0. Taula de continguts

1. Introducció

2. Informació del problema

3. Anàlisi exploratori descriptiu

En aquesta secció es presenta l'anàlisi exploratori descriptiu (AED) realitzat sobre el conjunt de dades proporcionat per al problema de predicció clínica. L'objectiu principal d'aquesta anàlisi és comprendre millor les característiques de les dades, identificar possibles problemes com valors perduts o *outliers*, i preparar les dades per a l'ajustament dels models predictius.

3.1 Anàlisi de les variables

Observem primer de tot la taula de dades per tenir una visió general de les variables disponibles i la seva tipologia. L'obtenim amb la comanda `df.describe().T`, que ens proporciona estadístiques descriptives per a les variables numèriques i categòriques. El resultat ha estat:

Variable	N	Mean (SD)	Min	25%	Median	75%	Max	Missing (%)
Age	9000	26.04 (10.01)	13.00	19.00	25.00	31.00	64.00	0.00%
Sex (0=F, 1=M)	9000	0.58 (0.49)	0.00	0.00	1.00	1.00	1.00	0.00%
BMI	9000	28.11 (5.43)	15.00	24.40	28.00	31.70	49.60	0.00%
Duration Untreated Psychosis	8872	19.22 (19.55)	0.30	6.40	12.50	24.30	125.00	1.42%
Family History	9000	0.12 (0.32)	0.00	0.00	0.00	0.00	1.00	0.00%
Initial Response	9000	41.84 (30.16)	0.00	10.10	38.20	72.30	100.00	0.00%
Prior Antipsychotics	9000	0.41 (0.67)	0.00	0.00	0.00	1.00	2.00	0.00%
TRS (Target)	9000	0.32 (0.46)	0.00	0.00	0.00	1.00	1.00	0.00%
Lymphocyte count	7009	1.80 (0.60)	0.50	1.38	1.80	2.20	4.02	22.12%
Neutrophil count	7015	5.01 (1.47)	1.50	4.01	5.02	6.01	9.96	22.06%
Triglycerides	6547	152.01 (61.10)	40.00	108.05	151.10	194.60	394.60	27.26%
Glucose	6381	95.86 (18.31)	65.00	82.20	95.50	108.30	159.60	29.10%
Alkaline Phosphatase	6062	85.17 (24.83)	30.00	68.20	84.70	101.90	179.30	32.64%
IL-17A	8999	2.66 (0.80)	-0.20	2.12	2.65	3.21	5.38	0.01%
CCL23	9000	3.78 (1.05)	-0.20	3.08	3.78	4.49	7.69	0.00%
TWEAK	9000	4.19 (1.24)	-0.54	3.37	4.19	5.04	8.92	0.00%
HLA-DRB1*04:02	9000	0.02 (0.15)	0.00	0.00	0.00	0.00	1.00	0.00%
HLA-B*15:02	9000	0.03 (0.18)	0.00	0.00	0.00	0.00	1.00	0.00%
HLA-A*31:01	9000	0.05 (0.21)	0.00	0.00	0.00	0.00	1.00	0.00%
Polygenic Risk Score	8999	0.030 (0.14)	-0.44	-0.07	0.02	0.11	0.58	0.01%
Del 22q11.2	9000	0.009 (0.09)	0.00	0.00	0.00	0.00	1.00	0.00%
Ki Whole Striatum	9000	0.0130 (0.002)	0.0080	0.0113	0.0129	0.0145	0.0200	0.00%
Ki Associative Striatum	9000	0.0130 (0.002)	0.0071	0.0113	0.0128	0.0146	0.0210	0.00%
SUVRc Whole Striatum	9000	1.18 (0.27)	0.80	0.97	1.16	1.36	2.00	0.00%
SUVRc Assoc. Striatum	9000	1.18 (0.27)	0.80	0.96	1.16	1.37	2.00	0.00%

Taula 1: Resum estadístic de totes les variables numèriques i categòriques del conjunt d'entrenament. Es mostra el recompte (N), mitjana i desviació estàndard, quartils i percentatge de valors perduts.

Obeservem que el conjunt de dades conté un total de 9000 mostres i diverses variables predictives, així com la variable objectiu `TRS (Target)`. Algunes variables presenten valors perduts, especialment les mèdiques com el recompte de limfòcits i neutròfils, triglicèrids, glucosa i fosfatasa alcalina, que gestionarem més endavant a la secció 3.3.

Pel que fa les distribucions de les variables, la majoria semblen tenir una distribució aproximadament normal, o de combinacions de normals, com per exemple:

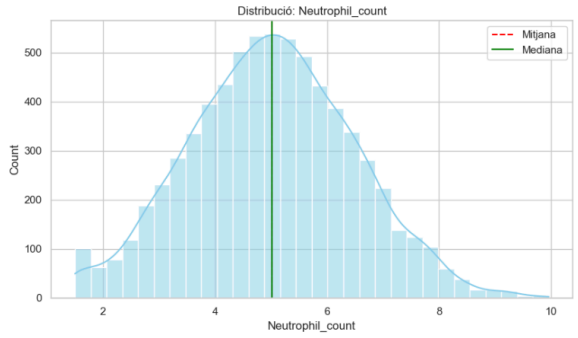

Figura 1: Distribució de `Neutrophil count`.

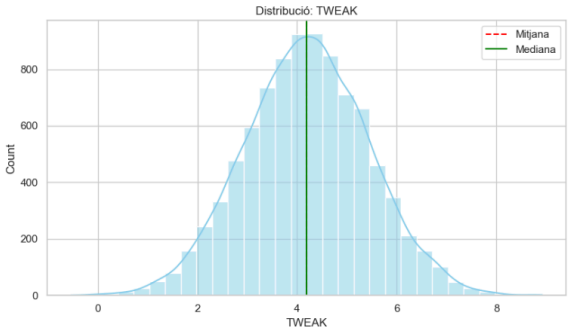

Figura 2: Distribució de `TWEAK`.

