Pràctica Bloc III: Aerolínies i Satisfacció

Universitat de les Illes Balears *Grau en Enginyeria Informàtica Assignatura: 21722 - Intel·ligència Artificial*

- **Nom i Llinatges:** Jaume Cardell Aguaded
- **Nom i Llinatges:** Jose María Macías Martínez
- **Data:** 14 de gener 2026
- **Curs:** 2025-2026

Índex

---

1. Introducció i Objectius

Objectiu

L'objectiu principal d'aquesta pràctica és aplicar tècniques d'aprenentatge automàtic (**Machine Learning**) per predir la satisfacció dels passatgers d'una aerolínia. A través d'aquest exercici, compararem el rendiment de cinc models diferents vistos a l'assignatura:

1. Perceptró
2. Regressió Logística
3. Màquines de Vectors de Suport (SVM)
4. Arbres de Decisió

5. Boscos Aleatoris (Random Forest)

Aquests models seran empleats mitjançant la llibreria `Scikit-learn`.

Descripció del Problema

Les aerolínies modernes recullen grans quantitats de dades per millorar l'experiència del client. Treballarem amb el dataset **Airline Passenger Satisfaction**, que conté informació demogràfica, detalls del vol i puntuacions de serveis (Wifi, menjar, comoditat, etc.).

La nostra variable objectiu (*target*) és **`satisfaction`**, que classifica el passatger en:

- `Satisfied`
- `Neutral or Dissatisfied`

---

2. Metodologia

Per comparar els diferents models y resoldre el problema, seguirem els següents passos obligatoris:

1. **Preprocessament:** Neteja de valors nuls (`NaN`), codificació de variables categòriques i normalització de dades.
2. **Anàlisi Exploratòria (EDA):** Estudi de les dades i aplicació d'algoritmes de **Clustering** (K-Means) per detectar patrons de comportament.
3. **Entrenament i Ajust:** Cerca dels millors hiperparàmetres per a cada model.
4. **Avaluació:** Comparació de mètriques i discussió dels resultats.

---

3. Práctica

3.1. Preprocessament de les dades:

La finalitat d'aquest primer procés es preparar l'informació que utilitzaran els diferents models d'aprenentatge que entrenarem posteriorment. El següent codi de Python càrrega les dades del ficher CSV enviat com a paràmetre, després mostra un exemple de les primeres files de dades per tal de assegurar les dades que conté el fitxer.

In [116…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
filename = 'apsd.csv' # Ruta arxiu CSV

# 1. Carrega el dataset
df = pd.read_csv(filename)

# 2. Mostra primeres files de dades
print("\nMostra les primeres files del dataset:")
display(df.head())

# 3. Mostra descripció estadística del dataset
print("\nDescripció estadística del dataset:")
df_description = df.describe()
display(df_description)
```

```
Mostra les primeres files del dataset:
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	I s
0	0	19556	Female	Loyal Customer	52	Business travel	Eco	160	
1	1	90035	Female	Loyal Customer	36	Business travel	Business	2863	
2	2	12360	Male	disloyal Customer	20	Business travel	Eco	192	
3	3	77959	Male	Loyal Customer	44	Business travel	Business	3377	
4	4	36875	Female	Loyal Customer	49	Business travel	Eco	1182	

5 rows × 25 columns

```
Descripció estadística del dataset:
```

	Unnamed: 0	id	Age	Flight Distance	Inflight serv
count	25976.000000	25976.000000	25976.000000	25976.000000	25976.000
mean	12987.500000	65005.657992	39.620958	1193.788459	2.724
std	7498.769632	37611.526647	15.135685	998.683999	1.335
min	0.000000	17.000000	7.000000	31.000000	0.000
25%	6493.750000	32170.500000	27.000000	414.000000	2.000
50%	12987.500000	65319.500000	40.000000	849.000000	3.000
75%	19481.250000	97584.250000	51.000000	1744.000000	4.000
max	25975.000000	129877.000000	85.000000	4983.000000	5.000

El valor **Unnamed:0** representa la columna que conté l'índex de fila. Es a dir la

fila a la que correspón cada dada.

Tractament de valors faltants: NaN

Aquesta part del codi mostra la informació del dataset carregat i posteriorment fa un conteig per tal de saber el nombre de columnes afectades per qualque Not a Number.

En aquest cas el total de columnes afectades es de 83 (de un total de 25976 columnes): Com es tracta de un nombre inferior al 0,33% podriem descartar les columnes afectades pero en aquest cas, per tal d'evitar la pérdua de cap dada, hem decidit emplenar les dades faltants amb la mitjana de la resta de dades.

In [117…

```
# Solucions per a valors nuls (NaN):
print("\nInformació del Dataset")
df.info()

print("\nConteig de NaN") # NaN = Not a Number (valors nuls)
print("Nombre de NaN per columna:")
print(df.isnull().sum()[df.isnull().sum() > 0])

filas_nan = df[df.isnull().any(axis=1)] # Filtrar files amb qualcun

print(f"Se han trobat {len(filas_nan)} files amb NaN.") # Imprimir
print(filas_nan[['id', 'Arrival Delay in Minutes']]) # Això realmen

# Decidim com tractar els NaN: Emplenarem amb la mitjana de la colu
# En aquest cas, només hi ha NaN a la columna 'Arrival Delay in Min
mitjana_retras = df['Arrival Delay in Minutes'].mean() # Calculem l
print("Mitjana retràs:", mitjana_retras)
df['Arrival Delay in Minutes'] = df['Arrival Delay in Minutes'].fil

# Verificam que ja no hi ha cap NaN a les dades:
print(f"NaN actual: {df['Arrival Delay in Minutes'].isnull().sum()}
```

```
Informació del Dataset
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25976 entries, 0 to 25975
Data columns (total 25 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Unnamed: 0                        25976 non-null  int64  
 1   id                                25976 non-null  int64  
 2   Gender                            25976 non-null  object 
 3   Customer Type                     25976 non-null  object 
 4   Age                               25976 non-null  int64  
 5   Type of Travel                    25976 non-null  object 
 6   Class                             25976 non-null  object 
 7   Flight Distance                   25976 non-null  int64  
 8   Inflight wifi service             25976 non-null  int64  
 9   Departure/Arrival time convenient  25976 non-null  int64  
 10  Ease of Online booking            25976 non-null  int64  
 11  Gate location                     25976 non-null  int64  
 12  Food and drink                    25976 non-null  int64  
 13  Online boarding                   25976 non-null  int64  
 14  Seat comfort                      25976 non-null  int64  
 15  Inflight entertainment            25976 non-null  int64  
 16  On-board service                  25976 non-null  int64  
 17  Leg room service                  25976 non-null  int64  
 18  Baggage handling                  25976 non-null  int64  
 19  Checkin service                   25976 non-null  int64  
 20  Inflight service                  25976 non-null  int64  
 21  Cleanliness                       25976 non-null  int64  
 22  Departure Delay in Minutes        25976 non-null  int64  
 23  Arrival Delay in Minutes          25893 non-null  float64
 24  satisfaction                      25976 non-null  object 
dtypes: float64(1), int64(19), object(5)
memory usage: 5.0+ MB

Conteig de NaN
Nombre de NaN per columna:
Arrival Delay in Minutes    83
dtype: int64
Se han trobat 83 files amb NaN.
           id  Arrival Delay in Minutes
516    107365                       NaN
656    108648                       NaN
1071    16797                       NaN
1224    30090                       NaN
1589    41924                       NaN
...       ...                       ...
24072   21780                       NaN
24133   64934                       NaN
24301  125688                       NaN
25128   64706                       NaN
25468   85927                       NaN

[83 rows x 2 columns]
Mitjana retràs: 14.74085660217047
NaN actual: 0
```

Per tant en aquest punt no queda cap columna amb valors nuls.

Tractament de variables categòricas.

Amb l'objectiu de poder entrenar els diferents models de inteligència artificial convertirem les dades de tipus text a nombres mitjançant la lliberia `Scikit-learn`.

Posteriorment mostram el valor número assignat a cada un dels valors originals del dataset.

