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:

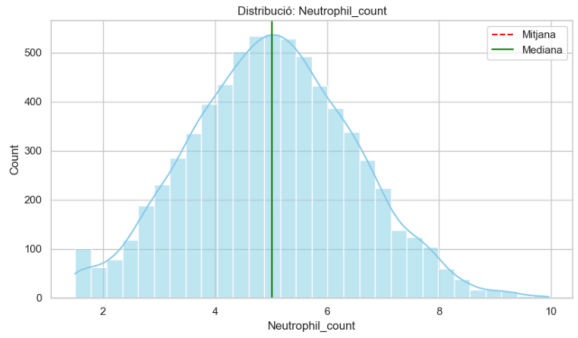

Figura 1: Distribució de `Neutrophil count`.

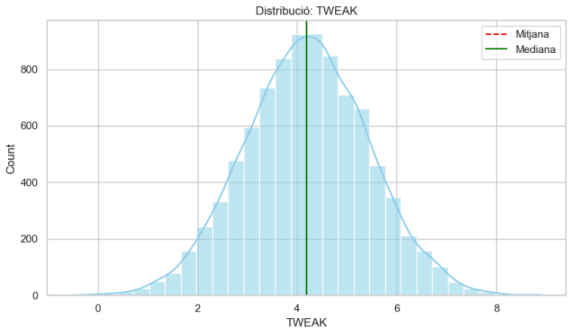

Figura 2: Distribució de `TWEAK`.

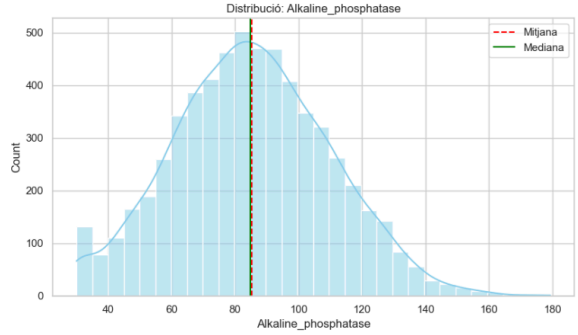

Figura 3: Distribució de `Alkaline phosphatase`

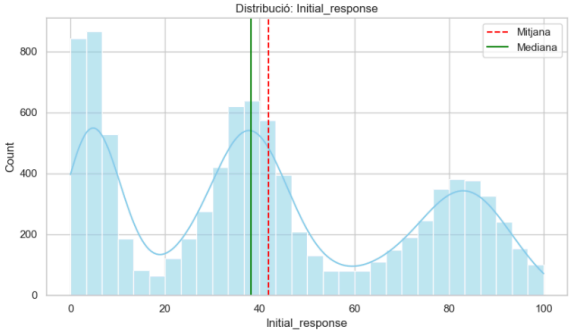

Figura 4: Distribució de `Initial response`.

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:

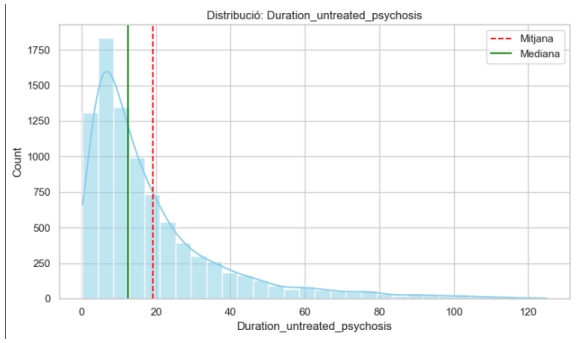

Figura 5: Distribució de `Duration_Untreated_Psychosis`.

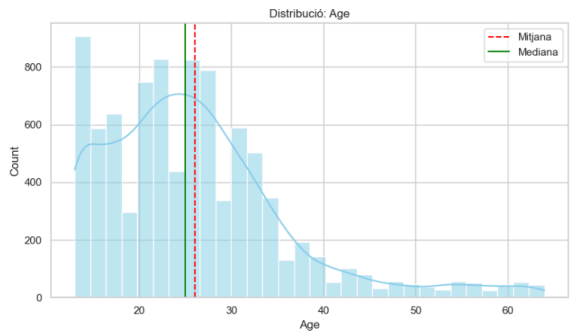

Figura 6: Distribució de `Age`.

