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

```
Choose Files 2 files
```
- **bank.zip**(application/x-zip-compressed) - 579043 bytes, last modified: 6/5/2025 - 100% done
- **bank-additional.zip**(application/x-zip-compressed) - 444572 bytes, last modified: 6/5/2025 - 100% done

```
Saving bank.zip to bank.zip
Saving bank-additional.zip to bank-additional.zip
```

```
# Downloading and unziping bank marketing dataset directly from UCI
!wget https://archive.ics.uci.edu/static/public/222/bank.zip
!unzip bank.zip
```

```
--2025-06-05 11:23:48--  https://archive.ics.uci.edu/static/public/222/bank.zip
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 404 Not Found
2025-06-05 11:23:49 ERROR 404: Not Found.

Archive:  bank.zip
  inflating: bank-full.csv
  inflating: bank-names.txt
  inflating: bank.csv
```

```
import pandas as pd

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

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcc
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unkno
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unkno
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unkno
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unkno
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unkno

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

```
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.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report
```

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

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcc
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unkno
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unkno
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unkno
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unkno
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unkno

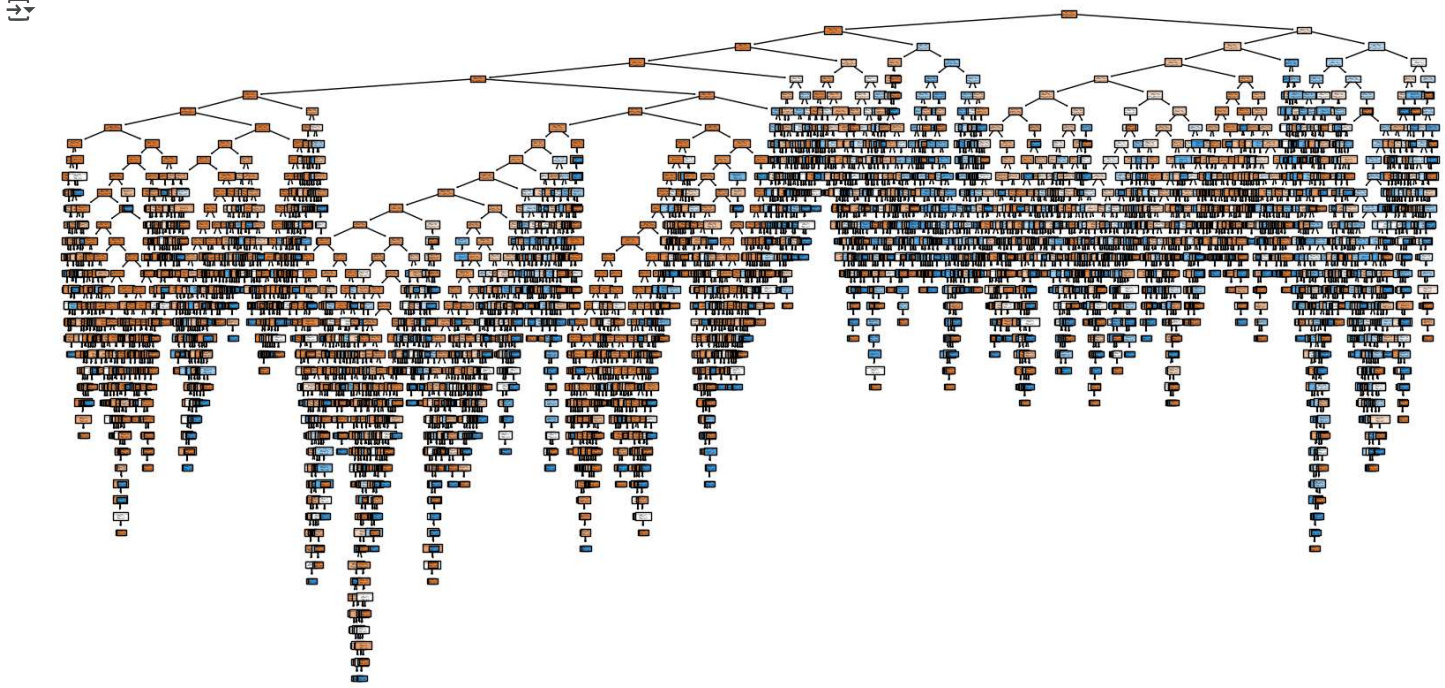
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# One-hot encoding for categorical features
df_encoded = pd.get_dummies(df, drop_first=True)

# Features and target
X = df_encoded.drop('y_yes', axis=1)
y = df_encoded['y_yes']
```

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)


print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8705075749198274
Classification Report:
                precision    recall  f1-score   support