In [118…

```
from sklearn.preprocessing import LabelEncoder # Scikit-learn

if df['Class'].dtype == 'object':
    class_mapping = {'Eco': 0, 'Eco Plus': 1, 'Business': 2}
    df['Class'] = df['Class'].map(class_mapping)
    print(f" -> Columna 'Class' (Manual): {class_mapping}")

cat_cols = df.select_dtypes(include=['object']).columns
print(f"Resta de variables a transformar automàticament: {list(cat_

le = LabelEncoder()

for col in cat_cols:
    df[col] = le.fit_transform(df[col])
    # Imprimim el mapping per saber què és cada número
    mapping = dict(zip(le.classes_, le.transform(le.classes_)))
    print(f" -> Columna '{col}': {mapping}")

print(df.info())
display(df.head())
```

```
 -> Columna 'Class' (Manual): {'Eco': 0, 'Eco Plus': 1, 'Business':
2}
Resta de variables a transformar automàticament: ['Gender', 'Custome
r Type', 'Type of Travel', 'satisfaction']
 -> Columna 'Gender': {'Female': np.int64(0), 'Male': np.int64(1)}
 -> Columna 'Customer Type': {'Loyal Customer': np.int64(0), 'disloy
al Customer': np.int64(1)}
 -> Columna 'Type of Travel': {'Business travel': np.int64(0), 'Pers
onal Travel': np.int64(1)}
 -> Columna 'satisfaction': {'neutral or dissatisfied': np.int64(0),
'satisfied': np.int64(1)}
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25976 entries, 0 to 25975
Data columns (total 25 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Unnamed: 0                        25976 non-null  int64
 1   id                                25976 non-null  int64
 2   Gender                            25976 non-null  int64
 3   Customer Type                     25976 non-null  int64
 4   Age                               25976 non-null  int64
 5   Type of Travel                    25976 non-null  int64
 6   Class                             25976 non-null  int64
 7   Flight Distance                   25976 non-null  int64
 8   Inflight wifi service             25976 non-null  int64
 9   Departure/Arrival time convenient  25976 non-null  int64
 10  Ease of Online booking            25976 non-null  int64
 11  Gate location                     25976 non-null  int64
 12  Food and drink                    25976 non-null  int64
 13  Online boarding                   25976 non-null  int64
 14  Seat comfort                      25976 non-null  int64
 15  Inflight entertainment            25976 non-null  int64
 16  On-board service                  25976 non-null  int64
 17  Leg room service                  25976 non-null  int64
 18  Baggage handling                  25976 non-null  int64
 19  Checkin service                   25976 non-null  int64
 20  Inflight service                  25976 non-null  int64
 21  Cleanliness                       25976 non-null  int64
 22  Departure Delay in Minutes        25976 non-null  int64
 23  Arrival Delay in Minutes          25976 non-null  float64
 24  satisfaction                      25976 non-null  int64
dtypes: float64(1), int64(24)
memory usage: 5.0 MB
None
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Infligl wi servic
0	0	19556	0	0	52	0	0	160	
1	1	90035	0	0	36	0	2	2863	
2	2	12360	1	1	20	0	0	192	
3	3	77959	1	0	44	0	2	3377	
4	4	36875	0	0	49	0	0	1182	

5 rows × 25 columns

Les variables categòriques afectades i els valors adoptats són:

1. *Class*: Eco(0), Eco Plus(1) i Business(2).
2. *Gender*: Female(0) i Male(1).
3. *Customer Type*: Loyal Customer(0) i Disloyal Customer(1).
4. *Type of Travel*: Business travel(0) i Personal Travel(1).
5. *Satisfaction*: Neutral or Dissatisfied(0) i Satisfied(1).

3.2. **Anàlisi Exploratòria de dades (EDA):**

Estudi de les dades i aplicació d'algoritmes de **Clustering** (K-Means) per detectar patrons de comportament.

3.2.1 Normalització de dades

Hem de normalitzar les dades per tal de crear un bon gráfic de K-Means. En el cas de que no normalitzar les dades els nombres grans tendríen molt més valor que els nombres petits. Les dades estan compreses entre 0 i 1 (empleant MinMaxScaler)

In [119…

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# 1. Selecció columnes per clustering:
# Excluim el 'target' (satisfaction): per tal de saber si s'agrupen
# També excloem l Unnamed : 0 (Columna índex)
dades = df.drop(['satisfaction','Unnamed: 0'], axis=1)

# 2. Normalització:
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(dades)

X_scaled_df = pd.DataFrame(X_scaled, columns=dades.columns)
```

```
# 3. Prova per veure si l'escalat ha funcionat.
print("--- Abans de l'escalat: ---")
print(df[['Flight Distance', 'Age','Gender','Class','Type of Travel

print("\n--- Després de l'escalat ---")
print(X_scaled_df[['Flight Distance', 'Age','Gender','Class','Type
```

```
--- Abans de l'escalat: ---
    Flight Distance  Age  Gender  Class  Type of Travel
0               160   52       0      0               0
1              2863   36       0      2               0
2               192   20       1      0               0
3              3377   44       1      2               0
4              1182   49       0      0               0
5               311   16       1      0               0
6              3987   77       0      2               0
7              2556   43       0      2               0
8               556   47       1      0               0
9              1744   46       0      2               0
10             1235   47       0      0               0
11              325   33       0      2               0
12             1009   46       0      2               0
13              451   60       0      2               0
14              925   52       0      2               0
15               83   50       1      0               1
16              728   31       0      0               0
17             1075   52       1      1               1
18             1927   43       0      0               1
19             3799   50       0      2               0

--- Després de l'escalat ---
    Flight Distance       Age  Gender  Class  Type of Travel
0          0.026050  0.576923     0.0    0.0             0.0
1          0.571890  0.371795     0.0    1.0             0.0
2          0.032512  0.166667     1.0    0.0             0.0
3          0.675687  0.474359     1.0    1.0             0.0
4          0.232431  0.538462     0.0    0.0             0.0
5          0.056543  0.115385     1.0    0.0             0.0
6          0.798869  0.897436     0.0    1.0             0.0
7          0.509895  0.461538     0.0    1.0             0.0
8          0.106018  0.512821     1.0    0.0             0.0
9          0.345921  0.500000     0.0    1.0             0.0
10         0.243134  0.512821     0.0    0.0             0.0
11         0.059370  0.333333     0.0    1.0             0.0
12         0.197496  0.500000     0.0    1.0             0.0
13         0.084814  0.679487     0.0    1.0             0.0
14         0.180533  0.576923     0.0    1.0             0.0
15         0.010501  0.551282     1.0    0.0             1.0
16         0.140751  0.307692     0.0    0.0             0.0
17         0.210824  0.576923     1.0    0.5             1.0
18         0.382876  0.461538     0.0    0.0             1.0
19         0.760905  0.551282     0.0    1.0             0.0
```

3.2.2 K-Means

Aplicam l'algoritme de K_MEANS per agrupar en diferents grups sense tenir

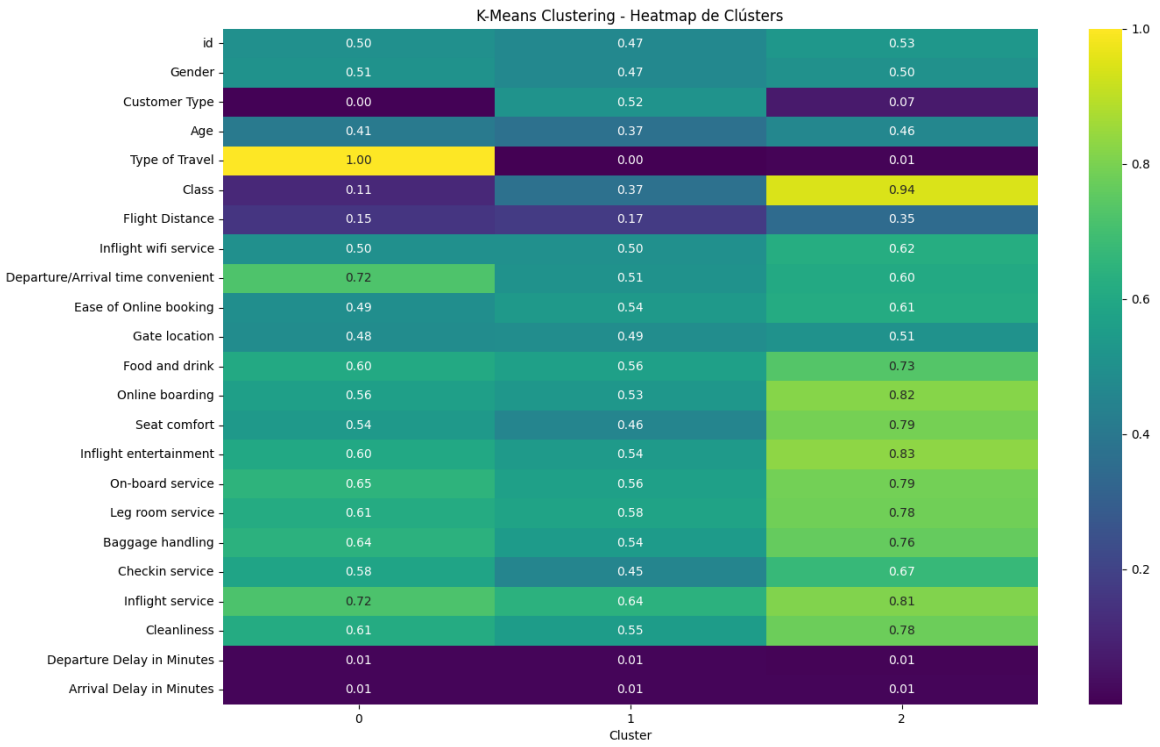
en compte l'objectiu (que en aquest cas es la satisfacció del clients).

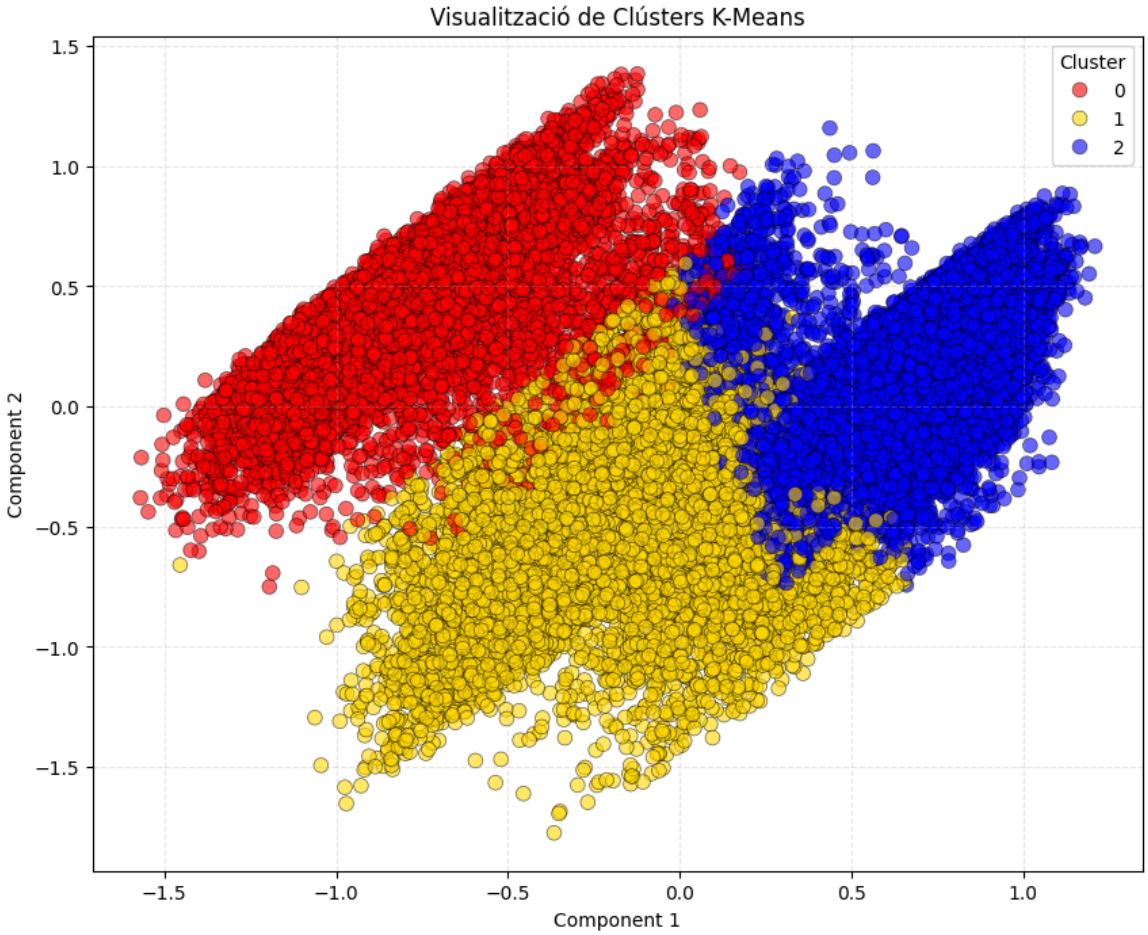
Hem triat que la K Òptima en aquest cas es 3, ja que es amb aquest nombre de grups com queda una major diferència entre els grups formats.

