Quiz Arbol De Decision.

Integrantes: Juan David Echeverria

1. Cargar y explorar dataset

```
import numpy as np
import pandas as pd
```

```
# cargar y ver las 10 primeras lineas
df = pd.read_csv("telco_custom.csv")
df.head(10)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic
7	6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL

10 rows × 21 columns

2. Revisar estructura y resumen

```
# informacion general
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
```

```
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

```
# estadisticas descriptivas
df.describe(include="all")
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Interne
count	7043	7043	7043.000000	7043	7043	7043.000000	7043	7043	
unique	7043	2	NaN	2	2	NaN	2	3	
top	3186-AJIEK	Male	NaN	No	No	NaN	Yes	No	
freq	1	3555	NaN	3641	4933	NaN	6361	3390	
mean	NaN	NaN	0.162147	NaN	NaN	32.371149	NaN	NaN	
std	NaN	NaN	0.368612	NaN	NaN	24.559481	NaN	NaN	
min	NaN	NaN	0.000000	NaN	NaN	0.000000	NaN	NaN	
25%	NaN	NaN	0.000000	NaN	NaN	9.000000	NaN	NaN	
50%	NaN	NaN	0.000000	NaN	NaN	29.000000	NaN	NaN	
75%	NaN	NaN	0.000000	NaN	NaN	55.000000	NaN	NaN	
max	NaN	NaN	1.000000	NaN	NaN	72.000000	NaN	NaN	

11 rows × 21 columns

4. Ver distribucion de la variable

```
# contar cuantos clientes se fueron o no
df["Churn"].value_counts()
```

Churn	count
No	5174
Yes	1869

dtype: int64

```
# ver proporcion porcentual
df["Churn"].value_counts(normalize=True) * 100
```

Churn	proportion
No	73.463013
Yes	26.536987

dtype: float64

5. Limpieza y preparacion de datos

```
# verificar valores nulos
df.isnull().sum()
```

	0
customerID	0
gender	0
SeniorCitizen	0
Partner	0
Dependents	0
tenure	0
PhoneService	0
MultipleLines	0
InternetService	0
OnlineSecurity	0
OnlineBackup	0
DeviceProtection	0
TechSupport	0
StreamingTV	0
StreamingMovies	0
Contract	0
PaperlessBilling	0
PaymentMethod	0
MonthlyCharges	0
TotalCharges	0
Churn	0

dtype: int64

```
# codificar variables categoricas
from sklearn.preprocessing import LabelEncoder
df_encoded = df.copy()
```

```
# verificar que variables son de tipo texto (object)
for col in df_encoded.columns:
  if df_encoded[col].dtype == "object":
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df_encoded[col])
```

6. Separar variables de entrada (x) y salida (y)

```
x = df_encoded.drop("Churn", axis=1) # las demas variables predictoras
y = df_encoded["Churn"] # variable a predecir
```

7. Dividir en 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=0.2, random_state=42)
```

8. Crear y entgrenar el arbol de decision

```
from sklearn.tree import DecisionTreeClassifier
modelo = DecisionTreeClassifier(max_depth=5)
modelo.fit(x_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)
```

9. Realizar predicciones

```
# ver prediccion comparada con la prueba
y_pred = modelo.predict(x_test)
y_pred_df = pd.DataFrame({"y_test": y_test, "y_pred": y_pred})
y_pred_df.head(10)
```

	y_test	y_pred
185	1	1
2715	0	0
3825	0	0
1807	1	1
132	0	0
1263	1	0
3732	0	0
1672	0	0
811	1	0
2526	1	0

Pasos siguientes: Generar código con y_pred_df | New interactive sheet

10. Evaluar el modelo

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# calcular precision
precision = accuracy_score(y_test, y_pred)
print("Precision: ", precision)
```