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:

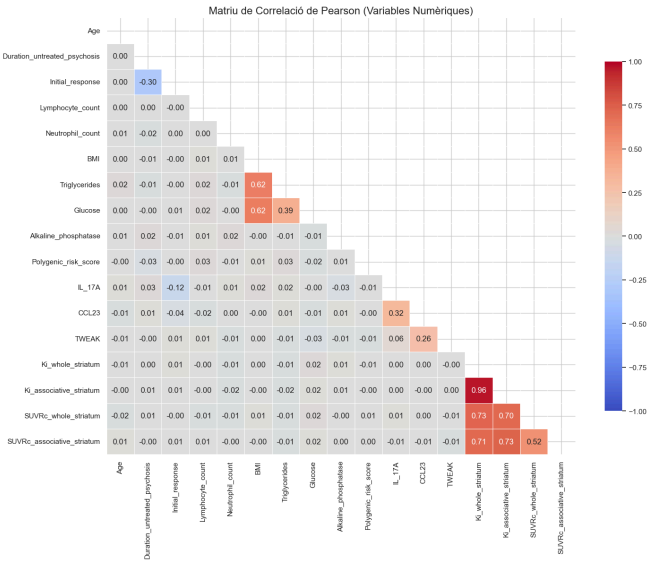

Figura 7. Matriu de Correlació de Pearson

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:

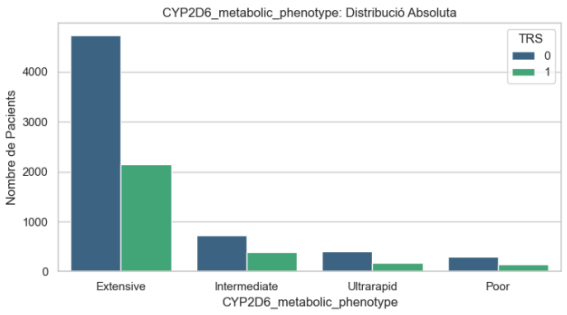

Figura 8: Distribució de la variable objectiu TRS

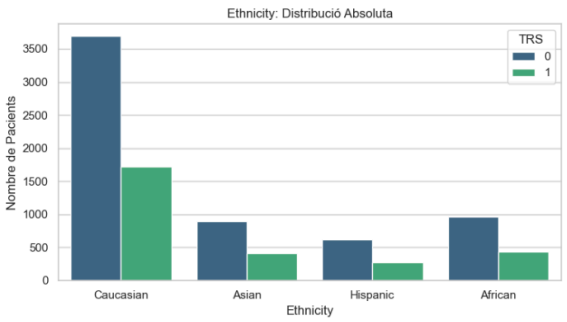

Figura 9: Distribució de la variable objectiu TRS

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:

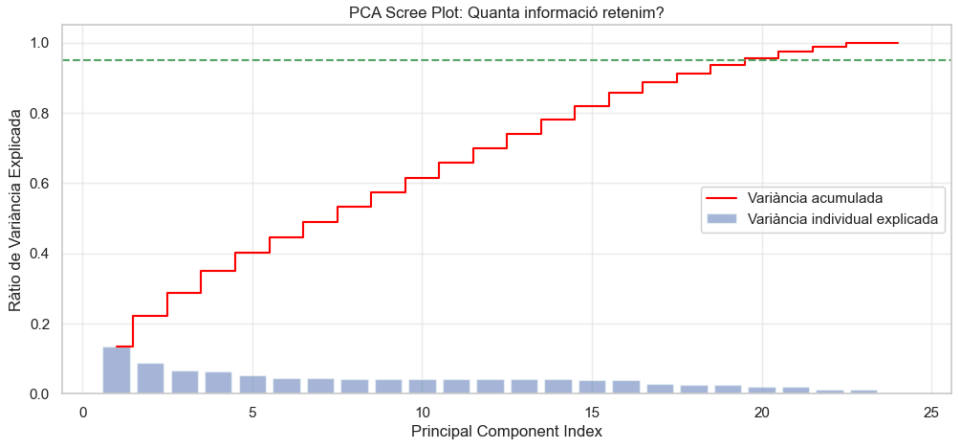

Figura 10: Gràfica de Scree Plot de la PCA

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.[SKL25svc]

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:

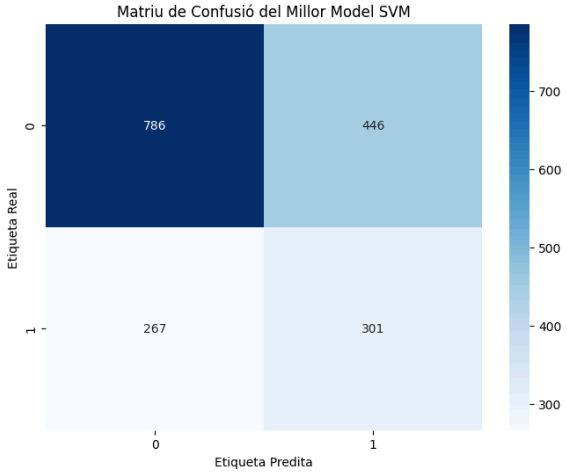

Figura 11: Matriu de confusió del model SVM

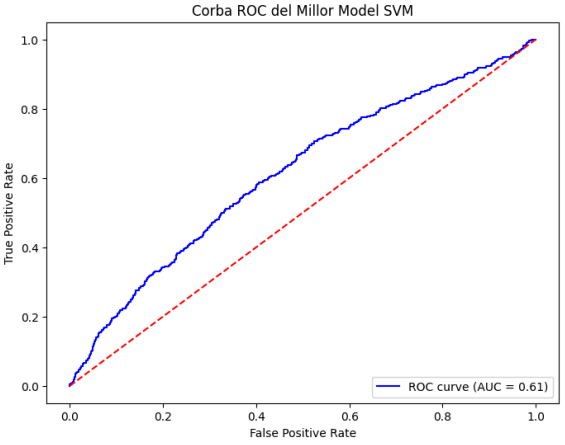

Figura 12: Corba ROC del model SVM

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. En aquest cas en concret, intentarem buscar la C que millori el recall, que en un context mèdic és més important per minimitzar els falsos negatius (pacients amb TRS no detectats). Per tant, provarem 15 valors de C entre 0.01 i 0.1 per trobar el millor compromís entre recall i precisió.

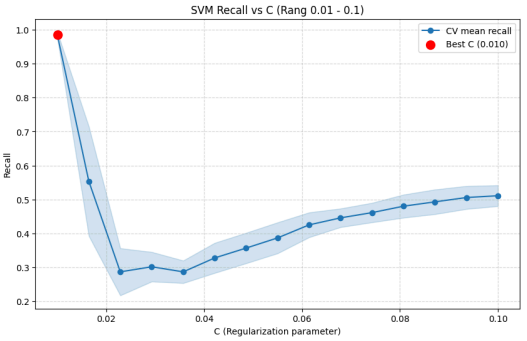

Figura 13: Gràfic del recall en funció de `C`

Ens retorna que el millor valor de C és `0.01`, que dona un recall de 1.0. Això és impossible, ja que vol dir que tots els pacients amb TRS han estat classificats correctament, per tant hi ha algun error en el càlcul. El segon millor valor és el de `0.1`, que ja l'havíem provat abans i ens donava un recall de 0.53. Per tant, provem amb un valor intermedi entre `0.01` i `0.1`, concretament `0.06`, per veure si podem millorar el recall sense sacrificar massa la precisió.

Classe	Prec.	Rec.	F1	Supp.
`0`	0.74	0.69	0.71	1232
`1`	0.42	0.49	0.45	568
`accuracy`		0.62		1800
`macro avg`	0.58	0.59	0.58	1800
`weighted`	0.64	0.62	0.63	1800

Taula X: Resultats del model SVM amb `C=0.06`

Veiem que amb aquest nou valor de `C=0.06`, el model aconsegueix un millor equilibri entre precisió i recall, amb una **accuracy** del 62% i un F1-score ponderat del 63%. Això indica que el model és capaç de detectar més casos de TRS (recall del 49%) sense sacrificar massa la precisió (42%).

Aquesta configuració sembla oferir un millor compromís per a l'objectiu mèdic de minimitzar els falsos negatius, tot mantenint una bona qualitat en les prediccions positives. També consultem els valors d'accuracy al train i al test per aquest nou model amb `C=0.06`, obtenint que l'accuracy al train és de 0.6804 i l'accuracy al test és de 0.6222. La diferència entre ambdós conjunts és ara d'aproximadament el 5.8%, indicant una millora en la generalització del model, tot i que encara hi ha marge per a una millor optimització.