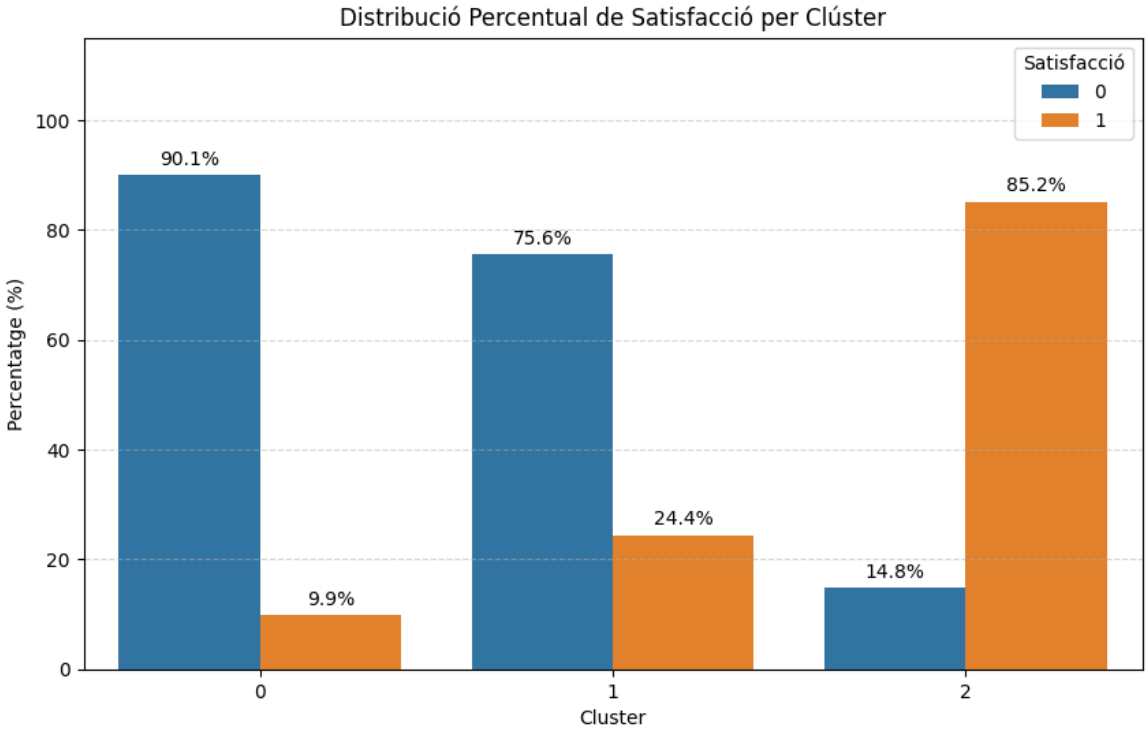
In [120…

```
import warnings
warnings.filterwarnings('ignore') # Silenciam warnings per netejar

# 1. Aplicam K-Means:
k_optim = 3
kmeans = KMeans(n_clusters=k_optim, random_state=42, n_init=10)

clusters_assign = kmeans.fit_predict(X_scaled)
df['Cluster'] = clusters_assign

# --- Intercanvi de grups ---
canvi_ordre = {0: 1, 1: 2, 2: 0} # No es necessari, pero per major
df['Cluster'] = df['Cluster'].map(canvi_ordre)

# 2. Heat Map:
df_para_grafico = pd.DataFrame(X_scaled, columns=dades.columns)
df_para_grafico['Cluster'] = df['Cluster'] # Le pegamos la etiqueta
cluster_summary_norm = df_para_grafico.groupby('Cluster').mean()

# 3. Plot Heatmap:
plt.figure(figsize=(15, 10))
sns.heatmap(cluster_summary_norm.T, annot=True, cmap='viridis', fmt
plt.title('K-Means Clustering - Heatmap de Clústers')
plt.show()

# 4. PCA per reducció de dimensions a 2D
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_scaled)

pca_df = pd.DataFrame(data=principal_components, columns=['PC1', 'P
pca_df['Cluster'] = df['Cluster']

colors = ['red', 'gold', 'blue'] # Colors personalitzats per als cl

plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='PC1',
    y='PC2',
    hue='Cluster',
    data=pca_df,
    palette=colors,  # ¡Aquí aplicamos tus colores!
    s=60,
    alpha=0.6,
    edgecolor='black',    # Borde negro para que el amarillo se vea
    linewidth=0.5
)

plt.title('Visualització de Clústers K-Means')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
```

```
plt.legend(title='Cluster')
plt.grid(True, linestyle='--', alpha=0.3)
plt.show()

# 5. Gráfic de barres amb percentatges de satisfacció per clúster
counts = df.groupby(['Cluster', 'satisfaction']).size().reset_index
total_por_cluster = counts.groupby('Cluster')['count'].transform('s
counts['percentage'] = (counts['count'] / total_por_cluster) * 100

plt.figure(figsize=(10, 6))
ax = sns.barplot(x='Cluster', y='percentage', hue='satisfaction', d

for container in ax.containers:
    ax.bar_label(container, fmt='%.1f%%', padding=3)

plt.title('Distribució Percentual de Satisfacció per Clúster')
plt.ylabel('Percentatge (%)')
plt.ylim(0, 115) # Espaciat adalt.
plt.legend(title='Satisfacció', loc='upper right')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.show()
```







Hem generat 3 gráfics que mostren informació molt útil:

1. Heat map - Clustering

Aquest primer gràfic mostra la mitjana de les dades normalitzades de cada un dels tres grups: variant el color de fons depenent del seu valor. Els grups resultants de l'algoritme de K-Means son 3:

1. *Type of Travel: Personal Travel*: aquest primer grup agrupa tots o quasi tots els viatgers que viatgen per motiu de NO feina, ja que l'atribut *Type of Travel* es 1.00.

2. *Type of Travel: Business Travel - Not Business Class*: el segon grup conté els passatgers que volen per motiu de feina pero que la classe del seu passatge es *ECO* o *ECO PLUS*.

3. *Type of Travel: Business Travel - Business Class*: aquest darrer grup agrupa la majoria de viatgers de viatje per feina i a més el tipus de seient del vol es Business class. També cal destacar que aquest grup conté la mitjana de distància considerablement més gran que els altres grups.

2. Clústers K-Means

Aquest gràfic mostra els 3 grups com a diferents "nùvols" ben diferenciats entre ells.

3. Distribució percentual de satisfacció.

Aquest histograma mostra els percentatges de clients satisfets (color taronja) vs els clients no satisfets/neutrals (color blau). Mostra clarament com els grups 0 i 1 contenten un baix índex de satisfacció de clients mentre que el grup 2 manté una satisfacció superior al 85%.

3.3 Entrenament i Ajust

Cerca dels millors *Hiperparàmetres* per a cada un dels models, entrenaments dels mateixos i obtenció dels resultats.

Preparació de les dades per els diferents experiments:

In [121…

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler

# 1. Definim X(Variables) i y(Objectiu)
X = df.drop(['id', 'satisfaction','Unnamed: 0'], axis=1, errors='ig
y = df['satisfaction'] # target

# 2. DIVISIÓ TRAIN / TEST (70% Entrenar, 30% Examen)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# 3. NORMALITZACIÓ ESPECÍFICA PER A CLASSIFICACIÓ
scaler_clas = StandardScaler()

X_train_scaled = scaler_clas.fit_transform(X_train)
X_test_scaled = scaler_clas.transform(X_test)
```

A l'anterior pas hem:

1. Separat les diferents característiques de les dades i l'objectiu.
2. Eliminat la dada de Unnamed: 0 que no aporta informació.
3. Dividit el conjunt de dades entre entrenament i test.
4. Normalitzat les dades per la clasificació.

Ara ja tenim les dades preparades per poder aplicar els diferents models.

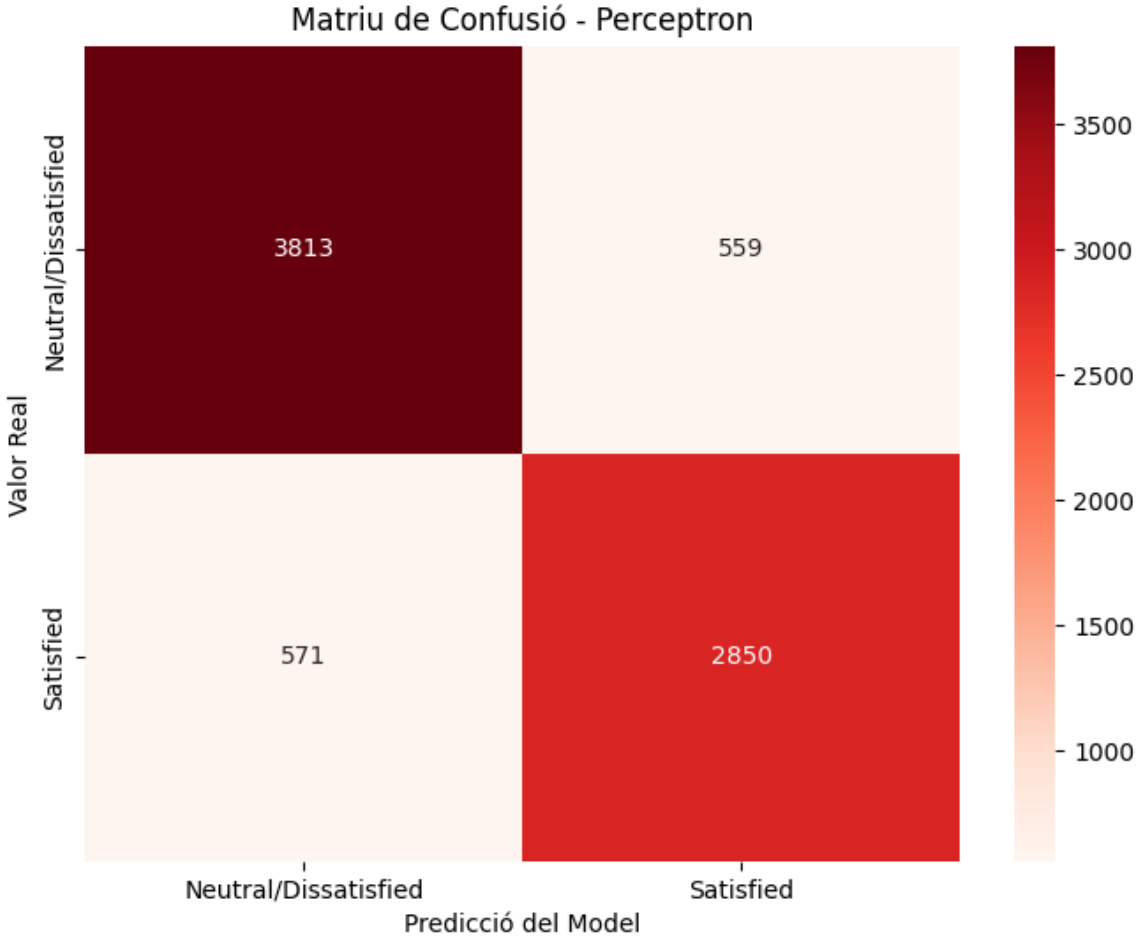
1. Perceptró

In [122…

