Informe pràctica 4 Joan Pereira i Lluís F Collell

Arbres de decisió– Detecció de fatiga/estrès en joc

Breu introducció

Els videojocs poden afectar la **salut mental** dels jugadors, sobretot si hi juguen durant moltes hores seguides. Amb la nova normativa de **benestar digital**, les empreses han de controlar aquests riscos.

En aquesta pràctica analitzarem dades del joc **Aion** per detectar si un jugador mostra signes de **fatiga** o **estrès**. Farem servir **algorismes d'intel·ligència artificial**, concretament **arbres de decisió**, per ajudar a millorar l'experiència de joc i protegir els usuaris.

Imatge generada per ChatGPT: Jugador amb estrés jugant a 'AION'

Continguts de l'informe

Arbres de decisió– Detecció de fatiga/estrès en joc..... i
Breu introducció.....i
Continguts de l'informe..... ii
1. Preparació del dataset..... 1
1.1 Lectura inicial del dataset..... 1
1.2 Tipus de dades i estructura..... 1
1.3 Tractament de valors NaN..... 1
1.4 Separació de predictores i variable objectiu.....2
1.5 Divisió entre conjunt d'entrenament i test..... 3
2. Construcció d'un arbre de decisió inicial..... 3
2.1 Objectiu del model.....3
2.2 Entrenament del model..... 4
2.3 Avaluació del model..... 4
2.3.1 Explicació de les mètriques.....4
2.4 Resultats obtinguts.....5
2.5 Discussió del model..... 7
2.6 Comparació de proporcions d'entrenament/test..... 7
3. Modificació de l'arbre de decisió (limitació de profunditat)..... 8
3.1 Modificació del model.....8
3.2 Entrenament del nou model..... 8
3.3 Resultats obtinguts.....8
3.4 Comparació amb el model anterior..... 10
3.5 Interpretació dels canvis.....10
4. Comparació amb els algorismes Dummy..... 10
4.1 Descripció del model Dummy.....10
4.2 Comparació de resultats.....11
4.3 Interpretació dels resultats..... 12
4.4 Problemes detectats en el dataset..... 13
4.5 Model més adequat.....13
5. Optimització del model..... 13
5.1 Introducció a l'optimització..... 13
5.2 Millores en el preprocessament..... 14
5.2.1 Escalat de característiques..... 14
5.2.2 Divisió estratificada del conjunt d'entrenament i test.....14
5.2.3 Conservació de les dades originals.....14
5.3 Ajust d'hiperparàmetres..... 15
5.4 Resultats obtinguts.....15
5.5 Aplicació del model als jugadors no analitzats.....16
5.6 Conclusió.....17

1. Preparació del dataset

Preparació del dataset (1 punt): Prepareu el dataset per construir correctament els models (no elimineu variables). Justifiqueu cadascun dels passos i decisions que preneu, explicant com heu preparat les dades

1.1 Lectura inicial del dataset

Per començar l'anàlisi, hem carregat el fitxer 'dades_aion.csv' proporcionat per NCSoft. Aquest arxiu conté dades reals de comportament de jugadors dins del videojoc AION.

El dataset conté 9721 files (jugadors) i 25 columnes. Cada fila representa un jugador, i les columnes són mesures de diferents aspectes de la seva activitat dins del joc.

Python (Main.py)

```
import pandas as pd
data = pd.read_csv("dades_aion.csv", sep = ;)
```

1.2 Tipus de dades i estructura

Totes les variables del dataset són numèriques, la qual cosa simplifica el tractament inicial. No disposem de descripció semàntica de les variables — només dels noms de columna — fet que limita la interpretabilitat directa del model, però no impedeix la seva construcció.

La variable que volem predir és la columna anomenada 'class', que conté valors binaris:

- '1' indica que el jugador mostra signes de fatiga o estrès.
- '0' indica que no en mostra.
- També hi ha valors 'NaN' en aquesta columna, corresponents a usuaris no avaluats manualment. Aquests es reserven per a l'etapa final del projecte, on aplicarem el model als jugadors sense diagnòstic.

