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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


from google.colab import files
uploaded = files.upload()
```

Choose Files bank.csv

- **bank.csv**(text/csv) - 461474 bytes, last modified: 1/8/2025 - 100% done

```
Saving bank.csv to bank (1).csv
```

```
df = pd.read_csv("bank.csv", sep=';')
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutco
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unkno
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	fail
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	fail
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unkno
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unkno

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Check for missing values
print(df.isnull().sum())

# Encode categorical features
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = le.fit_transform(df[col])

# Display the cleaned and encoded data
df.head()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	10	1	0	0	1787	0	0	0	19	10	79	1	-1	0	3	0
1	33	7	1	1	0	4789	1	1	0	11	8	220	1	339	4	0	0
2	35	4	2	2	0	1350	1	0	0	16	0	185	1	330	1	0	0
3	30	4	1	2	0	1476	1	1	2	3	6	199	4	-1	0	3	0
4	59	1	1	1	0	0	1	0	2	5	8	226	1	-1	0	3	0

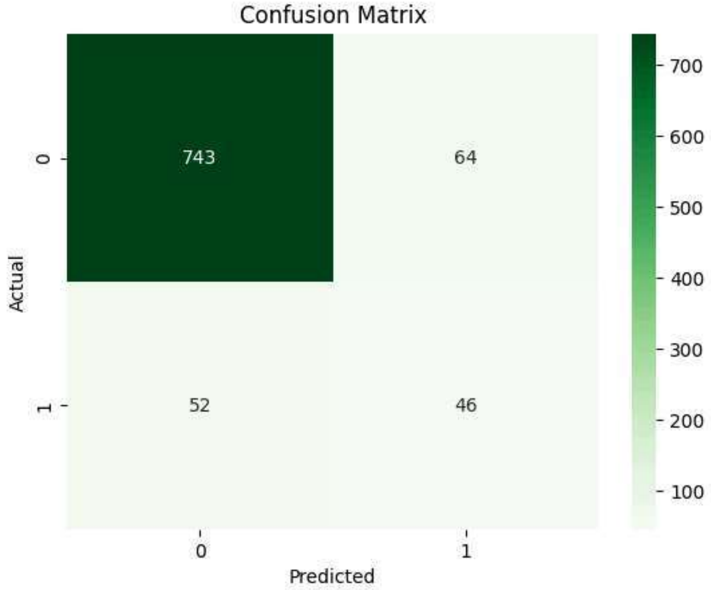
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Split the data into features (X) and target (y)
X = df.drop('y', axis=1)
y = df['y']

# Train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the Decision Tree model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

from sklearn.tree import DecisionTreeClassifier

# Train the decision tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
# Evaluation metrics
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
Accuracy: 0.8718232044198895

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.92      0.93       807
           1       0.42      0.47      0.44        98

    accuracy                           0.87       905
   macro avg       0.68      0.70      0.68       905
weighted avg       0.88      0.87      0.88       905
```



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

plt.figure(figsize=(20,10))
plot_tree(dt_model, feature_names=X.columns, class_names=['No', 'Yes'], filled=True, rounded=True)
plt.title("Decision Tree Classifier for Bank Marketing Dataset")
plt.show()
```