Observem la matriu de confusió i la corba ROC actualitzades:

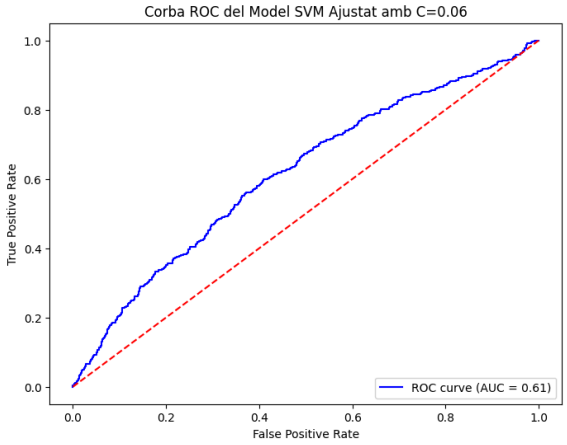

Figura 14: Corba ROC

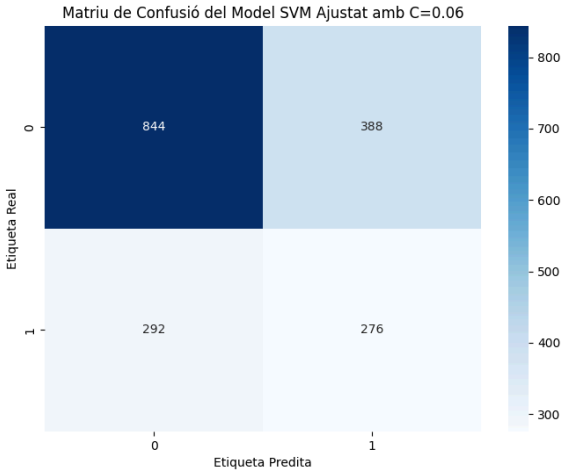

Figura 15: Matriu de confusió

Comparant el model inicial amb el nou model ajustat amb **`C=0.06`**, s'observa una millora en la robustesa del classificador, tot i que el rendiment global continua sent moderat. L'**AUC** es manté en **0.61**, cosa que indica una capacitat de discriminació estable, amb una corba ROC clarament per sobre de la diagonal aleatòria però encara lluny d'un classificador ideal.

Pel que fa a la matriu de confusió, el model ajustat amb `C=0.06` obté **844 Veritables Negatius** i **388 Falsos Positius** per a la classe 0, la qual cosa reflecteix una bona capacitat per identificar pacients no TRS sense caure en un excés d'errors positius. Per a la classe 1 (TRS), el model detecta **276 Veritables Positius** i deixa **292 Falsos Negatius**, mantenint un compromís raonable entre la detecció de casos positius i el control d'alertes falses. En conjunt, la reducció de C fins a 0.06 ha permès obtenir una frontera de decisió més generalitzable i menys sensible al soroll, fent el model més conservador però també més fiable, encara que segueixi existint marge de millora en la sensibilitat a la classe TRS.

En conclusió, el model SVM ajustat amb `C=0.06` ofereix un millor equilibri entre precisió i recall, essent més adequat per a l'objectiu mèdic de minimitzar els falsos negatius. Tot i això, el rendiment global del model continua sent moderat, indicant la necessitat d'explorar altres models o tècniques per millorar la capacitat predictiva en aquest context.

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 han estat considerats tenint en compte les característiques del conjunt de dades, és a dir, tendència al sobreajustament i desbalanceig de classes. Els hiperparàmetres que he considerat són:

Resumit en una taula, els valors provats han sigut:

Hiperparàmetre	Valors provats	Significat
`max_depth`	[3, 4, 5]	Profunditat màxima dels arbres: controla la complexitat del model per evitar sobreajustaments.
`learning_rate`	[0.01, 0.05, 0.1]	Taxa d'aprenentatge: controla la contribució de cada arbre, amb valors baixos per evitar sobreajustaments.
`n_estimators`	[100, 200, 300]	Nombre d'arbres: equilibri entre rendiment i temps de càlcul.
`scale_pos_weight`	[ratio_xgb]	Pes de la classe positiva: gestiona el desbalanceig de la variable TRS per donar més influència a la classe minoritària.
`subsample`	[0.8, 0.9]	Fració de mostres per arbre: introdueix diversitat i robustesa.
`colsample_bytree`	[0.8, 0.9]	Fració de característiques per arbre: introdueix diversitat en les característiques.
`min_child_weight`	[3, 5, 7]	Pes mínim requerit en un node fill: controla la creació de fulles per regularitzar el model.
`reg_alpha`	[0.1, 0.5, 1]	Regularització L1: penalitza les característiques no rellevants per evitar sobreajustament.
`reg_lambda`	[0.1, 0.5, 1]	Regularització L2: penalitza els pesos grans per suavitzar el model.
`gamma`	[0.1, 0.2]	Pèrdua mínima per partició: regularitza la complexitat dels arbres.

