```
# Datos del alumno

**Nombre**:Natalia


**Apellidos**:Camarano


**Grupo:**A2
```

Datos del alumno

Nombre:Natalia

Apellidos:Camarano

Grupo:A2

Recordamos que el objetivo de este proyecto es predecir si un trabajador va a abandonar la empresa o no (variable `left`).

El objetivo de este sprint es entrenar 3 árboles de decisión:

- Uno sin aplicar ningún criterio de poda (sin hiperparámetros)
- Otros dos especificando diferentes hiperparámetros.

Para este caso continuaremos con los datos empleados en secciones anteriores.

Lo primero de todo será importar las librerías necesarias.

```
import os
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (accuracy_score,
                             f1_score,
                             roc_auc_score,
                             roc_curve,
                             confusion_matrix,
                             classification_report)
from sklearn.model_selection import (cross_val_score,
                                     GridSearchCV,
                                     RandomizedSearchCV,
                                     learning_curve,
                                     validation_curve,
                                     train_test_split)
from sklearn.pipeline import make_pipeline
from sklearn.utils import resample
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')

%matplotlib inline
```

Y las específicas para este sprint.

```
pip install dtreeviz
```

```
Collecting dtreeviz
  Downloading dtreeviz-2.2.2-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: graphviz>=0.9 in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (0.20.3)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (2.1.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (1.26.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (1.3.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (3.7.1)
Requirement already satisfied: colour in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (0.1.5)
Requirement already satisfied: pytest in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (7.4.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->dtreeviz) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->dtreeviz) (2024.1)
Requirement already satisfied: iniconfig in /usr/local/lib/python3.10/dist-packages (from pytest->dtreeviz) (2.0.0)
Requirement already satisfied: pluggy<2.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from pytest->dtreeviz) (1.5.0)
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /usr/local/lib/python3.10/dist-packages (from pytest->dtreeviz) (1.2.2)
Requirement already satisfied: tomli>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pytest->dtreeviz) (2.0.1)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->dtreeviz) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->dtreeviz) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->dtreeviz) (3.5.0
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->dtreeviz
Downloading dtreeviz-2.2.2-py3-none-any.whl (91 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 91.8/91.8 kB 5.1 MB/s eta 0:00:00
Installing collected packages: dtreeviz
Successfully installed dtreeviz-2.2.2
```

```
from sklearn import tree
import dtreeviz
from sklearn.tree import export_text
```