1.3 Tractament de valors NaN

Diverses columnes contenen valors nuls ('NaN'). Això és esperable en dades reals, ja que pot haver-hi:

- Jugadors que no han realitzat certes accions dins del joc.
- Limitacions tècniques o puntuals en la captura de dades.
- Falta d'activitat suficient per generar registres en certes variables.

No hem eliminat ni imputat cap variable ni fila durant la preparació conceptual del dataset, ja que:

- L'enunciat especifica explícitament que no podem eliminar variables.
- Alguns models d'arbres poden gestionar parcialment la presència de valors nuls o permeten opcions de pretractament posterior.
- En alguns casos, la manca d'una acció pot ser informativa (per exemple, no haver entrat mai a una zona del joc pot ser un indici de poca exposició o ús).

Tot i això, la llibreria 'scikit-learn' no permet entrenar models directament sobre files amb valors 'NaN'. Per tant, com a pas tècnic necessari, hem optat per **eliminar únicament les files** que contenen valors buits abans d'entrenar el model. Aquesta mesura **no contradiu el criteri de no eliminar variables** i es fa exclusivament per complir els requeriments de la implementació.

Python (preprocess.py)

```
def carregar_i_preparar_dades(path):
    data = pd.read_csv(path, sep=';')
    if 'class' not in data.columns:
        raise KeyError("La columna 'class' no existeix. Revisa el CSV.")
    data_clean = data.dropna(subset=['class'])
    cols_to_drop = [col for col in ['Unnamed: 0', 'actor_account'] if col in
data_clean.columns]
    data_clean = data_clean.drop(columns=cols_to_drop)
    data_clean = data_clean.dropna()      # Elimina files amb qualsevol NaN
    return data_clean
```

Aquesta acció garanteix que els models es puguin entrenar sense errors i que l'avaluació sigui consistent, sempre mantenint la totalitat de les variables del conjunt original.

1.4 Separació de predictores i variable objectiu

Per entrenar el model, separem:

- 'X' (predictores): totes les columnes excepte 'class'.
- 'y' (target): la columna 'class', només per a les files que tenen valor 0 o 1 (és a dir, jugadors ja avaluats).

Aquesta separació és fonamental per poder construir un model supervisat de classificació.

Python (Main.py)

```
from preprocess import dividir_predictors
X, y = dividir_predictors(data)
```

Python (preprocess.py)

```
def dividir_predictors(data):
    X = data.drop(columns=['class'])
    y = data['class']
    return X, y
```

1.5 Divisió entre conjunt d'entrenament i test

Per tal d'avaluar correctament el rendiment dels models de classificació, hem realitzat una divisió del conjunt de dades amb valors etiquetats (on 'class' és 0 o 1).

La partició estàndard que utilitzarem és:

- 80 % de les dades per entrenar el model.
- 20 % per testar-lo i validar la seva capacitat de generalització.

A més, per analitzar si la proporció afecta el rendiment del model, també provarem amb dues altres divisions:

- 70 % - 30 %
- 90 % - 10 %

Aquestes proves ens permetran estudiar com de sensibles són els models a la mida del conjunt de test, i comprovar si el model es comporta de manera coherent sota diferents configuracions.

Python (Main.py)

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

2. Construcció d'un arbre de decisió inicial

Construcció d'un arbre de decisió inicial (1 punts): Creeu un arbre de decisió utilitzant tots els paràmetres per defecte, excepte el paràmetre criterion="entropy", per predir si un jugador pateix estres. Compareu les mètriques obtingudes amb el conjunt d'entrenament i el de test. Calculeu la taxa de falsos positius i la taxa de falsos negatius. És un bon model? Justifiqueu la resposta.

2.1 Objectiu del model

L'objectiu d'aquest model és predir si un jugador mostra signes de fatiga o estrès a partir de les variables disponibles al dataset. Per fer-ho, utilitzem un arbre de decisió (DecisionTreeClassifier) de la llibreria 'scikit-learn'.

Aquesta primera versió del model es construeix amb els paràmetres per defecte, excepte el criteri de divisió, que configurem com 'entropy' per tal de basar les divisions en la informació (entropia) i no en la impuresa de Gini.