```
Precision:  0.794889992902768
```

```
# mostrar matriz de confusion
matriz = confusion_matrix(y_test, y_pred)
print("Matriz de confusion: \n", matriz)
```

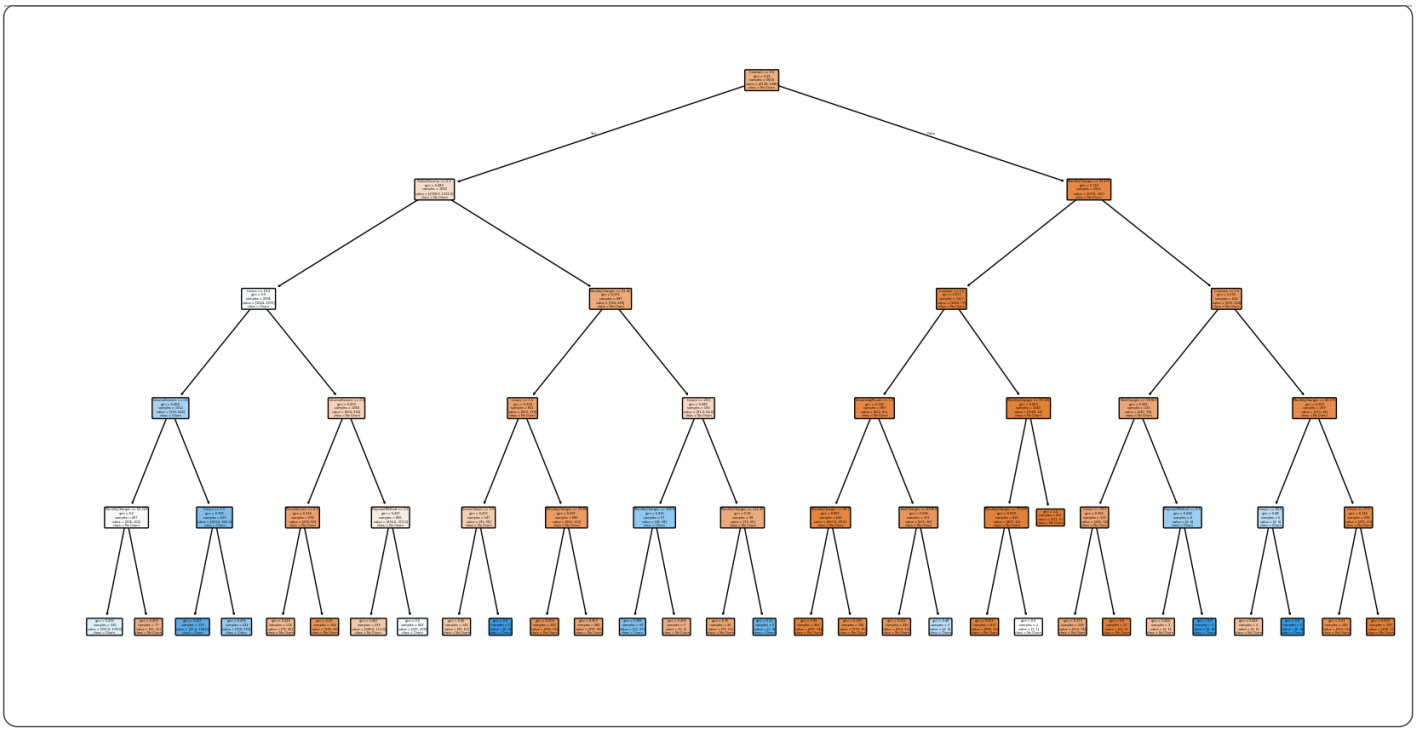
```
Matriz de confusion:
 [[884 152]
 [137 236]]
```

```
# reporte detallado
reporte = classification_report(y_test, y_pred)
print("Reporte de clasificacion: \n", reporte)
```

```
Reporte de clasificacion:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86      1036
           1       0.61      0.63      0.62       373

    accuracy                           0.79      1409
   macro avg       0.74      0.74      0.74      1409
weighted avg       0.80      0.79      0.80      1409
```

11. Visualizar el arbol

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(20,10))
plot_tree(modelo,
          feature_names = x.columns,
          class_names = ["No Churn", "Churn"],
          filled=True,
          rounded=True)
plt.show()
```