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

No s'han considerat arbres més profunds ni taxes d'aprenentatge més altes per evitar sobreajustaments, donada la mida i complexitat del conjunt de dades. El paràmetre `scale_pos_weight` s'ha establert amb el valor del ratio entre les classes per donar més pes a la classe minoritària (TRS). Les regularitzacions L1 i L2 s'han inclòs per penalitzar característiques no rellevants i pesos grans, respectivament, ajudant a prevenir l'overfitting.

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

```
{'colsample_bytree': 0.9, 'gamma': 0.1, 'learning_rate': 0.01, 'max_depth': 5,
'min_child_weight': 3, 'n_estimators': 300, 'reg_alpha': 1, 'reg_lambda': 1,
'scale_pos_weight': np.float64(2.171806167400881), 'subsample': 0.8}
```

Aquest model utilitza **300 arbres** amb una **profunditat màxima de 5** per capturar relacions complexes entre les dades. La **taxa d'aprenentatge de 0.01** assegura que cada arbre contribueixi de manera gradual al model final, ajudant a prevenir l'overfitting. Els paràmetres `subsample=0.8` i `colsample_bytree=0.9` introdueixen diversitat en les mostres i característiques utilitzades per a cada arbre, millorant la robustesa del model. A més, la configuració `scale_pos_weight=2.17` compensa el desequilibri entre classes, donant més pes a la classe minoritària (TRS). Les regularitzacions L1 i L2 amb valors de 1 penalitzen característiques no rellevants i pesos grans, respectivament, ajudant a prevenir l'overfitting. [XGB25]

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

Classe	Prec.	Rec.	F1	Supp.
0	0.75	0.58	0.65	1232
1	0.39	0.59	0.47	568
accuracy		0.58		1800
macro avg	0.57	0.58	0.56	1800
weighted	0.64	0.58	0.60	1800

Taula 8: Resultats de classificació per al model XGBoost.

El model XGBoost presenta un rendiment moderat amb una **accuracy** del 58% i un F1-score ponderat del 60%. Tot i que la precisió per a la classe positiva (TRS) se situa en el 39%, el model assoleix un **recall** del 59%, xifra que indica una capacitat notable per identificar pacients amb resistència al tractament, malgrat l'elevat nombre de falsos positius generats.

Pel que fa a la classe no TRS (0), el model demostra una robustesa superior en precisió (75%), encertant la majoria de casos negatius, encara que el seu recall (58%) suggereix que una part d'aquests pacients es classifiquen erròniament com a positius. Aquests resultats reflecteixen un compromís similar al del model SVM, prioritzant la detecció de la classe minoritària per sobre de la precisió global, un factor clau en el context clínic de la malaltia.

Observem la matriu de confusió i la corba ROC:

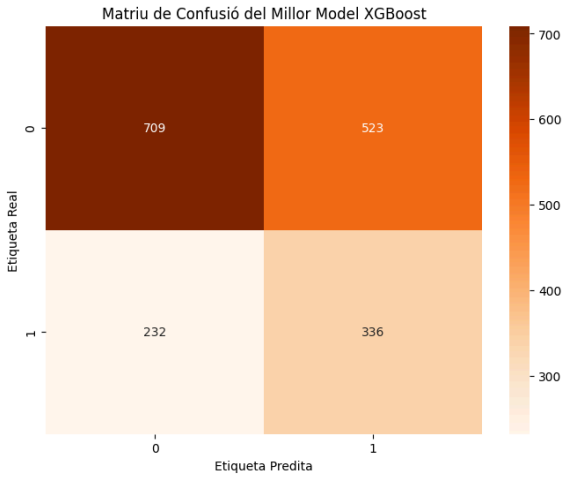

Figura 16: Matriu de confusió del model XGBoost

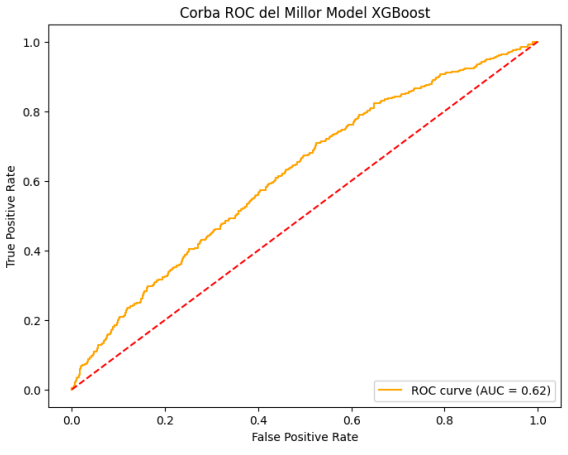

Figura 17: Corba ROC del model XGBoost

El model prioritza la sensibilitat, identificant correctament **336 casos positius** (True Positives). Tot i així, genera un nombre elevat de falsos positius (523), el que confirma que el model tendeix a ser "agressiu" en la classificació per no perdre pacients de risc. L'**AUC de 0.62** indica que el model té una capacitat de discriminació superior a l'atzar. La corba puja de forma constant per sobre de la línia de referència, demostrant que el model és capaç de separar les classes tot i la complexitat i el desbalanceig de les dades. En resum, el model és millor detectant els casos de TRS (recall) que no pas sent precís en les seves prediccions positives, un comportament esperat en un context mèdic on es volen minimitzar els falsos negatius.

Revisem si hi ha overajustament observant les corbes d'aprenentatge:

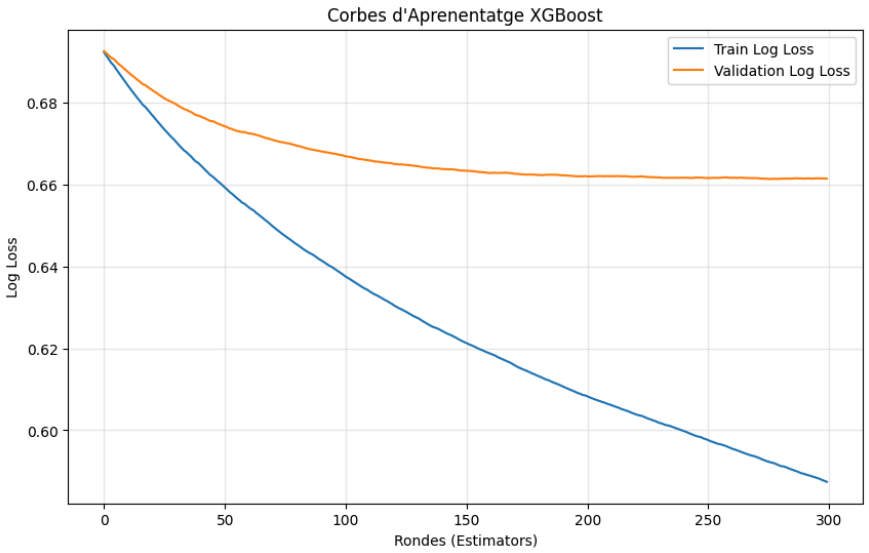

Figura 18: Corbes d'aprenentatge del model XGBoost

Podem veure clarament com l'error de validació redueix i arriba a estabilitzar-se al final de l'entrenament, és a dir, quan arriba a entrenar-se l'arbre número 300. Això indica que el model no pateix d'overajustament, ja que l'error de validació no augmenta després d'un cert punt. Per tant, podem concloure que el model XGBoost està ben ajustat als dades d'entrenament i generalitza bé al conjunt de prova, tot i la complexitat del conjunt de dades i el desbalanceig de classes.

4.3 Regressió logística personalitzada

En aquesta secció, desenvolupem un model de regressió logística des de zero, implementant l'algorisme d'optimització de descens de gradient per ajustar els pesos del model. Aquest model tindrà com a objectiu classificar els pacients en funció de la variable `TRS`, utilitzant les mateixes característiques que en els models anteriors. El model serà de mini-batch gradient descent, és a dir, en cada iteració s'utilitzarà un subconjunt aleatori de dades per calcular el gradient i actualitzar els pesos.

4.3.1 Preprocessament de les dades

Abans d'entrenar el model de regressió logística, és necessari realitzar un preprocessament adequat de les dades per assegurar que el model pugui aprendre correctament. Els passos seguits són:

1. Partició de 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_rl, X_test_rl, y_train_rl, y_test_rl = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

2. Com que el model és sensible a variables correlacionades, eliminem les característiques que presenten una alta correlació entre elles. Això es fa per evitar problemes de multicol·linealitat que podrien afectar la convergència del model. En concret, considerarem una correlació alta quan el coeficient de correlació de Pearson sigui superior a 0.5 en valor absolut, que són:

    - Triglycerides
    - Glucose
    - Ki_associative_striatum
    - SUVRc_whole_striatum
    - SUVRc_associative_striatum

3. 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 de regressió logística les utilitzi correctament.