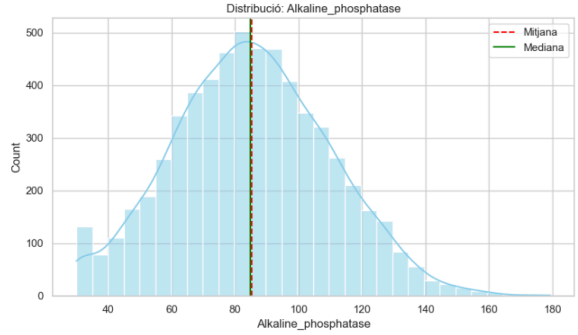

Figura 3: Distribució de `Alkaline phosphatase`

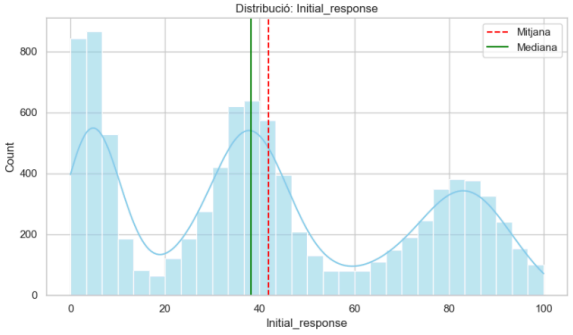

Figura 4: Distribució de `Initial response`.

Aquestes variables es poden utilitzar directament en els models predictius, ja que no requereixen transformacions addicionals per a la seva normalització. No obstant això, algunes variables com el `Duration_Untreated_Psychosis` o `Age` mostren una distribució més asimètrica:

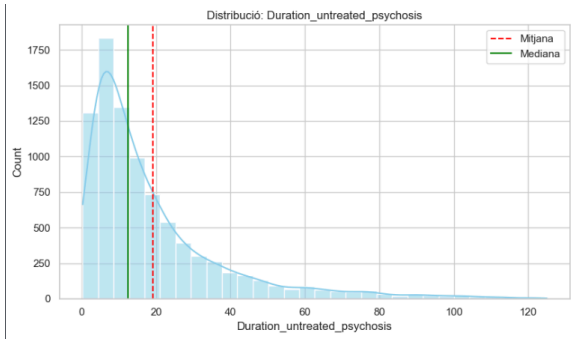

Figura 5: Distribució de `Duration_Untreated_Psychosis`.

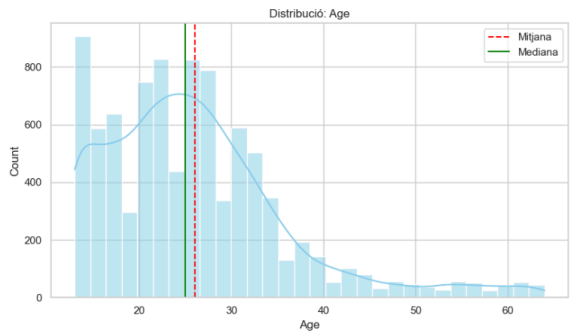

Figura 6: Distribució de `Age`.

En aquest cas podem veure unes cues llargues cap a la dreta, indicant la presència d'alguns valors extrems. Comprovem ràpidament amb el skewness que aquestes variables no són normals. Si tenen un skewness major a 1 o menor a -1, es consideren altament asimètriques. Comprovem:

```
ANÀLISI D'ASIMETRIA (SKEWNESS):
--------------------------------------------------------------------------------
                  Variable  Skewness  Kurtosis                                Estat
Duration_untreated_psychosis  2.150479  5.323006 🔴 MOLT ASIMÈTRICA (Requereix Log/BoxCox)
                       Age  1.258793  2.065980 🔴 MOLT ASIMÈTRICA (Requereix Log/BoxCox)
  SUVRc_associative_striatum  0.457536 -0.422680                    🟢 NORMAL (Simètrica)
        SUVRc_whole_striatum  0.434121 -0.408815                    🟢 NORMAL (Simètrica)
        Polygenic_risk_score  0.405186  0.323084                    🟢 NORMAL (Simètrica)
     Ki_associative_striatum  0.328552 -0.022675                    🟢 NORMAL (Simètrica)
...
```

Aquest desequilibri en la distribució de les dades pot afectar el rendiment dels models predictius, especialment aquells que assumeixen normalitat en les variables. Per tant, considerarem aplicar

transformacions com el logaritme o Box-Cox a aquestes variables abans de l'ajustament dels models.

Mirem la correlació entre les variables numèriques per identificar possibles relacions lineals que puguin ser útils per a la predicció. Utilitzem un mapa de calor per visualitzar aquestes correlacions:

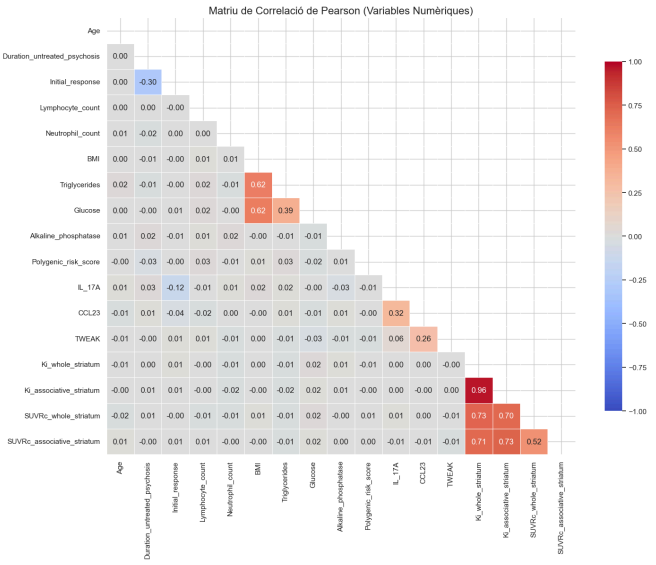

Figura 7. Matriu de Correlació de Pearson

Veiem que en termes generals, les correlacions entre les variables molt baixes o pràcticament 0, amb algunes excepcions:

Aquesta imatge mostra una **matriu de correlació de Pearson** (heatmap), que mesura la relació lineal entre diferents variables numèriques del teu dataset. Els valors van de **-1** (blau fosc, correlació negativa perfecta) a **+1** (vermell fosc, correlació positiva perfecta), passant per **0** (gris/blanc, sense correlació).

Aquí tens la interpretació detallada dels patrons més rellevants:

1. Redundància a les Variables avall dreta (Vermell Fosc) Aquest és el grup més destacat de la gràfica. Les variables relacionades amb l'estriat (*striatum*) mostren correlacions extremadament altes.
    - **`Ki_whole_striatum`** vs. **`Ki_associative_striatum`:** 0.96
    - **`Ki`** vs. **`SUVRc`**: ~0.70 - 0.73
2. Clúster Metabòlic (Vermell Mig):
    - **`Triglycerides`** vs. **`Glucose`**: 0.62
    - **`Glucose`** vs. **`BMI`**: 0.39
3. Relació entre no tratament i resposta inicial (Blau Fosc):
    - **`Duration_untreated_psychosis`** vs. **`Initial_response`**: -0.30

Aquestes correlacions suggereixen que certes variables estan fortament relacionades i podrien ser redundants en els models predictius. Gestionarem aquestes relacions en la fase de selecció de característiques per optimitzar el rendiment dels models.