Cargamos las matrices de train y test que hemos calculado en el sprint 1.

```
from google.colab import drive
drive.mount('/content/drive')
#https://drive.google.com/file/d/1S7DtU2HEFXkqyJp_RcILNSgEsY3huIL9/view?usp=sharing
```

```
Mounted at /content/drive
```

```
##Descarga manual: https://drive.google.com/file/d/1S7DtU2HEFXkqyJp_RcILNSgEsY3huIL9/view?usp=sharing
##Descarga manual: https://drive.google.com/file/d/1O1BRGwSd81TOOpaQvoo5BHDsrPEruz93/view?usp=sharing
##Descarga manual: https://drive.google.com/file/d/1u3yI5L_2YI77uF_WT9ysIzw8dydVlQSh/view?usp=sharing
##Descarga manual: https://drive.google.com/file/d/1eWo_m2gbihSuUQeOhH8I8Q0uudluQKBT/view?usp=sharing
```

```
#Descargamos los ficheros de Google Drive (si lo ejecutais en un entorno diferente a Google Colab, tenéis que intalar previamente wget e
!wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1S7DtU2HEFXkqyJp_RcILNSgEsY3huIL9' -O 'y_train.npy'
!wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1O1BRGwSd81TOOpaQvoo5BHDsrPEruz93' -O 'y_test.npy'
!wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1u3yI5L_2YI77uF_WT9ysIzw8dydVlQSh' -O 'X_train.npy'
!wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1eWo_m2gbihSuUQeOhH8I8Q0uudluQKBT' -O 'X_test.npy'
```

```
--2024-09-17 21:50:56--  https://drive.google.com/uc?export=download&id=1S7DtU2HEFXkqyJp_RcILNSgEsY3huIL9
Resolving drive.google.com (drive.google.com)... 172.253.122.113, 172.253.122.101, 172.253.122.100, ...
Connecting to drive.google.com (drive.google.com)|172.253.122.113|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1S7DtU2HEFXkqyJp_RcILNSgEsY3huIL9&export=download [following]
--2024-09-17 21:50:56--  https://drive.usercontent.google.com/download?id=1S7DtU2HEFXkqyJp_RcILNSgEsY3huIL9&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 142.251.16.132, 2607:f8b0:4004:c17::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|142.251.16.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 96120 (94K) [application/octet-stream]
Saving to: ‘y_train.npy’

y_train.npy         100%[===================>]  93.87K  --.-KB/s    in 0.01s

2024-09-17 21:50:59 (9.64 MB/s) - ‘y_train.npy’ saved [96120/96120]

--2024-09-17 21:50:59--  https://drive.google.com/uc?export=download&id=1O1BRGwSd81TOOpaQvoo5BHDsrPEruz93
Resolving drive.google.com (drive.google.com)... 172.253.122.113, 172.253.122.101, 172.253.122.100, ...
Connecting to drive.google.com (drive.google.com)|172.253.122.113|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1O1BRGwSd81TOOpaQvoo5BHDsrPEruz93&export=download [following]
--2024-09-17 21:50:59--  https://drive.usercontent.google.com/download?id=1O1BRGwSd81TOOpaQvoo5BHDsrPEruz93&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 142.251.16.132, 2607:f8b0:4004:c17::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|142.251.16.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24128 (24K) [application/octet-stream]
Saving to: ‘y_test.npy’

y_test.npy          100%[===================>]  23.56K  --.-KB/s    in 0s

2024-09-17 21:51:01 (63.9 MB/s) - ‘y_test.npy’ saved [24128/24128]

--2024-09-17 21:51:01--  https://drive.google.com/uc?export=download&id=1u3yI5L_2YI77uF_WT9ysIzw8dydVlQSh
Resolving drive.google.com (drive.google.com)... 172.253.122.113, 172.253.122.101, 172.253.122.100, ...
Connecting to drive.google.com (drive.google.com)|172.253.122.113|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1u3yI5L_2YI77uF_WT9ysIzw8dydVlQSh&export=download [following]
--2024-09-17 21:51:01--  https://drive.usercontent.google.com/download?id=1u3yI5L_2YI77uF_WT9ysIzw8dydVlQSh&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 142.251.16.132, 2607:f8b0:4004:c17::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|142.251.16.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1631992 (1.6M) [application/octet-stream]
Saving to: ‘X_train.npy’

X_train.npy         100%[===================>]   1.56M  --.-KB/s    in 0.07s

2024-09-17 21:51:03 (22.9 MB/s) - ‘X_train.npy’ saved [1631992/1631992]

--2024-09-17 21:51:03--  https://drive.google.com/uc?export=download&id=1eWo_m2gbihSuUQeOhH8I8Q0uudluQKBT
Resolving drive.google.com (drive.google.com)... 172.253.122.113, 172.253.122.101, 172.253.122.100, ...
Connecting to drive.google.com (drive.google.com)|172.253.122.113|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1eWo_m2gbihSuUQeOhH8I8Q0uudluQKBT&export=download [following]
```

```
--2024-09-17 21:51:03--  https://drive.usercontent.google.com/download?id=1eWo_m2gbihSuUQeOhH8I8Q0uudluQKBT&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 142.251.16.132, 2607:f8b0:4004:c17::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|142.251.16.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 408128 (399K) [application/octet-stream]
```

Verificamos que las dimensiones se corresponden con la partición de 80% de los datos para el conjunto de train y 20% de los datos para test.

```
X_train, X_test,y_train,y_test = np.load("X_train.npy"),np.load("X_test.npy"),np.load("y_train.npy"),np.load("y_test.npy")

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(11999, 17)
(3000, 17)
(11999,)
(3000,)
```

