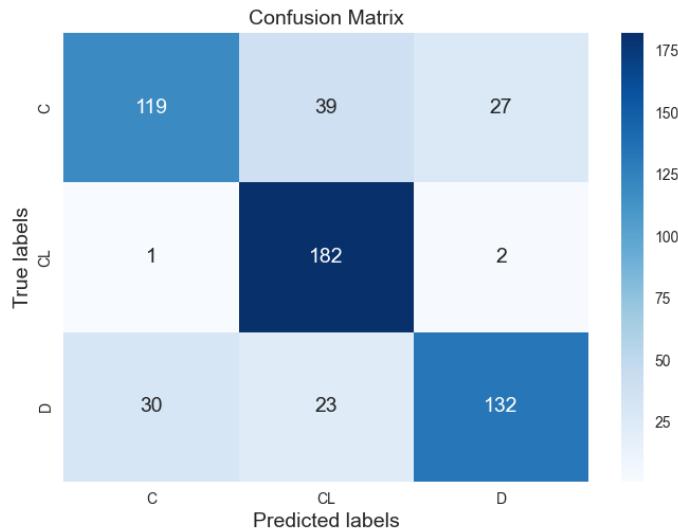


Figura 37: Matriu de confusió del model KNN - Train

A més, com que la funció també ploteja la matriu de confusió, podem veure les classes predites en relació amb les classes reals. D'aquí efectivament veiem que la classe 'CL' es prediu amb moltíssima exactitud (de les 185 mostres només falla en 3), mentre que li costa un pèl més predir correctament les altres dues classes, tot i que segueix sent un rendiment prou bo, ja que de les 185 en prediu bé 119 i 132 per a les classes 'C' i 'D', respectivament.

Finalment, si veiem els resultats de les diferents mètriques en la partició de train i test al fer cross validation (entenent com a test la partició de validació creuada que simula dades no vistes), obtenim els següents resultats:

Metric	Train	Test
Precision_macro	1.000	0.792
Recall_macro	1.000	0.780
Accuracy	1.000	0.781
F1_macro	1.000	0.772

Taula 9: Average Cross-Validation Scores

Com es pot observar, les mètriques d'entrenament són perfectes (1.000) en comparació amb les de validació (Test), que són significativament més baixes. Això indica un cas evident d'overfitting, en què el model KNN està excessivament ajustat a les dades d'entrenament, resultant en una pèrdua de generalització quan s'enfronta a noves dades en la validació creuada. Aquesta situació es manifesta en la major eficàcia del model per a predir les dades que ja ha vist (dades d'entrenament), però una eficàcia notablement menor en predir dades que no ha vist (dades de validació).

4.6.2 Arbre de Decisió - Entrenament amb Train

En aquest model fem servir `DecisionTreeClassifier` i configarem el model amb els paràmetres que hem obtingut anteriorment (veure 7), que han estat `criterion='gini'`, `max_depth=10`, `min_samples_split=5`, i a més afegim una llavor per a què sempre que executi el codi em doni els mateixos resultats, en aquest cas he declarat `random_state=33`.

Un cop inicialitzat el model, l'entrenem amb les dades d'entrenament (`X_train`, `y_train`). A continuació, utilitzem la funció `evaluate_model` per avaluar el rendiment del model en el conjunt d'entrenament.

Tot i que els resultats no són dolents (veure taula ??), agafant com a exemple els passos que es van seguir en el Laboratori 5 d'aquesta assignatura, he creat una llista anomenada `tree_models` en què cada arbre dins d'aquesta llista ha estat creat amb diferents valors de `ccp_alpha` obtinguts del `cost_complexity_pruning_path` de `tree_model`. És a dir, cada arbre en `tree_models` està podat a un nivell diferent. En aquest mateix codi plotejem el gràfic 38, que ens mostra la relació entre la impuresa total de les fulles d'un arbre de decisió i el valor de alpha efectiu utilitzat en el procés de poda de complexitat de costos.

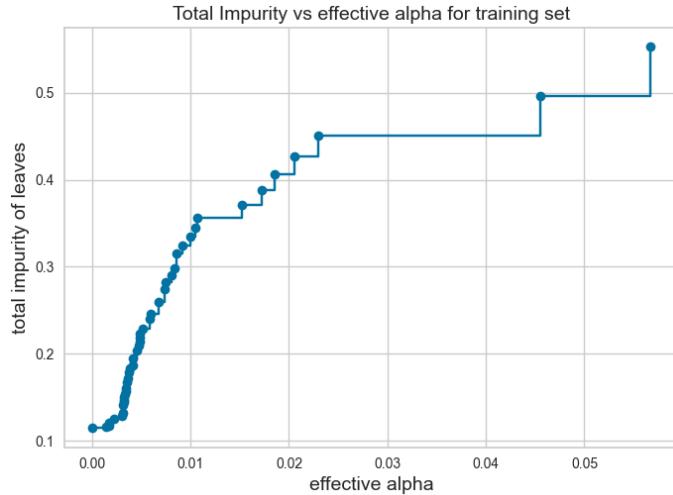


Figura 38: Arbre de Decisió - Impuresa vs Alpha

Com podem veure, quan alpha és proper a zero, l'impuresa disminueix ràpidament amb l'augment d'alpha, és a dir, la poda està eliminant fulles amb poca contribució a la reducció de l'impuresa. Hi ha diversos nivells on l'increment de alpha no canvia l'impuresa, suggerint que aquests increments no estan provocant més poda significativa en l'arbre. També podem veure alguns salts grans en l'impuresa, especialment cap al final del gràfic, que poden indicar que s'està realitzant una poda agressiva, on s'estan perdent fulles que són importants per a la capacitat predictiva de l'arbre. Així doncs, d'aquest gràfic podem deduir que l'objectiu seria seleccionar un valor d'alpha que mantingui la impuresa baixa però eviti la complexitat excessiva. Per tant, segurament el millor valor d'alpha es troba entre el 0.00 i 0.01, ja que en aquesta regió és on obtenim la major reducció de la impuresa sense una pèrdua significativa de la capacitat predictiva.

Per tal que sigui el propi codi el que decideixi el millor valor d'alpha i no fer-ho a ull o manualment, seleccionem el millor model d'aquesta llista basant-nos en el rendiment mitjançant validació creuada segons la mètrica `F1_macro` en el conjunt d'entrenament. Per tant, anomenem `best_model_index` a l'índex del model amb la millor puntuació i `tree_models[best_model_index]` és el model optimitzat dins d'aquesta sèrie d'arbres amb diferents nivells de poda, seleccionat per tenir el millor rendiment en la validació creuada. Si mirem el valor d'alpha del millor model de la llista, veiem que efectivament es troba en la regió on havíem predit: `ccp_alpha=0.005984555984555988`. Així doncs, si evaluem el nou model (`evaluate_model(tree_models[best_model_index], X_train, y_train)`), podem comparar els resultats d'aquest model òptim amb els primers resultats obtinguts amb el primer arbre de decisió creat:

	Precision	Recall	F1-score
C	0.68	0.65	0.67
CL	0.78	0.88	0.82
D	0.74	0.68	0.70
Accuracy			0.74
Macro Avg	0.73	0.74	0.73
Weighted Avg	0.73	0.74	0.73

Taula 10: Classification Report - `tree_model`

	Precision	Recall	F1-score
C	0.71	0.72	0.71
CL	0.80	0.84	0.82
D	0.78	0.72	0.75
Accuracy			0.76
Macro Avg	0.76	0.76	0.76
Weighted Avg	0.76	0.76	0.76

Taula 11: Classification Report - `tree_models[best_model_index]`

Com podem veure hi ha una diferència entre els resultats obtinguts en els dos models d'Arbre de Decisió, per la qual cosa veiem que ha estat útil fer aquest tros de codi extra per millorar aquest model. Si ens fixem en els resultats, podem veure que en el cas del `tree_model`, les mètriques per a cada classe són les següents:

- Classe C: Una precisió de 0.68, sensibilitat de 0.65 i puntuació F1 de 0.67.
- Classe CL: Una precisió de 0.78, sensibilitat de 0.88 i puntuació F1 de 0.82.
- Classe D: Una precisió de 0.74, sensibilitat de 0.68 i puntuació F1 de 0.70.

La precisió general del model és de 0.74, amb una mitjana macro i una mitjana ponderada de 0.73 en totes les mètriques.

Pel que fa al model `tree_models[best_model_index]`, les mètriques són lleugerament millors:

- Classe C: Una precisió de 0.71, sensibilitat de 0.72 i puntuació F1 de 0.71.
- Classe CL: Una precisió de 0.80, sensibilitat de 0.84 i puntuació F1 de 0.82.
- Classe D: Una precisió de 0.78, sensibilitat de 0.72 i puntuació F1 de 0.75.

En aquest cas, la precisió general és de 0.76, amb una mitjana macro i una mitjana ponderada que igualen aquesta xifra, 0.76, en totes les mètriques.

Comparant els dos models, `tree_models[best_model_index]` supera el `tree_model` en gairebé totes les mètriques individuals i generals. Això indica que el procés de poda basat en la validació creuada ha millorat el rendiment del model en l'equilibri entre l'overfitting i la captura de regularitats en les dades. En particular, l'augment en la puntuació F1 per a la classe D (de 0.70 a 0.75) i les millors puntuacions de precisió i sensibilitat per a les classes C i CL suggereixen que la millora és consistent a través de les diferents classes.

També cal remarcar, però, que hi ha hagut una certa disminució en el *recall* per a la classe CL. En el `tree_model`, la sensibilitat per a la classe D és de 0.88, mentre que en el `tree_models[best_model_index]` disminueix a 0.84. Com que el recall mesura la proporció de positius reals que han estat correctament identificats pel model, aquesta disminució indica que un percentatge més gran de casos positius reals no estan sent capturats pel nou model.

Aquest canvi podria ser atribuït a diversos factors, com ara:

1. La poda de l'arbre pot haver eliminat alguns nodes que capturaven informació específica rellevant per a la classe CL. Això pot haver conduit a una generalització que, mentre millora la precisió i redueix l'overfitting, també pot conduir a una menor capacitat per a reconèixer tots els positius reals d'aquesta classe.
2. La distribució de les classes pot haver influït en la selecció del model, ja que com hem vist al llarg del treball, la classe CL és la menys representada en el dataset, per la qual cosa el model amb poda pot haver priorititzat la precisió en les classes més predominants, que no la sensibilitat en aquesta classe que és menys comuna.
3. Els canvis en el valor de ccp_alpha poden haver conduït a una solució de compromís on la millora en altres mètriques, com la precisió general i la puntuació F1, s'ha donat prioritat sobre el recall de la classe CL.

Malgrat això, el `tree_models[best_model_index]` encara mostra una millora general a través de totes les mètriques, incloent la precisió i la puntuació F1 per aquesta classe. Això suggerix que la pèrdua de recall en aquesta classe pot ser un efecte secundari acceptable dins de la millora del rendiment global.

Per tant, la selecció de `tree_models[best_model_index]` com a model final és la millor opció en aquest punt del treball ja que aquest proporciona una millor capacitat predictiva general, com ho demostra la millora en la precisió general del model del 74% al 76%.

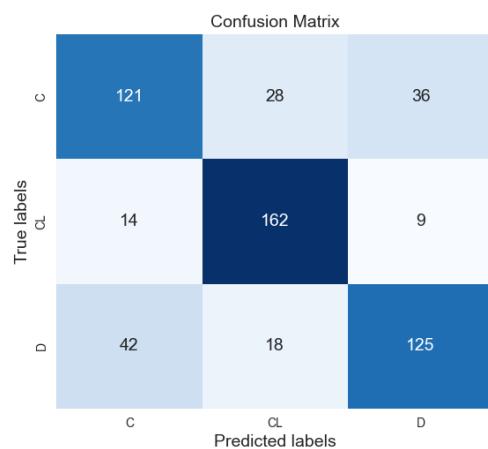


Figura 39: Matriu de confusió - `tree_model`

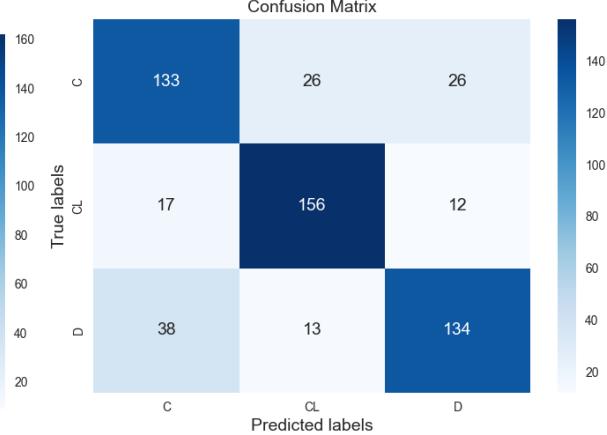


Figura 40: Matriu de confusió - `tree_models[best_model_index]`

També podem veure aquesta diferència en les matrius de confusió (figures 39 i 40), en que podem veure que fa justícia als resultats obtinguts en els dos classification reports. Finalment, cal destacar que s'ha obtingut un F1-score mitjà del model de 0.759, el qual és un pèl pitjor que l'obtingut pel KNN (0.772), per tant ara cal veure quin rendiment dóna el model SVM per aquesta mètrica, i així escollir quin és el millor per al conjunt de train.

