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

Choose Files 2 files

- **bank.csv**(text/csv) - 461474 bytes, last modified: 4/29/2025 - 100% done
- **bank-full.csv**(text/csv) - 4610348 bytes, last modified: 4/29/2025 - 100% done

```
Saving bank.csv to bank.csv
Saving bank-full.csv to bank-full.csv
```

```
import pandas as pd

# Load the CSV file
df = pd.read_csv('bank.csv', sep=';')  # separator is semicolon
df = pd.read_csv('bank-full.csv', sep=';')  # separator is semicolon
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	m
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	
1	44	technician	single	secondary	no	29	yes	no	unknown	5	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	
4	33	unknown	single	unknown	no	1	no	no	unknown	5	

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

```
# Check for missing values and basic info
df.info()
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

	0
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

```
# Convert categorical columns using one-hot encoding
df_encoded = pd.get_dummies(df, drop_first=True)

# Check the updated dataframe
df_encoded.head()
```

	age	balance	day	duration	campaign	pdays	previous	job_blue-collar	job_entrepreneur	...
0	58	2143	5	261	1	-1	0	False	False	
1	44	29	5	151	1	-1	0	False	False	
2	33	2	5	76	1	-1	0	False	True	
3	47	1506	5	92	1	-1	0	True	False	
4	33	1	5	198	1	-1	0	False	False	

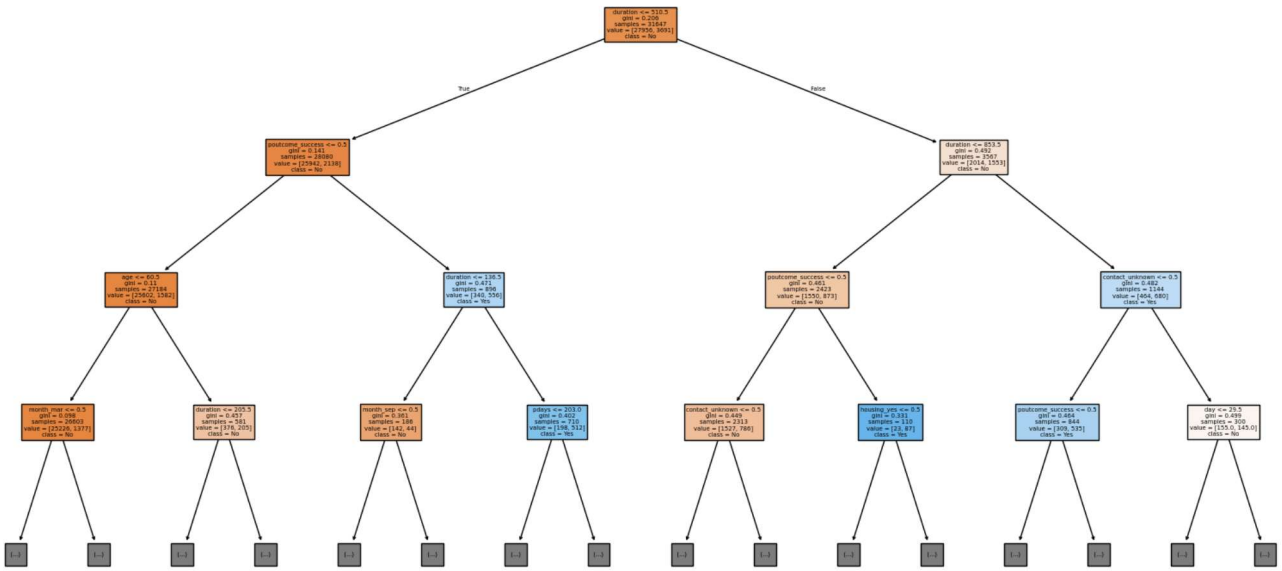
5 rows × 43 columns

```
# Features and target
X = df_encoded.drop('y_yes', axis=1)  # 'y_yes' is the target column (Yes=1, No=0)
y = df_encoded['y_yes']

# Split data
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)


from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Create model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8720141551164848
```