4. 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`.

5. Transformació de les variables que no segueixen una distribució normal: Per millorar la convergència del model, apliquem la transformació Yeo-Johnson a les variables numèriques que no segueixen una distribució normal, que són: `Age` i `Duration_Untreated_Psychosis`. Aquesta transformació ajuda a estabilitzar la variància i a fer que les dades siguin més semblants a una distribució normal.

6. Escalat de les dades: Finalment, escalem totes les característiques utilitzant l'estandardització (StandardScaler), que transforma les dades perquè tinguin una mitjana de 0 i una desviació estàndard d'1. Això és important per al model de regressió logística, ja que ajuda a millorar la convergència durant l'entrenament.

D'aquesta manera, les dades estan preparades per ser utilitzades en l'entrenament del model de regressió logística personalitzada.

4.3.2 Implementació del model

El model que programarem ha de ser de l'estil mini-batch gradient descent, i a més de `sklearn`. Per tant, implementarem els següents mètodes[SKL25log]:

- `__init__`: per inicialitzar els pesos i altres paràmetres del model.
- `_sigmoid`: per calcular la funció sigmoide, que transforma les sortides lineals en probabilitats entre 0 i 1. Aquest mètode és essencial per a la regressió logística. Serà privat, ja que només s'utilitzarà dins de la classe del model.
- `_compute_class_weights`: per calcular els pesos de les classes basant-se en la distribució de la variable objectiu `TRS`. Això ajudarà a gestionar el desbalanceig de classes durant l'entrenament, problema present en el nostre conjunt de dades.
- `_compute_loss`: calcula el cross-entropy loss entre les prediccions i les etiquetes reals. Primer de tot força que els valors mai siguin 0. Després calcula la loss de la seguent manera: $-\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{y}^{(i)})+(1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$, on m és el nombre d'exemples, $y^{(i)}$ és l'etiqueta real i $\hat{y}^{(i)}$ és la predicció del model per a l'exemple i. Estarà ponderat pels pesos de les classes calculats prèviament per gestionar el desbalanceig. També inclou les regularitzacions L1 i L2 per evitar l'overfitting, seguint les fórmules donades a les lliçons de teoria:
  - Regularització L1: Afegeix $\lambda\sum|\omega|$ al càlcul de la loss, on λ és el paràmetre de regularització i ω són els pesos del model. Penalitza el valor absolut dels pesos, cosa que pot portar a que alguns pesos esdevinguin exactament zero, promovent l'esparsitat.
  - Regularització L2: Afegeix $\frac{\lambda}{2}\sum\omega^2$ al càlcul de la loss. Penalitza els pesos grans, ajudant a mantenir els pesos petits i evitant que el model es sobreajusti als dades d'entrenament.
- `_compute_gradient`: per calcular el gradient de la funció de pèrdua respecte als pesos del model. Aquest gradient s'utilitzarà per actualitzar els pesos durant l'entrenament. Aquesta funció també inclou les regularitzacions L1 i L2 per evitar l'overfitting, seguint les fórmules donades a les lliçons de teoria:
  - Regularització L1: Afegeix $\lambda\cdot sign(\omega)$ al càlcul del gradient, on λ és el paràmetre de regularització i ω són els pesos del model. Aquesta penalització afegeix una constant positiva o negativa depenent del signe del pes, promovent l'esparsitat en els pesos.
  - Regularització L2: Afegeix $\lambda\cdot\omega$ al càlcul del gradient. Aquesta penalització és proporcional al valor del pes, ajudant a mantenir els pesos petits i evitant que el model es sobreajusti als dades d'entrenament.

- `fit`: És el mètode principal per entrenar el model utilitzant mini-batch gradient descent. Aquest mètode actualitza els pesos del model en funció del gradient calculat per cada mini-batch de dades. Funciona de tal manera:

    1. Inicialitza amb els valors x_train i y_train i calcula els pesos de les classes.
    2. Per a cada època, reordena aleatòriament les dades d'entrenament per garantir que els mini-batches siguin diferents en cada època.
    3. Després, divideix les dades en mini-batches de la mida especificada.
    4. Prediu, calcula el gradient i actualitza els pesos i biaix per a cada mini-batch.

- `predict`: per fer prediccions binàries (0 o 1) basades en un llindar donat pel paràmetre `threshold`, que per defecte és 0.5.

- `get_params`: per obtenir els paràmetres actuals del model, això és necessari per utilitzar el model amb `GridSearchCV`.