3.2 Desbalanceig de la classe objectiu

La variable objectiu és `TRS`, que indica si un pacient desenvolupa resistència als tractaments antipsicòtics. La proporció de pacients la mirem executant `ratio_trs = df['TRS'].value_counts(normalize=True)`. El resultat és que el percentatge d'individus a la dataset que desenvolupa TRS és aproximadament del 31.5%, mentre que el 68.4% restant no desenvolupa resistència. Això indica un desbalanceig en les classes, que haurem de gestionar a l'hora de modelar. Aquest desbalanceig es denota en les variables categòriques:

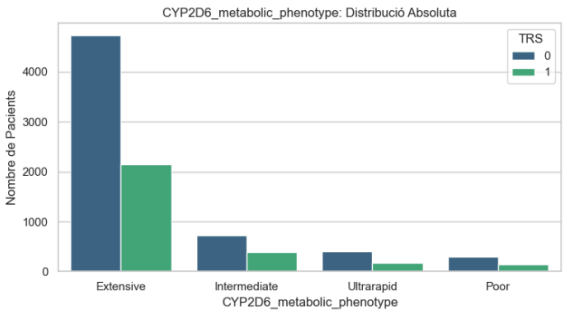

Figura 8: Distribució de la variable objectiu TRS

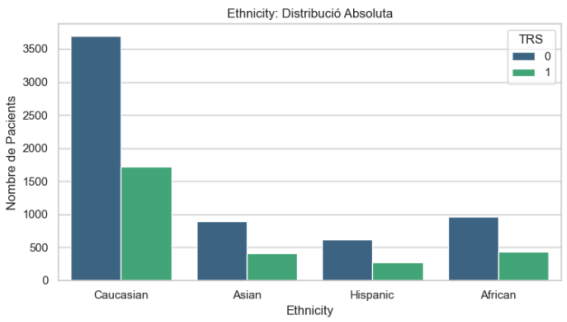

Figura 9: Distribució de la variable objectiu TRS

Veiem no hi ha cap biaix que provoqui el desbalanceig, ja que la distribució de les altres variables categòriques és força equilibrada. Per tant, per gestionar aquest desbalanceig en la fase d'ajustament dels models, considerarem tècniques com assignar pesos a les classes.

3.3 Valors perduts

Com podem veure a la Taula 1, algunes variables tenen un percentatge significatiu de valors perduts. Veiem-ho en més detall:

Variable	Total Missings	Percentatge (%)
Alkaline_phosphatase	2938	32.64%
Glucose	2619	29.10%
Triglycerides	2453	27.26%
Lymphocyte_count	1991	22.12%
Neutrophil_count	1985	22.06%
Duration_untreated_psychosis	128	1.42%
Polygenic_risk_score	1	0.01%
IL_17A	1	0.01%

Taula 2: Valors perduts per variable.

Experimentalment he provat d'imputar els valors perduts amb diferents tècniques, com la mitjana, mediana, KNN i regressió. Després d'avaluar el rendiment dels models amb aquestes diferents imputacions, he observat que la imputació mitjançant KNN amb k=5 ofereix els millors resultats en termes de precisió i robustesa del model. Per tant, he decidit utilitzar aquesta tècnica per gestionar els valors perduts en les variables mèdiques.

3.4 *Outliers*

He fet servir el mètode del IQR per detectar valors extrems en les variables numèriques. Aquest mètode defineix els *outliers* com aquells valors que es troben fora de l'interval $[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$, on $Q1$ i $Q3$ són el primer i tercer quartil, respectivament, i IQR és el rang interquartílic ($Q3 - Q1$). La taula resultant mostra el nombre d'*outliers* detectats per variable:

Variable	n outliers	Outliers (%)
Duration_untreated_psychosis	651	7.23
Age	360	4.00
Polygenic_risk_score	125	1.39
Ki_associative_striatum	83	0.92
Ki_whole_striatum	71	0.79
CCL23	59	0.66
TWEAK	58	0.64
IL_17A	57	0.63
BMI	39	0.43
SUVRc_whole_striatum	35	0.39
Neutrophil_count	32	0.36
SUVRc_associative_striatum	29	0.32
Lymphocyte_count	26	0.29
Alkaline_phosphatase	23	0.26
Triglycerides	22	0.24
Glucose	16	0.18

Taula 3: Variables amb valors extrems (outliers).

Com veiem, la variable amb més outliers és `Duration_untreated_psychosis`, amb un total de 651 valors extrems, representant el 7.23% del total de mostres. Aquesta variable mostra una distribució molt asimètrica, amb una cua llarga cap a la dreta, indicant que hi ha alguns pacients amb períodes molt llargs sense tractament. Aquesta variable podria beneficiar-se d'una transformació logarítmica per reduir l'impacte dels outliers en els models predictius. El mateix passa amb la variable `Age`, que també presenta una quantitat significativa d'outliers (360 valors, 4.00%). La resta de variables tenen un nombre relativament baix d'outliers, tots per sota de l'1% del total de mostres.

Per tant, he decidit no eliminar aquests outliers, ja que podrien contenir informació rellevant sobre pacients amb característiques extremes. En lloc d'això, aplicaré transformacions adequades a aquestes variables per minimitzar el seu impacte en els models predictius. Pel que fa a les altres variables amb pocs outliers, no aplicaré cap acció específica, ja que la seva presència és mínima i no afectarà significativament els resultats dels models. A més, en un context mèdic, eliminar valors extrems podria conduir a la pèrdua d'informació important sobre pacients amb condicions rares o greus, que poden ser crucials per a la predicció.