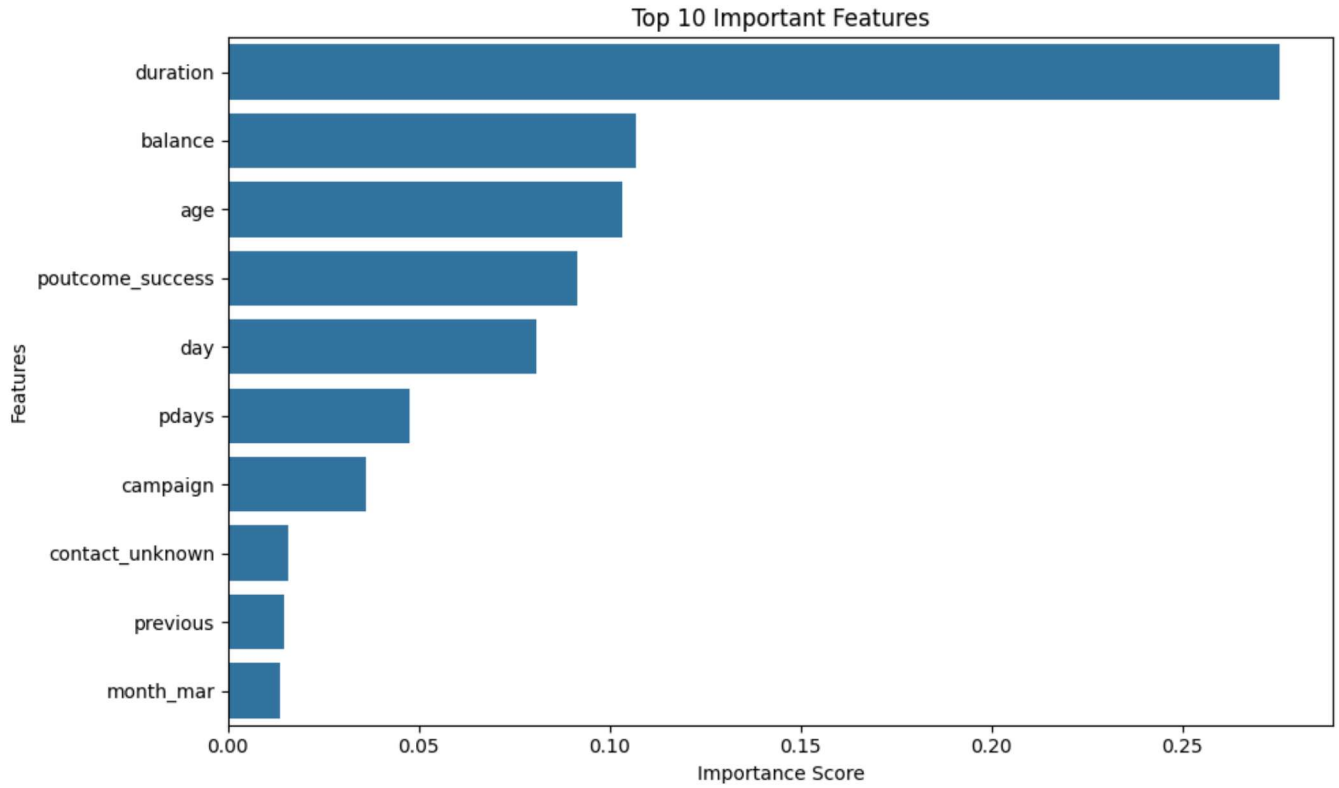
```
Confusion Matrix:
 [[11086   880]
 [  856   742]]

Classification Report:
              precision    recall  f1-score   support

       False       0.93      0.93      0.93     11966
        True       0.46      0.46      0.46      1598

    accuracy                           0.87     13564
   macro avg       0.69      0.70      0.69     13564
weighted avg       0.87      0.87      0.87     13564
```

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

plt.figure(figsize=(20, 10))
plot_tree(model, filled=True, feature_names=X.columns, class_names=["No", "Yes"],
plt.show()
```