```
from sklearn import linear_model
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score,classification_report, c

# Cerca de hiperparàmetres per al Perceptró

# 1. Definim possibles valors dels hiperparàmetres
param_grid = {
    'penalty': ['l2', 'l1', 'elasticnet', None],
    'alpha': [0.0001, 0.001, 0.01, 0.1, 1],
    'max_iter': [1000, 2000, 3000],
}

# 2. GridSearchCV
perceptron = linear_model.Perceptron(random_state=42)
grid = GridSearchCV(estimator=perceptron, param_grid=param_grid, cv

# 3. Entrenem (4 x 5 x 3)
grid.fit(X_train_scaled, y_train)

# 4. Resultats
best_model = grid.best_estimator_

print(f"\nHiperparàmetres selccionats: {grid.best_params_}")
print(f"Accuracy mitjana al Train: {grid.best_score_:.4f}")

# 5. Comprovació final amb el Test Set
y_pred_pt = best_model.predict(X_test_scaled)
acc_test = accuracy_score(y_test, y_pred_pt)
print(f"Accuracy final al Test Set: {acc_test:.4f}")

# 6. Resultats gràfics:
print("Informe de classificació (Perceptron):\n",classification_rep

cm = confusion_matrix(y_test, y_pred_pt)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds',
            xticklabels=['Neutral/Dissatisfied', 'Satisfied'],
            yticklabels=['Neutral/Dissatisfied', 'Satisfied'])

plt.title("Matriu de Confusió - Perceptron")
plt.xlabel("Predicció del Model")
plt.ylabel("Valor Real")
```

```
plt.show()
```

```
Hiperparàmetres selccionats: {'alpha': 0.0001, 'max_iter': 1000, 'pe
nalty': None}
Accuracy mitjana al Train: 0.8314
Accuracy final al Test Set: 0.8550
Informe de classificació (Perceptron):
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      4372
           1       0.84      0.83      0.83      3421

    accuracy                           0.85      7793
   macro avg       0.85      0.85      0.85      7793
weighted avg       0.85      0.85      0.85      7793
```



Selecció d'hiperparàmetres

Per tal de seleccionar la configuració més òptima del model de manera objectiva, hem implementat un procés d'optimització automàtica utilitzant *GridSearchCV de Scikit-learn* basat en validació creuada.

Aquesta tècnica ens permet explorar l'espai d'hiperparàmetres definits i seleccionar aquella combinació que ofereix la millor precisió mitjana sobre els conjunts de validació.

Els hiperparàmetres a avaluar són:

1. *penalty* (Tipus de regularització): Aquest paràmetre defineix el mètode utilitzat per penalitzar la complexitat del model i evitar el overfitting.

2. alpha (Intensitat de la regularització): Constant que multiplica el terme de penalització. Controla la "força" amb la qual restringim l'aprenentatge del model.

3. max_iter (Màxim d'iteracions): Com que el Perceptró és un algorisme iteratiu que només convergeix si les dades són linealment separables, aquest paràmetre actua com a condició de parada de seguretat.

Mètriques d'avaluació

1. *Accuracy*: Calculada com la proporció de prediccions correctes sobre el total ((TP + TN) / (TP + TN + FP + FN)). S'ha utilitzat com a mètrica base per tenir una visió general del rendiment.

2. *Matriu de Confusió*: Aquesta eina és fonamental perquè ens permet descompondre les prediccions en quatre categories: Veritables Positius, Veritables Negatius, Falsos Positius i Falsos Negatius. En el context de la satisfacció dels passatgers, la matriu ens permet veure si el model té tendència a "sobreestimar" la satisfacció (més Falsos Positius) o a "ignorar" passatgers satisfets (més Falsos Negatius) de forma gràfica i clara.

3. *Classification-Report*: Aquesta eina de aporta una visió detallada del rendiment del model. En lloc de donar-nos una única xifra, ens genera un informe complet amb les mètriques clau per a cada classe ("Neutral/Dissatisfied" i "Satisfied"): La seva utilitat principal en aquesta pràctica és assegurar-nos que el model no funcioni bé només globalment (accuracy), sinó que sigui robust i consistent en totes dues classes, evitant biaixos cap a la classe majoritària.

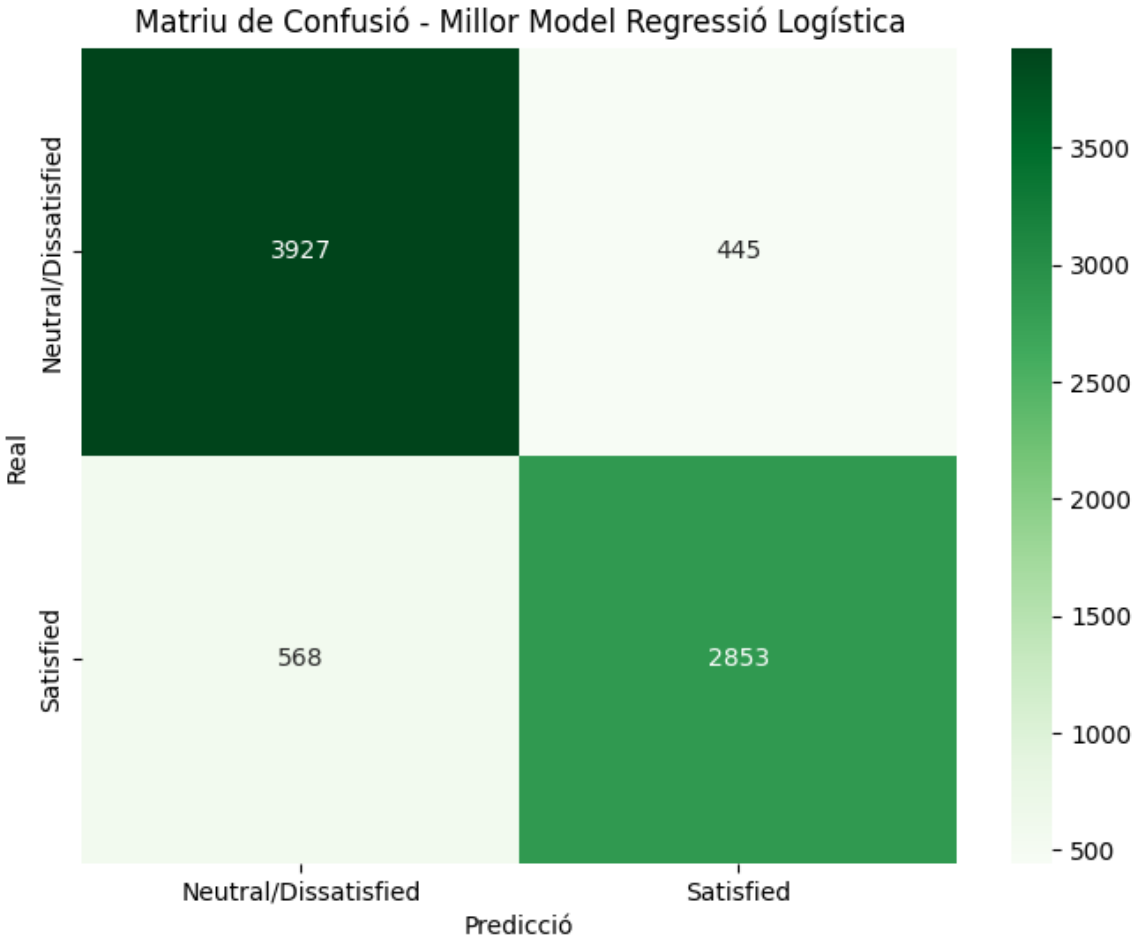
2. Regressió Logística

In [123…

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report,
import matplotlib.pyplot as plt
import seaborn as sns

# Definició de hiperparàmetres que provarem:
param_grid = [
    {
        'solver': ['lbfgs'],
        'penalty': ['l2', None],
        'max_iter': [100, 1000, 3000],
        'C': [0.1, 1, 10]
    },
    {
        'solver': ['liblinear'],
        'penalty': ['l1', 'l2'],
        'max_iter': [100, 1000, 3000],
        'C': [0.1, 1, 10]
    },
        {
        'solver': ['saga'],
        'penalty': ['l1', 'l2'],
        'max_iter': [100, 1000, 3000],
        'C': [0.1, 1, 10]
    }
]

log_reg = LogisticRegression(random_state=42)

# Millor combinació de hiperparàmetres amb GridSearchCV
grid_search = GridSearchCV(
    estimator=log_reg,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=1,          # Control d'error div/0.
)

grid_search.fit(X_train_scaled, y_train)

best_model = grid_search.best_estimator_
print(f"\Millors hiperparàmetres: {grid_search.best_params_}")
print(f"Millor precisió a cross-validation: {grid_search.best_score

y_pred_rl = best_model.predict(X_test_scaled)
acc_test_lr = accuracy_score(y_test, y_pred_rl)

print(f"\nPrecisió final al Test Set: {acc_test_lr:.4f}")
print("\nInforme de classificació:")
print(classification_report(y_test, y_pred_rl))

cm_lr = confusion_matrix(y_test, y_pred_rl)
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm_lr, annot=True, fmt='d', cmap='Greens',
            xticklabels=['Neutral/Dissatisfied', 'Satisfied'],
            yticklabels=['Neutral/Dissatisfied', 'Satisfied'])
plt.title(f"Matriu de Confusió - Millor Model Regressió Logística")
plt.xlabel("Predicció")
plt.ylabel("Real")
plt.show()
```

```
\Millors hiperparàmetres: {'C': 0.1, 'max_iter': 100, 'penalty': Non
e, 'solver': 'lbfgs'}
Millor precisió a cross-validation: 0.8706

Precisió final al Test Set: 0.8700

Informe de classificació:
              precision    recall  f1-score   support

           0       0.87      0.90      0.89      4372
           1       0.87      0.83      0.85      3421

    accuracy                           0.87      7793
   macro avg       0.87      0.87      0.87      7793
weighted avg       0.87      0.87      0.87      7793
```



Selecció d'hiperparàmetres

Per tal de fer la selecció dels valors dels millors hiperparàmetres hem fet ús

de l'eina GridSearhcV.