3.5 Estudi de dimensionalitat

He realitzat una anàlisi de components principals (PCA) per avaluar la dimensionalitat del conjunt de dades i identificar possibles reduccions de dimensions que puguin millorar l'eficiència dels models predictius. La PCA és una tècnica estadística que transforma les variables originals en un nou conjunt de variables no correlacionades, anomenades components principals, que capturen la major part de la variància present en les dades. Com que no tolera valors perduts, he utilitzat el conjunt de dades amb els valors imputats mitjançant KNN. També eliminem la variable objectiu `TRS` abans d'aplicar la PCA, ja que aquesta tècnica només s'aplica a les variables predictives i la variable `id_patient`, que és un identificador únic per a cada pacient i no aporta informació rellevant per a la predicció. L'apliquem fent ús de la lliberieria `sklearn.decomposition.PCA`, i els resultats obtinguts són els següents:

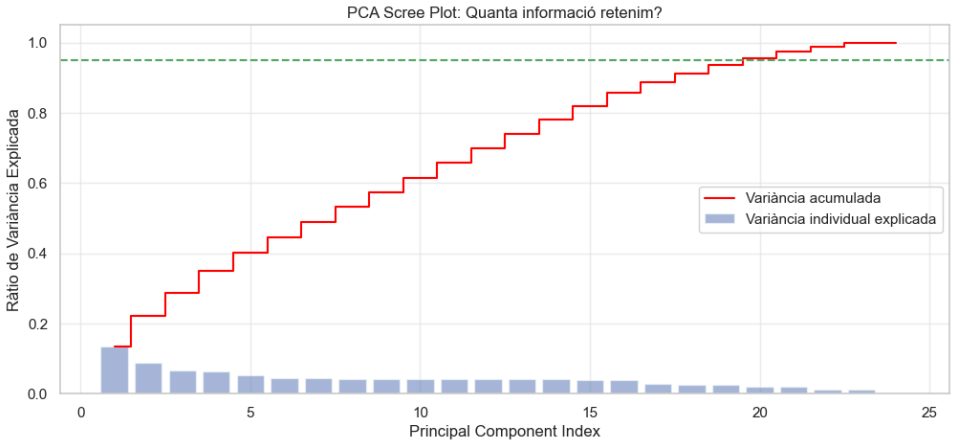

Figura 10: Gràfica de Scree Plot de la PCA

Podem veure que calen 20 components per explicar el 95% de la variància total del conjunt de dades. Això indica que hi ha una certa redundància entre les variables originals, ja que només es necessiten 20 components per capturar la major part de la informació. No obstant això, com que el nombre original de variables no és molt alt, he decidit no reduir la dimensionalitat en aquesta fase i mantenir totes les variables originals per a l'ajustament dels models. Això permetrà als models aprofitar tota la informació disponible i potencialment millorar el rendiment predictiu.

3.6 Partició del conjunt de dades

En models de machines learning, és fonamental dividir el conjunt de dades en subconjunts d'entrenament i prova per avaluar el rendiment dels models de manera objectiva. He utilitzat una divisió del 80% per a l'entrenament i del 20% per a la prova, assegurant-me que ambdues particions mantinguin la mateixa proporció de la variable objectiu `TRS` (estratificació). Això es fa per garantir que els models es trenin i s'avaluïn en mostres representatives de totes les classes. He utilitzat la funció `train_test_split` de la biblioteca `sklearn.model_selection`, amb el paràmetre `stratify` per assegurar aquesta estratificació.

Aquesta divisió es farà de manera individual en cada model, per assegurar que qualsevol preprocessament específic del model no afecti la partició dels dades.

4. Ajustament de models

Els 3 models bàsics a ajustar són Support Vector Machine (SVM), XGBoost i una regressió logística personalitzada. A continuació es detallen els passos seguits per a cada model, incloent el preprocessament, l'ajustament del model i els resultats finals obtinguts.

4.1 SVM

Support Vector Machine (SVM) és un model de classificació potent que busca trobar l'hiperplà que millor separa les classes en l'espai de característiques. Aquest SVM serà ajustat amb la biblioteca `sklearn.svm.SVC`, provant diferents nuclis i ajustant els hiperparàmetres mitjançant una cerca en quadrícula (`GridSearchCV`).

4.1.1 Preprocessament

El processament de les dades per al model SVM inclou els següents passos:

1. Particionem les dades: Tal i com s'ha descrit a la secció 3.6, dividim el conjunt de dades en un 80% per a l'entrenament i un 20% per a la prova, assegurant-nos que ambdues particions mantinguin la mateixa proporció de la variable objectiu `TRS`:

```
X_train_svm, X_test_svm, y_train_svm, y_test_svm = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

2. Codificació de variables categòriques: Utilitzem l'encodificació One-Hot per a les variables categòriques, que són `Ethnicity` i `CYP2D6_metabolic_phenotype`. Això crea variables binàries per a cada categoria, permetent que el model SVM les utilitzi correctament-

3. Imputació de valors perduts: Com s'ha descrit a la secció 3.3, utilitzem la imputació KNN amb k=5 per gestionar els valors perduts en les variables mèdiques. Com que encara no hem escalat les dades, fem un escalat temporal per a la imputació amb `StandardScaler`. Després de la imputació, desfem l'escalat temporal amb `inverse_transform`.

4. Transformem les variables asimètriques: Apliquem una transformació logarítmica a les variables `Duration_untreated_psychosis` i `Age` per reduir la seva asimetria i millorar la normalitat de la distribució. Ho fem amb la classe PowerTransformer de `sklearn.preprocessing`, utilitzant el mètode 'yeo-johnson', que és robust amb valors negatius i zero.

5. Escalat de les dades: Finalment, escalem totes les variables numèriques utilitzant l'estandardització `StandardScaler` per assegurar que tinguin una mitjana de 0 i una desviació estàndard de 1. Això és especialment important per als models SVM, ja que són sensibles a l'escala de les característiques.

4.1.2 Ajustament del model

Per tal de trobar els millors hiperparàmetres per al model SVM, he utilitzat una cerca en quadrícula (`GridSearchCV`) amb validació creuada de 5 plecs. Els hiperparàmetres que he considerat són:

- `C`: El paràmetre de regularització, que controla la compensació entre maximitzar el marge i minimitzar l'error de classificació. Com que no he eliminat outliers, he provat valors baixos de C per evitar sobreajustaments a aquests valors extrems i obtenir una milor generalització. No té cap sentit provar valors alts de C en aquest cas.
- `kernel`: El tipus de nucli a utilitzar. He provat els nuclis RBF i polinòmic per veure quin s'adapta millor a les dades. No he provat el nucli lineal, ja que les dades no semblen linealment separables.
- `gamma`: El paràmetre del nucli RBF, que controla l'abast de la influència d'un sol exemple d'entrenament. He provat valors baixos per evitar sobreajustaments i també valors predefinits com 'scale' i 'auto'.
- `degree`: El grau del nucli polinòmic. He provat graus baixos per evitar models massa complexos.
- `class_weight`: El pes de les classes per gestionar el desbalanceig de la variable objectiu. He provat d'utilitzar diferents pesos perquè la classe minoritària tingui més influència en l'ajustament del model, ja que només són un 31.5% del total d'observacions.

Resumit en una taula, els valors provats han sigut:

Hiperparàmetre	Valors provats	Significat
`C`	[0.001, 0.01, 0.1, 1]	Paràmetre de regularització: controla el compromís entre marge gran i errors de classificació al train.
`gamma`	['scale', 'auto', 0.01]	Coeficient del nucli (RBF/poly): determina l'abast d'influència de cada mostra sobre la frontera de decisió.
`kernel`	['rbf', 'linear', 'poly']	Tipus de nucli utilitzat: lineal, radial (RBF) o polinòmic.
`class_weight`	['balanced', {0: 1, 1: 2}, `class_weights`]	Ponderació de cada classe per tractar el desbalanceig; 'balanced' ajusta pesos segons la freqüència.
`degree`	[2, 3, 4]	Grau del polínom quan `kernel='poly'`; controla la complexitat de la frontera polinòmica.

Taula 4: Espai de cerca d'hiperparàmetres per al model SVM.

Per tant, executem la graella de cerca amb aquests paràmetres i seleccionem el model amb la millor puntuació de validació creuada:

```
grid_search = GridSearchCV(
    estimator=svm_base,
    param_grid=param_grid,
    cv=5,
    scoring='f1_macro',
    verbose=2,
    n_jobs=-1
)
```

Amb aquest codi indiquem que es faci una cerca en quadrícula amb validació creuada de 5 plecs, utilitzant la mètrica F1 macro per avaluar el rendiment dels models. El paràmetre `verbose=2` permet veure el progrés de la cerca, i `n_jobs=-1` utilitza tots els nuclis disponibles del processador per accelerar el càlcul. El resultat de la cerca ens proporciona els millors hiperparàmetres per al model SVM:

```
{'C': 0.1, 'class_weight': 'balanced', 'degree': 3, 'gamma': 'auto', 'kernel': 'poly'}
```

Aquest model utilitza un **kernel polinòmic de grau 3** per capturar relacions no lineals complexes entre les dades. La configuració `C=0.1` aplica una **regularització forta**, prioritzant una frontera de decisió simple amb un marge ample per evitar l'overfitting. En concret, la frontera de decisió es defineix mitjançant el kernel polinòmic. La funció de kernel específica per a aquesta configuració és:

$$K(x, y) = (\gamma \cdot \langle x, y \rangle + r)^d$$

On:

- γ: Es calcula com $\frac{1}{n_{features}}$ quan s'utilitza `gamma='auto'`, on $n_{features}$ és el nombre de característiques del conjunt de dades. Aquest paràmetre controla l'abast de la influència de cada punt de dades, és a dir, com de lluny pot arribar la influència d'un punt d'entrenament en la frontera de decisió.
- d: Correspon al grau del polinomi
- r: És el terme independent `coef0`, que per defecte és 0.0.

Finalment, el paràmetre `class_weight='balanced'` compensa automàticament el desequilibri entre classes, mentre que `gamma='auto'` escala la influència de cada punt segons l'invers del nombre de característiques del conjunt de dades.[SKL25]

Les mètriques d'avaluació del model SVM ajustat al conjunt de prova són les següents:

Classe	Prec.	Rec.	F1	Supp.
`0`	0.75	0.64	0.69	1232
`1`	0.40	0.53	0.46	568
`accuracy`		0.60		1800
`macro`	0.57	0.58	0.57	1800
`weighted`	0.64	0.60	0.62	1800

Taula 5: Resultats del model

Per a la classe negativa (no TRS), la **Precision** del 75% i el **Recall** del 64% indiquen una bona capacitat d'identificació base. En canvi, per a la classe positiva (TRS), la Precision cau al 40%, reflectint dificultats pel desequilibri de dades.

Amb una **accuracy** del 60% i un **weighted F1-score** del 62%, el model mostra un rendiment moderat, sent més eficaç en la predicció de la classe majoritària que en la detecció de pacients amb TRS.

Observem la matriu de confusió i la corba ROC:

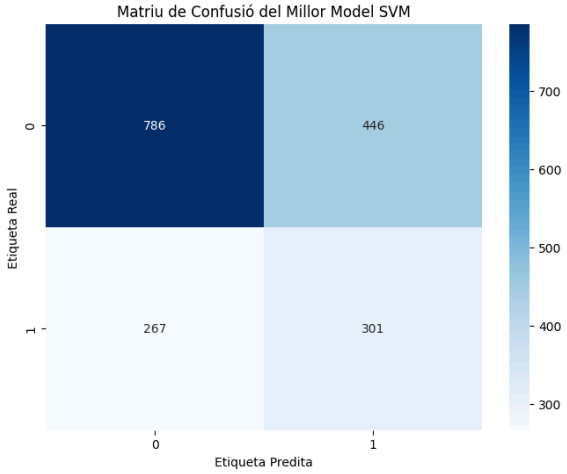

Figura 11: Matriu de confusió del model SVM

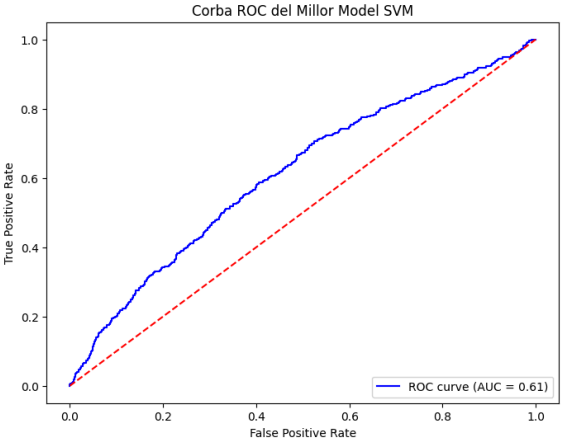

Figura 12: Corba ROC del model SVM

Per mirar si el model pateix d'algun sobreajustament, comparem l'accuracy al conjunt d'entrenament i al conjunt de prova. Trobem els valors executant la comanda `train_acc = best_svm.score(X_train_svm_final, y_train_svm)` i `test_acc = best_svm.score(X_test_svm_final, y_test_svm)`, obtenint els següents resultats que l'accuracy al train és de 0.6786 i l'accuracy al test és de 0.6039. Tenim una diferència de gairebé el 7.5% entre ambdós conjunts.

Això indica que hi ha una diferència notable entre l'accuracy al conjunt d'entrenament i al conjunt de prova, suggerint un cert grau de sobreajustament. El model sembla adaptar-se massa bé als exemples d'entrenament, però no generalitza tan bé als nous exemples del conjunt de prova. Per mitigar aquest sobreajustament, intentem provar valors més baixos de `C` en la cerca en quadrícula, mantenint la resta de hiperparàmetres iguals. Provarem valors entre 0.01 i 0.1 perquè és el rang que la cerca en grid ha hagut de decidir. El gràfic resultant ha estat la figura 13

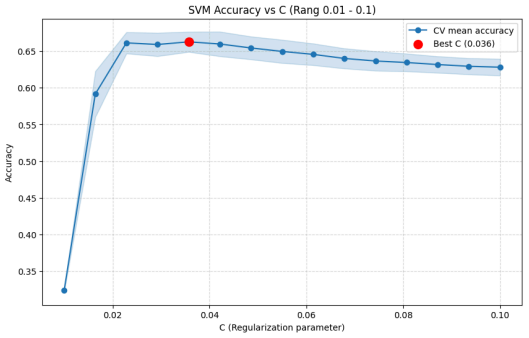

Figura 13: Gràfic d'accuracy en funció de `C`