CUESTION 1 - Árbol con hiperparámetros por defecto

```
# Generamos un árbol de decisión para clasificación
clf_tree = tree.DecisionTreeClassifier(class_weight="balanced")

# El parámetro class_weight='balanced' emplea los valores de y para ajustar automáticamente los pesos de forma inversamente proporcional
```

```
# Entrenamos el modelo con los datos de entrenamiento
clf_tree.fit(X_train, y_train)
# Evaluar el rendimiento del modelo con los datos de prueba
y_pred = clf_tree.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print(f"F1 Score: {f1_score(y_test, y_pred, average='weighted')}")
#print(f"ROC AUC Score: {roc_auc_score(y_test, clf_tree.predict_proba(X_test), multi_class='ovr')}")
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
```

```
Accuracy: 0.984
F1 Score: 0.9840753478526832
Confusion Matrix:
[[2252   34]
 [  14  700]]
```

```
# Obtenemos las predicciones para el conjunto de test
y_pred = clf_tree.predict(X_test)

# Imprimimos las predicciones obtenidas
print("Predicciones para el conjunto de test:")
print(y_pred)
```

```
Predicciones para el conjunto de test:
[0 0 0 ... 0 0 1]
```

```
# ¿Cual es la profundidad del árbol entrenado?
tree_depth = clf_tree.tree_.max_depth

# Imprimir la profundidad del árbol
print(f"La profundidad del árbol entrenado es: {tree_depth}")
```

```
La profundidad del árbol entrenado es: 21
```

```
# Evaluar la capacidad predictiva del modelo
# Hacemos las predicciones para el conjunto de test
y_pred = clf_tree.predict(X_test)

# Imprimimos las métricas de evaluación del modelo
print("Evaluación de la capacidad predictiva del modelo:")

# 1. Accuracy (precisión general)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# 2. F1-Score (para medir precisión y recall balanceados)
f1 = f1_score(y_test, y_pred, average='weighted')
print(f"F1 Score: {f1:.4f}")

# 3. ROC AUC Score (Área bajo la curva ROC)
```

```
#roc_auc = roc_auc_score(y_test, clf_tree.predict_proba(X_test), multi_class='ovr')
#print(f"ROC AUC Score: {roc_auc:.4f}")

# 4. Matriz de confusión
conf_matrix = confusion_matrix(y_test, y_pred)
print(f"Confusion Matrix:\n{conf_matrix}")

# 5. Informe de clasificación (precisión, recall, F1 para cada clase)
class_report = classification_report(y_test, y_pred)
print(f"Classification Report:\n{class_report}")
```

```
Evaluación de la capacidad predictiva del modelo:
Accuracy: 0.9840
F1 Score: 0.9841
Confusion Matrix:
[[2252   34]
 [  14  700]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2286
           1       0.95      0.98      0.97       714

    accuracy                           0.98      3000
   macro avg       0.97      0.98      0.98      3000
weighted avg       0.98      0.98      0.98      3000
```

CUESTION 2 - Árbol modificando algún hiperparámetro

Sugerencia: ¿Qué ocurre si limitamos la profundidad del árbol?

```
# CUESTIÓN 2 - Árbol modificando el hiperparámetro 'max_depth'
# Limitamos la profundidad del árbol a 3 niveles, por ejemplo
clf_tree_limited = tree.DecisionTreeClassifier(max_depth=3, class_weight="balanced")

# Entrenamos el modelo con los datos de entrenamiento
clf_tree_limited.fit(X_train, y_train)

# Obtenemos las predicciones para el conjunto de test
y_pred_limited = clf_tree_limited.predict(X_test)

# Imprimimos las métricas de evaluación para el árbol limitado
print("Evaluación del árbol con profundidad limitada (max_depth=3):")

# 1. Accuracy (precisión general)
accuracy_limited = accuracy_score(y_test, y_pred_limited)
print(f"Accuracy: {accuracy_limited:.4f}")

# 2. F1-Score (para medir precisión y recall balanceados)
f1_limited = f1_score(y_test, y_pred_limited, average='weighted')
print(f"F1 Score: {f1_limited:.4f}")

# 3. ROC AUC Score (Área bajo la curva ROC)
#roc_auc_limited = roc_auc_score(y_test, clf_tree_limited.predict_proba(X_test), multi_class='ovr')
#print(f"ROC AUC Score: {roc_auc_limited:.4f}")

# 4. Matriz de confusión
conf_matrix_limited = confusion_matrix(y_test, y_pred_limited)
print(f"Confusion Matrix:\n{conf_matrix_limited}")

# 5. Informe de clasificación (precisión, recall, F1 para cada clase)
class_report_limited = classification_report(y_test, y_pred_limited)
print(f"Classification Report:\n{class_report_limited}")

# Comparar la profundidad del árbol original vs el árbol limitado
print(f"Profundidad del árbol original: {clf_tree.tree_.max_depth}")
print(f"Profundidad del árbol limitado: {clf_tree_limited.tree_.max_depth}")
```

```
Evaluación del árbol con profundidad limitada (max_depth=3):
Accuracy: 0.9097
F1 Score: 0.9129
Confusion Matrix:
[[2060  226]
 [  45  669]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.90      0.94      2286
           1       0.75      0.94      0.83       714
```

```
    accuracy                           0.91      3000
   macro avg       0.86      0.92      0.88      3000
weighted avg       0.92      0.91      0.91      3000

Profundidad del árbol original: 21
Profundidad del árbol limitado: 3
```