1. solver (Algorisme d'Optimització) lbfgs: robust per a la majoria de conjunts de dades, però està limitat a la regularització L2 o a cap penalització (None).
   liblinear: És ideal per a datasets més petits. saga: És especialment eficient per a grans conjunts de dades i és l'únic solver que suporta totes les penalitzacions (L1, L2 i ElasticNet).

2. penalty (Regularització: L1, L2, None) Per tal d'evitar el overfitting. l2 (Ridge): Penalitza la suma dels quadrats dels coeficients (θ). Manté tots els coeficients petits però diferents de zero. És l'estàndard per reduir la variància del model sense perdre informació. l1 (Lasso): Penalitza la suma del valor absolut dels coeficients. Té la propietat matemàtica de portar alguns coeficients exactament a zero. Això actua com una selecció automàtica de característiques, eliminant variables del dataset que no aporten informació (soroll) i simplificant el model. None: S'inclou per avaluar el rendiment del model "pur" (sense regularització) i determinar si les dades són suficients per generalitzar bé per si soles.

3. C (Invers de la força de regularització) Valors petits (ex: 0.1): Impliquen una regularització forta. El model tolera més errors en l'entrenament per buscar una solució més simple (coeficients més petits). Això redueix el risc d'overfitting però pot causar underfitting (alt biaix). Valors grans (ex: 10): Impliquen una regularització dèbil. El model intenta ajustar-se molt fidelment a les dades d'entrenament. Això redueix el biaix però augmenta dràsticament el risc de tenir alta variància (overfitting). Hem seleccionat: (0.1, 1, 10) per explorar sistemàticament l'espai entre aquests dos extrems.

4. max_iter (Iteracions màximes) Els algorismes basats en gradient necessiten iterar fins a convergir al mínim global de la funció de cost. 100: Valor per defecte. Pot ser insuficient si les dades no estan perfectament escalades o si la superfície de la funció de cost és complexa. 1000 i 3000: Augmentar aquest valor assegura que, fins i tot en configuracions difícils (com una regularització forta o moltes característiques), el solver tingui temps suficient per convergir i no es detingui prematurament abans de trobar la solució òptima.

Mètriques d'avaluació

Per avaluar els resultats de Regresió Logística hem empleat els mateixos mètodes que amb el Perceptró:

1. *Accuracy*: Calculada com la proporció de prediccions correctes sobre el total ((TP + TN) / (TP + TN + FP + FN)). S'ha utilitzat com a mètrica base

per tenir una visió general del rendiment.

2. *Matriu de Confusió*: Aquesta eina és fonamental perquè ens permet descompondre les prediccions en quatre categories: Veritables Positius, Veritables Negatius, Falsos Positius i Falsos Negatius. En el context de la satisfacció dels passatgers, la matriu ens permet veure si el model té tendència a "sobreestimar" la satisfacció (més Falsos Positius) o a "ignorar" passatgers satisfets (més Falsos Negatius) de forma gràfica i clara.

3. *Classification-Report*: Aquesta eina de aporta una visió detallada del rendiment del model. En lloc de donar-nos una única xifra, ens genera un informe complet amb les mètriques clau per a cada classe ("Neutral/Dissatisfied" i "Satisfied"): La seva utilitat principal en aquesta pràctica és assegurar-nos que el model no funcioni bé només globalment (accuracy), sinó que sigui robust i consistent en totes dues classes, evitant biaixos cap a la classe majoritària.

3. SVM

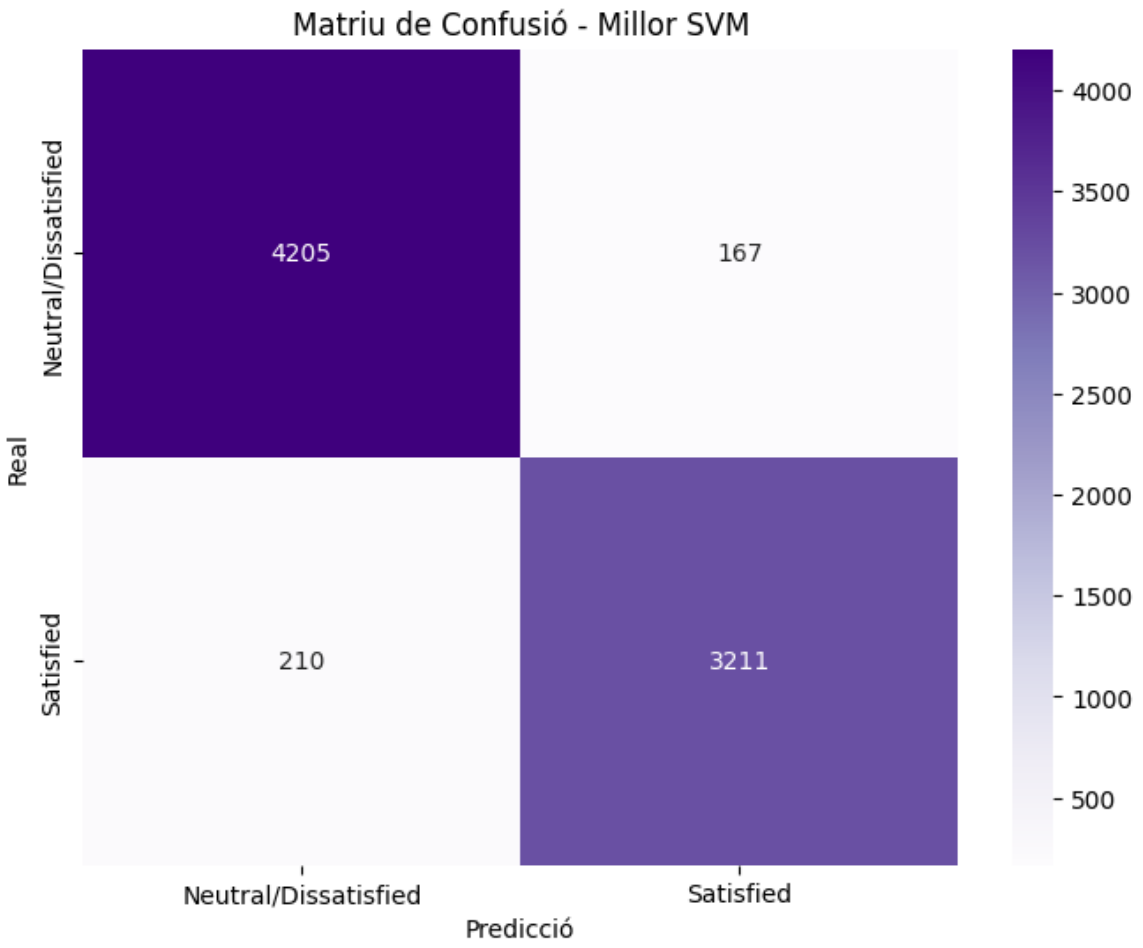
In [124…

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report,
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Hiperparàmetres a provar
param_grid = [
    {
        'kernel': ['rbf'],
        'C': [0.1, 1, 10],
        'gamma': ['scale', 0.1]
    },
    {
        'kernel': ['linear'],
        'C': [1, 10]
    },
    {
        'kernel': ['poly'],
        'degree': [2, 3],
        'C': [1]
    }
]

svm = SVC(random_state=42)

grid_search = GridSearchCV(
    estimator=svm,
    param_grid=param_grid,
    cv=3,
    scoring='accuracy',
    n_jobs=-1,         # Paral·lelització: per anar més ràpid. (No
    verbose=2
)

#  Entrenar
grid_search.fit(X_train_scaled, y_train)

#  Resultats
best_model = grid_search.best_estimator_
print(f"\nMillors hiperparàmetres: {grid_search.best_params_}")
print(f"Millor precisió: {grid_search.best_score_:.4f}")

y_pred_svm = best_model.predict(X_test_scaled)
acc_test_svm = accuracy_score(y_test, y_pred_svm)

print(f"\nPrecisió final al Test Set: {acc_test_svm:.4f}")
print("\nInforme de classificació:")
print(classification_report(y_test, y_pred_svm))