A més, també podem veure els dos arbres gràficament (figura ??), en què podem observar clarament com el segon arbre (amb poda) és més senzill i conté menys nodes i divisions que el primer arbre.

Per tant, podem veure manera visual com s'han eliminat certes divisions que aportaven poc a la millora del rendiment del model.

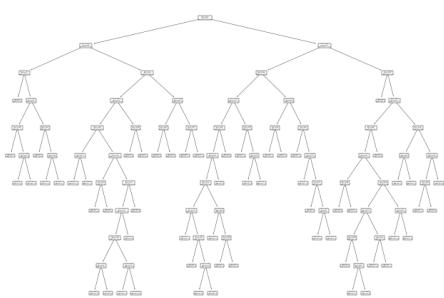


Figura 41: Arbre de Decisió - `tree_model`

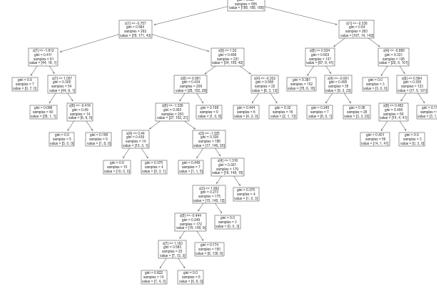


Figura 42: Arbre de Decisió - `tree_models[best_model_index]`

El primer arbre (sense poda) és més complex i profund, amb molts més nodes i divisions. Això ens indica visualment que el model ha après a ajustar-se molt detalladament a les dades del train, per tant pot tenir un rendiment molt alt en les dades de train però generalitzar pitjor en les dades de test.

Finalment, si visualitzem els dos models en fer cross validation, obtenim els següents valors de les mètriques:

Model	Precisio	Recall	Accuracy	F1-score
<code>tree_model</code>	Train: 0.941 Test: 0.737	Train: 0.939 Test: 0.735	Train: 0.939 Test: 0.735	Train: 0.939 Test: 0.731
<code>tree_models[best_model_index]</code>	Train: 0.864 Test: 0.766	Train: 0.862 Test: 0.762	Train: 0.862 Test: 0.762	Train: 0.862 Test: 0.759

Taula 12: Comparació dels resultats de validació creuada - Arbres de Decisió

Com podem veure, el model original `tree_model` mostra una diferència significativa entre les mètriques d'entrenament i de validació, en què hi ha una clara presència d'overfitting. Aquest overfitting es manifesta en l'alta precisió en les dades d'entrenament en comparació amb les dades de validació, indicant que el model està excessivament ajustat a les dades d'entrenament i, per tant, perd una certa capacitat de generalització.

D'altra banda, el model `tree_models[best_model_index]` mostra una menor diferència entre les mètriques d'entrenament i de validació, indicant que la poda de l'arbre ha ajudat a reduir l'overfitting. Aquest model està menys ajustat a les dades d'entrenament i mostra una millor capacitat de generalització, com ho demostra la millora en les mètriques de validació.

En resum, els resultats de la validació creuada indiquen que la poda de l'arbre ha estat efectiva per reduir l'overfitting. El model `tree_models[best_model_index]`, amb poda, mostra una millor capacitat de generalització sense perdre massa rendiment en l'entrenament. Aquesta millora es

reflecteix en un equilibri més adequat entre les mètriques d'entrenament i de validació, el que indica que el model està capturant millor les tendències generals de les dades sense ajustar-se excessivament a les de l'entrenament.

4.6.3 SVM - Entrenament amb Train

Per tal d'entrenar correctament aquest model, en primer lloc hem d'importar la classe `SVC` de la biblioteca `sklearn.svm`, que és l'implementació de SVM per a tasques de classificació. A continuació, instanciem l'objecte `SVC` amb els paràmetres que hem trobat anteriorment, que concretament han estat: `kernel='rbf'`, `C=1.0`, `gamma='scale'` i `random_state=33`. Així doncs, amb el model SVM ja configurat, podem procedir a entrenar-lo utilitzant el conjunt de dades d'entrenament (`svm.fit(X_train, y_train)`):

Aquest mètode ajusta el model SVM als exemples d'entrenament proporcionats, trobant els hiperplans òptims que divideixen les classes.

Finalment, avaluem el rendiment del model entrenat utilitzant la funció `evaluate_model`, en què obtenim els següents resultats: **F1-score for the model:** 0.8322636743060194

	Precision	Recall	F1-score
C	0.78	0.76	0.77
CL	0.89	0.94	0.91
D	0.83	0.81	0.82
Accuracy	0.83		
Macro Avg	0.83	0.83	0.83
Weighted Avg	0.83	0.83	0.83

Taula 13: Classification Report - SVM

El rendiment del model SVM, després de la validació creuada, mostra una puntuació F1, precisió i accuracy uniformes de 0.83. Aquestes mètriques indiquen que el model SVM no només classifica les instàncies correctament en general, sinó que també manté un balanç entre la identificació de casos positius i la precisió amb la qual fa aquestes identificacions ,i dels 3 models que hem provat és el que mostra uns valors d'aquestes mètriques més alts que la resta.

Com podem veure, té una precisió de 0.78 per a la classe C (el 78% de les instàncies classificades com a C són correctes), mentre que les altres puntuacions de precisió per a les classes CL i D són les més altes que hem obtingut fins al moment (0.89 i 0.83, respectivament), doncs suggereixen que el model és molt fiable quan prediu una instància com a positiva en aquestes categories.

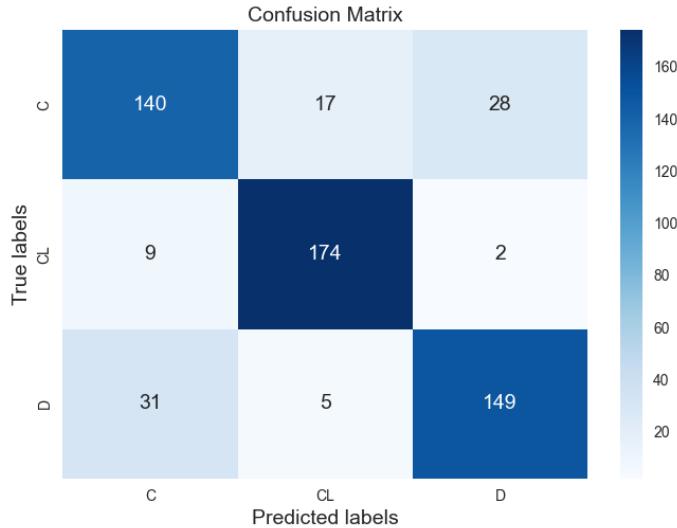


Figura 43: Matriu de confusió del model SVM - Train

Si visualitzem la matriu de confusió generada per la funció d'avaluació del model, podem veure que és la que millors resultats dóna per les classes 'C' i 'D', mentre que el KNN prediu lleugerament millor la classe 'CL'.

Finalment, si ens centrem en els resultats de les diferents mètriques al fer validació creuada, obtenim el següent:

Mètrica	Train	Test
Precisio	0.899	0.838
Recall	0.900	0.834
Accuracy	0.900	0.834
F1-score	0.899	0.832

Taula 14: Valors mitjans de les Mètriques - Cross Validation

Com podem veure, aquestes mètriques mostren una diferència entre els valors obtinguts en el conjunt d'entrenament i en el de validació, tot i que no és una diferència tan gran com en el cas dels models anteriors. Per tant, de la taula anterior podem concloure que segueix havent-hi un cert overfitting, ja que les mètriques d'entrenament són superiors a les de validació, però aquesta diferència no és excessiva, el que suggereix que, mentre el model està lleugerament ajustat a les dades d'entrenament, encara manté una bona capacitat de generalitzar a noves dades.

Així doncs, si el comparem amb els models KNN i Arbre de Decisió, el SVM mostra un millor equilibri entre les mètriques d'entrenament i de validació, suggerint que és menys propens a l'overfitting i pot ser una millor opció per a la classificació en aquest conjunt de dades.

Per tant, donat el seu millor rendiment en la validació creuada i una menor tendència a l'overfitting, així com uns millors resultats en gairebé totes les mètriques, el model SVM sembla ser la millor elecció entre els tres models analitzats per al problema de classificació de la variable Status, ja que en el conjunt d'entrenament hem pogut veure la seva capacitat per a adaptar-se i generalitzar

eficaçment, lo qual el fa preferible als altres models analitzats. Tot i així, ara cal veure com rendeixen amb un conjunt de dades nou, que es tracta de la partició de test, i veure si el SVM continua sent el millor o hi ha un altre model que dona la sorpresa i dona un major rendiment amb aquest nou conjunt de dades.

5 Selecció del model

Un cop ja hem entrenat els diferents models en el conjunt de train, hem vist que, a priori, la millor opció sembla ser el Support Vector Machine (SVM), ja que és el que dona com a resultat un valor major de f1-score, que com hem explicat anteriorment és la mètrica que volem maximitzar. Tot i això, ho hem de contrastar amb la partició del test, ja que d'aquesta manera decidirem quin dels 3 models és el millor pel problema proposat de classificació, doncs des que vam particionar el dataset, aquest conjunt de dades no ha estat modificat en cap moment, excepte per imputar els missings (amb la informació del train), per tant, serveix com una representació honesta de dades noves, per la qual cosa el fet de contrastar els resultats amb aquesta partició ens proporciona una mesura objectiva i imparcial del rendiment real del model.

Per tal d'optimitzar el codi he creat una funció `evaluate_final_model(model, X_test, y_test)` que fa la mateixa funció que `evaluate_model()` però, en comptes de fer Cross Validation, prediu els resultats i els compara amb el conjunt de test, donant com a output el Classification Report i la matriu de confusió que hem anat veient fins al moment per tal de veure com rendeix cada model amb conjunts de dades noves.

Així doncs, podem procedir a avaluar els tres models diferents en la partició de test.

5.1 Resultats en partició de test

5.1.1 KNN - Resultats amb Test

En avaluar el KNN amb el conjunt de prova, obtenim el següent report de classificació:

	Precision	Recall	F1-score
C	0.74	0.55	0.63
CL	0.09	0.40	0.14
D	0.69	0.56	0.62
Accuracy	0.55		
Macro Avg	0.51	0.51	0.47
Weighted Avg	0.68	0.55	0.60

Taula 15: Test Classification Report - KNN

Com podem observar, els resultats mostrats a la taula per la classe 'C' indiquen una precisió de 0.74, la qual és relativament alta comparada amb les altres classes. Això suggereix que quan el model prediu una instància com a pertanyent a la classe C, hi ha una probabilitat bastant considerable que aquesta predicción sigui correcta. No obstant això, la sensibilitat (recall) de 0.55 per a la mateixa classe indica que el model no està capturant un percentatge significatiu d'instàncies reals de la classe C, és a dir, moltes d'elles es perdren o es classifiquen incorrectament com a pertanyent a altres classes.