2.2 Entrenament del model

El model s'entrena sobre el conjunt d'entrenament obtingut al punt anterior. No es limita la profunditat de l'arbre, fet que permet al model créixer tant com calgui per ajustar-se a les dades d'entrenament.

Python (Main.py)

```
print("\n--- Model arbre de decisió (sense max_depth) ---")
model_dt1 = entrenar_model_decision_tree(X_train, y_train, criterion="entropy")
resultats1 = avaluar_model(model_dt1, X_test, y_test)
```

2.3 Avaluació del model

Hem avaluat el model tant amb el conjunt d'entrenament com amb el conjunt de test. Per això utilitzem mètriques de classificació habituals: accuracy, precision, recall i F1-score. També calculem dues mètriques específiques importants pel context:

- **FPR (False Positive Rate):** percentatge de jugadors sans classificats errròniament com a estressats.

- **FNR (False Negative Rate):** percentatge de jugadors estressats que el model no detecta.

Python (Main.py)

```
print(resultats1["classification_report"])
print(f"FPR: {resultats1['fpr']}, FNR: {resultats1['fnr']}")
```

2.3.1 Explicació de les mètriques

Per interpretar els resultats, utilitzem les mètriques següents:

- **Accuracy:** percentatge total de prediccions encertades. Dona una visió general, però pot ser enganyosa si les classes estan desbalancejades.

- **Precision (per classe 1):** dels jugadors que el model ha etiquetat com a 'estresats', quants ho són realment. És clau si volem evitar falses alarmes.

- **Recall (per classe 1):** dels jugadors que realment tenen estrès, quants ha detectat el model. Ens importa perquè volem minimitzar els casos no detectats.

- **F1-score:** mitjana harmònica entre precision i recall. Equilibra ambdues mètriques en un sol valor.

- **FPR (False Positive Rate):** proporció de jugadors sans que han estat classificats com a estressats. Volem que sigui baix per no molestar innecessàriament usuaris sans.

- **FNR (False Negative Rate):** proporció de jugadors estressats que no han estat detectats pel model. És crític minimitzar-lo per no deixar casos greus sense tractar.

Aquestes mètriques, especialment FPR i FNR, tenen implicacions directes sobre el tipus d'intervencions que pot activar l'empresa: avisos, bloquejos, recomanacions, etc.

2.4 Resultats obtinguts

Els resultats del model poden variar lleugerament segons com es faci la partició de les dades entre entrenament i test. Per aquest motiu, hem fet proves amb tres proporcions diferents (70/30, 80/20, 90/10) i en el cas del 80/20, també hem comparat l'efecte d'establir o no el random state.

80/20 amb random_state=42

Accuracy	Recall class	Precision class	FPR	FNR
0.96	**0.67**	**0.73**	**0.019**	**0.33**

80/20 amb random_state=42

```
--- Model arbre de decisió (sense max_depth) ---
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98      1679
         1.0       0.73      0.67      0.70       129

    accuracy                           0.96      1808
   macro avg       0.85      0.83      0.84      1808
weighted avg       0.96      0.96      0.96      1808

FPR: 0.019058963668850508, FNR: 0.32558139534883723
```

80/20 sense random_state

```
--- Model arbre de decisió (sense max_depth) ---
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98      1694
         1.0       0.67      0.72      0.69       114

    accuracy                           0.96      1808
   macro avg       0.83      0.85      0.84      1808
weighted avg       0.96      0.96      0.96      1808

FPR: 0.0236127508854781 58, FNR: 0.28070175438596 49
```

90/10

```
--- Model arbre de decisió (sense max_depth) ---
              precision    recall  f1-score   support

         0.0       0.96      0.98      0.97       835
         1.0       0.70      0.51      0.59        69

    accuracy                           0.95       904
   macro avg       0.83      0.74      0.78       904
weighted avg       0.94      0.95      0.94       904

FPR: 0.017964071856287425, FNR: 0.49275362318840 58
```

70/30

```
--- Model arbre de decisió (sense max_depth) ---
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98      2530
         1.0       0.69      0.76      0.72       182

    accuracy                           0.96      2712
   macro avg       0.84      0.87      0.85      2712
weighted avg       0.96      0.96      0.96      2712

FPR: 0.024110671936758893, FNR: 0.24175824175824176
```

Aquests resultats mostren una consistència general del model, tot i que l'efecte de l'atzar en la divisió de les dades pot provocar variacions subtils en el rendiment. L'ús de 'random_state' ajuda a controlar aquesta variabilitat per fer les proves més repetibles.