Ens retorna que el millor valor de C és `0.036`, per tant ajustem el model final amb aquest valor i obtenim les següents mètriques:

Classe	Prec.	Rec.	F1	Supp.
`0`	0.73	0.82	0.77	1232
`1`	0.46	0.35	0.40	568
`accuracy`		0.67		1800
`macro avg`	0.60	0.58	0.58	1800
`weighted`	0.65	0.67	0.65	1800

Taula 6: Mètriques

Les mètriques mostren una millora en l'accuracy global del model, que ha augmentat al 67%. La classe negativa (no TRS) manté un bon rendiment amb un F1-score del 77%, mentre que la classe positiva (TRS) mostra una lleugera millora amb un F1-score del 40%. La macro avg F1-score també ha augmentat al 58%, indicant una millor equilibració entre les classes. No obstant això, el model encara té dificultats per predir correctament la classe minoritària (TRS), com es reflecteix en el baix Recall del 35%. Això suggereix que, tot i la millora, el model encara té marge de millora en la identificació de pacients que desenvolupen TRS.

Observem la matriu de confusió i la corba ROC actualitzades:

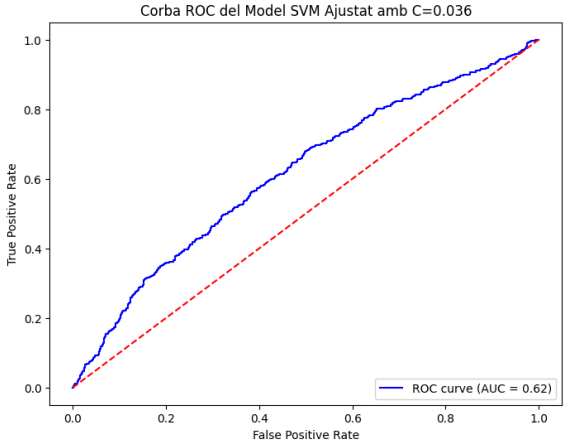

Figura 14: Corba ROC

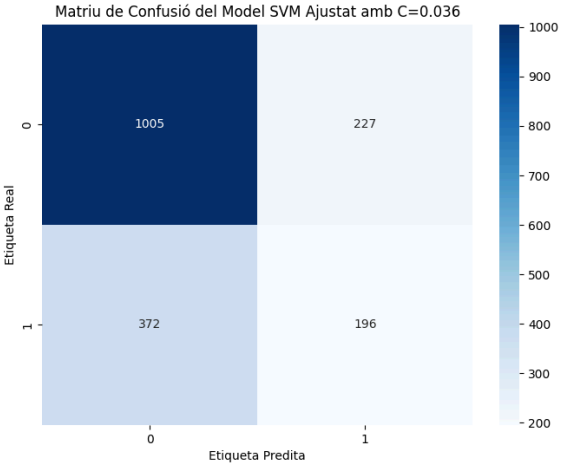

Figura 15: Matriu de confusió

Comparant el model inicial amb el nou model ajustat `C=0.036`, s'observa una millora en la robustesa del classificador malgrat que el rendiment general segueix sent moderat. L'**AUC:** ha passat de 0.61 a **0.62**. Tot i ser un increment lleuger, indica una millor capacitat del model per separar les classes i una corba ROC més allunyada de la línia aleatòria en comparació amb el primer model.

El model ajustat també ha reduït dràsticament els errors de la classe 0 (falsos positius), passant de 446 a **227**. Això demostra que la reducció del paràmetre `C` ha permès crear una frontera de decisió més simple i menys propensa al soroll de les dades. Tot i així, aquesta simplificació ha penalitzat el *recall* de la classe minoritària, identificant menys casos positius (196 vs 301). És a dir, el model és ara més conservador en la predicció de la classe 1 (TRS), prioritzant la qualitat de les prediccions positives per sobre de la quantitat.

En conclusió, el model ajustat és més conservador i té un millor rendiment en la classe majoritària, aconseguint una frontera de decisió més generalitzable i menys "sobreajustada" al soroll inicial. No obstant això, encara hi ha marge de millora en la detecció de la classe minoritària, que és crucial en aquest context mèdic.

4.2 XGBoost

XGBoost és un model d'ensamblatge basat en arbres de decisió que utilitza l'algorisme de gradient boosting per millorar la precisió de les prediccions. Aquest model serà ajustat amb la biblioteca `xgboost`, provant diferents hiperparàmetres mitjançant una cerca en quadrícula (`GridSearchCV`).

4.2.1 Preprocessament

El XGBoost és menys sensible a l'escala de les dades i als valors extrems, per la qual cosa el preprocessament serà més senzill que en el cas de l'SVM. Els passos seguits són:

1. Particionem les dades: Tal i com s'ha descrit a la secció 3.6, dividim el conjunt de dades en un 80% per a l'entrenament i un 20% per a la prova, assegurant-nos que ambdues particions mantinguin la mateixa proporció de la variable objectiu `TRS`:

```
X_train_xgb, X_test_xgb, y_train_xgb, y_test_xgb = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

2. Codificació de variables categòriques: Utilitzem l'encodificació One-Hot per a les variables categòriques, que són `Ethnicity` i `CYP2D6_metabolic_phenotype`. Això crea variables binàries per a cada categoria, permetent que el model XGBoost les utilitzi correctament.

3. Imputació de valors perduts: Com s'ha descrit a la secció 3.3, utilitzem la imputació KNN amb k=5 per gestionar els valors perduts en les variables mèdiques. Serà necessari escalar temporalment les dades amb `StandardScaler` per a la imputació, i després desfem l'escalat amb `inverse_transform`.