La classe CL, d'altra banda, presenta una precisió extremadament baixa de 0.09, juntament amb un recall de 0.40. Encara que aquesta sensibilitat pugui ser acceptable, la baixa precisió i un F1-score de només 0.14 indiquen clarament que el model no classifica gens bé les instàncies d'aquesta classe. Per a la classe D, els resultats són bastant millors que els de la classe CL, amb una precisió de 0.69 i una sensibilitat de 0.56, resultant en una puntuació F1 de 0.62. Aquests números indiquen un

rendiment moderat, suggerint que el model té un cert grau d'habilitat en la detecció i classificació d'instàncies de la classe D, encara que encara hi ha un ampli marge de millora.

Per altra banda, l'accuracy general del model KNN és de només 0.55, la qual cosa indica que més del 40% de totes les prediccions realitzades pel model són incorrectes. Aquesta puntuació reflecteix la capacitat general del model per classificar correctament les instàncies a través de totes les classes i evidencia limitacions significatives en la seva aplicabilitat com a solució fiable per a la classificació de la variable Status.

La mitjana macro i la mitjana ponderada, per la seva banda, són de 0.47 i 0.60, respectivament. La mitjana macro és una mitjana aritmètica simple de les mètriques calculades per a cada classe, tractant totes les classes per igual independentment del seu nombre d'instàncies, per tant, el valor obtingut de 0.47 en aquesta mètrica indica que el model no té un rendiment uniforme a través de totes les classes.

Per contra, la mitjana ponderada de 0.60 té en compte el suport de cada classe, és a dir, pondera la contribució de cada classe a la mètrica basant-se en la freqüència de les instàncies d'aquesta classe en el conjunt de dades. El valor més alt de la mitjana ponderada respecte a la mitjana macro suggerix que el model té un millor rendiment en les classes amb més instàncies. Aquesta diferència entre les dues mitjanes pot ser indicativa d'un model que està sobreajustat a les classes més predominants en el conjunt de dades a costa de les classes menys representades.

Com que el que realment importa és el rendiment global tenint en compte la freqüència de cada classe, la mitjana ponderada és una millor mesura del rendiment global del model, per tant un valor d'aquesta de 0.60 no és excel·lent però tampoc és massa dramàtic.

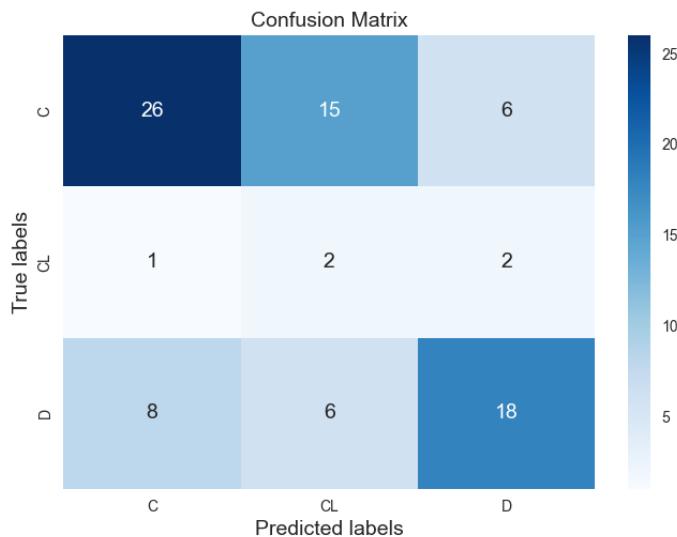


Figura 44: Matriu de confusió del model KNN - Test

Si observem la matriu de confusió obtinguda, veiem que solament hi ha 5 instàncies de la classe CL, la qual cosa justifica en part la mala predicció de la mateixa, ja que només que predigui 2 o 3 malament, les mètriques d'avaluació ja donen un número realment baix. Per altra banda, veiem que tant les classes D com C es prediuen la majoria de les vegades de manera correcta, però el nombre d'instàncies predites erròneament, tot i no ser major del 50%, no era l'esperat, ja que especialment

en el cas de la classe C, aquesta molts cops es prediu com a C.

Finalment, si visualitzem la corba ROC d'aquest model (figura 45), veiem que la corba blava representa la classe C amb un AUC de 0.70, la verda correspon a la classe CL amb un AUC de 0.72, i la línia teixida en blau mostra la classe D amb un AUC de 0.76. Aquests valors d'AUC indiquen que la capacitat del model per distingir entre la classe positiva i la negativa és bastant moderada. La classe D, amb el major AUC de 0.76, és on el model KNN rendeix millor, tot i que cap de les classes arriba a un valor d'AUC que es consideri excel·lent (per exemple, ≥ 0.80).

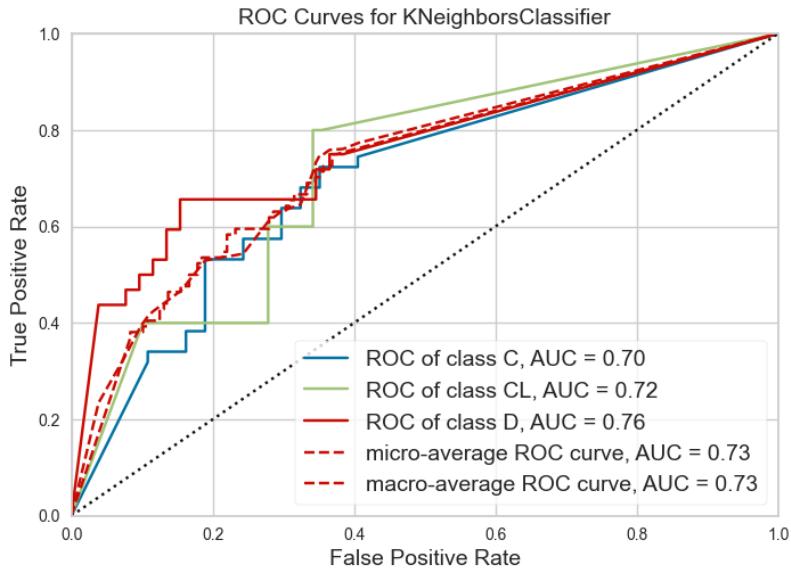


Figura 45: Corba ROC del model KNN

El micro-average AUC ((línia discontinua vermella)) dóna una visió del rendiment global del model, ponderant cada classe per la quantitat d'instàncies. En aquest cas, l'AUC micro-mitjana és de 0.73, el que indica un rendiment global acceptable en el conjunt de dades. Per altra banda, el Macro-average ROC curve, a diferència del micro-average, calcula la mitjana de l'AUC de cada classe sense ponderar pel nombre d'instàncies. També té un valor de 0.73, el que suggereix que el model té un rendiment homogeni entre les classes quan no es té en compte el seu suport. El fet que les corbes micro i macro mitjanes tinguin el mateix valor és interessant i pot indicar que el conjunt de dades està relativament balancejat, o que les diferències en el rendiment del model entre les classes més i menys freqüents s'equilibren entre si.

La línia discontinua negra representa la corba ROC d'un classificador totalment aleatori. Una corba ROC que està significativament per sobre d'aquesta línia indica un model amb un poder predictiu millor que el d'un model aleatori. Com podem veure, totes les corbes ROC del model KNN estan per sobre de la línia discontinua, el que confirma que el model té una capacitat predictiva millor que la casualitat. No obstant això, encara hi ha molt marge de millora, ja que les corbes estan força allunyades de la cantonada superior esquerra, que representaria un rendiment perfecte.

Per tant, un cop fet l'anàlisi complet per aquest primer model, hem vist que el KNN no ens proporciona el nivell de rendiment que desitjaríem en el conjunt de test, tot i que tampoc dona

resultats excessivament dolents. És per això que en aquest punt sembla que no és el model adient per a aquest problema de classificació, per la qual cosa ara cal veure si els següents models donen millors resultats, i escollir el que rendeixi millor dels tres.

5.1.2 Arbre de Decisió - Resultats amb Test

En avaluar l'Arbre de Decisió en el conjunt de test, obtenim el següent report:

	Precision	Recall	F1-score
C	0.72	0.62	0.67
CL	0.17	0.40	0.24
D	0.59	0.59	0.59
Accuracy			0.60
Macro Avg	0.50	0.54	0.50
Weighted Avg	0.64	0.60	0.61

Taula 16: Test Classification Report - Decision Tree

Com podem veure en els resultats obtinguts, a primera vista podem veure una clara diferència en les prediccions de la classe CL, ja que en aquest model les mètriques obtingudes, tot i seguir sent extremadament baixes, són considerablement millors que les que hem obtingut amb el KNN (sobretot en quant a precisió i recall). Si ens centrem en la classe C, veiem que aquesta té una precisió de 0.72, indicant que aproximadament el 72% de les instàncies que el model ha predit com a pertanyent a la classe C eren correctes. No obstant això, el recall d'aquesta categoria és de 0.62, suggerint que el model deixa una quantitat considerable d'instàncies d'aquesta classe sense detectar, però és major que l'obtingut anteriorment amb el KNN.

En comparació, la classe D té una precisió i sensibilitat iguals de 0.59, resultant en una puntuació F1 de 0.59. Aquesta simetria entre precisió i sensibilitat pot indicar un equilibri entre el reconeixement de les instàncies positives i la capacitat de discriminació del model, però ambdues mètriques no són òptimes ni molt menys, en concret la precisió del KNN per aquesta classe era significativament major (0.69).

Per altra banda, la precisió general del model és del 60%, el que significa que 6 de cada 10 prediccions són correctes. Aquesta xifra no és dolenta, però tampoc compleix les expectatives en un model centrat en l'estat d'un pacient, per la qual hauria estat millor obtenir un Accuracy lleugerament superior.

Un altre cop, la diferència entre la mitjana macro i la ponderada suggereix que el model està tenint un millor rendiment en les classes amb més instàncies.

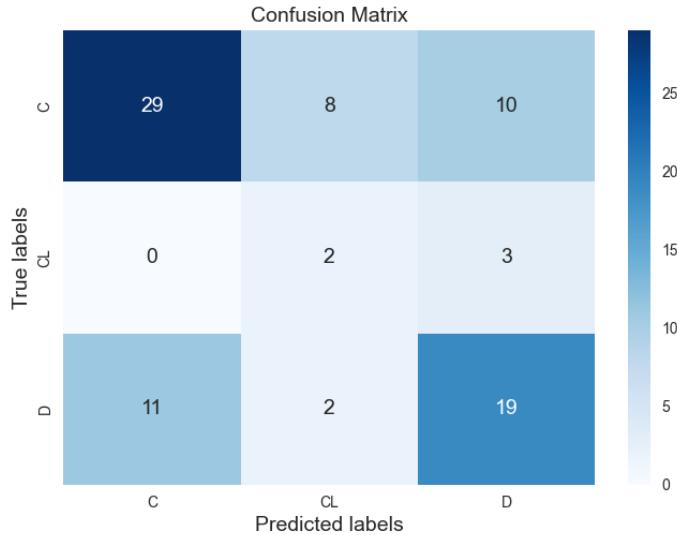


Figura 46: Matriu de confusió del model Decision Tree - Test

Si ens fixem en la matriu de confusió (figura 46), veiem els problemes ja mencionats, concretament podem observar que moltes de les instàncies 'D' han estat predites com a 'C', mentre que una gran part de 'C' han estat predites com a 'CL' i com a 'D', per la qual cosa tampoc és un model amb un gran rendiment, doncs caldrà veure si el SVM el millora, ja que fins al moment els resultats no han estat excessivament satisfactoris.

Finalment, si observem la corba ROC d'aquest model (figura 47), veiem que els valors de l'àrea sota la corba eren lleugerament millors els obtinguts en el KNN, per la qual cosa, si el model SVM no millora significativament aquests dos models, haurem de considerar la corba ROC a l'hora de seleccionar-ne un, ja que en aquest cas hem obtingut uns resultats millors en el model anterior.

Com podem observar, la corba ROC de la classe C té un AUC de 0.67, la classe CL un AUC de 0.64, i la classe D un AUC de 0.73. Aquests valors estan moderadament per sobre de 0.5, que és el rendiment d'un classificador aleatori, però cap d'ells s'acosta a 1, que seria un rendiment perfecte. Això indica que el model té una capacitat limitada per distingir entre les classes positives i negatives. En particular, podem veure que efectivament és la classe CL la que té un valor de AUC més baix (0.64), mostrant les dificultats del model per distingir aquesta classe de les altres.