         False       0.93      0.92      0.93      7952
          True       0.46      0.48      0.47      1091

      accuracy                           0.87      9043
     macro avg       0.70      0.70      0.70      9043
  weighted avg       0.87      0.87      0.87      9043
```

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

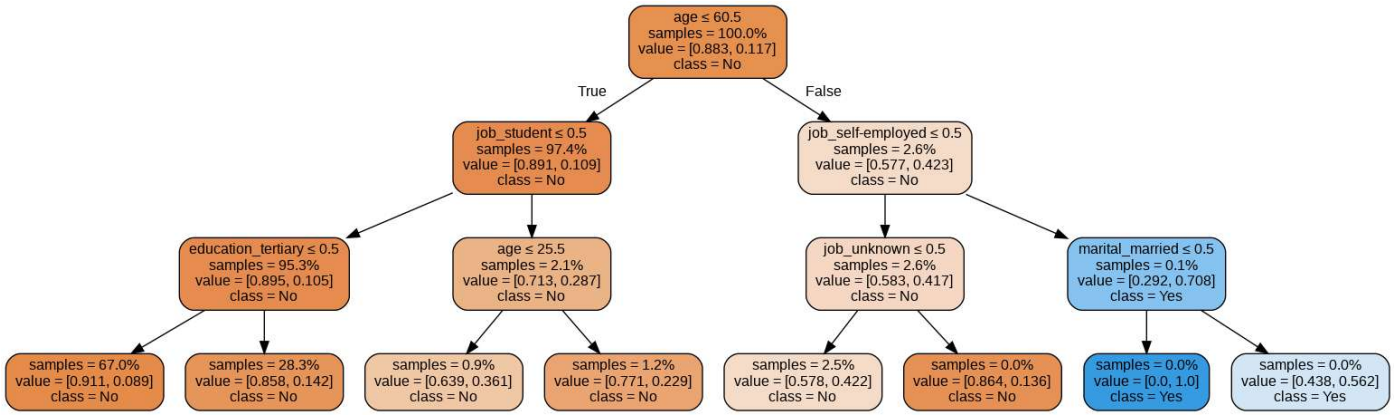


```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("bank-full.csv", sep=';')

# Select features
features = ['age', 'job', 'marital', 'education', 'y']
df_small = df[features].copy()
```

```
# Encode target variable
le_y = LabelEncoder()
df_small['y'] = le_y.fit_transform(df_small['y'])  # yes=1, no=0