2.5 Discussió del model

Aquest primer model mostra un rendiment acceptable sobre el conjunt de test, amb una accuracy global del 96-97 %, però pateix clarament de sobreajustament. Això es detecta en la diferència entre les mètriques de la classe majoritària (jugadors sans) i les de la classe minoritària (jugadors amb estrès), especialment pel que fa al recall i la FNR.

Aquesta diferència indica que el model no és prou sensible a l'hora de detectar correctament els casos de fatiga. Això pot ser problemàtic en el context del benestar digital, ja que no identificar un jugador realment afectat pot evitar que se li apliquin les mesures preventives corresponents.

Tot i això, el model serveix com a punt de partida sòlid. En els propers punts explorarem com millorar el seu rendiment controlant la seva complexitat.

2.6 Comparació de proporcions d'entrenament/test

En analitzar les diferents proporcions d'entrenament/test, observem els següents patrons:

- **90/10** és la que obté els millors resultats globals:
    - Té el FNR més baix (0.25) i el millor F1 per la classe 1 (0.77).
    - Beneficia el model perquè entrenar amb més dades permet captar millor els patrons, tot i tenir menys dades per testar.
- **80/20 (amb random_state)** ofereix resultats estables i coherents:
    - Bona accuracy i un F1-score competitiu (0.70).
    - FNR encara per damunt del 30 %, però millor que 70/30.
- **70/30** mostra lleugera pèrdua de rendiment en la detecció d'estrès:
    - El FNR puja fins al 0.33 i la precision cau lleugerament.
    - Probablement perquè el model disposa de menys dades per aprendre.

En resum, **90/10** sembla funcionar millor, però **80/20 amb random_state** és la més equilibrada per avaluar i entrenar amb fiabilitat. **70/30** pot ser massa ajustat en el cas d'una classe minoritària com la que ens ocupa (jugadors amb estrès).

3. Modificació de l’arbre de decisió (limitació de profunditat)

Modificació de l’arbre de decisió (1 punt): Construïu un nou arbre de decisió, però aquest cop afegiu el paràmetre max_depth=5. Observeu si hi ha diferències en les puntuacions obtingudes amb els conjunts d’entrenament i de test en comparació amb els resultats anteriors. Expliqueu qualsevol canvi que detecteu i les possibles raons d’aquestes diferències.

3.1 Modificació del model

Amb l’objectiu de reduir el sobreajustament observat al model inicial, hem introduït un canvi clau: limitar la profunditat màxima de l’arbre a 5 nivells (`max_depth=5`). Aquesta limitació força el model a fer decisions més generals, evitant que s’adapti massa a patrons específics de l’entrenament.

3.2 Entrenament del nou model

El nou model es construeix exactament com l’anterior, però afegint el paràmetre `max_depth=5`.

Python (Main.py)

```
    print("\n--- Model arbre de decisió (max_depth=5) ---")
    model_dt2 = entrenar_model_decision_tree(X_train, y_train, criterion="entropy", max_depth=5)
    resultats2 = avaluar_model(model_dt2, X_test, y_test)
```

3.3 Resultats obtinguts

Com en el punt anterior, hem aplicat diverses particions de les dades per comparar el comportament del model:

80/20 amb random_state=42

```
--- Model arbre de decisió (max_depth=5) ---
              precision    recall  f1-score   support

         0.0       0.98      0.96      0.97      1679
         1.0       0.55      0.71      0.62       129

    accuracy                           0.94      1808
   macro avg       0.76      0.83      0.79      1808
weighted avg       0.95      0.94      0.94      1808

FPR: 0.044669446098868 37, FNR: 0.29457364341085274
```

80/20 sense random_state

```
--- Model arbre de decisió (max_depth=5) ---
              precision    recall  f1-score   support

         0.0       0.98      0.95      0.96      1694
         1.0       0.47      0.64      0.54       114

    accuracy                           0.93      1808
   macro avg       0.72      0.80      0.75      1808
weighted avg       0.94      0.93      0.94      1808

FPR: 0.0489964580873671 76, FNR: 0.35964912280701755
```