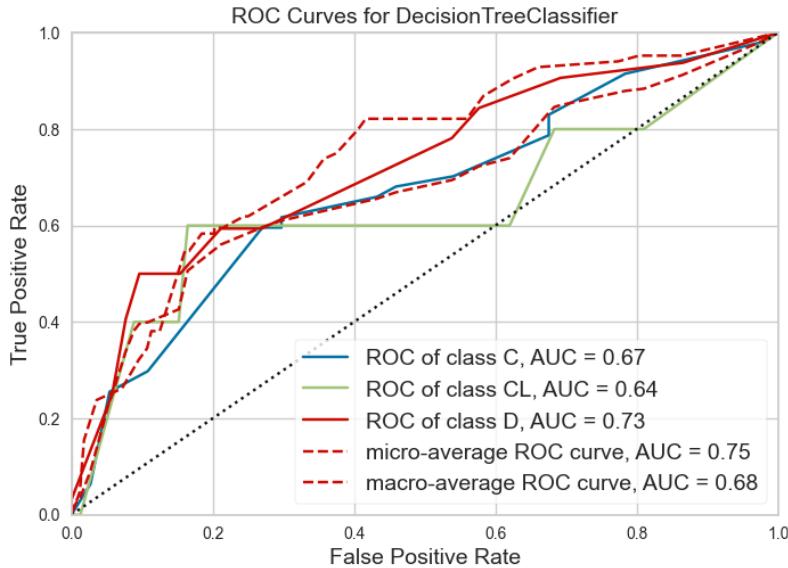


Figura 47: Corba ROC del model Decision Tree

La corba ROC micro-mitjana té un AUC de 0.75, que indica que el model té una bona capacitat de classificació quan es considera el volum d'instàncies. Per altra banda, la corba ROC macro-mitjana té un valor de 0.68, que indica un rendiment més moderat, mostrant que hi ha coherència en el rendiment a través de les classes però amb marge de millora. Comparant les corbes micro i macro mitjanes, podem veure que la corba micro-mitjana està lleugerament per sobre de la corba macro-mitjana, el que suggerix que el model està tenint un millor rendiment en les classes amb més instàncies, tal i com hem deduït en analitzar el report obtingut.

Així doncs, el rendiment global del model Decision Tree es pot considerar acceptable però no òptim. Encara que supera el rendiment d'un classificador aleatori, el model mostra un potencial de millora en la capacitat de distingir entre les classes. En general mostra un rendiment més o menys similar al del KNN en el conjunt de test (pitjors resultats en la corba ROC però millors valors en la majoria de les mètriques del Classification Report). Tot i així, encara hi ha un gran marge de millora, com ara la necessitat d'un recall més alt o una millor precisió en la classe CL. Per tant, ara ens falta veure si l'últim model, l'SVM, supera als dos anteriors o no, i llavors podrem escollir el millor d'entre els tres.

5.1.3 SVM - Resultats amb Test

Finalment, en avaluar el model final de SVM s'obté el següent report:

	Precision	Recall	F1-score
C	0.82	0.66	0.73
CL	0.13	0.40	0.20
D	0.71	0.69	0.70
Accuracy			0.65
Macro Avg	0.55	0.58	0.54
Weighted Avg	0.73	0.65	0.69

Taula 17: Test Classification Report

Si ens fixem en la classe C, observem que té una precisió de 0.82, la qual és la més alta entre les tres classes i la més alta obtinguda fins al moment, per lo que el model SVM sembla que és bastant precís a l'hora de predir que una instància pertany a la classe C. Tot i això, té un recall de 0.66, és a dir, hi ha un percentatge significatiu d'instàncies reals de la classe C que no són detectades pel model. Cal remarcar, però, que malgrat no sigui un recall molt elevat (com ara ≈ 0.80), és el més elevat que s'ha obtingut fins al moment per a aquesta classe. Finalment, la puntuació F1 (que és de 0.73) és evidentment superior a les obtingudes pel KNN i per l'Arbre de Decisió, ja que al ser la precisió i el recall els més elevats fins al moment, era d'esperar que el f1-score també ho fos. Per tant, sense cap dubte aquest model és el que mostra un millor rendiment en la classificació de la classe C de Status.

Per altra banda, la classe CL continua mostrant resultats molt dolents, amb una precisió de només 0.13, indicant que la majoria de les instàncies predites com CL són incorrectes. Com hem vist per a tots els models, aquesta és una classe que difícilment es predirà de manera correcta ja que hi ha molt poques instàncies de la mateixa en la partició de test (únicament 5), per lo que tant el model SVM com els altres que he provat estan lluny de ser fiables per a la detecció d'instàncies de la classe CL.

La classe D, però, presenta una precisió de 0.71 i un recall de 0.69, resultants en una puntuació F1 de 0.70. Aquests valors indiquen que el model SVM té una capacitat raonable per classificar la classe D, amb un bon equilibri entre la identificació de casos positius i la prevenció de falsos positius. Tot i que encara hi ha marge de millora, és el valor més alt obtingut fins al moment per a aquesta classe, lo qual ens fa la idea que segurament aquest sigui el model més adient entre els 3 per al problema de classificació, ja que és el que millor prediu les classes 'D' i 'C'.

Com podem veure, la precisió general (accuracy) del model és del 65%, el que significa que aproximadament dos terços de les prediccions són correctes. Aquesta xifra pot ser acceptable depenent de l'aplicació específica, però cal tenir en compte que la classe CL fa que baixi substancialment, és a dir, realment hauria de ser un valor més alt si no tinguessim en compte la classe CL, la qual al cap i a la fi només té 5 instàncies, i per a l'estudi mèdic és més important distingir entre D i C, que no pas entre C i CL, per exemple. Per tant, l'Accuracy general és bo tenint en compte això i, a més, és el més alt dels tres models diferents.

Si ens fixem en les mitjanes macro i ponderades, veiem que la mitjana macro és de 0.55 i la mitjana ponderada de 0.69, per tant reflecteixen el contrast entre un rendiment equitatiu entre les classes (macro) i un rendiment influenciat per la freqüència de cada classe (ponderada). La discrepància entre aquestes dues mètriques pot ser indicativa de l'incomplet equilibri en el tractament de les classes, particularment la CL, que està significativament subrepresentada en termes de rendiment, per tant si ens centrem en la mitjana ponderada veiem que és un valor bastant acceptable(0.69).

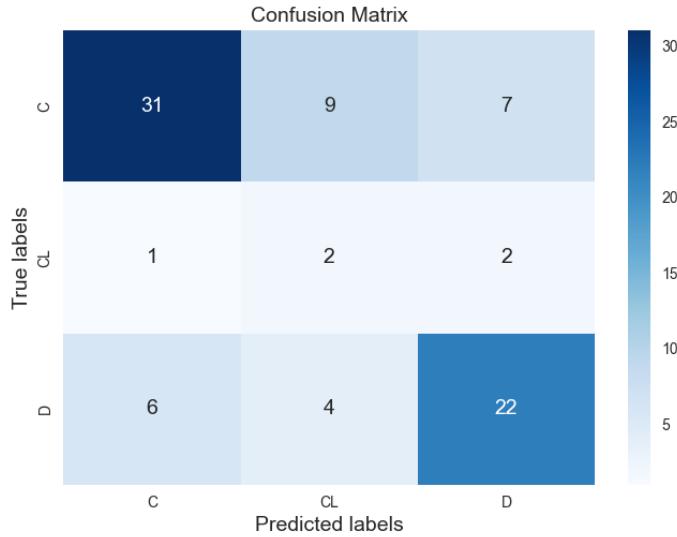


Figura 48: Matriu de confusió del model SVM - Test

Si observem la matriu de confusió, podem veure clarament que és el que millor diferencia les classes C i D d'entre els tres models diferents, ja que és el que té els valors més alts d'instàncies C que s'han predit correctament com a tal (31) i el mateix per a la classe D (22). Observant la corba ROC (figura 49), veiem que sense dubte és el model amb uns valors més grans d'AUC per a totes les classes; la corba per a la classe C té un AUC de 0.81, la classe CL té un AUC de 0.75, i la classe D un 0.82. Aquests valors indiquen una bona capacitat discriminativa per a cada classe individual. En particular, les classes C i D tenen valors d'AUC bastant alts, propers a 0.8, el que indica que el model té una bona capacitat per distingir correctament les instàncies positives d'aquestes classes de les negatives. La corba ROC micro-mitjana té un AUC de 0.84, el qual reflecteix un rendiment global molt bo del model en la classificació, suggerint que quan es consideren totes les classes conjuntament segons la seva freqüència, aquest model té una excel·lent capacitat de discriminació.

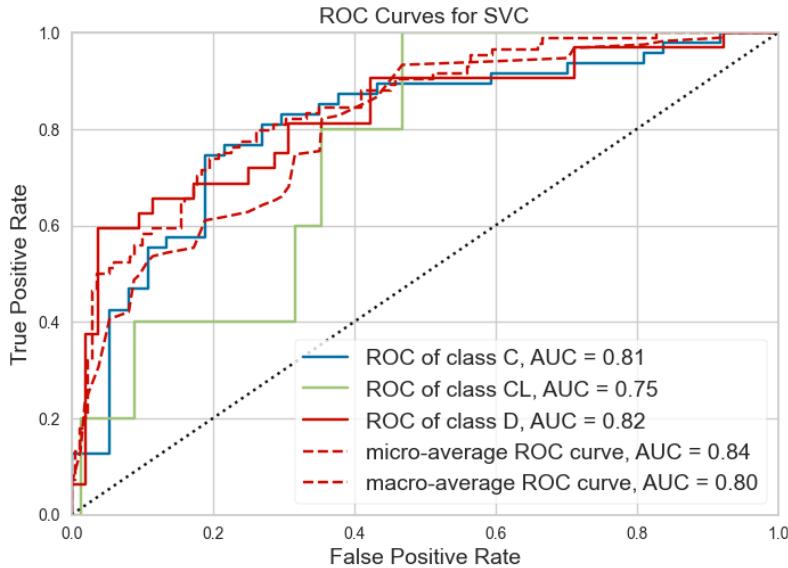


Figura 49: Corba ROC del model SVM

En quant a la corba ROC macro-mitjana, aquesta té un AUC de 0.80, sent lleugerament inferior al de la micro-mitjana, però encara indica un rendiment consistent i relativament alt a través de totes les classes. Com podem veure, la proximitat de les corbes ROC a la cantonada superior esquerra indiquen una taxa més alta de TP respecte als FN.

Per tant, el model SVM demostra ser robust en la discriminació de les diferents classes. Els valors AUC proporcionen una confirmació quantitativa del bon rendiment del model, amb indicacions especialment positives per a les classes C i D.

Així doncs, aquest és un model amb una capacitat bastant notable per a classificar correctament les instàncies de les classes C i D, sent evidentment superior als models anteriors, per tant, arribats a aquest punt decideixo que **el model escollit és el SVM**, per lo que podem procedir a analitzar-lo en profunditat i justificar bé aquesta decisió.

5.2 Model escollit - SVM

L'elecció del model SVM com a millor opció entre els tres models analitzats (KNN, Arbre de Decisió i SVM) es basa en l'anàlisi de l'apartat anterior sobre diversos factors i mètriques que he utilitzat per avaluar el rendiment de cada model en la tasca de classificació de les classes C, CL i D de la variable objectiu 'Status'.

En primer lloc cal destacar les mètriques específiques per a cada classe que proporciona el report de classificació del model SVM. Per a la classe C, aquest model mostra una precisió realment alta de 0.82, lo qual significa que aproximadament el 82% de les instàncies que el model prediu com a pertanyents a la classe C, són correctes. A més, tot i que el recall d'aquesta classe (0.66) no és extremadament alt, és el més elevat entre tots els models que he provat, lo qual vol dir que el model SVM és millor per identificar correctament les instàncies reals de la classe C en comparació amb els altres models. Això es mostra també en un millor valor F1-score (0.73) per a la classe C respecte als altres models, sent aquest el més alt obtingut, a banda d'indicar un bon equilibri entre la precisió i recall.

Per a la classe CL, tots els models (incloent el SVM) han mostrat resultats realment baixos, ja que aquesta classe té molt poques instàncies en el conjunt de dades de test (concretament 5). Malgrat això, és important remarcar que el model SVM ha aconseguit una precisió molt similar a la de l'Arbre de Decisió (0.13 a 0.17), que ha estat aquesta la millor de les tres.