4.2.2 Ajustament del model

Primer de tot definim un model base d'XGBoost amb els paràmetres per defecte:

```
xgb_base = xgb.XGBClassifier(
    objective='binary:logistic',
    random_state=42,
    n_jobs=-1,
)
```

Per tal de trobar els millors hiperparàmetres per al model XGBoost, he utilitzat una cerca en quadrícula (`GridSearchCV`) amb validació creuada de 5 plecs [XGB25]. Els hiperparàmetres que he considerat són:

- `n_estimators`: El nombre d'arbres a construir. He provat valors entre 50 i 300 per trobar un equilibri entre rendiment i temps de càlcul.
- `max_depth`: La profunditat màxima dels arbres. He provat valors entre 3 i 7 per controlar la complexitat del model.
- `learning_rate`: La taxa d'aprenentatge, que controla la contribució de cada arbre al model final. He provat valors baixos per evitar sobreajustaments, de 0.01 a 0.2.
- `subsample`: La fracció de mostres utilitzades per a entrenar cada arbre. He provat valors entre 0.6 i 1.0 per introduir diversitat en els arbres.
- `colsample_bytree`: La fracció de característiques utilitzades per a entrenar cada arbre. He provat valors entre 0.6 i 1.0 per introduir diversitat en les característiques.
- `scale_pos_weight`: El pes de la classe positiva per gestionar el desbalanceig de la variable objectiu. He provat diferents pesos perquè la classe minoritària tingui més influència en l'ajustament del model. Els valors provats han sigut 1, la proporció entre les classes i aquesta per 2
- `gamma`: El paràmetre de regularització que controla la complexitat dels arbres. He provat els valors de 0, 1 i 5 per veure quin s'adapta millor a les dades.

Resumit en una taula, els valors provats han sigut:

Aquí tens la taula d'hiperparàmetres per al model XGBoost, seguint exactament l'estil i l'estructura que has utilitzat anteriorment per al model SVM:

Hiperparàmetre	Valors provats	Significat
`max_depth`	[3,5,7]	Profunditat màxima de l'arbre: controla la complexitat del model i el risc de sobreajustament.
`learning_rate`	[0.01, 0.1, 0.2]	Taxa d'aprenentatge: escala la contribució de cada nou arbre per suavitzar el procés d'optimització.
`n_estimators`	[50,100,200,300]	Nombre d'arbres: quantitat total d'iteracions de boosting a realitzar.
`scale_pos_weight`	[1, ratio, ratio*1.5, ratio*2]	Control del desbalanceig: pondera la classe positiva per millorar la sensibilitat en classes minoritàries.
`subsample`	[0.6, 0.8, 1.0]	Mostreig de files: proporció de dades d'entrenament utilitzades per a cada arbre per afegir robustesa.
`colsample_bytree`	[0.6, 0.8, 1.0]	Mostreig de columnes: proporció de característiques seleccionades a l'atzar per a cada arbre.
`gamma`	[0,1,5]	Regularització conservadora: pèrdua mínima requerida per realitzar una partició addicional en un node.
`eval_metric`	['logloss', 'auc']	Mètrica d'avaluació: criteri utilitzat per monitorar el rendiment durant la validació.

Taula 7: Espai de cerca d'hiperparàmetres per al model XGBoost.

El resultat de la cerca ens proporciona els millors hiperparàmetres per al model XGBoost:

```
{'colsample_bytree': 1.0, 'eval_metric': 'logloss', 'gamma': 5, 'learning_rate': 0.01,
'max_depth': 7, 'n_estimators': 300, 'scale_pos_weight': np.float64(2.171806167400881),
'subsample': 0.6}
```

Aquest model utilitza una profunditat màxima de 7 per als arbres, amb una taxa d'aprenentatge baixa de 0.01 per evitar sobreajustaments. El nombre d'arbres és de 300, i s'utilitza un mostreig de files del 60% i un mostreig de columnes del 100% per introduir diversitat en els arbres. El paràmetre `scale_pos_weight` s'ha ajustat a la proporció entre les classes, sense multiplicar, per donar més pes a la classe minoritària. Finalment, el paràmetre `gamma` s'ha establert en 5 per aplicar una regularització més forta i evitar particions innecessàries en els arbres. Aquest model, de base, em fa sospitar que hi ha molt de sobreajustament, ja que té molts arbres i una profunditat alta, però la taxa d'aprenentatge baixa podria ajudar a mitigar-ho, per tant caldrà analitzar-ho més endavant.

Observem les mètriques d'avaluació del model XGBoost ajustat al conjunt de prova:

Classe	Prec.	Rec.	F1	Supp.
0	0.74	0.63	0.68	1232
1	0.40	0.52	0.45	568
accuracy		0.60		1800
macro	0.57	0.58	0.57	1800
weighted	0.63	0.60	0.61	1800

Taula 5: Resultats del model

Podem veure que el model XGBoost té un rendiment similar al model SVM, amb una **accuracy** del 60%, és a dir, el 60% de les prediccions són correctes. Veiem que assigna correctament el 63% de la classe 0 (no TRS) i el 52% de la classe 1 (TRS).

També veiem que de les que assigna a la classe 1, només el 40% són correctes. Això indica que el model té dificultats per identificar correctament els pacients amb TRS, ja que hi ha molts falsos positius. Pel que fa a les que assigna a la classe 0, el 74% són correctes, però hi ha un 37% de falsos negatius, és a dir, pacients amb TRS que són classificats com a no TRS. Més enllà, l'**F1-score ponderat** és del 61%, indicant un rendiment moderat en general.

Comprovem la matriu de confusió i la corba ROC:

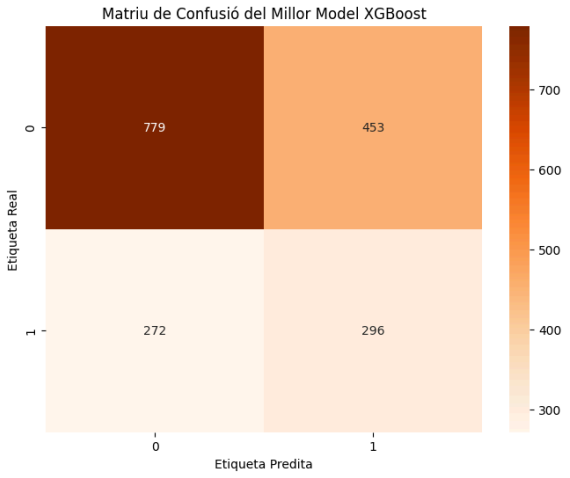

Figura 16: Matriu de confusió del model XGBoost

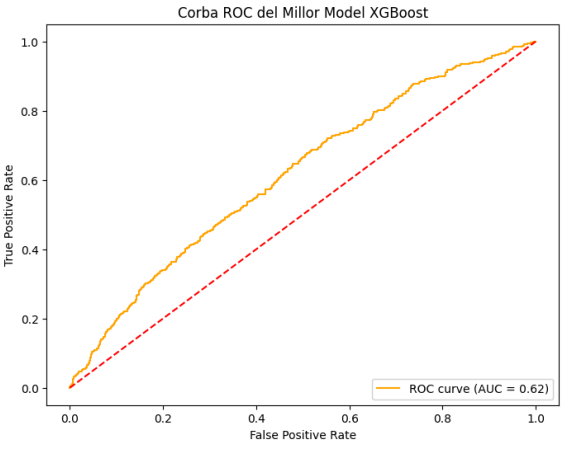

Figura 17: Corba ROC del model XGBoost

Veiem que els resultats són molt similars als del model SVM, amb una AUC de 0.62, indicant una capacitat moderada per separar les classes. La matriu de confusió mostra que hi ha un nombre significatiu de falsos positius i falsos negatius, suggerint que el model té dificultats per identificar correctament els pacients amb TRS. Aquest comportament és similar al del model SVM, indicant que ambdós models tenen limitacions en la seva capacitat per predir aquesta condició mèdica específica. Comprovem també si hi ha sobreajustament comparant l'accuracy al conjunt d'entrenament i al conjunt de prova. Obtenim que el train té un accuracy de 0.8149 i el test de 0.5972, per tant hi ha una diferència molt gran entre ambdós conjunts, indicant un sobreajustament molt elevat. Per tant, caldrà ajustar els hiperparàmetres per tal de reduir aquest sobreajustament. Visualitzem també les corbes d'aprenentatge per veure com evoluciona l'error en funció del nombre d'arbres, ens pot donar un indicador de si cal augmentar o reduir el nombre d'arbres i quants:

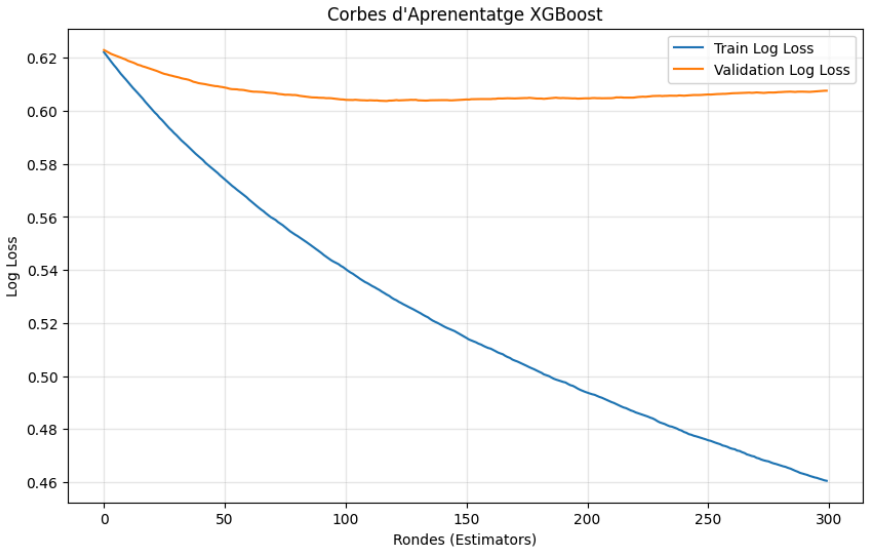

Figura 18: Corbes d'aprenentatge del model XGBoost

Veiem que a partir de la ronda 100, l'error de validació comença a creixer, indicant que el model comença a sobreajustar-se. Per tant, podem reduir el nombre d'arbres a 100 per tal de reduir el sobreajustament. Mirem de refinar ara la profunditat màxima dels arbres, provant valors de l'1 al 10, per veure quin és el millor valor per tal de reduir el sobreajustament. Mantenim la resta d'hiperparàmetres de moment. El gràfic resultant és la figura 19

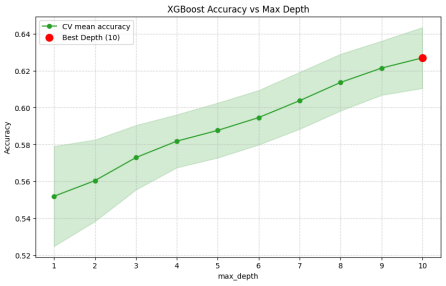

Figura 19: Gràfic de profunditat màxima vs accuracy

Veiem que l'**accuracy** augmenta a mida que la profunditat augmenta; per tant, no sembla que reduir la profunditat ajudi a reduir el sobreajustament. Tot i així, per simplicitat, establim la profunditat a **7**, que és un valor intermedi.

Provem doncs a mirar de reduir el valor gamma, ja que aquest paràmetre controla la complexitat dels arbres i podria ajudar a reduir el sobreajustament. Provem diferents valors del 0 al 10, per veure si això ajuda a reduir el sobreajustament:

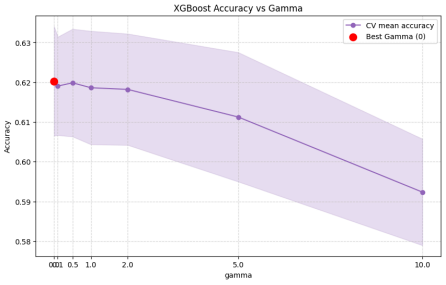

Figura 20: Gràfic de gamma vs accuracy

Veiem ara que l'**accuracy** disminueix a mida que `gamma` augmenta; per tant, sembla que reduir el valor de `gamma` era el culpable del sobreajustament. Establim doncs `gamma` a **0**.

Ens queda un model final amb els següents hiperparàmetres ajustats:

```
{'colsample_bytree': 1.0, 'eval_metric': 'logloss', 'gamma': 0, 'learning_rate': 0.01,
'max_depth': 7, 'n_estimators': 100, 'scale_pos_weight': np.float64(2.171806167400881),
'subsample': 0.8}
```

També augmentem el `subsample` a 0.8 per tal de reduir el sobreajustament, ja que amb un valor més alt, cada arbre veu més dades i per tant és menys propens a sobreajustar-se.

4.3 Regressió logística personalitzada

4.3.1 Preprocessament

4.3.2 Ajustament del model

4.3.3 Resultat final

5. Model final

6. Model Card

7. Conclusions

8. Referències

[SKL25]: Scikit-learn developers. (n.d.). SVC — scikit-learn 1.7.2 documentation. Recuperat el 21 de desembre de 2025, de https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[XGB25]: XGBoost developers. (s.f.). XGBoost Parameters. Recuperat el 21 de desembre de 2025, de https://xgboost.readthedocs.io/en/stable/parameter.html