# One-hot encode categorical features
X = pd.get_dummies(df_small.drop('y', axis=1), drop_first=True)
y = df_small['y']


from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(max_depth=3, random_state=42)
model.fit(X, y)
```

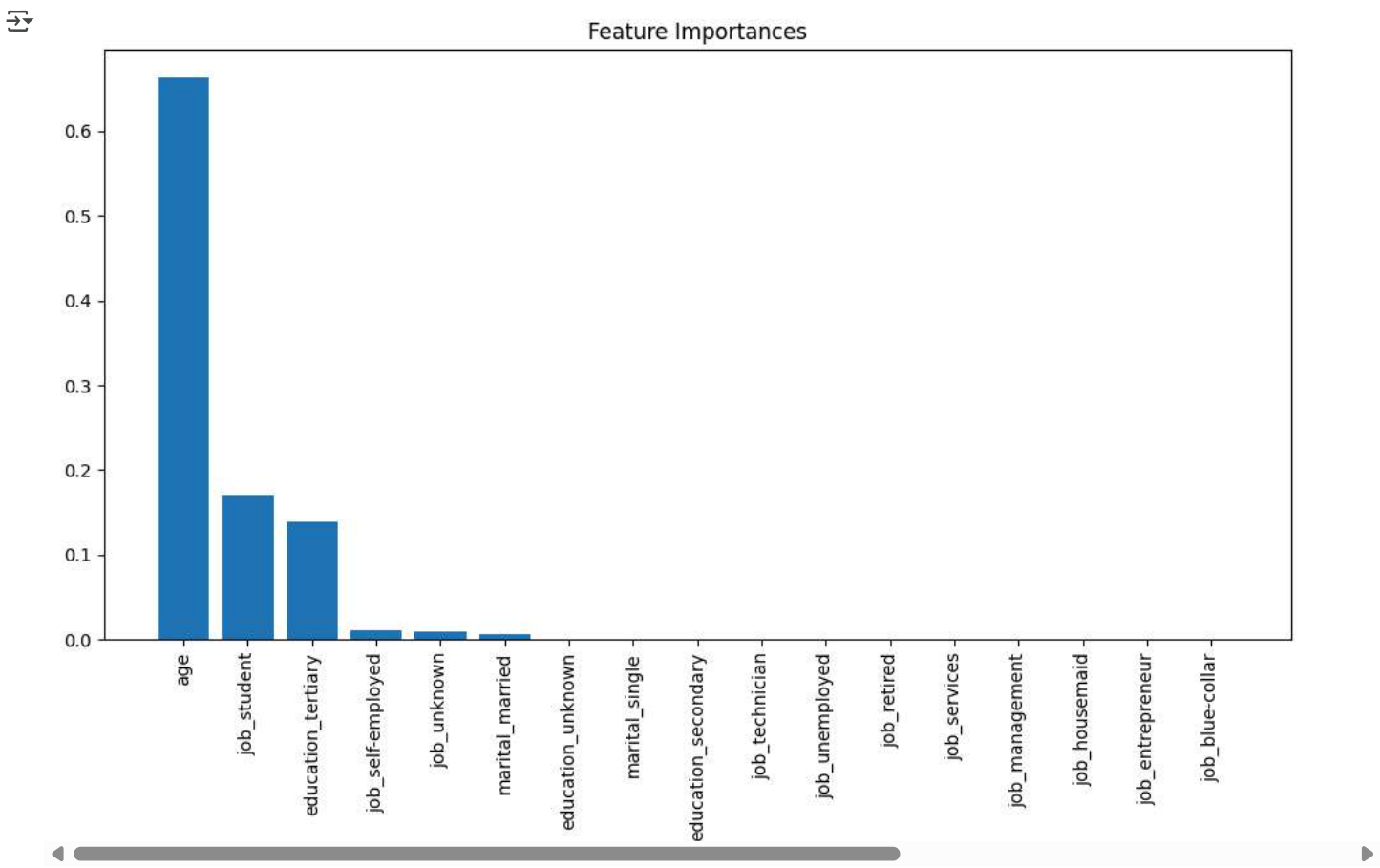
```
DecisionTreeClassifier(max_depth=3, random_state=42)
```

```
from sklearn.tree import export_graphviz

dot_data = export_graphviz(
    model,
    out_file=None,
    feature_names=X.columns,
    class_names=["No", "Yes"],
    filled=True,
    rounded=True,
    special_characters=True,
    impurity=False,
    proportion=True
)

# Add fontsize in DOT source to improve readability
dot_data = 'digraph Tree {\nnode [fontsize=12, shape=box]; edge [fontsize=12];\n' + dot_data.split('digraph Tree {',1)[1]


import graphviz
from IPython.display import Image

graph = graphviz.Source(dot_data)
graph.render("decision_tree_readable", format="png")  # Saves the image file
Image("decision_tree_readable.png")                   # Display inline
```

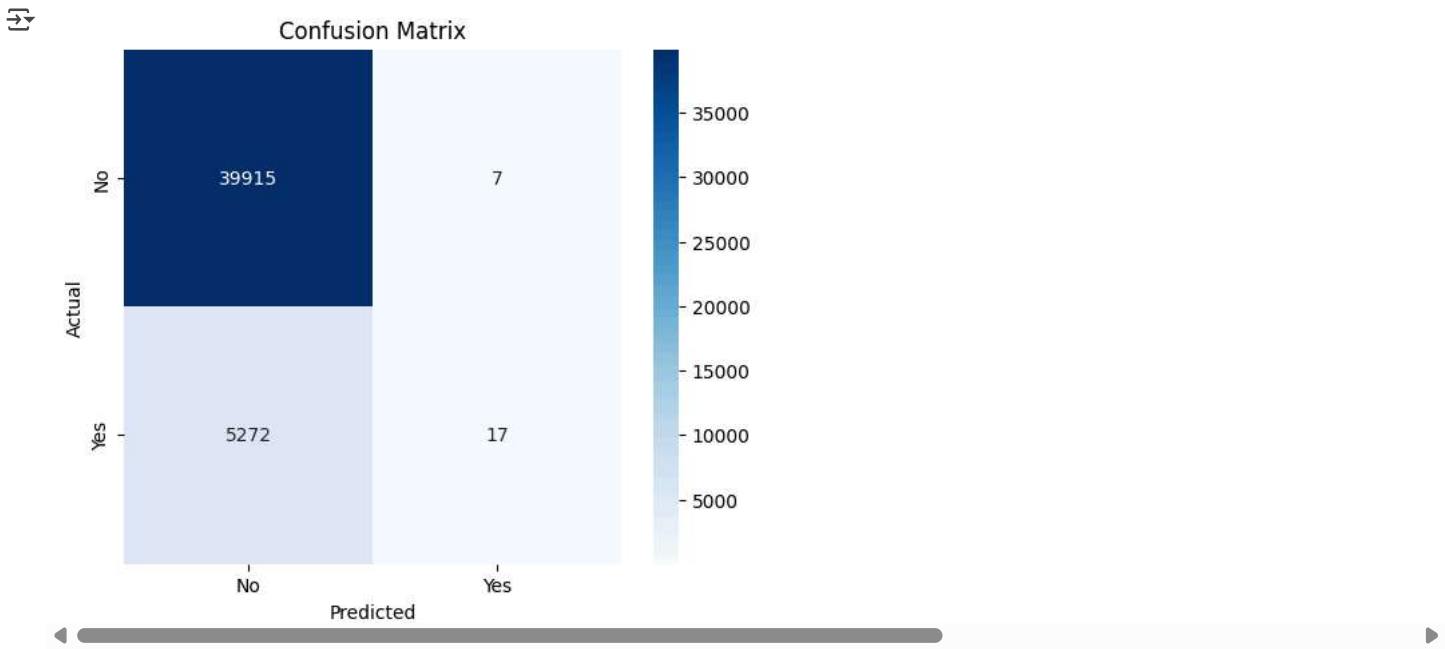