Pel que fa a la classe D, el model SVM ha obtingut una precisió de 0.71, que és superior a la del KNN i similar a la de l'Arbre de Decisió. El recall per a aquesta classe és de 0.69, el que indica que el model SVM és efectiu en identificar instàncies reals de la classe D. El resultat F1 de 0.70 per a la classe D és també notablement millor que els altres models, destacant l'equilibri que aquest model aconsegueix entre la precisió i el recall.

De la mateixa manera, el SVM ha obtingut l'accuracy més alt entre els tres models, indicant la seva eficàcia global en aquesta tasca de classificació complexa.

A més, les mètriques de mitjana macro i ponderada mostren que el model SVM té un millor rendiment en les classes més freqüents, i també ha obtingut els valors més alts d'AUC per a totes les classes. Concretament, les classes C i D tenen valors d'AUC bastant alts (propers a 0.8), que indiquen que el model és molt bo en la discriminació d'aquestes dues classes en particular. Això és fonamental, ja que les classes C i D són les més rellevants des d'un punt de vista mèdic, doncs en la realitat nosalters volem predir si el pacient sobrevisuirà o no, per la qual cosa el que era realment important era distingir la classe D de les altres, la qual cosa s'ha complert satisfactoriament (en certa part, ja que evidentment hi ha molt marge de millora i es podria crear un model bastant millor i més òptim).

Per a poder comparar en una única taula els tres models i conoure finalment les diferències observades entre els tres, a l'hora de predir els valors de y de cada model pel conjunt de test, he guardat els valors de les mètriques considerades importants (accuracy, precision, recall i f1-score) en una taula `results_df` fent servir la funció `save_results(clf, X_test, y_test, nclf, df)`, obtinguda de la secció 5 de Laboratori feta a classe. D'aquesta manera, arribats a aquest punt podem comparar directament les mitjanes de cada mètrica per als tres models diferents:

	Test Acc	Precision Score (W)	Recall Score (W)	F1 Score (W)
KNN	0.547619	0.684559	0.547619	0.599777
Decision Tree	0.595238	0.641766	0.595238	0.613212
SVM	0.654762	0.734743	0.654762	0.686090

Taula 18: Mètriques obtingudes per a cada model

Com podem veure, el model SVM té l'Accuracy més alt, amb un valor de 0.654762, seguit de prop per l'Arbre de Decisió (0.595238) i el KNN (0.547619), per tant d'aquí podem extreure que el SVM és el millor dels models en termes de precisió general de les prediccions.

El Precision Score ponderat mesura la precisió mitjana de les prediccions del model per a totes les classes, donant més pes a les classes amb més instàncies. En aquesta mètrica, el model SVM obte la puntuació més alta (0.734743), el que significa que les seves prediccions són més precises en promig per a totes les classes.

El Recall Score ponderat, per altra banda, mesura la capacitat del model per a identificar correctament les instàncies positives de totes les classes, ponderant les classes amb més instàncies. Un altre cop, el model SVM està al capdavant amb un valor de 0.654762, indicant que és el millor en la identificació de les instàncies positives.

Finalment, el F1 Score ponderat (que combina la precisió i el recall), proporciona una visió equilibrada del rendiment del model. El model SVM també obte la puntuació més alta en aquesta

mètrica (0.686090), indicant que té un equilibri superior entre la precisió i el recall. Com s'ha explicat anteriorment, aquesta és la mètrica que inicialment volíem maximitzar, ja que a l'hora de predir l'estat dels pacients hem decidit que el més important era minimitzar tant els falsos positius com els falsos negatius, per la qual cosa la millor opció era tenir un bon balanç entre precisió i recall, que és justament el que ens dona la mètrica f1-score. Com que el model que ens ha donat millors resultats per a aquesta mètrica ha estat el SVM, no hi ha cap dubte que és el model més adient d'entre els tres per a aquest problema.

Així doncs, la decisió d'escol·lir el model SVM es basa en el fet que destaca en totes les mètriques que havíem considerat importants anteriorment (Accuracy, Precision , Recall i F1 Score) i mostra un rendiment superior en comparació amb els altres dos models (KNN i Arbre de Decisió). A més, és especialment bo en la classificació de les classes C i D, que són les més rellevants per a aquest estudi mèdic. Per tant, basant-me en aquest anàlisi exhaustiu de les mètriques i el rendiment general, és bastant raonable concloure que el model SVM és la millor opció per a la tasca de classificació en aquest dataset.

5.3 Anàlisi de les limitacions i capacitats del model

L'anàlisi exhaustiu de les mètriques i el rendiment del model SVM en la classificació de les classes C, CL i D de la variable objectiu 'Status' ha revelat un rendiment destacat. No obstant això, és fonamental reconèixer tant les capacitats com les limitacions d'aquest model per comprendre plenament el seu potencial i les àrees de millora en futures aplicacions.

5.4 Capacitats

- **Alta Precisió en Classificació (classes C i D):** El model SVM ha demostrat una capacitat notable per assolir una bona precisió en la classificació de les classes C i D. Això s'atribueix principalment a la seva capacitat de crear un hiperplà òptim que separa de manera efectiva les diferents classes. L'ús del kernel RBF ha permès al model gestionar eficaçment les relacions no lineals entre les característiques, cosa que resulta en una alta precisió en la classificació de les classes C i D.
- **Bon Balanç entre Precisió i Recall:** El model SVM ha obtingut un alt F1-score, lo qual demostra un molt bon equilibri entre precisió (capacitat de classificar correctament les instàncies positives) i recall (la capacitat de detectar totes les instàncies positives). Aquest equilibri és crític en l'àmbit mèdic en el qual estem tractant, on tant els falsos positius com els falsos negatius tenen conseqüències significatives. D'aquesta manera, amb un valor de C equilibrat (1.0), el model ha aconseguit un balanç entre la maximització del marge i l'error de classificació, resultant en un bon F1 Score.
- **Bona Generalització:** L'alt valor d'accuracy del SVM en el conjunt de test ha indicat una forta capacitat d'aquest de generalització. A diferència dels models que poden fer un overfitting massa exagerat a les dades del train, el SVM, amb una elecció adequada de paràmetres, pot proporcionar prediccions fiables i consistentes en dades que no ha vist mai, la qual cosa és realment important per a l'aplicació en nous conjunts de dades mèdiques.

5.5 Limitacions

- **Interpretabilitat:** Una limitació significativa del SVM és la seva falta de transparència en la interpretació dels resultats. A diferència d'altres models que proporcionen una visió clara

de com es prenen les decisions, el SVM, especialment amb el kernel RBF, pot fer que el model sigui menys interpretable en comparació amb aquests models més simples (com ara els arbres de decisió), lo qual pot ser una limitació sobretot en contextos mèdics on s'espera una major transparència en les decisions del model.

- **Escalabilitat en Grans Conjunts de Dades:** El SVM pot ser computacionalment exigent, especialment en el cas de grans conjunts de dades. Això es deu al fet que la complexitat del càcul dels vectors de suport augmenta amb el nombre d'instàncies, fent que el model sigui menys pràctic per a aplicacions amb enormes volums de dades. Per tant, si en aquest context mèdic arribessin moltes dades noves de molts pacients, aquest model seria computacionalment més exigent que altres més senzills.
- **Context mèdic - Importància dels errors:** Les mètriques obtingudes pel model SVM en aquest estudi, tot i ser prou bones, haurien de ser un punt de millora significatiu en un context mèdic on la precisió i la fiabilitat són crítiques. Per tant, s'ha de reconeixer que la classificació de l'estat dels pacients és una tasca molt delicada i que qualsevol marge d'error pot tenir conseqüències importants per a la salut dels pacients. Per tant, buscar maneres de millorar les mètriques, minimitzant els falsos positius i falsos negatius, és un objectiu fonamental per garantir una classificació més precisa i fiable en futurs estudis i aplicacions mèdiques. És per aquesta raó que l'objectiu principal d'aquest model és l'aprenentatge i la comprensió dels conceptes de la classificació de dades mitjançant SVM, ja que no està destinat a un ús professional o clínic. Per tant, el seu enfocament és purament educatiu, i els resultats obtinguts han de ser considerats en aquest context.

6 Bonus 1 - Explainable Boosting Machine (EBM)

6.1 Definició del Model

6.1.1 Motivació

Les Màquines de Boosting Explicables (EBM) són una classe de models de màquina d'aprenentatge que combinen la potència predictiva dels mètodes d'ensemble amb la interpretabilitat dels models lineals o arbres de decisió. Aquestes característiques les converteixen en una opció atractiva per a tasques que requereixen tant precisió com transparència en les prediccions. És per això que, un cop haver acabat la part 'obligatòria' de la pràctica, crec que pot ser interessant veure com rendeix aquest model en els conjunts de train i test. Per tant, aquesta secció simplement serveix per veure com rendiria un model EBM en la tasca de classificació de la variable Status, i a més podem obtenir nova informació, com ara quines són les variables més importants en cada conjunt. Així doncs, podem procedir a analitzar més en profunitat aquest nou model.

6.1.2 Complexitat i Interpretabilitat

- **Complexitat:** Els EBM són menys complexos que els models basats en xarxes neuronals profundes però més complexos que els models lineals simples. La seva complexitat ve de l'ús de múltiples models petits (normalment arbres de decisió) per a cada feature, lo que permet capturar interaccions no lineals entre característiques i la variable objectiu.
- **Interpretabilitat:** El gran avantatge del model EBM és la seva interpretabilitat, ja que permet visualitzar l'impacte individual de cada característica en la predicción. Això es fa mitjançant funcions de forma que mostren com cada variable afecta la sortida del model, fent-los útils per a aplicacions en què la comprensió del model és crucial.

6.1.3 Hiperparàmetres

Els EBM permeten l'ajust d'hiperparàmetres específics per optimitzar el rendiment del model. Per al nostre cas d'ús, ens centrarem en la troballa dels hiperparàmetres òptims per a `max_bins` i `learning_rate`:

- **max_bins:** Aquest hiperparàmetre controla el nombre de 'bins' utilitzats per modelar les variables numèriques. Un nombre més gran de 'bins' permet millorar la capacitat del model per capturar tendències dins de les dades, però també pot incrementar el risc d'overfitting.
- **learning_rate:** Aquest hiperparàmetre ajusta la contribució de cada model en l'ensemble durant el procés de boosting. Una taxa d'aprenentatge més baixa pot conduir a un model més robust mitjançant l'increment del nombre d'iteracions necessàries per convergir, mentre que una taxa d'aprenentatge més alta pot accelerar la convergència però pot provocar un overfitting de les dades de train.

6.1.4 Volum de Dades

Els EBM són flexibles i poden treballar amb diferents quantitats de dades. No necessiten tantes dades com altres models més complexos, però tenen millor rendiment quan poden aprendre de moltes dades. Com que el nostre dataset és petit, això pot limitar la capacitat de l'EBM per aprendre tota la complexitat de les dades, ja que és important ajustar bé el model per tal d'evitar que s'adapti massa als casos concrets que ha vist i no pugui predir bé en situacions noves.

6.1.5 Adequació al Nostre Context

En el context del nostre problema de predicció de supervivència de pacients amb cirrosi hepàtica, el model EBM teòricament ofereix un molt bon equilibri entre precisió i interpretabilitat, ja que pot capturar interaccions no lineals entre les característiques clíniques, essencial per a aquest tipus d'aplicacions mèdiques, mentre que al mateix temps proporcionen una comprensió clara de com cada característica influeix en la predicció. Això és fonamental en l'àmbit mèdic, on les decisions basades en models han de ser transparents i fàcilment interpretables tant per als professionals de la salut com per als pacients. Per tant, a priori sembla que és una bona opció, ara cal veure com rendeix en les particions de train i test.

6.1.6 Selecció d'Hiperparàmetres

Com hem fet en els tres models anteriors, per tal de determinar la configuració òptima dels hiperparàmetres, hem utilitzat la tècnica *Grid Search* amb validació creuada de 10 divisions per garantir la fiabilitat dels resultats, en què provem totes les combinacions possibles dels hiperparàmetres escollits i es seleccionen aquells que maximitzen la mètrica de rendiment *f1-score*. En el nostre cas, hem provat els hiperparàmetres `learning_rate` amb els valors {0.01, 0.1, 1} i `max_bins` amb els valors {128, 256, 512}.