# Matriu de Confusió
cm_svm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(cm_svm, annot=True, fmt='d', cmap='Purples',
            xticklabels=['Neutral/Dissatisfied', 'Satisfied'],
            yticklabels=['Neutral/Dissatisfied', 'Satisfied'])
plt.title(f"Matriu de Confusió - Millor SVM")
plt.xlabel("Predicció")
plt.ylabel("Real")
plt.show()
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
[CV] END .....................C=1, gamma=scale, kernel=rbf; total
time=   2.3s
[CV] END .....................C=1, gamma=scale, kernel=rbf; total
time=   2.3s
[CV] END .....................C=1, gamma=scale, kernel=rbf; total
time=   2.3s
[CV] END ...................C=0.1, gamma=scale, kernel=rbf; total
time=   2.9s
[CV] END ...................C=0.1, gamma=scale, kernel=rbf; total
time=   3.0s
[CV] END .......................C=1, gamma=0.1, kernel=rbf; total
time=   2.9s
[CV] END ...................C=0.1, gamma=scale, kernel=rbf; total
time=   3.1s
[CV] END .....................C=0.1, gamma=0.1, kernel=rbf; total
time=   3.6s
[CV] END .....................C=0.1, gamma=0.1, kernel=rbf; total
time=   3.7s
[CV] END .....................C=0.1, gamma=0.1, kernel=rbf; total
time=   3.7s
[CV] END ....................C=10, gamma=scale, kernel=rbf; total
time=   2.2s
[CV] END .......................C=1, gamma=0.1, kernel=rbf; total
time=   2.6s
[CV] END .......................C=1, gamma=0.1, kernel=rbf; total
time=   2.7s
[CV] END ....................C=10, gamma=scale, kernel=rbf; total
time=   2.1s
[CV] END ....................C=10, gamma=scale, kernel=rbf; total
time=   2.1s
[CV] END .....................C=10, gamma=0.1, kernel=rbf; total
time=   2.5s
[CV] END .....................C=10, gamma=0.1, kernel=rbf; total
time=   2.7s
[CV] END .....................C=10, gamma=0.1, kernel=rbf; total
time=   2.7s
[CV] END .....................C=1, degree=2, kernel=poly; total
time=   2.3s
[CV] END .....................C=1, degree=2, kernel=poly; total
time=   2.2s
[CV] END .....................C=1, degree=2, kernel=poly; total
time=   2.2s
[CV] END ..............................C=1, kernel=linear; total
time=   4.3s
[CV] END ..............................C=1, kernel=linear; total
time=   4.4s
[CV] END .....................C=1, degree=3, kernel=poly; total
```

```
time=   2.2s
[CV] END ..............................C=1, kernel=linear; total
time=   4.3s
[CV] END ........................C=1, degree=3, kernel=poly; total
time=   1.8s
[CV] END ........................C=1, degree=3, kernel=poly; total
time=   1.6s
[CV] END .............................C=10, kernel=linear; total
time=   9.4s
[CV] END .............................C=10, kernel=linear; total
time=   9.4s
[CV] END .............................C=10, kernel=linear; total
time=   9.7s

Millors hiperparàmetres: {'C': 10, 'gamma': 'scale', 'kernel': 'rb
f'}
Millor precisió: 0.9453

Precisió final al Test Set: 0.9516

Informe de classificació:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      4372
           1       0.95      0.94      0.94      3421

    accuracy                           0.95      7793
   macro avg       0.95      0.95      0.95      7793
weighted avg       0.95      0.95      0.95      7793
```



Selecció d'hiperparàmetres

L'elecció de la graella de paràmetres (param_grid) per a l'algorisme SVM respon a la necessitat de trobar un equilibri entre la capacitat de generalització del model (evitar overfitting) i la seva precisió en les dades d'entrenament, així com d'explorar diferents transformacions de l'espai de característiques.

1. kernel (Nucli) Hem seleccionat tres tipus per cobrir diferents naturaleses de les dades: linear: Aquest nucli no projecta les dades, sinó que busca directament un hiperplà separador en l'espai original. És útil com a línia base (baseline) per comprovar si el problema de satisfacció és linealment separable, la qual cosa indicaria una solució senzilla i eficient. rbf: És el nucli més utilitzat per defecte per a problemes no lineals. Calcula la similitud basant-se en la distància euclidiana entre punts, definint la influència d'un exemple segons una funció gaussiana ($K(x,z)=\exp(-\gamma\|x-z\|$)

2. Poly: Intenta modelar fronteres de decisió corbes utilitzant polinomis de grau d ($K(x,z)=(\langle x,z\rangle+r)$

3. C (Paràmetre de Regularització) El paràmetre C controla l'equilibri (trade-off) entre aconseguir un marge de separació el més ample possible i classificar correctament els punts d'entrenament. Actua sobre la funció

de cost afegint una penalització per als errors de classificació.

4. gamma (Coeficient del Kernel) Aquest paràmetre és específic per als kernels no lineals com rbf, poly i sigmoid. Defineix l'abast de la influència d'un sol exemple d'entrenament.

5. degree (Grau del Polinomi) Únicament rellevant quan el kernel='poly'. Determina la complexitat de la funció polinòmica utilitzada per transformar l'espai. Hem provat graus 2 i 3 per permetre al model capturar relacions quadràtiques o cúbiques entre les característiques, sense augmentar excessivament la complexitat computacional ni el risc d'overfitting que comportarien graus superiors.

Mètriques del SVM

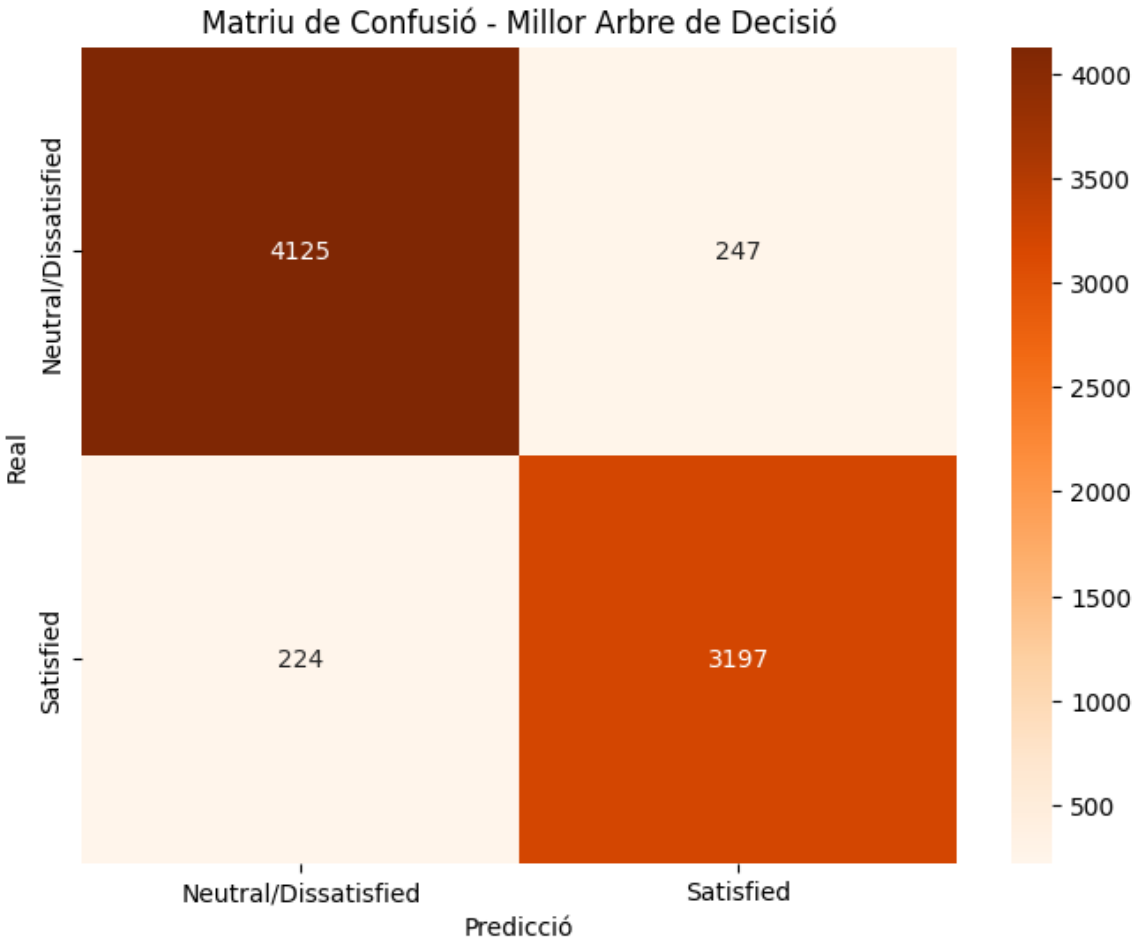
```
En aquest model també hem empleat la metrica de
Accuray per evaluar la millor selecció
d'hiperparàmetres.

Tornam mostrar gràficament els resultats mitjançant
la matriu de confusió i l'informe de classificació.
```

4. Arbres de decisió

In [125…

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report,
import matplotlib.pyplot as plt
import seaborn as sns

#  Hiperparàmetres a provar
param_grid = {
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 10, 50],
    'min_samples_leaf': [1, 5, 10]
}

dt_model = DecisionTreeClassifier(random_state=42)

grid_search_dt = GridSearchCV(
    estimator=dt_model,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

grid_search_dt.fit(X_train_scaled, y_train)

best_dt = grid_search_dt.best_estimator_
print(f"\nMillors hiperparàmetres: {grid_search_dt.best_params_}")
print(f"Millor precisió (CV): {grid_search_dt.best_score_:.4f}")

y_pred_dt = best_dt.predict(X_test_scaled)
acc_test_dt = accuracy_score(y_test, y_pred_dt)

print(f"\nPrecisió final al Test Set: {acc_test_dt:.4f}")
print("\nInforme de classificació:")
print(classification_report(y_test, y_pred_dt))

cm_dt = confusion_matrix(y_test, y_pred_dt)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Oranges',
            xticklabels=['Neutral/Dissatisfied', 'Satisfied'],
            yticklabels=['Neutral/Dissatisfied', 'Satisfied'])
plt.title(f"Matriu de Confusió - Millor Arbre de Decisió")
plt.xlabel("Predicció")
plt.ylabel("Real")
plt.show()
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits

Millors hiperparàmetres: {'max_depth': 10, 'min_samples_leaf': 1, 'm
in_samples_split': 2}
Millor precisió (CV): 0.9419

Precisió final al Test Set: 0.9396

Informe de classificació:
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      4372
           1       0.93      0.93      0.93      3421

    accuracy                           0.94      7793
   macro avg       0.94      0.94      0.94      7793
weighted avg       0.94      0.94      0.94      7793
```



Hiperparàmetres

1. max_depth (Profunditat Màxima) Controla la longitud màxima de l'arbre (nombre de nivells des de l'arrel fins a la fulla més llunyana). 10, 20, 30.

2. min_samples_split (Mínim de mostres per dividir) Defineix quantes mostres ha de tenir un node intern per poder ser dividit. Si un node té molt poques mostres, dividir-lo crearà regles molt específiques per a casos concrets (soroll). Augmentar aquest valor (ex: 50) obliga

l'algorisme a aprendre regles més generals que afectin grups més grans de passatgers, reduint la variància del model. 2, 5, 50.