12. Importancia de variables

```
importancia = pd.Series(modelo.feature_importances_, index=x.columns)
importancia.sort_values(ascending=False).head(10)
```

	0
Contract	0.536974
tenure	0.140369
OnlineSecurity	0.139314
InternetService	0.077209
MonthlyCharges	0.070560
PaymentMethod	0.013093
TotalCharges	0.006422
SeniorCitizen	0.005853
PhoneService	0.005123
StreamingMovies	0.005083

dtype: float64

13. Guardar resultados opcional

```
df_result = x_test.copy()
df_result["Real"] = y_test
df_result["Predicho"] = y_pred
df_result.to_csv("resultados_churn.csv", index=False)
```

14. Incluir un nuevo resultado

1. crear un nuevo registro con las columnas reales

```
nuevo_cliente = {
 'gender': 'Male',
 'SeniorCitizen': 0,
 'Partner': 'No',
 'Dependents': 'No',
 'tenure': 6,
 'PhoneService': 'Yes',
 'MultipleLines': 'No',
 'InternetService': 'Fiber optic',
 'OnlineSecurity': 'No',
 'OnlineBackup': 'No',
 'DeviceProtection': 'Yes',
 'TechSupport': 'No',
 'StreamingTV': 'Yes',
 'StreamingMovies': 'Yes',
 'Contract': 'Month-to-month',
 'PaperlessBilling': 'Yes',
 'PaymentMethod': 'Electronic check',
 'MonthlyCharges': 78.9,
 'TotalCharges': 475.0
}
nuevo_df = pd.DataFrame([nuevo_cliente])
# codificar el dataset
for col in nuevo_df.columns:
  if nuevo_df[col].dtype == "object":
    le = LabelEncoder()
    le.fit(df[col].astype(str))
    nuevo_df[col] = le.fit_transform(nuevo_df[col].astype(str))
```

3. Alinear con las columnas del modelo predecir

```
nuevo_df = nuevo_df.reindex(columns=x.columns, fill_value=0)
prediccion = modelo.predict(nuevo_df)
probabilidad = modelo.predict_proba(nuevo_df)
```

4. Mostrar resultado legible

```
if prediccion[0] == 1:
  print("El modelo predice que este cliente PROBABLEMENTE se dara de baja")
else:
  print("El modelo predice que este cliente permancera en la compañia")
print("Probabiolidad de baja: ", np.round(probabilidad[0][1]*100, 2), "%")
```

```
El modelo predice que este cliente permancera en la compañia
Probabiolidad de baja:  28.57 %
```

REFLEXION FINAL

- ¿Qué atributos del cliente parecen más críticos para la predicción?
  - Segun la ikmportancia de las variables la que mas influye en la prediccion es "Contract", el tipo de contrato probablemente los contratos con mayor duracion tienen menor probabilidad de darse de baja a comparacion de contratos por mes. Luego la seguridad y la antiguedad tambien parecen ser un factor donde la presencia o ausencia de seguridad y el tiempo que ha estado un cliente en la empresa influye.
- ¿Cómo cambiarían el resultado si el contrato fuera anual, o si tuviera soporte técnico activo?

- Dado que el contrato es la variable mas importante el aumentar este tiempo podria reducir significativamente el porcentaje de baja del cliente.
- En este caso soporte tecnico no se evidencia una importancia es decir que no es muy relevante en el modelo de prediccion, en este caso y el entrenamiento y deicisiones que se tomaron no cambiaria el resultado.

• ¿Qué acciones podría implementar la empresa para evitar la baja de este cliente?

- Segun el modelo y las variables con mayor importancia seria: ofrecer un contrato con un largo plazo, proporcionar servicios de seguridad y de internet y como la antiguedad tambien es fundamental mi recomendacion es mantener al cliente satisfecho con una mejor experiencia.