CUESTION 3 - Árbol modificando algún hiperparámetro

Sugerencia: ¿Y si cambiamos el criterio para evaluar la impureza de las particiones?

```
# Cambiamos el criterio a "entropy"
clf_tree_entropy = tree.DecisionTreeClassifier(criterion="entropy", class_weight="balanced")

# Entrenamos el modelo con los datos de entrenamiento
clf_tree_entropy.fit(X_train, y_train)

# Obtenemos las predicciones para el conjunto de test
y_pred_entropy = clf_tree_entropy.predict(X_test)

# Imprimimos las métricas de evaluación para el árbol con criterio "entropy"
print("Evaluación del árbol con criterio de impureza 'entropy':")

# 1. Accuracy (precisión general)
accuracy_entropy = accuracy_score(y_test, y_pred_entropy)
print(f"Accuracy: {accuracy_entropy:.4f}")

# 2. F1-Score (para medir precisión y recall balanceados)
f1_entropy = f1_score(y_test, y_pred_entropy, average='weighted')
print(f"F1 Score: {f1_entropy:.4f}")

# 3. ROC AUC Score (Área bajo la curva ROC)
#roc_auc_entropy = roc_auc_score(y_test, clf_tree_entropy.predict_proba(X_test), multi_class='ovr')
#print(f"ROC AUC Score: {roc_auc_entropy:.4f}")

# 4. Matriz de confusión
conf_matrix_entropy = confusion_matrix(y_test, y_pred_entropy)
print(f"Confusion Matrix:\n{conf_matrix_entropy}")

# 5. Informe de clasificación (precisión, recall, F1 para cada clase)
class_report_entropy = classification_report(y_test, y_pred_entropy)
print(f"Classification Report:\n{class_report_entropy}")

# Comparar la profundidad del árbol original vs el árbol con criterio "entropy"
print(f"Profundidad del árbol original (gini): {clf_tree.tree_.max_depth}")
print(f"Profundidad del árbol con criterio 'entropy': {clf_tree_entropy.tree_.max_depth}")
```

```
Evaluación del árbol con criterio de impureza 'entropy':
Accuracy: 0.9853
F1 Score: 0.9854
Confusion Matrix:
[[2255   31]
 [  13  701]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2286
           1       0.96      0.98      0.97       714

    accuracy                           0.99      3000
   macro avg       0.98      0.98      0.98      3000
weighted avg       0.99      0.99      0.99      3000

Profundidad del árbol original (gini): 21
Profundidad del árbol con criterio 'entropy': 23
```

CUESTION 4 - ¿Qué modelo tiene mayor poder predictivo?