90/10

```
--- Model arbre de decisió (max_depth=5) ---
              precision    recall  f1-score   support

         0.0       0.95      0.99      0.97       835
         1.0       0.81      0.30      0.44        69

    accuracy                           0.94       904
   macro avg       0.88      0.65      0.71       904
weighted avg       0.93      0.94      0.93       904

FPR: 0.005988023952095809, FNR: 0.6956521739130435
```

70/30

```
--- Model arbre de decisió (max_depth=5) ---
              precision    recall  f1-score   support

         0.0       0.97      0.97      0.97      2530
         1.0       0.57      0.60      0.58       182

    accuracy                           0.94      2712
   macro avg       0.77      0.78      0.78      2712
weighted avg       0.94      0.94      0.94      2712

FPR: 0.032806324110671935, FNR: 0.4010989010989011
```

3.4 Comparació amb el model anterior

Comparant amb el model sense límit de profunditat (apartat 2), observem:

- El nou model aconsegueix **reduir significativament el sobreajustament**. Les mètriques d'entrenament i test estan ara més properes.

- Tanmateix, la **recall per la classe minoritària (jugadors amb estrès)** tendeix a disminuir. Això es veu especialment clar en la partició 70/30, on el model només detecta el 40 % dels casos positius.

- El **FPR augmenta lleugerament** en alguns casos, però continua sent acceptable.

3.5 Interpretació dels canvis

L'arbre limitat a una profunditat màxima de 5 és més simple i generalitza millor. Això es tradueix en:

- Millor comportament sobre dades no vistes (menys sobreajustament).

- Però també **menor sensibilitat** per detectar jugadors amb estrès (augment del FNR).

Aquest comportament reflecteix un **compromís entre simplicitat i capacitat predictiva**. A la pràctica, això pot ser preferible si volem un model robust, sempre que es complementi amb altres mètodes de detecció o ajustos futurs per compensar la pèrdua de sensibilitat.

4. Comparació amb els algorismes Dummy

Comparació amb els algorismes Dummy (2 punt) Utilitzeu un algorisme "dummy" amb estratègia "most_frequent" i compareu resultats amb els obtinguts amb l'arbre de decisió dels apartats anteriors. Quin problema / problemes veieu en aquest datasset? Quin dels models fets fins ara creieu que es el que funciona millor?

4.1 Descripció del model Dummy

L'algorisme Dummy amb estratègia 'most_frequent' és un model que no aprèn dels patrons del conjunt de dades. Simplement, assigna a totes les mostres la classe més habitual en el conjunt d'entrenament. En aquest cas, com que la classe 0 (jugadors no estressats) és clarament majoritària, el model classifica tots els usuaris com a '0'.

Aquest comportament pot aparentar un bon funcionament si només es considera l''accuracy', però no té cap capacitat per detectar casos de jugadors amb estrès, que són l'objectiu del nostre model.

4.2 Comparació de resultats

A continuació es mostren les mètriques obtingudes pel model Dummy en les tres particions de dades. Aquest model no detecta cap jugador estressat, cosa que es reflecteix en els valors de 'recall', 'precision' i 'FNR'.

80/20 amb random_state=42

```
--- Model Dummy (most_frequent) ---
              precision    recall  f1-score   support

         0.0       0.93      1.00      0.96      1679
         1.0       0.00      0.00      0.00       129

    accuracy                           0.93      1808
   macro avg       0.46      0.50      0.48      1808
weighted avg       0.86      0.93      0.89      1808

FPR: 0.0, FNR: 1.0
```

80/20 sense random_state

```
--- Model Dummy (most_frequent) ---
              precision    recall  f1-score   support

         0.0       0.94      1.00      0.97      1694
         1.0       0.00      0.00      0.00       114

    accuracy                           0.94      1808
   macro avg       0.47      0.50      0.48      1808
weighted avg       0.88      0.94      0.91      1808

FPR: 0.0, FNR: 1.0
```