```
import matplotlib.pyplot as plt
import numpy as np

importances = model.feature_importances_
indices = np.argsort(importances)[::-1]
features = X.columns

plt.figure(figsize=(12,6))
plt.title("Feature Importances")
plt.bar(range(len(importances)), importances[indices], align='center')
```

```
plt.xticks(range(len(importances)), [features[i] for i in indices], rotation=90)
plt.show()
```




```
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_pred = model.predict(X)
cm = confusion_matrix(y, y_pred)

plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No','Yes'], yticklabels=['No','Yes'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```




```
import graphviz
from google.colab import files

# Render the graph to PDF
graph.render("decision_tree_readable", format="pdf")

# Download the PDF file
files.download("decision_tree_readable.pdf")
```

```
# Import required libraries again for clarity
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset again (already loaded previously)
df = pd.read_csv('bank-full.csv', sep=';')

# Select a simplified feature set for interpretability
selected_features = ['age', 'job', 'marital', 'education']
df_selected = df[selected_features + ['y']]

# One-hot encoding for categorical features
df_encoded = pd.get_dummies(df_selected, drop_first=True)

# Features and target
X = df_encoded.drop('y_yes', axis=1)
y = df_encoded['y_yes']

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

# Decision Tree model with limited depth
clf = DecisionTreeClassifier(max_depth=3, random_state=42)
clf.fit(X_train, y_train)

# Predictions
y_pred = clf.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, output_dict=True)
conf_matrix = confusion_matrix(y_test, y_pred)

accuracy, report, conf_matrix
```

```
(0.8794647793873714,
 {'False': {'precision': 0.8797034085878708,
   'recall': 0.9996227364185111,
   'f1-score': 0.9358370614551448,
   'support': 7952.0},
  'True': {'precision': 0.5714285714285714,
   'recall': 0.0036663611365719525,
   'f1-score': 0.007285974499089253,
   'support': 1091.0},
  'accuracy': 0.8794647793873714,
  'macro avg': {'precision': 0.7255659900082211,
   'recall': 0.5016445487775415,
   'f1-score': 0.47156151797711704,
   'support': 9043.0},
  'weighted avg': {'precision': 0.8425113431957669,
   'recall': 0.8794647793873714,
   'f1-score': 0.8238112695864003,
   'support': 9043.0}},
 array([[7949,    3],
        [1087,    4]]))
```

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import matplotlib.pyplot as plt

# Step 1: Train model
clf = DecisionTreeClassifier(max_depth=4, random_state=42)
clf.fit(X_train, y_train)

# Step 2: Predictions
y_pred = clf.predict(X_test)

# Step 3: Evaluation
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Step 4: Plot the tree
plt.figure(figsize=(20,10))
plot_tree(clf, feature_names=X.columns, class_names=['No', 'Yes'], filled=True)
plt.title("Decision Tree Visualization (Depth = 4)")
plt.show()
```

```
Accuracy: 0.8793541966161672

Classification Report:
              precision    recall  f1-score   support

       False       0.88      1.00      0.94      7952
        True       0.50      0.00      0.01      1091

    accuracy                           0.88      9043
   macro avg       0.69      0.50      0.47      9043
weighted avg       0.83      0.88      0.82      9043


Confusion Matrix:
[[7949    3]
 [1088    3]]
```

Decision Tree Visualization (Depth = 4)