Comparar los diferentes árboles entrenados empleando las métricas más adecuadas para un problema desblanaceado como el nuestro.

```
# Comparación de los diferentes árboles entrenados
# Modelos: clf_tree (gini), clf_tree_entropy (entropy), clf_tree_limited (profundidad limitada)

# 1. Evaluar el árbol con criterio Gini (por defecto)
```

```
y_pred_gini = clf_tree.predict(X_test)
f1_gini = f1_score(y_test, y_pred_gini, average='weighted')
#roc_auc_gini = roc_auc_score(y_test, clf_tree.predict_proba(X_test), multi_class='ovr')
roc_auc_gini = roc_auc_score(y_test, clf_tree.predict_proba(X_test)[:, 1])
print("Árbol con criterio Gini (por defecto):")
print(f"F1 Score: {f1_gini:.4f}")
print(f"ROC AUC Score: {roc_auc_gini:.4f}")
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred_gini)}")
print(f"Classification Report:\n{classification_report(y_test, y_pred_gini)}")
print("\n")

# 2. Evaluar el árbol con criterio Entropy
y_pred_entropy = clf_tree_entropy.predict(X_test)
f1_entropy = f1_score(y_test, y_pred_entropy, average='weighted')
#roc_auc_entropy = roc_auc_score(y_test, clf_tree_entropy.predict_proba(X_test), multi_class='ovr')
roc_auc_entropy = roc_auc_score(y_test, clf_tree_entropy.predict_proba(X_test)[:, 1])
print("Árbol con criterio Entropy:")
print(f"F1 Score: {f1_entropy:.4f}")
print(f"ROC AUC Score: {roc_auc_entropy:.4f}")
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred_entropy)}")
print(f"Classification Report:\n{classification_report(y_test, y_pred_entropy)}")
print("\n")

# 3. Evaluar el árbol con profundidad limitada
y_pred_limited = clf_tree_limited.predict(X_test)
f1_limited = f1_score(y_test, y_pred_limited, average='weighted')
#roc_auc_limited = roc_auc_score(y_test, clf_tree_limited.predict_proba(X_test), multi_class='ovr')
roc_auc_limited = roc_auc_score(y_test, clf_tree_limited.predict_proba(X_test)[:, 1])
print("Árbol con profundidad limitada (max_depth=3):")
print(f"F1 Score: {f1_limited:.4f}")
print(f"ROC AUC Score: {roc_auc_limited:.4f}")
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred_limited)}")
print(f"Classification Report:\n{classification_report(y_test, y_pred_limited)}")
print("\n")

# Comparación final de las métricas F1-Score y ROC AUC Score
print("Comparación de F1-Score y ROC AUC Score:")
print(f"Árbol con Gini:\t\t F1: {f1_gini:.4f}, ROC AUC: {roc_auc_gini:.4f}")
print(f"Árbol con Entropy:\t F1: {f1_entropy:.4f}, ROC AUC: {roc_auc_entropy:.4f}")
print(f"Árbol con Prof. Limitada:\t F1: {f1_limited:.4f}, ROC AUC: {roc_auc_limited:.4f}")
```

```
Árbol con criterio Gini (por defecto):
F1 Score: 0.9841
ROC AUC Score: 0.9828
Confusion Matrix:
[[2252   34]
 [  14  700]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2286
           1       0.95      0.98      0.97       714

    accuracy                           0.98      3000
   macro avg       0.97      0.98      0.98      3000
weighted avg       0.98      0.98      0.98      3000



Árbol con criterio Entropy:
F1 Score: 0.9854
ROC AUC Score: 0.9841
Confusion Matrix:
[[2255   31]
 [  13  701]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2286
           1       0.96      0.98      0.97       714

    accuracy                           0.99      3000
   macro avg       0.98      0.98      0.98      3000
weighted avg       0.99      0.99      0.99      3000



Árbol con profundidad limitada (max_depth=3):
F1 Score: 0.9129
ROC AUC Score: 0.9291
Confusion Matrix:
[[2060  226]
 [  45  669]]
Classification Report:
              precision    recall  f1-score   support
           0       0.98      0.90      0.94      2286
           1       0.75      0.94      0.83       714

    accuracy                           0.91      3000
   macro avg       0.86      0.92      0.88      3000
weighted avg       0.92      0.91      0.91      3000



Comparación de F1-Score y ROC AUC Score:
Árbol con Gini:          F1: 0.9841, ROC AUC: 0.9828
Árbol con Entropy:       F1: 0.9854, ROC AUC: 0.9841
Árbol con Prof. Limitada:        F1: 0.9129, ROC AUC: 0.9291
```

CUESTION 5 - Representación del árbol

En este apartado se valorará más la capacidad de analizar la información que nos transmite la representación visual y en formato literal (export_text()), que el propio gráfico en sí. Por lo tanto, si el árbol que has elegido es demasiado profundo, te recomiendo emplear para este apartado uno más simple, limitando la profundidad, para que la representación y análisis sea legible y entendible.