```
import seaborn as sns

# Feature importance
importances = model.feature_importances_
features = X.columns
feat_imp = pd.Series(importances, index=features).sort_values(ascending=False)

# Plot
plt.figure(figsize=(10,6))
sns.barplot(x=feat_imp[:10], y=feat_imp.index[:10])
plt.title("Top 10 Important Features")
```

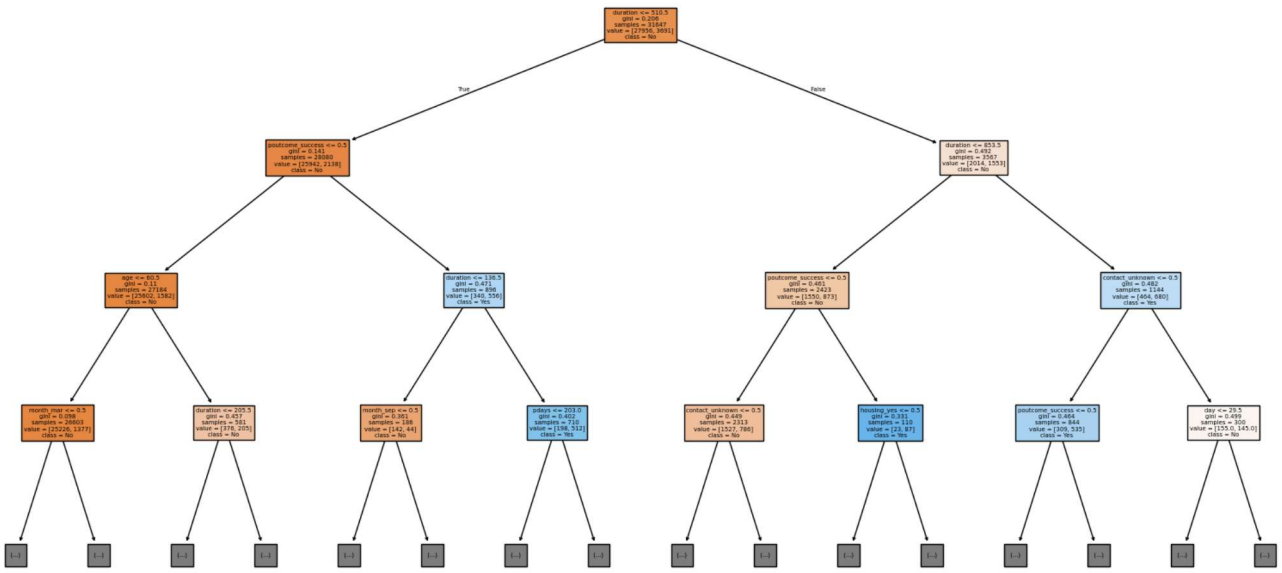
```
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.tight_layout()
plt.show()
```




```
!pip install fpdf
!pip install pillow
```

```
Collecting fpdf
  Downloading fpdf-1.7.2.tar.gz (39 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: fpdf
  Building wheel for fpdf (setup.py) ... done
  Created wheel for fpdf: filename=fpdf-1.7.2-py2.py3-none-any.whl size=40704 sha256=6bc
  Stored in directory: /root/.cache/pip/wheels/65/4f/66/bbda9866da446a72e206d6484cd97381
Successfully built fpdf
Installing collected packages: fpdf
Successfully installed fpdf-1.7.2
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (11.2.1
```

```
# Save decision tree as PNG
plt.figure(figsize=(20, 10))
plot_tree(model, filled=True, feature_names=X.columns, class_names=["No", "Yes"],
plt.savefig("decision_tree.png")
```




```
with open("model_report.txt", "w") as f:
    f.write("Accuracy: {:.2f}\n".format(accuracy_score(y_test, y_pred)))
    f.write("\nConfusion Matrix:\n")
    f.write(str(confusion_matrix(y_test, y_pred)))
    f.write("\n\nClassification Report:\n")
    f.write(classification_report(y_test, y_pred))
```