Així doncs, utilitzant la classe `ExplainableBoostingClassifier`, després d'entrenar i avaluar totes les combinacions possibles, els resultats han indicat que els hiperparàmetres òptims són un `learning_rate` de 0.1 i un `max_bins` de 512.

6.2 Entrenament amb Train

Per entrenar el nou model EBM utilitzant el conjunt de dades de train, he configurat el model amb els hiperparàmetres trobats en l'apartat anterior (`learning_rate` de 0.1 i `max_bins` de 512), i he desactivat les interaccions entre característiques ('`interactions=0`') per simplificar el model i fer-lo més interpretable, d'aquesta manera el model només considerarà l'efecte individual de cada característica en la predicció, sense tenir en compte com la combinació de característiques pot influir en el resultat. A més, he fixat una llavor (`random_state=33`) per assegurar la reproductibilitat dels resultats.

A continuació, el model s'ha ajustat amb les dades de train utilitzant la funció 'fit' i després s'ha evaluat amb la funció `evaluate_model`. Els resultats de la validació creuada per la precisió, el recall, l'accuracy i el f1-score mostren un rendiment elevat en el conjunt d'entrenament però una caiguda en el conjunt de test. Això suggereix que, tot i que el model aprèn bé les dades d'entrenament, podria estar fent overfitting, ja que el rendiment disminueix quan s'aplica a dades no vistes. Per tant, si ens centrem en les diferències entre les mètriques obtingudes en el train i en el test (aquest últim de la validació creuada, no és el test real), podem treure les següents conclusions:

- **Precisió:** El model té una alta precisió en el conjunt d'entrenament (0.913) però disminueix en el conjunt de test (0.775). Això indica que, mentre que el model és capaç d'identificar correctament la majoria de les classes positives en les dades d'entrenament, la seva capacitat per fer-ho en dades noves és més baixa.
- **Recall:** Una situació similar s'observa en el recall, amb un valor elevat en l'entrenament (0.913) i una disminució en el test (0.768).
- **Accuracy:** L'exactitud segueix el mateix patró, essent alta en l'entrenament (0.913) i més baixa en el test (0.768).

- **F1-score:** La puntuació F1 mostra també una alta puntuació en l'entrenament (0.912) i una més baixa en el test (0.764).

Per tant, podem veure com rendeix en el conjunt de train i posteriorment ja veurem si les mètriques disminueixen molt o no en el conjunt de test. Així doncs, si ens fixem en el *Classification Report*, observem que la classe 'CL' és la que té un millor rendiment en totes les mètriques, amb una precisió de 0.81, un recall de 0.88 i una puntuació F1 de 0.84. Això indica que el model és més àgil en la identificació correcta i en la recuperació dels casos de la classe 'CL' en comparació amb les altres classes, o almenys en aquesta partició, ja que en tots els models anteriors hem vist que, al ser una classe amb tantes poques mostres en el test, les mètriques de la mateixa donen uns valors molt baixos.

	Precision	Recall	F1-score
C	0.71	0.71	0.71
CL	0.81	0.88	0.84
D	0.78	0.71	0.75
Accuracy	0.77		
Macro Avg	0.77	0.77	0.77
Weighted Avg	0.77	0.77	0.77

Taula 19: Classification Report - EBM

La classe 'C' i la classe 'D', per altra banda, tenen una precisió i un recall de 0.71, amb una puntuació F1 de 0.71 per a 'C' i lleugerament superior, 0.75, per a 'D'. Això ens indica que el model és bastant consistent en la seva capacitat de predicció per aquestes classes, equilibrant de manera similar la precisió i el recall, però amb un marge de millora en ambdues mètriques en comparació amb 'CL'.

L'accuracy del model és de 0.77, la qual cosa significa que al voltant del 77% de les prediccions totals són correctes a través de totes les classes. La mitjana macro i la mitjana ponderada de totes les mètriques (precisió, recall i F1-score) també són de 0.77, reafirmant que el model té un rendiment relativament homogeni a través de les classes en el conjunt d'entrenament.

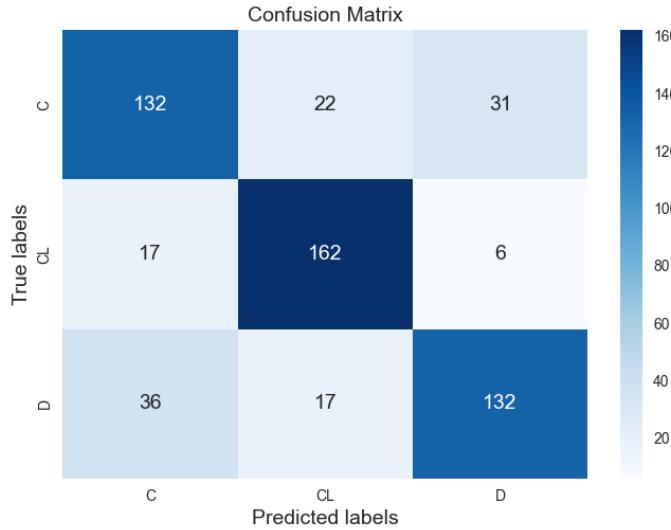


Figura 50: Matriu de confusió del model EBM - Train

Podem reafirmar aquests resultats veient la matriu de confusió en aquesta partició de train, en què podem comprovar que la classe CL és la que té una millor taxa de predicción correcta, amb la major quantitat de veritables positius i la menor quantitat de falsos negatius. La classe C i la classe D mostren un rendiment similar en termes de veritables positius, però la classe C té més falsos negatius comparada amb la classe D.

L'existència de falsos negatius significatius en les classes C i D ens suggereix que el model pot estar tenint dificultats per distingir entre aquestes dues classes, o que hi ha característiques que són similars entre aquestes dues classes que el model confon.

6.3 Importància global dels features

Aquest gràfic mostra la importància global de les característiques (Global Term/Feature Importances) en el model EBM, on les barres representen la importància relativa de cada característica en el model, és a dir, el pes, mesurat pel Mean Absolute Score (Weighted).

D'aquest gràfic podem observar quines són les característiques tenen més pes en les decisions del model. Per exemple, l'edat (Age) i els nivells de bilirrubina (Bilirubin) en els pacients semblen ser les dues característiques més importants, indicant que tenen una gran influència en les prediccions del model. Per tant, aquestes són les que més contribueixen al model.

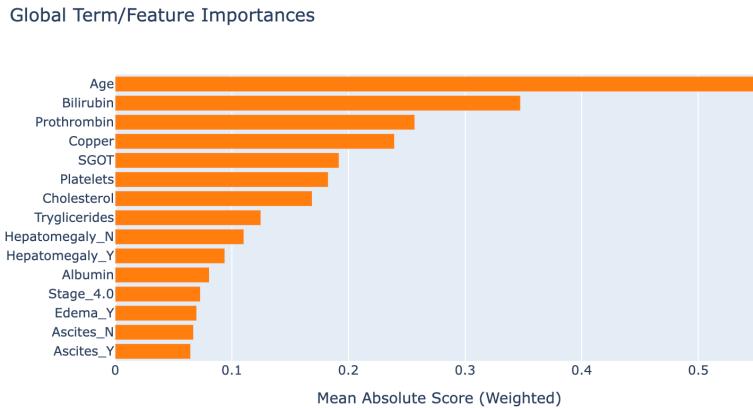


Figura 51: Importància global dels features - EBM

Així doncs, aquest gràfic seria una bona eina per incloure a la Model Card, ja que proporciona informació valuosa sobre el funcionament intern del model EBM i pot servir per a discutir el seu ús, la seva interpretació i els potencials biaixos.

6.4 Resultats amb Test

Un cop vist el rendiment del model en el conjunt d'entrenament, ja podem fer servir la funció `evaluate_final_model` per a veure com rendeix en dades no vistes, és a dir, en el conjunt de Test. Així doncs, un cridem a la funció anterior amb aquest model, obtenim els següents resultats:

	Precision	Recall	F1-score
C	0.84	0.66	0.74
CL	0.17	0.40	0.24
D	0.71	0.78	0.75
Accuracy	0.69		
Macro Avg	0.57	0.61	0.57
Weighted Avg	0.75	0.69	0.71

Taula 20: Test Classification Report - EBM

Com podem veure, per la classe C la precisió és relativament alta (0.84), indicant que quan el model prediu la classe C, és correcte un 84% de les vegades. El recall és més baix a 0.66, el que suggereix que de tots els casos reals de la classe C, el model només n'identifica correctament aproximadament dos terços. Per tant, el f1-score resultant d'aquestes dues mètriques és bastant acceptable (0.74). En quant a la classe CL, aquesta mostra uns valors de les mètriques molt baixos, però com hem vist al llarg del treball, el fet que tan sols tingui 5 mostres en el conjunt de test, fa que les mètriques siguin menys estables i més susceptibles a variacions degudes a la mida de la mostra. Per a la classe D, la precisió i el recall són bastant més equilibrats, amb valors de 0.71 i 0.78, respectivament. Això porta a una puntuació F1 de 0.75, el que és comparable amb la classe C, però amb un millor equilibri entre precisió i recall.

Per altra banda, l'accuracy del model és de 0.69, el que significa que, en general, el model fa una predicción correcta el 69% de las veces a través de todas las clases, siendo el valor más alto entre todos los modelos probados (ya que la accuracy del SVM era 0.65). La media ponderada es más alta (0.75) para la precisión y 0.71 para la puntuación F1, reflejando que el modelo tiene un mejor rendimiento en las clases con más soporte.

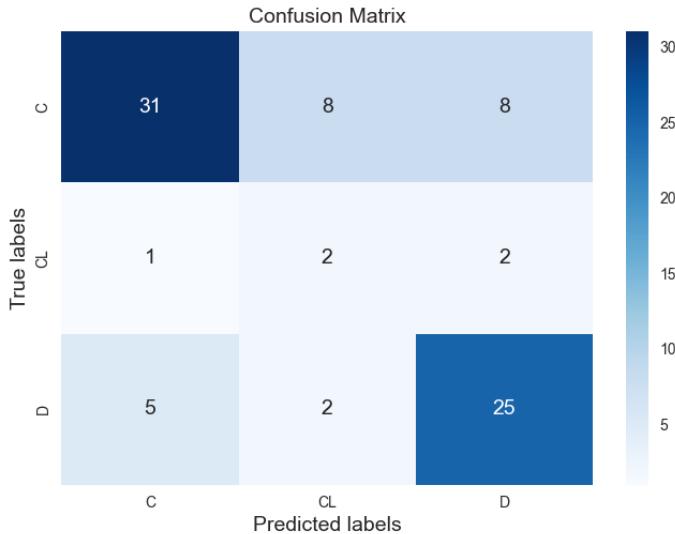


Figura 52: Matriu de confusió del model EBM - Test

Si observem la matriu de confusió obtenida, podem comprovar que la classe C i la classe D mostren un nombre més alt de veritables positius comparat amb la classe CL, lo qual com hem vist es deu a què la classe CL té molt menys casos i per tant és més difícil per al model aprendre a identificar-la correctament, per tant podem veure com això provoca que aquesta classe sigui la que presenta més problemes per al model, tant en termes de precisió com de recall. Això es pot veure en el baix nombre de veritables positius i el nombre igualment baix de casos incorrectament classificats com a CL.

En resum, mentre que la capacitat del model EBM per classificar les classes C i D és acceptable, hi ha clarament espai per a la millora, especialment en la identificació de la classe CL i en la reducció de la confusió entre les classes C i D. Aquestes àrees d'error poden ser una prioritat per a la investigació futura i l'ajust del model.

Finalment, si ens fixem en la corba ROC que obtenim d'aquest model i els valors AUC correspondents, podem veure que la classe C mostra un AUC de 0.80, la qual cosa indica que el model té una bona capacitat de distingir entre la classe C i les altres classes. Com hem vist anteriorment, un valor d'AUC de 0.80 significa que hi ha un 80% de probabilitat que el model classifiqui correctament una muestra aleatoria positiva per sobre d'una muestra negativa. Per a la classe CL i D, que també indica una buena capacidad de discriminación para estas clases, todo lo que podemos ver es que la línea de la curva ROC para la clase CL comienza a incrementarse tarde en comparación con las otras, es decir, no hay ningún modelo en donde haya muchos TP en comparación con los FP.