3. min_samples_leaf (Mínim de mostres per fulla) Garanteix que cada fulla tingui un mínim d'observacions. 1, 5, 10.

Mètriques d'avaluació

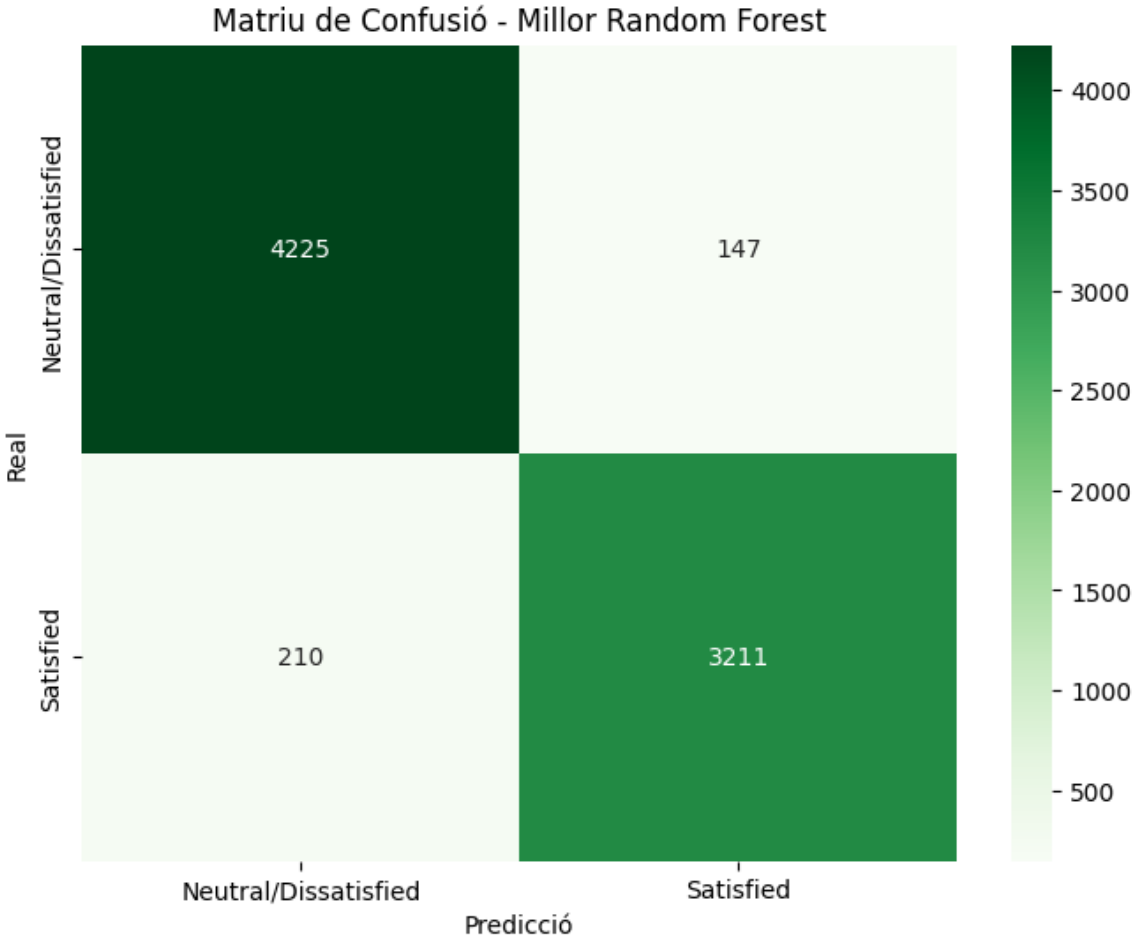
Per l'avaluació dels arbres de decisió hem empleat Accuracy per trobar el millor model i per tal de representar gràficament hem mostrat els resultats mitjançant la matriu de confusió i l'informe de classificació.

5. Boscos aleatoris

In [126…

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report,
import matplotlib.pyplot as plt
import seaborn as sns

# Hiperparàmetres:
param_grid = {
    'n_estimators': [50, 100, 200], # Nombre d'arbres al bosc
    'max_depth': [None, 10, 20], # control de la profunditat
    'min_samples_split': [2, 10],        # Regularització dels node
    'min_samples_leaf': [1, 5, 10]
}

rf_model = RandomForestClassifier(random_state=42, n_jobs=-1)

grid_search_rf = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

grid_search_rf.fit(X_train_scaled, y_train)

best_rf = grid_search_rf.best_estimator_
print(f"\nMillors hiperparàmetres: {grid_search_rf.best_params_}")
print(f"Millor precisió (CV): {grid_search_rf.best_score_:.4f}")

y_pred_rf = best_rf.predict(X_test_scaled)
acc_test_rf = accuracy_score(y_test, y_pred_rf)

print(f"\nPrecisió final al Test Set: {acc_test_rf:.4f}")
print("\nInforme de classificació (Random Forest):")
print(classification_report(y_test, y_pred_rf))

cm_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Greens',
            xticklabels=['Neutral/Dissatisfied', 'Satisfied'],
            yticklabels=['Neutral/Dissatisfied', 'Satisfied'])
plt.title(f"Matriu de Confusió - Millor Random Forest")
plt.xlabel("Predicció")
plt.ylabel("Real")
plt.show()
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits

Millors hiperparàmetres: {'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 200}
Millor precisió (CV): 0.9515

Precisió final al Test Set: 0.9542

Informe de classificació (Random Forest):
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      4372
           1       0.96      0.94      0.95      3421