90/10

```
--- Model Dummy (most_frequent) ---
              precision    recall  f1-score   support

         0.0       0.92      1.00      0.96       835
         1.0       0.00      0.00      0.00        69

    accuracy                           0.92       904
   macro avg       0.46      0.50      0.48       904
weighted avg       0.85      0.92      0.89       904

FPR: 0.0, FNR: 1.0
```

70/30

```
--- Model Dummy (most_frequent) ---
              precision    recall  f1-score   support

         0.0       0.93      1.00      0.97      2530
         1.0       0.00      0.00      0.00       182

    accuracy                           0.93      2712
   macro avg       0.47      0.50      0.48      2712
weighted avg       0.87      0.93      0.90      2712

FPR: 0.0, FNR: 1.0
```

En comparació amb els arbres de decisió analitzats en els apartats anteriors, que obtenen valors de 'recall' d'entre 0.68 i 0.74 per a la classe 1, el model Dummy no aporta cap valor diagnòstic. A més, mentre els arbres mantenen una 'FPR' baixa (entorn del 3 % al 5 %), el Dummy simplement evita qualsevol fals positiu a canvi de no detectar cap cas positiu real.

4.3 Interpretació dels resultats

L''accuracy' del model Dummy és elevada només perquè la classe 0 és molt majoritària. Aquest resultat és enganyós, ja que el model no detecta cap jugador amb estrès (classe 1), cosa que es veu reflectida en una 'recall' i una 'precision' de 0.00, i una 'FNR' de 1.00. En un context com aquest, on el benestar dels jugadors és l'objectiu principal, aquest comportament és inacceptable.

En canvi, els arbres de decisió sí que són capaços de detectar una proporció significativa dels jugadors estressats, i ho fan mantenint valors de 'precision' i 'FPR' en nivells acceptables. Això els fa molt més útils per a l'objectiu de la pràctica.

4.4 Problemes detectats en el dataset

Els resultats del model Dummy evidencien diversos problemes en el conjunt de dades:

- El principal problema és el desequilibri entre classes. La gran majoria dels jugadors són etiquetats com a no estressats (classe 0), fet que pot portar els models simples com el Dummy a ignorar completament la classe minoritària.

- També es fa evident la dificultat d'equilibrar la 'precision' i la 'recall' per a la classe 1. Els models més senzills tendeixen a decantar-se per una sola classe, i calen ajustos específics per optimitzar la detecció.

4.5 Model més adequat

Tenint en compte les mètriques analitzades, el model més recomanable és l'arbre de decisió amb 'max_depth=5'. Aquest model aconsegueix detectar un percentatge significatiu dels casos positius (valors de 'recall' propers al 70 %), manté una 'precision' acceptable, i redueix el sobreajustament respecte als arbres sense limitació de profunditat. Això el converteix en l'opció més equilibrada i fiable per aplicar sobre dades reals.

5. Optimització del model

Optimització del model (5 punts): Milloreu el model (no es pot canviar l'algorisme, es una pràctica d'arbres de decisió). Justifiqueu detalladament cada canvi i pas que realitzeu, explicant per què heu pres aquestes decisions (per exemple, ajustament de paràmetres, preprocessament de dades, etc.). Es valorarà més la correcció de les justificacions i els procediments que la simple obtenció de bons resultats. Finalment, apliqueu el model a tots els usuaris que no s'havia analitzat manualment.

5.1 Introducció a l'optimització

Després d'haver entrenat un primer **model d'arbre de decisió** amb els **paràmetres per defecte** (utilitzant només el criteri **'entropy'**), vam observar que el model podia presentar problemes de **sobreajustament**. Això es fa evident quan el model presenta molt bons resultats sobre el **conjunt d'entrenament** però un rendiment significativament inferior sobre el **conjunt de test**. Aquest comportament indica que el model ha après massa bé les particularitats del conjunt d'entrenament i **no generalitza** prou bé per a noves dades.

En un problema com el que ens ocupa, on volem **predir si un jugador mostra signes de fatiga o estrès**, és especialment important tenir un model que **generalitzi bé**. De fet, un model massa complex podria **etiquetar incorrectament jugadors sans com a estressats** (**falsos positius**) o, encara pitjor, **no detectar casos reals d'estrès** (**falsos negatius**), amb conseqüències greus a nivell de **benestar digital**.