- `set_params`: per establir nous valors als paràmetres del model, també necessari per a `GridSearchCV`.

Una vegada implementats aquests mètodes, el model de regressió logística estarà llest per ser entrenat i avaluat amb les dades preprocesades.

4.3.3 Ajustament del model

Com que hem programat el model de regressió logística de manera que sigui compatible amb l'ús de la gird, doncs utilitzarem `GridSearchCV` per trobar els millors hiperparàmetres per al nostre model personalitzat. Els hiperparàmetres que considerarem són:

Hiperparàmetre	Valors provats	Significat
`learning_rate`	[0.0001, 0.005, 0.01, 0.05, 0.1]	Taxa d'aprenentatge: controla la velocitat amb què el model actualitza els pesos a cada iteració.
`batch_size`	[16, 32, 64]	Mida del lot: nombre de mostres utilitzades en cada actualització dels pesos en l'entrenament.
`n_iterations`	[300, 500, 700]	Nombre d'iteracions: quantitat màxima de passades d'optimització sobre les dades d'entrenament.
`regularization`	['l1', 'l2', None]	Tipus de regularització: L1 fomenta la sparsitat dels pesos, L2 els manté petits; None implica absència de regularització explícita.
`lambda_reg`	[0.01, 0.1, 1]	Força de la regularització: coeficient que controla la intensitat del terme de penalització en la funció de pèrdua.
`class_weight`	['balanced', {0: 1, 1: 2}]	Pes de les classes: ajusta la importància relativa de cada classe per tractar el desbalanceig, donant més pes a la classe minoritària.

Taula X: Espai de cerca d'hiperparàmetres per al model de regressió logística.

Com que per sobre de tot volem evitar l'overfitting, he considerat valors baixos per a la taxa d'aprenentatge, valors mitjans de nombre d'iteracions i he inclòs opcions de regularització L1 i L2. També he inclòs diferents mides de lot per observar com afecten la convergència del model. El paràmetre `class_weight` s'ha establert amb l'opció 'balanced' i un pes personalitzat per donar més importància a la classe minoritària (TRS). El resultat obtingut ha estat:

```
{'batch_size': 64, 'class_weight': {0: 1, 1: 2}, 'lambda_reg': 1, 'learning_rate': 0.0001,
'n_iterations': 300, 'regularization': 'l2'}```
```

El model utilitza una **mida de lot de 64** per equilibrar l'eficiència computacional i l'estabilitat del gradient. La **taxa d'aprenentatge de 0.0001** assegura actualitzacions molt petites dels pesos, ajudant a prevenir oscil·lacions i millorant la convergència. S'han realitzat **300 iteracions**, suficients per permetre que el model

aprengui sense sobreajustar-se. La regularització L2 amb un **lambda de 1** penalitza els pesos grans, ajudant a mantenir-los petits i evitant l'overfitting. Finalment, el pes personalitzat `{0: 1, 1: 2}` dóna més importància a la classe minoritària (TRS), millorant la capacitat del model per detectar aquests casos.

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

	precision	recall	f1-score	support
0	0.74	0.68	0.71	1232
1	0.41	0.48	0.44	568
accuracy			0.62	1800
macro avg	0.57	0.58	0.57	1800
weighted avg	0.63	0.62	0.62	1800

El model de regressió logística presenta un rendiment moderat, amb una **accuracy** del 62% i un F1-score ponderat també del 62%, cosa que indica un equilibri discret entre precisió i recall global.

En el cas de la classe **TRS (1)**, el model identifica correctament el **48% dels pacients (recall)**, amb una precisió del 41%, la qual cosa reflecteix una capacitat limitada per detectar de forma fiable els pacients resistents al tractament tot i una lleugera millora respecte als models previs.

Per a la classe **no TRS (0)**, s'obté una precisió del **74%** i un recall del **68%**, mostrant una millor capacitat per classificar correctament els casos negatius.

Aquest patró de resultats manté un compromís similar als models anteriors, prioritzant parcialment la detecció de la classe minoritària per sobre de l'optimització de la precisió global, cosa rellevant en el context clínic on les conseqüències de no detectar un cas TRS poden ser elevades.

Taula X: Mètriques de classificació del model de regressió logística

5. Model final

6. Model Card

7. Conclusions

8. Referències

[SKL25svc]: Scikit-learn developers. (s.f.). 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

[SKL25log]: scikit-learn Developers. (s. f.). LogisticRegression — scikit-learn 1.8.0 documentation. Scikit-learn. Recuperat el 23 de desembre de 2025, de https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html