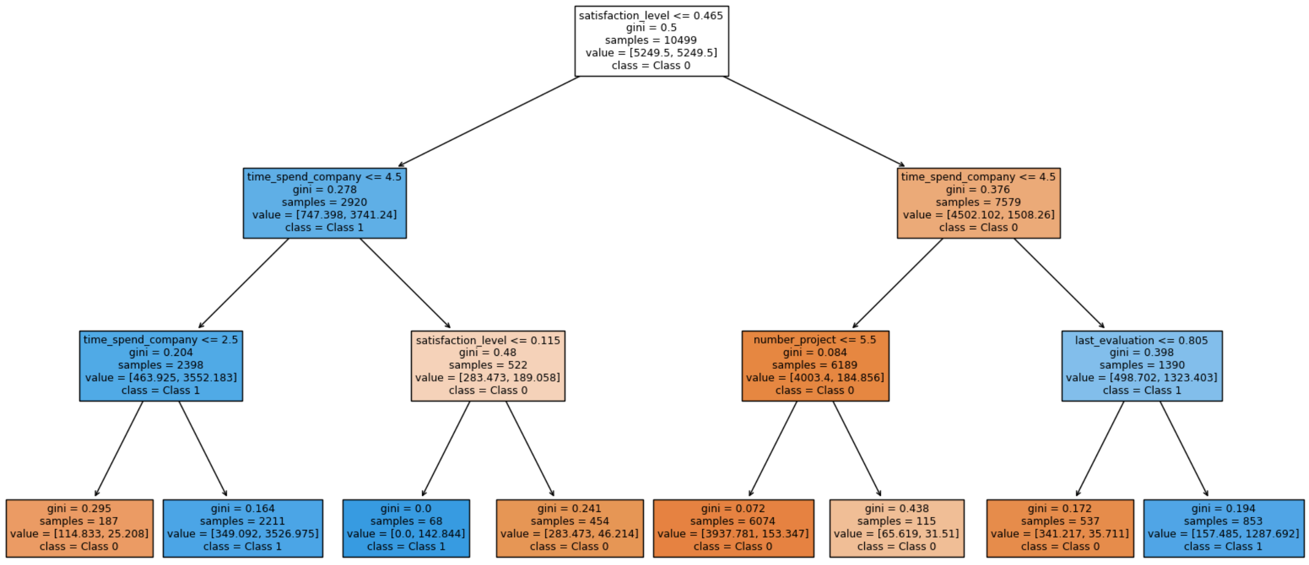
```
import pandas as pd

# Cargar los datos
df = pd.read_csv('Rotacion_empleados.csv')

# Mostrar las primeras filas para ver las columnas y datos
print(df.head())

# 1. Renombrar la variable 'sales' a 'department'
df.rename(columns={'sales': 'department'}, inplace=True)

# 2. Recodificar la variable 'salary' con los valores: low=0, medium=1, high=2
df['salary'] = df['salary'].map({'low': 0, 'medium': 1, 'high': 2})

# 3. Convertir la variable 'department' a variables dummies
df = pd.get_dummies(df, columns=['department'], drop_first=True)

# Mostrar el DataFrame transformado
print(df.head())

# Separar características (X) y etiqueta (y)
X = df.drop('left', axis=1)  # Asumiendo que la etiqueta se llama 'target'
y = df['left']

# Dividir en conjunto de entrenamiento y prueba
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, random_state=42)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
   satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                0.38             0.53               2                   157
1                0.80             0.86               5                   262
2                0.11             0.88               7                   272
3                0.72             0.87               5                   223
4                0.37             0.52               2                   159

   time_spend_company  Work_accident  left  promotion_last_5years  sales  \
0                   3              0     1                      0  sales
1                   6              0     1                      0  sales
2                   4              0     1                      0  sales
3                   5              0     1                      0  sales
4                   3              0     1                      0  sales

   salary
0     low
1  medium
2  medium
3     low
4     low
   satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                0.38             0.53               2                   157
1                0.80             0.86               5                   262
2                0.11             0.88               7                   272
3                0.72             0.87               5                   223
4                0.37             0.52               2                   159
```

```
   time_spend_company  Work_accident  left  promotion_last_5years  salary  \
0                   3              0     1                      0       0
1                   6              0     1                      0       1
2                   4              0     1                      0       1
3                   5              0     1                      0       0
4                   3              0     1                      0       0

   department_RandD  department_accounting  department_hr  \
0             False                  False          False
1             False                  False          False
2             False                  False          False
3             False                  False          False
4             False                  False          False

   department_management  department_marketing  department_product_mng  \
0                  False                 False                   False
1                  False                 False                   False
2                  False                 False                   False
3                  False                 False                   False
4                  False                 False                   False

   department_sales  department_support  department_technical
0              True               False                 False
1              True               False                 False
2              True               False                 False
3              True               False                 False
4              True               False                 False
(10499, 17)
(4500, 17)
(10499,)
(4500,)
```

```
"""
from sklearn import tree
from sklearn.tree import export_text
import matplotlib.pyplot as plt
"""
# Supongamos que clf_tree_limited es el modelo de árbol de decisión entrenado con profundidad limitada
# Si no lo has hecho aún, aquí tienes cómo entrenarlo:
clf_tree_limited = tree.DecisionTreeClassifier(max_depth=3, class_weight="balanced")
clf_tree_limited.fit(X_train, y_train)

# 1. Exportar el árbol a formato texto
tree_rules = export_text(clf_tree_limited, feature_names=list(X.columns))
print("Árbol en formato texto:")
print(tree_rules)

# 2. Representar gráficamente el árbol
fig, ax = plt.subplots(figsize=(20, 10))  # Ajustar el tamaño para una mejor visualización
tree.plot_tree(clf_tree_limited, feature_names=X.columns, class_names=['Class 0', 'Class 1'], filled=True, ax=ax)
plt.show()
```

```
Árbol en formato texto:
|--- satisfaction_level <= 0.47
|   |--- time_spend_company <= 4.50
|   |   |--- time_spend_company <= 2.50
|   |   |   |--- class: 0
|   |   |--- time_spend_company >  2.50
|   |   |   |--- class: 1
|   |--- time_spend_company >  4.50
|   |   |--- satisfaction_level <= 0.11
|   |   |   |--- class: 1
|   |   |--- satisfaction_level >  0.11
|   |   |   |--- class: 0
|--- satisfaction_level >  0.47
|   |--- time_spend_company <= 4.50
|   |   |--- number_project <= 5.50
|   |   |   |--- class: 0
|   |   |--- number_project >  5.50
|   |   |   |--- class: 0
|   |--- time_spend_company >  4.50
|   |   |--- last_evaluation <= 0.81
|   |   |   |--- class: 0
|   |   |--- last_evaluation >  0.81
|   |   |   |--- class: 1
```