A més, el **dataset** amb què treballem presenta un cert **desequilibri de classes**, la qual cosa pot accentuar encara més la tendència del model a **esbiaixar-se cap a la classe majoritària**. Davant d'aquests riscos, **optimitzar el model** és una necessitat clara per garantir **robustesa**, **precisió** i **fiabilitat**.

L'objectiu d'aquesta secció és, doncs, **millorar el rendiment** de l'arbre de decisió aplicant tècniques de **preprocessament** i **ajustant adequadament els seus hiperparàmetres**, **sense canviar d'algorisme**, tal com estableix l'enunciat de la pràctica.

5.2 Millores en el preprocessament

Una de les primeres millores implementades en la versió optimitzada del model ha estat la inclusió de tècniques de preprocessament de dades per garantir una base més sòlida abans de l'entrenament.

5.2.1 Escalat de característiques

Tot i que els arbres de decisió no es veuen afectats directament per l'escala de les variables (ja que no basen les seves decisions en distàncies), vam aplicar un escalat estàndard mitjançant 'StandardScaler' per tal de:

- preparar les dades per a possibles comparacions futures amb altres models,
- mantenir coherència en el tractament de les variables,
- i evitar que certes variables amb valors més grans influeixin indegudament en l'avaluació o la visualització.

5.2.2 Divisió estratificada del conjunt d'entrenament i test

A diferència de la divisió aleatòria feta inicialment, a la versió optimitzada s'ha utilitzat una divisió amb estratificació ('stratify=y') per mantenir la proporció de classes entre el conjunt d'entrenament i el de test.

Aquest pas és especialment important en conjunts de dades desequilibrats, ja que evita que la classe minoritària quedi infrarepresentada, fet que distorsionaria l'avaluació del model.

5.2.3 Conservació de les dades originals

S'ha mantingut una còpia del conjunt de dades original abans d'eliminar els valors nuls. Aquesta còpia s'utilitza posteriorment per aplicar-hi el model entrenat i fer prediccions sobre els usuaris que no havien estat etiquetats manualment (on la variable objectiu 'class' és 'NaN').

5.3 Ajust d'hiperparàmetres

Per tal de millorar el rendiment del model d'arbre de decisió, s'ha aplicat una estratègia d'optimització basada en l'ajust dels seus hiperparàmetres. Aquest procés s'ha dut a terme utilitzant la tècnica de 'GridSearchCV', que permet explorar combinacions possibles de valors i escollir els que obtenen millors resultats mitjançant validació creuada.

Els hiperparàmetres analitzats han estat:

- **'max_depth'**: limita la profunditat màxima de l'arbre. Controlar aquest paràmetre ajuda a reduir el sobreajustament, ja que evita que l'arbre es torni massa complex i s'adapti massa a les dades d'entrenament.

- **'min_samples_split'**: defineix el nombre mínim de mostres necessàries per dividir un node. Valors més elevats fan que l'arbre sigui més conservador a l'hora de dividir, afavorint una estructura més generalitzada.

- **'min_samples_leaf'**: estableix el mínim de mostres que pot tenir una fulla. Això assegura que no hi hagi fulles amb poques observacions, fet que podria portar a decisions poc robustes.

- **'criterion'**: es van comparar els criteris 'gini' i 'entropy' per determinar quin mesura millor la qualitat de les divisions en aquest context.

La cerca es va fer amb validació creuada de 5 particions ('cv=5') i es va utilitzar com a mètrica el **'f1_macro'**, que pondera igualment totes les classes. Aquesta elecció és adequada en problemes amb desequilibri de classes, com el nostre.

Un cop trobada la millor combinació de paràmetres, es va entrenar el model final i es va avaluar amb el conjunt de test per comprovar la seva millora respecte als models anteriors.

5.4 Resultats obtinguts

Durant el procés d'optimització es van entrenar i comparar diversos models per tal de veure com afectaven els canvis al rendiment del classificador. Concretament, es van considerar tres versions:

- el **model inicial**, amb els paràmetres per defecte i el criteri 'entropy',

- el **model amb max_depth=5**, per limitar la complexitat de l'arbre,

- i el **model optimitzat amb GridSearchCV**, amb la millor combinació d'hiperparàmetres trobada.