    accuracy                           0.95      7793
   macro avg       0.95      0.95      0.95      7793
weighted avg       0.95      0.95      0.95      7793
```



Hiperparàmetres Random-Forest

Hem empleat GridSearchCV per tal de trobar els millors valors dels seleccionats hiperparàmetres: n estimators: 50, 100, 200: nombre d'arbres al bosc. max_depth: none, 10, 20: control de la profunditat. min_samples_split: 2,10: regularització de nodes interns. min_samples_leaf: 1,5, 10: quantitat minima de mostres per ser fulla.

Mètriques d'evaluació Random-Forest

De igual forma per tal d'evaluar el rendiment dels Boscos aleatoris hem empleat l'Accuracy. La matriu de confusió i l'informe de classificació permetràn en el següent punt evaluar el rendiment amb profunditat.

Discussió crítica dels resultats

Anàlisi de rendiments:

1. Perceptró

   - Precisió: 0.8529
   - Recall: 0.8526
   - F1-Score: 0.8527
   - Accuracy: 0.8550

   Aquest model ha mostrat un rendiment força decent i molt equilibrat, amb mètriques consistents al voltant del 85%. Tot i que els resultats són acceptables per a una línia base, és el model que obté les puntuacions més baixes.

   Això s'explica per la simplicitat de l'algorisme: el Perceptró és un classificador lineal que intenta separar les classes amb un hiperplà recte. Com que el comportament dels passatgers i la seva satisfacció tenen relacions complexes i no lineals que una línia recta no pot capturar perfectament, el model té un sostre de rendiment ("biaix alt") que no pot superar, a diferència de models més complexos com el Random Forest o l'SVM.

2. Regresió logística

   - Precisió: 0.8694
   - Recall: 0.8661
   - F1-Score: 0.8675
   - Accuracy: 0.8700

   Aquest model mostra una petita millora respecte al Perceptró, amb un no molt gran augment generalitzat en totes les mètriques (aproximadament un +1.5% en Accuracy i F1-Score).

   La raó d'aquesta millora és que, tot i ser també un classificador lineal, la Regressió Logística és probabilística. En lloc de fer un tall sec com el Perceptró, utilitza la funció sigmoide per estimar la probabilitat de pertinença a una classe. Això li permet gestionar millor les dades que no són perfectament separables. Tot i això, encara es queda per sota dels models no lineals (Arbres, SVM, RF).

3. SVM

   - Precisió: 0.9515
   - Recall: 0.9502
   - F1-Score: 0.9508

- Accuracy: 0.9516

Aquest model representa un salt qualitatiu molt important respecte als anteriors, amb una millora de més de 8 punts percentuals. De fet, es queda molt a prop del millor model (Random Forest).

L'explicació d'aquest èxit rau en la capacitat de l'SVM per gestionar fronteres de decisió no lineals mitjançant l'ús de kernels (en aquest cas RBF). A diferència del Perceptró o la Regressió Logística, l'SVM pot projectar les dades a dimensions superiors per separar complexament els passatgers satisfets dels insatisfets. Això li permet capturar matisos en el comportament dels clients que els models lineals ignoren completament.

4. Arbre de decisió

   - Precisió: 0.9384
   - Recall: 0.9390
   - F1-Score: 0.9387
   - Accuracy: 0.9396

   Aquest model aconsegueix un rendiment excel·lent, superant clarament els models lineals i situant-se molt a prop dels millors classificadors (SVM i Random Forest).

   L'èxit de l'Arbre de Decisió en aquest conjunt de dades s'explica per la seva naturalesa no lineal. A diferència del Perceptró o la Regressió Logística, l'arbre no intenta traçar una línia recta, sinó que divideix l'espai de característiques en regions rectangulars basant-se en regles lògiques. Això li permet capturar patrons complexos en el comportament dels passatgers.

   No obstant això, es queda lleugerament per devall del Random Forest. Això és degut al fet que els arbres individuals tendeixen a tenir una alta variància i poden patir d'overfitting (ajustant-se massa a les dades d'entrenament i perdent capacitat de generalització).

5. Boscos aleatoris

   - Precisió: 0.9544
   - Recall: 0.9525
   - F1-Score: 0.9534
   - Accuracy: 0.9542

   Aquest model és el que ha obtingut el millor rendimt.

   El Random Forest entrena centenars d'arbres de decisió diferents (cadascun amb un subconjunt de dades i característiques diferents) i combina els seus resultats. Això li permet:

Mantenir la capacitat de capturar patrons no lineals (com un arbre individual).

Eliminar el problema de l'alta variància (overfitting) que patia l'Arbre de Decisió simple. En fer la mitjana de molts arbres, els errors individuals es cancel·len, donant lloc a un model molt més robust i generalitzable.

Possibles fonts de biaix o errors

1. Subjectivitat de les Enquestes: Les característiques d'entrada són en gran part puntuacions subjectives (0-5) sobre serveis (menjar, seient, wifi). Font d'error: Un "3" per a un passatger exigent pot significar una mala experiència, mentre que per a un altre pot ser acceptable. El model assumeix que el valor numèric té el mateix significat per a tothom, cosa que introdueix soroll (noise) en l'aprenentatge.

2. Biaix de Selecció: Encara que no podem controlar com es van recollir les dades, és important mencionar que les enquestes de satisfacció solen ser contestades per usuaris amb opinions extremes (molt contents o molt enfadats). Els passatgers amb una experiència "normal" sovint no responen (el mateix passa amb les enquestes per evaluar al professorat de la UIB), la qual cosa pot fer que el model no generalitzi bé per al passatger mitjà.

3. Desequilibri de Classes: És habitual que en les dades reals una de les dues categories target sigui més freqüent que l'altra. Si el model s'entrena amb aquest desequilibri sense corregir-lo, tendirà a desenvolupar un biaix estadístic a favor de la classe majoritària per tal de maximitzar la precisió global (Accuracy). Això provoca que el sistema tendesqui a triar la predominant, fallant sistemàticament en la detecció del grup minoritari.

Comparació entre models:

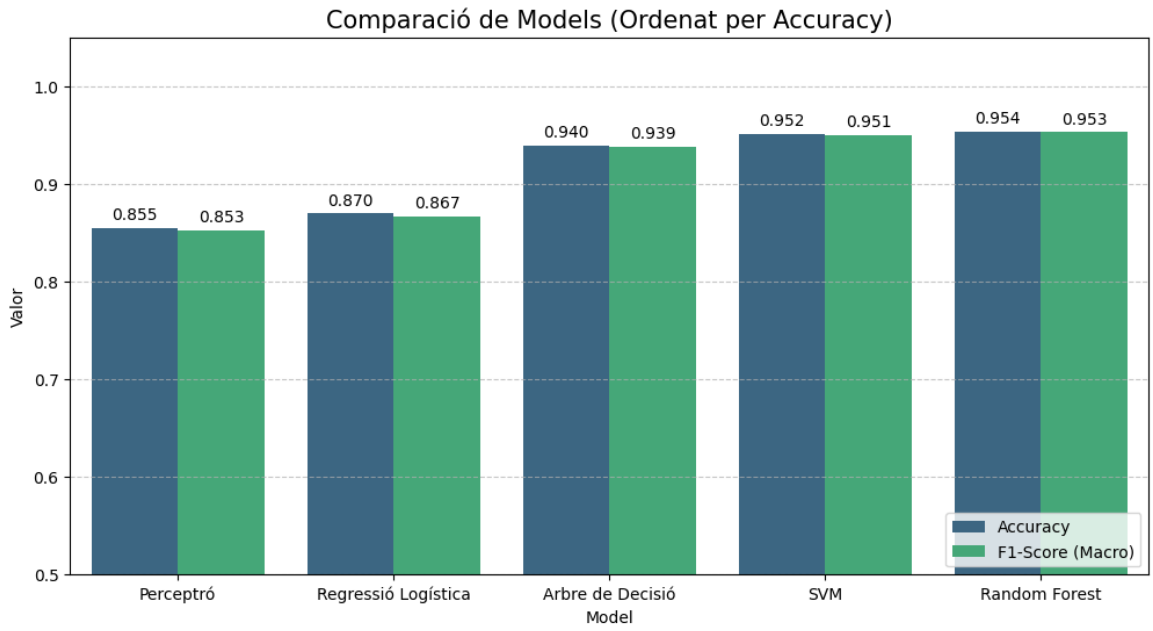
In [127…

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report

model_predictions = {
    'Perceptró': y_pred_pt,
    'Regressió Logística': y_pred_rl,
    'SVM': y_pred_svm,
    'Arbre de Decisió': y_pred_dt,
    'Random Forest': y_pred_rf
}

metrics_data = []
```

```
for model_name, preds in model_predictions.items():
    report = classification_report(y_test, preds, output_dict=True)
    metrics_data.append({
        'Model': model_name,
        'Accuracy': report['accuracy'],
        'Precision (Macro)': report['macro avg']['precision'],
        'Recall (Macro)': report['macro avg']['recall'],
        'F1-Score (Macro)': report['macro avg']['f1-score']
    })


comparison_df = pd.DataFrame(metrics_data).set_index('Model')
comparison_df = comparison_df.sort_values(by='Accuracy', ascending=
styled_df = comparison_df.style.background_gradient(cmap='Greens').

print("Taula comparativa de models (Ordenada per Accuracy):")
display(styled_df)

# Gràfic:
plt.figure(figsize=(12, 6))

df_melted = comparison_df.reset_index().melt(id_vars='Model',
                                             value_vars=['Accuracy'
                                             var_name='Métrica', va

ax = sns.barplot(data=df_melted, x='Model', y='Valor', hue='Métrica

for container in ax.containers:
    ax.bar_label(container, fmt='%.3f', padding=3, fontsize=10)

plt.title('Comparació de Models (Ordenat per Accuracy)', fontsize=1
plt.ylim(0.5, 1.05)
plt.legend(loc='lower right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
Taula comparativa de models (Ordenada per Accuracy):
```

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1-Score (Macro)
Perceptró	0.8550	0.8529	0.8526	0.8527
Regressió Logística	0.8700	0.8694	0.8661	0.8675
Arbre de Decisió	0.9396	0.9384	0.9390	0.9387
SVM	0.9516	0.9515	0.9502	0.9508
Random Forest	0.9542	0.9544	0.9525	0.9534



Les taules de valors esmentades anteriorment comparen els resultats obtinguts per cada model (amb els millors hiperparàmetres seleccionats).

El rendiment ha estat molt encertat per els 5 diferents models: essent el Perceptró el menys precís i el Random Forest el que més èxit ha tengut.

Com es poden veure a les gràfiques podem treure dues importants conclussions:

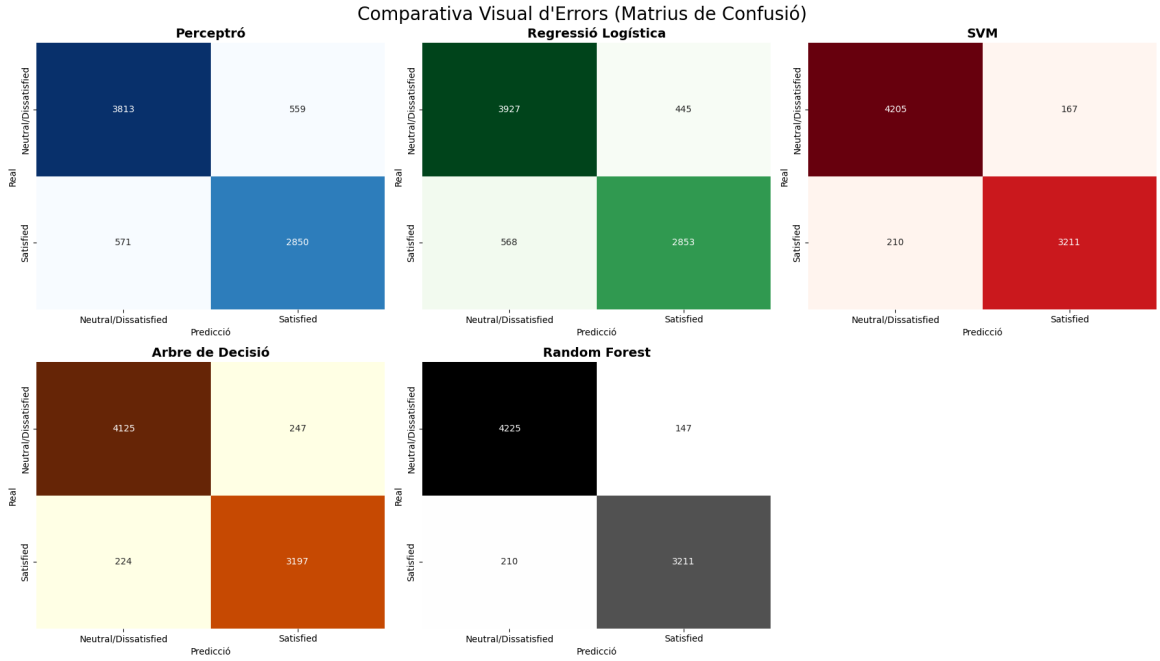
1. Dos grups diferenciats: El perceptró i regressió logística han tengut un rendiment parescut amb una mitjana de Accurancy de 0.8625. Per altre bande els arbres de decisió, SVM i el Random-Forest han estat en voltant al 0.95 de Accurancy.

2. Alt èxit a tots els models: Tot i que tenim dos models amb un percentatge d'èxit més baix respecte als altres tres, la tasa d'encert ha estat molt elevada per a tots els 5 casos.

In [130…

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
import math

preds_dict = {
    'Perceptró': y_pred_pt,
    'Regressió Logística': y_pred_rl,
    'SVM': y_pred_svm,
    'Arbre de Decisió': y_pred_dt,
    'Random Forest': y_pred_rf
}

models_names = list(preds_dict.keys())
n_models = len(models_names)
cols = 3
```

```
rows = math.ceil(n_models / cols)

fig, axes = plt.subplots(rows, cols, figsize=(18, 5 * rows))
axes = axes.flatten()

colors_cmap = ['Blues', 'Greens', 'Reds', 'YlOrBr', 'Greys'] # Colo

for i, model_name in enumerate(models_names):
    preds = preds_dict[model_name]
    cm = confusion_matrix(y_test, preds)
    current_cmap = colors_cmap[i % len(colors_cmap)]

    sns.heatmap(cm, annot=True, fmt='d', cmap=current_cmap, cbar=Fa
                xticklabels=['Neutral/Dissatisfied', 'Satisfied'],
                yticklabels=['Neutral/Dissatisfied', 'Satisfied'])

    axes[i].set_title(f"{model_name}", fontsize=14, fontweight='bol
    axes[i].set_xlabel('Predicció')
    axes[i].set_ylabel('Real')

for j in range(i + 1, len(axes)):
    axes[j].axis('off')

plt.tight_layout()
plt.suptitle("Comparativa Visual d'Errors (Matrius de Confusió)", f
plt.show()
```



Aclariments

1. La versió de Python empleada per l'execució del codi es *Python 3.9.6^.
2. S'ha empleat ús de la Inteligència Artificial per la realització de codi de gràfics i de codi Markdown: GOOGLE GEMINI (Versió gener 2026).