Empieza a programar o a crear código con IA.

NOTA: las matrices de train y test que hemos cargado al inicio del sprint no contienen información de los nombres de cada atributo. Esto se puede solventar cargando de nuevo los datos con los que comenzábamos el sprint 1.

Recordad que en el sprint 1 para obtener las matrices de train y test, previamente realizamos unas transformaciones de los datos. Recordamos los pasos:

- Renombrar la variable `sales` a `department`.
- Recodificar la variable `salary` con el siguiente criterio para conservar el sentido de orden que contiene la variable: low=0, medium=1, high=2.
- Transformar la otra variable de tipo *object* (`department`) a numérica codificando sus categorías como variables dummies.

Por lo tanto, a continuación os dejo un enlace para descargar el fichero original al que, para poder usarlo en este sprint, tendréis que hacer previamente las transformaciones descritas previamente.

```
#Descarga manual: https://drive.google.com/file/d/1ZwgdmI525zInDh1SQVm7FZt8kWxfrvOK/view?usp=sharing
#Descargamos los ficheros de Google Drive
!wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1ZwgdmI525zInDh1SQVm7FZt8kWxfrvOK' -O 'Rotacion_empleados.c
```

```
--2024-09-17 22:20:24--  https://drive.google.com/uc?export=download&id=1ZwgdmI525zInDh1SQVm7FZt8kWxfrvOK
Resolving drive.google.com (drive.google.com)... 142.251.16.138, 142.251.16.139, 142.251.16.100, ...
Connecting to drive.google.com (drive.google.com)|142.251.16.138|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1ZwgdmI525zInDh1SQVm7FZt8kWxfrvOK&export=download [following]
--2024-09-17 22:20:24--  https://drive.usercontent.google.com/download?id=1ZwgdmI525zInDh1SQVm7FZt8kWxfrvOK&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 142.251.179.132, 2607:f8b0:4004:c17::84
```