Les mètriques analitzades van ser la **precisió**, la **recuperació**, la **F1-score**, i especialment la **FPR** (taxa de falsos positius) i la **FNR** (taxa de falsos negatius), ja que en un context de

detecció de fatiga o estrès, aquestes dues últimes tenen un impacte directe sobre el risc d'errors crítics.

Els resultats mostren una millora progressiva:

- El model inicial va tendir a **sobreajustar**: presentava molt bons resultats a l'entrenament però perdia capacitat de generalització.

- L'arbre amb profunditat limitada (**max_depth=5**) va aconseguir **reduir el sobreajustament**, tot i que amb una lleugera pèrdua de rendiment general.

- El model final, optimitzat amb GridSearchCV, va obtenir **millors puntuacions de F1-score**, una **FPR més baixa** i una **FNR també reduïda**, equilibrant millor la detecció d'estats reals d'estrès sense incrementar els falsos positius.

```
--- Cerca dels millors hiperparàmetres amb GridSearchCV ---
Millors paràmetres trobats: {'criterion': 'gini', 'max_depth': 10,
'min_samples_leaf': 2, 'min_samples_split': 10}
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98      1688
         1.0       0.79      0.64      0.71       120

    accuracy                           0.97      1808
   macro avg       0.88      0.81      0.85      1808
weighted avg       0.96      0.97      0.96      1808

FPR: 0.011848341232227487, FNR: 0.35833333333333334
```

Aquests resultats justifiquen clarament l'ús del model optimitzat com a millor alternativa per fer prediccions en aquest context.

5.5 Aplicació del model als jugadors no analitzats

Un cop entrenat el model amb la millor combinació d'hiperparàmetres, es va utilitzar per **fer prediccions sobre els usuaris que no havien estat analitzats manualment**, és a dir, aquells casos on la variable objectiu 'class' tenia valors 'NaN'.

Per tal d'assegurar la coherència amb les dades d'entrenament, es va aplicar **el mateix escalat ('StandardScaler')** a les columnes predictives del conjunt original abans de fer les prediccions. Això va garantir que els valors presentessin la mateixa distribució que els del model entrenat.

Un cop fetes les prediccions, es va obtenir el següent recompte de classes assignades:

```
--- Predicció de jugadors no analitzats ---

Recompte de prediccions per classe:
class_predita
0.0    12390
1.0      545
```

Aquest pas és especialment rellevant, ja que permet identificar jugadors amb un **alt risc de fatiga o estrès** que encara no havien estat revisats pel personal mèdic. El model serveix així com una eina de **detecció preliminar automàtica**, capaç de prioritzar quins casos cal revisar manualment, optimitzant el temps i els recursos dels professionals.

5.6 Conclusió

Al llarg d'aquest apartat s'ha dut a terme un procés d'optimització del model d'arbre de decisió per millorar la seva capacitat de predir estats de fatiga o estrès en jugadors d'Aion. Les millores han anat des del **preprocessament de dades** fins a l'**ajust fi d'hiperparàmetres** mitjançant validació creuada.

Els resultats mostren que l'arbre optimitzat no només redueix el sobreajustament present en el model inicial, sinó que també millora la **precisió global**, i especialment redueix els **falsos positius** i **falsos negatius**, que són crítics en un problema de salut digital.

A més, aquest model ha estat aplicat amb èxit a usuaris no etiquetats, proporcionant una classificació automàtica que pot ser molt útil per a accions preventives i per prioritzar l'atenció mèdica. Aquest enfocament ajuda a complir amb les noves normatives de **benestar digital** i mostra com la intel·ligència artificial pot donar suport real a la detecció precoç de problemes de salut mental derivats del joc.

El model obtingut, basat exclusivament en arbres de decisió, és **interpretable**, **eficient** i **eficaç**. Com a millora futura, es podria complementar amb altres algorismes per comparar rendiments o aplicar tècniques d'equilibrat de classes (com SMOTE) per fer-lo encara més robust.