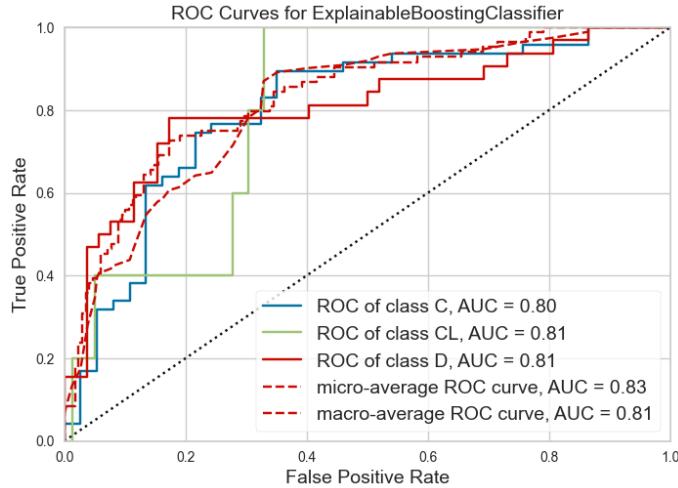


Figura 53: Corba ROC - EBM

El model EBM mostra un bon rendiment en la distinció entre les classes positives i negatives, com s'indica pels valors AUC propers a 0.80 o superiors per a totes les classes. El rendiment consistent a través de les classes i les corbes ROC mitjanes suggereix que el model és bastant robust i fiable en la seva capacitat de discriminació.

Així doncs, si comparem tots els models en la taula `results_df`, veiem que efectivament el model EBM supera als altres tres models (KNN, Decision Tree i SVM) en totes les mètriques de rendiment: accuracy, precisió, recall i la puntuació F1-Score totes ponderades (W) per la quantitat de suport (nombre de mostres per classe).

Model	Accuracy	Precisió	Recall	F1-Score
KNN	0.547619	0.684559	0.547619	0.599777
Arbre de Decisió	0.595238	0.641766	0.595238	0.613212
SVM	0.654762	0.734743	0.654762	0.686090
EBM	0.690476	0.750820	0.690476	0.711280

Taula 21: Comparació del Rendiment dels Models

Tot i això, com que es tracta d'una secció Bonus, vaig prendre la decisió de triar el Support Vector Machine (SVM) sobre els altres dos models inicials (KNN i Decision Tree) sense tenir en compte aquest últim model, ja que volia escollir un dels tres models inicials.

No obstant això, si haguéssim de considerar els quatre models, l'EBM seria l'escollit degut al seu rendiment superior obtingut en les diferents mètriques, a banda de tenir una gran interpretabilitat, que probablement és un dels seus punts més forts si considerem que la tasca és la predicció de l'estat d'un pacient. En un context mèdic, la capacitat d'explicar les decisions del model és molt valuosa, doncs les explicacions poden augmentar la confiança en les decisions assistides per la IA, facilitar la revisió clínica de les prediccions i potencialment revelar intuïcions sobre les relacions entre les característiques i els resultats de salut. D'aquesta manera, la transparència de model EBM pot ajudar a la col·laboració entre els desenvolupadors de models i els experts clínics, fomentant una

comprensió més profunda dels patrons dins de les dades mèdiques.

Per tant, encara que l'SVM ofereix un bon rendiment i és generalment un model robust, l'EBM porta un avantatge addicional que és una molt bona interpretació, la qual pot ser especialment important en l'àmbit de la salut. Aquesta interpretabilitat, combinada amb el seu alt rendiment, fa que l'EBM sigui molt atractiu per aplicacions mèdiques on es prenen decisions crítiques basades en les prediccions del model.

7 Bonus 2 - Clustering

7.1 Identificació de Clusters

Per a aquesta última secció Bonus, he decidit reutilitzar la taula `df_original`, la qual és una còpia de `df` abans de començar amb la Secció 2, i per tant aprofito que aquesta taula ja no conté cap outlier però no s'ha eliminat cap variable (exceptuant 'ID'). Tot i així, com que la imputació de missings la vaig fer després de particionar del dataset, he hagut de tornar a fer-ho, fent servir el mètode 'KNN' per les numèriques tal i com es va fer en l'apartat corresponent. Per a les categòriques, com que la funció que vaig crear per imputar proporcionalment estava pensada per a fer-la servir amb un conjunt de train i un de test, l'he hagut de modificar molt poc per tal que ja no rebi com a argument la partició de test, sinó que rebi un únic dataset, i imputi els missings de cada variable categòrica fent servir la imputació proporcional.

A continuació, he estandarditzat les variables numèriques per tal que totes contribueixin equitativament al procés de clustering. A més, he codificat les categòriques fent servir la funció creada en l'apartat de Recodificació de variables.

Així doncs, un cop fet aquest 'preprocessament' bàsic per a poder fer el clustering, ja podem visualitzar el dendrograma fent servir mètode de 'Ward', el qual minimitza la variància dins dels clústers que es van formant:

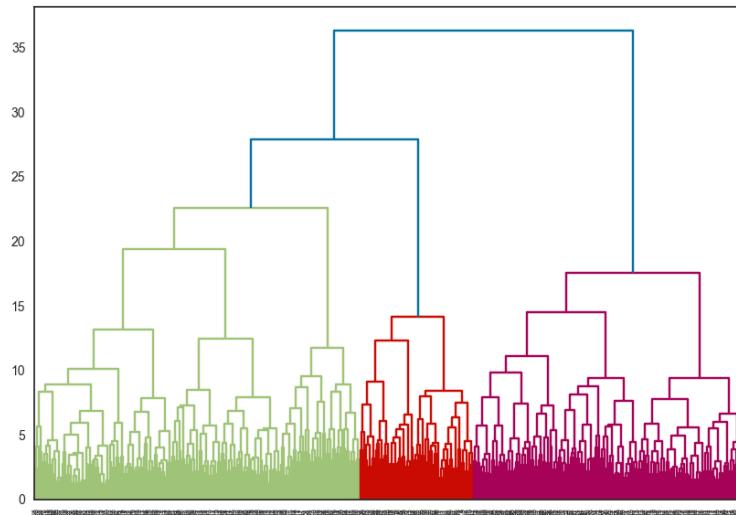


Figura 54: Dendrograma

Com podem veure, hi ha una gran diferència d'altura (distància en l'eix vertical) entre les dues últimes fusió que resultarien en reduir el nombre de clústers de 3 a 2, això pot indicar que hi ha una diferència significativa entre aquests dos grups, i per tant, poden representar diferències substancials entre els pacients.

Com que la variable objectiu Status té tres categories i una d'elles, 'D', indica la mort del pacient, mentre que les altres dues indiquen que el pacient sobrevis, pot ser útil veure si els dos clústers identificats poden correspondre a pacients amb diferents resultats en termes de supervivència, és a dir, veure si un clúster pot tenir una major proporció de pacients que moren (Status = D), mentre que l'altre pot estar més associat amb pacients que viuen o tenen un estat desconegut.

Així doncs, un bon valor de l'alçada de l'eix y per tallar el dendrograma seria 32, ja que al tallar en aquesta alçada obtindriem dos clusters, lo qual podem veure gràficament en el nou dendrograma tallat per aquesta alçada:

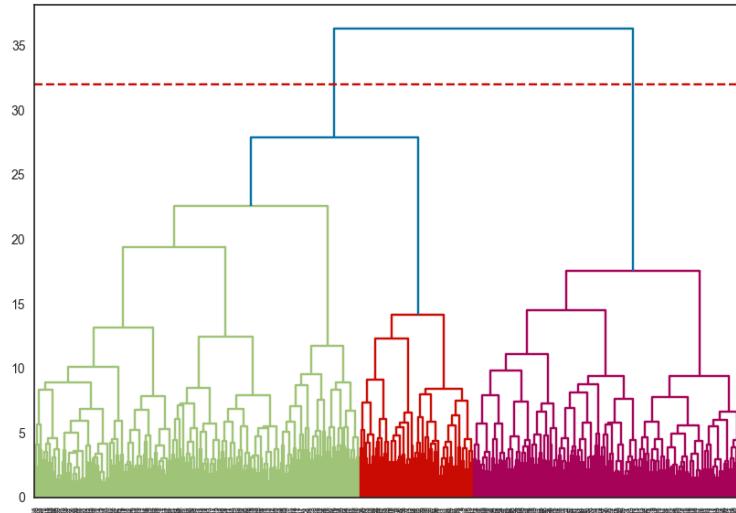


Figura 55: Dendrograma tallat ($y=32$)

7.2 Anàlisi dels Clusters

Un cop ja hem vist que la millor opció per a intentar treure conclusions del clustering es agafar únicament 2 clusters, podem procedir a assignar-los. Per tant, he utilitzat `fcluster` per assignar cada observació al clúster 1 o 2 basant-me en la matriu d'enllaç jeràrquic previament calculada (linked). D'aquesta manera, els clústers s'assignen de manera que només hi ha dos grups principals. Un cop crea els aquests dos grups, es calculen les estadístiques descriptives (mitjana, desviació estàndard, valors mínims i màxims) per a cada clúster separatadament, només per a les columnes numèriques (ja que l'output és molt extens), i s'obtenen els següents resultats:

Variable	Cluster 1		Cluster 2	
	Mean	Std	Mean	Std
N_Days	2749.99	975.34	1406.89	837.22
Age	48.18	9.85	51.59	10.64
Bilirubin	1.02	0.71	2.57	1.71
Cholesterol	298.14	83.08	330.41	85.01
Albumin	3.73	0.32	3.38	0.36
Copper	54.83	34.98	106.33	49.65
Alk_Phosphatase	1135.53	445.08	1473.03	595.16
SGOT	100.47	37.63	130.33	40.66
Tryglicerides	99.81	29.66	126.95	40.56
Platelets	279.90	86.65	235.59	87.49
Prothrombin	10.52	0.62	10.67	0.79

Taula 22: Estadístiques de les columnes numèriques - Valors obtinguts en cada Cluster

D'aquesta taula podem veure que hi ha certes diferències en els valors obtinguts per a les diferents variables numèriques. En primer lloc, les estadístiques del Cluster 1 mostren que els pacients que pertanyen a aquest tenen un valor mitjà de N_Days molt superior als del Cluster 2 (gairebé el doble). També veiem que tenen valors més baixos de Colesterol, Copper, Alk_Phosphatase, SGOT, Tryglicerides i sobretot de Bilirubina. Per tant, sembla que aquest clúster inclou els pacients amb un estat de salut relativament millor i una possible millor supervivència que els del segon cluster. Quant a l'edat dels pacients, veiem que el Cluster 2 té una mitjana de pacients amb més edat que, tot i que la diferència sigui de únicament 3 anys, també és un factor a tenir en compte, ja que durant el treball hem vist que els pacients d'edats més avançades eren més propensos a pertànyer a la classe D de Status.

Per tal d'analitzar-ho més en profunditat, podem visualitzar els valors mitjans i la desviació estàndard de les variables categòriques (no cal incloure els valors mínims i màxims ja que sempre són 0 i 1, en tots els casos). D'aquesta manera, podem veure si hi ha diferències entre els valors obtinguts per a cada classe de les diferents variables categòriques.

Feature	Cluster 1		Cluster 2	
	Mean	Std	Mean	Std
Drug_D-penicillamine	0.477987	0.501093	0.540541	0.499319
Drug_Placebo	0.522013	0.501093	0.459459	0.499319
Sex_F	0.924528	0.264986	0.876448	0.329707
Sex_M	0.075472	0.264986	0.123552	0.329707
Ascites_N	0.962264	0.191159	0.891892	0.311118
Ascites_Y	0.037736	0.191159	0.108108	0.311118
Hepatomegaly_N	0.704403	0.457753	0.335907	0.473221
Hepatomegaly_Y	0.295597	0.457753	0.664093	0.473221
Spiders_N	0.817610	0.387386	0.644788	0.479504
Spiders_Y	0.182390	0.387386	0.355212	0.479504
Edema_N	0.930818	0.254566	0.795367	0.404214
Edema_S	0.062893	0.243538	0.131274	0.338354
Edema_Y	0.006289	0.079305	0.073359	0.261230
Stage_1.0	0.113208	0.317847	0.011583	0.107206
Stage_2.0	0.396226	0.490658	0.115830	0.320641
Stage_3.0	0.339623	0.475077	0.401544	0.491160
Stage_4.0	0.150943	0.359125	0.471042	0.500127

Taula 23: Columnes categòriques - Valors obtinguts en cada Cluster

Com podem veure, en el Cluster 2 hi ha una major presència de pacients masculins en comparació al primer cluster. Veiem també que proporcionalment hi ha més pacients amb Ascites en el Cluster 2 en comparació amb el Cluster 1, la qual cosa ens reafirma que el primer cluster sembla que sigui de pacients amb millors condicions que no pas els que pertanyen al segon.

A més, podem veure que hi ha una gran diferència en el cas de la variable 'Hepatomegaly', ja que hi ha més pacients sense hepatomegalia en el Clúster 1, mentre que hi ha més pacients amb hepatomegalia en el Clúster 2, gairebé el doble de la proporció que en el Clúster 1. En el cas de Spiders, també podem veure que hi ha una proporció més gran de pacients amb aquesta condició en el Clúster 2, com també succeeix amb l'Edema, ja que hi ha més casos d'edema entre els pacients del Clúster 2.

Finalment, si ens fixem en el feature 'Stage', podem observar com hi ha una proporció significativament més gran de pacients en etapes més avançades (Stage 3 i 4) en el Clúster 2, mentre que el Clúster 1 conté més pacients en les etapes inicials (Stage 1 i 2). Com vam veure en analitzar les correlacions de les variables amb la variable objectiu Status, els pacients que es trobaven en etapes més avançades solien estar morts ('D' a Status), lo qual també es complia pels que patien malalties com L'Hepatomegalia, Spiders o Edema, que justament hi ha una gran quantitat de pacients amb aquestes malalties en el Cluster 2 en comparació al Cluster 1.

7.2.1 PCA

Per aprofundir encara més en l'anàlisi del clustering, he aplicat l'anàlisi de components principals (PCA) al conjunt de dades df_clustering. L'objectiu principal és visualitzar com les observacions es distribueixen en l'espai de les dues primeres components principals i com es relacionen amb els clústers prèviament creats.

Un cop calculats els components principals, he creat un gràfic de dispersió on cada punt representa una observació, i la seva coloració indica el clúster al qual pertany. Aquesta visualització permet identificar si les observacions d'un mateix clúster es concentren o es separen en l'espai de les dues primeres components.

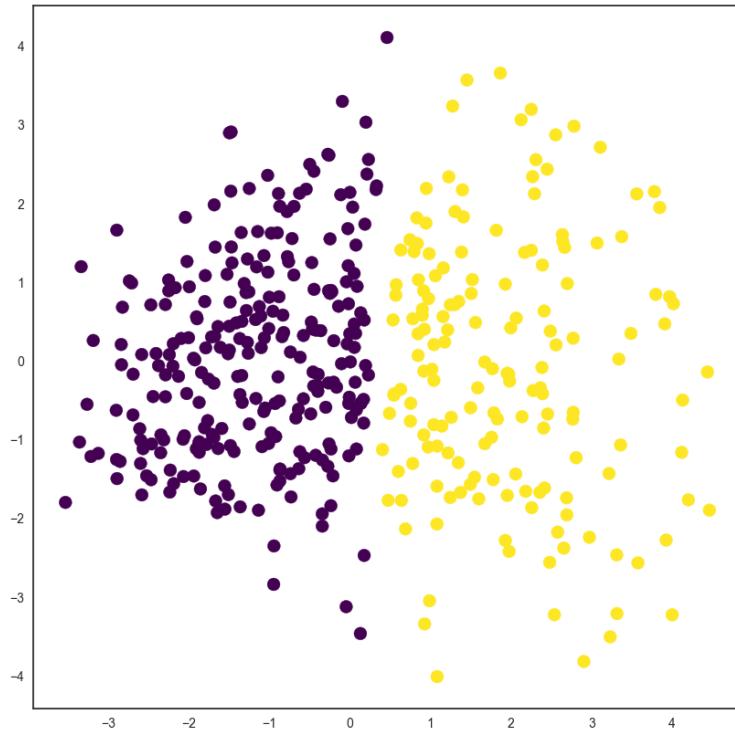


Figura 56: Visualització del PCA amb Coloració per Clúster

Observant la Figura 56, es pot veure clarament una separació distintiva entre els dos clústers. Aquesta separació indica que les observacions d'un mateix clúster comparteixen patrons similars en termes de les dues primeres components principals.

Per aprofundir en la interpretació del PCA, he examinat quines són les característiques (variables originals) que tenen major pes en les dues primeres components principals. Això em proporciona una comprensió de quines variables contribueixen més a la variabilitat entre les observacions.

Component Principal	Variables Destacades
PC1	Bilirubin, Copper, SGOT, Prothrombin, Alk.Phos
PC2	Cholesterol, Alk.Phos, Platelets, SGOT, Albumin

Taula 24: Variables Destacades en les Components Principals

La Taula 24 destaca les variables més rellevants per a cada component principal. Això suggereix que les observacions es diferencien principalment en funció d'aquestes variables en l'espai de les dues primeres components.

Així doncs, amb la informació proporcionada pel PCA, es poden extreure diverses conclusions addicionals sobre els clústers identificats:

- Els dos clústers estan clarament separats tant en els dos primers components com en termes de les variables destacades per cada component.
- Les variables com Bilirubin, Alk.Phos, SGOT i Copper tenen un fort impacte en la separació dels clústers.
- El Clúster 1 sembla estar associat amb valors més baixos en aquestes variables, indicant millors condicions de salut, mentre que el Clúster 2 té valors més alts, indicant situacions més crítiques dels pacients.

7.2.2 Conclusions

Així doncs, un cop fet el Clustering i el corresponent Profiling, hem identificat que el Cluster 1 està fortament associat amb pacients que sobreviuen, indicant possiblement una pertinença a les classes C i CL de Status. Per contra, el Cluster 2 sembla estar relacionat amb pacients que moren, suggerint que aquests majoritàriament pertanyen a la classe D de Status. Aquesta relació entre els clústers i les categories de "Status" suggereix que, fins i tot sense la variable Status directament, podem identificar clústers que capturen informació rellevant sobre l'estat dels pacients. Aquesta identificació de clústers pot ser utilitzada tant en la tasca específica per comprendre perfils de pacients amb diferents resultats de salut com fora d'aquest context per millorar la comprensió i el tractament de pacients amb malalties hepàtiques.

8 Model Card

Detalls del Model

Visió general

Aquest model SVM (Support Vector Machine) està dissenyat per a predir si les instàncies d'un dataset mèdic pertanyen a una de tres classes possibles, identificades com a C, CL i D, que representen diferents estats de la malaltia en pacients. Aquest model està entrenat amb un algorisme SVM que utilitza un kernel de tipus RBF (Radial Basis Function) per a tractar les relacions no lineals entre les característiques. És un algorisme no paramètric que busca el marge màxim per a separar les classes en l'espai de característiques.

Hiperparàmetres:

- C: 1.0
- Kernel: RBF
- Gamma: Scale

Creador: Pau Prat Moreno, pau.prat.moreno@estudiantat.upc.edu

Versió: 1.0

Data de Creació: 20-12-2023

Referències:

- <https://archive.ics.uci.edu/dataset/878/cirrhosis+patient+survival+prediction+dataset-1>
- <https://scikit-learn.org/stable/>

Ús Previst

Casos d'Ús

L'intenció d'ús d'aquest model SVM és com a eina educativa, que pot ser utilitzada per altres estudiants en aquest àmbit. Per tant, aquest model està única i exclusivament destinat a ser utilitzat en un entorn acadèmic per l'aprenentatge i no està pensat per a ser utilitzat en la realització de diagnòstics mèdics reals o decisions de negocis.

Usuaris Intencionats

- Estudiants en Machine Learning

Conjunt de dades

Variables numèriques

Variables categòriques

Mètriques

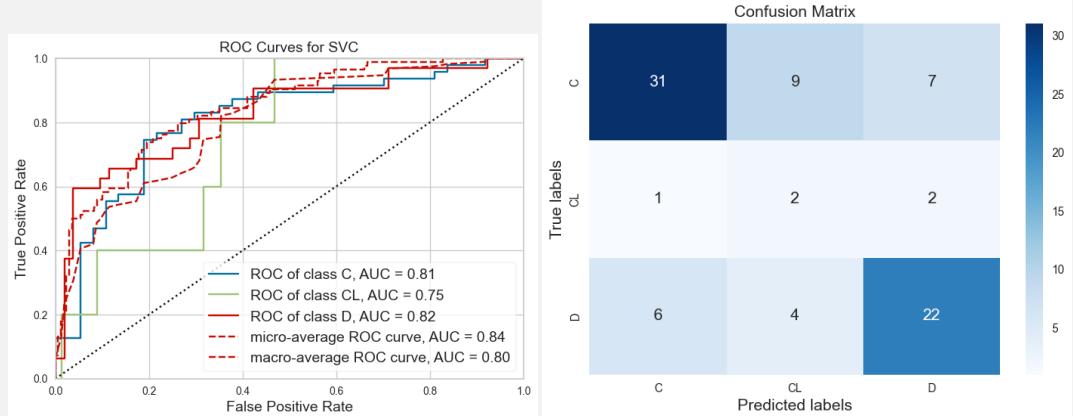
Mètriques de Rendiment

- **Accuracy:** 0.654762 - Indica un nivell decent de precisió general en el conjunt de test.
- **Precision:** 0.734743 - Reflecteix la capacitat del model per classificar correctament les instàncies positives.
- **Recall:** 0.654762 - Mostra la capacitat del model per identificar totes les instàncies positives.
- **F1 Score:** 0.686090 - Combina precisió i recall, suggerint un equilibri entre aquestes dues mètriques.

Aquestes mètriques mostren que el model SVM té un rendiment equilibrat i eficaç en aquesta aplicació específica, tot i que es reconeixen les limitacions en contextos fora de l'àmbit acadèmic.

Anàlisi Quantitativa

Corba ROC i matriu de confusió



Consideracions Ètiques

Consideracions Ètiques

El model SVM s'ha desenvolupat amb una perspectiva educativa i no està destinat a aplicacions clíniques. És crucial evitar l'ús indegut en entorns mèdics reals, on les decisions poden tenir conseqüències significatives per a la salut dels pacients. A més, és important tenir en compte els potencials biaixos en la selecció de dades que podrien influir en els resultats del model. Per a mitigar aquests riscos, recomano:

- **Supervisió Rigurosa:** Utilització del model sota la supervisió d'experts en dades per a assegurar la seva aplicació apropiada.
- **Transparència:** Mantenir una comunicació clara sobre les limitacions del model i no sobrestimar les seves capacitats predictives.

Advertències i Recomanacions

Limitacions del Model

- **Interpretació:** L'ús del kernel RBF pot complicar la interpretació dels resultats, sent una limitació en entorns que requereixen transparència.
- **Ajustament d'Hiperparàmetres:** El rendiment depèn fortament de l'elecció de C i gamma, on un ajust inadequat pot portar a un overfitting o a una generalització pobre.
- **Escalabilitat:** Aquest model es pot tornar ineficient amb grans volums de dades, incrementant els costos computacionals.
- **Desequilibri de Classes:** Amb classes desequilibrades, pot haver-hi un biaix cap a les classes majoritàries, afectant l'actuació en les classes menys representades.

Recomanacions

Malgrat els resultats obtinguts, hi ha àmbits de millora importants per a futures versions del model:

- **Diversificació de Dades:** Ampliar i diversificar el conjunt de dades d'entrenament per millorar la representativitat del model.
- **Escalabilitat:** Optimitzar el model per afrontar grans volums de dades sense comprometre el rendiment.
- **Abordar Desequilibris:** Implementar tècniques per gestionar desequilibris de classes i millorar la classificació de classes minoritàries.
- **Provar Diferents Models:** Realitzar proves amb altres algorismes de Machine Learning, ajustant paràmetres mínims per identificar el model més adequat per aquest problema.