Close

```
# **Task 3** – Decision Tree Classifier

**Dataset:** Bank Marketing Dataset (Kaggle)
**Objective:**
Build a Decision Tree Classifier to predict whether a customer w
to a term deposit.
This task includes dataset loading, preprocessing, feature engin
training, evaluation, and visualization.
```

Task 3 – Decision Tree Classifier

Dataset: Bank Marketing Dataset (Kaggle)
Objective:
Build a Decision Tree Classifier to predict whether a customer will subscribe to a term deposit.
This task includes dataset loading, preprocessing, feature engineering, model training, evaluation, and visualization.

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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving bank.csv to bank (1).csv

```
import pandas as pd

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

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previo
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	

```
print("Shape:", df.shape)
print("\nColumns:\n", df.columns)

print("\nDataset Info:")
print(df.info())

print("\nMissing Values:")
print(df.isnull().sum())
```

```
Shape: (11162, 17)

Columns:
 Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        11162 non-null  int64
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64
 12  campaign   11162 non-null  int64
 13  pdays      11162 non-null  int64
 14  previous   11162 non-null  int64
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
None
```

```
Missing Values:
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
deposit      0
dtype: int64
```

```
df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previo
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	

```
X = df.drop("deposit", axis=1)
y = df["deposit"].map({"yes":1, "no":0})
```

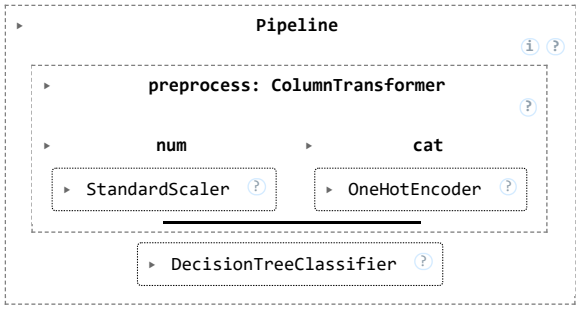
```
num_cols = X.select_dtypes(include=['int64','float64']).columns
cat_cols = X.select_dtypes(include=['object']).columns

num_cols, cat_cols
```

```
(Index(['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous'], dtype='object'),
 Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
        'month', 'poutcome'],
       dtype='object'))
```

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

preprocess = ColumnTransformer([
    ("num", StandardScaler(), num_cols),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols)
])
```

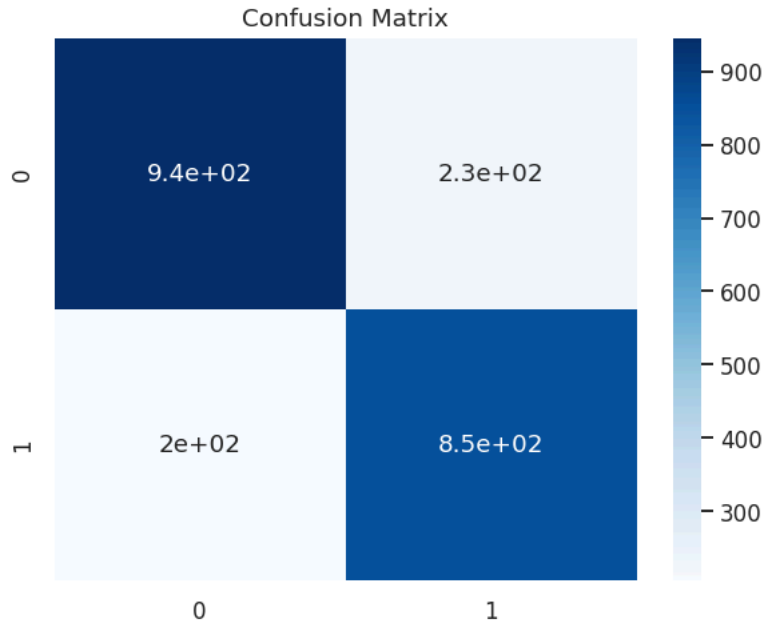
```
from sklearn.model_selection import train_test_split

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

```
from sklearn.tree import DecisionTreeClassifier

clf = Pipeline([
    ("preprocess", preprocess),
    ("model", DecisionTreeClassifier(max_depth=5, random_state=42))
])

clf.fit(X_train, y_train)
```

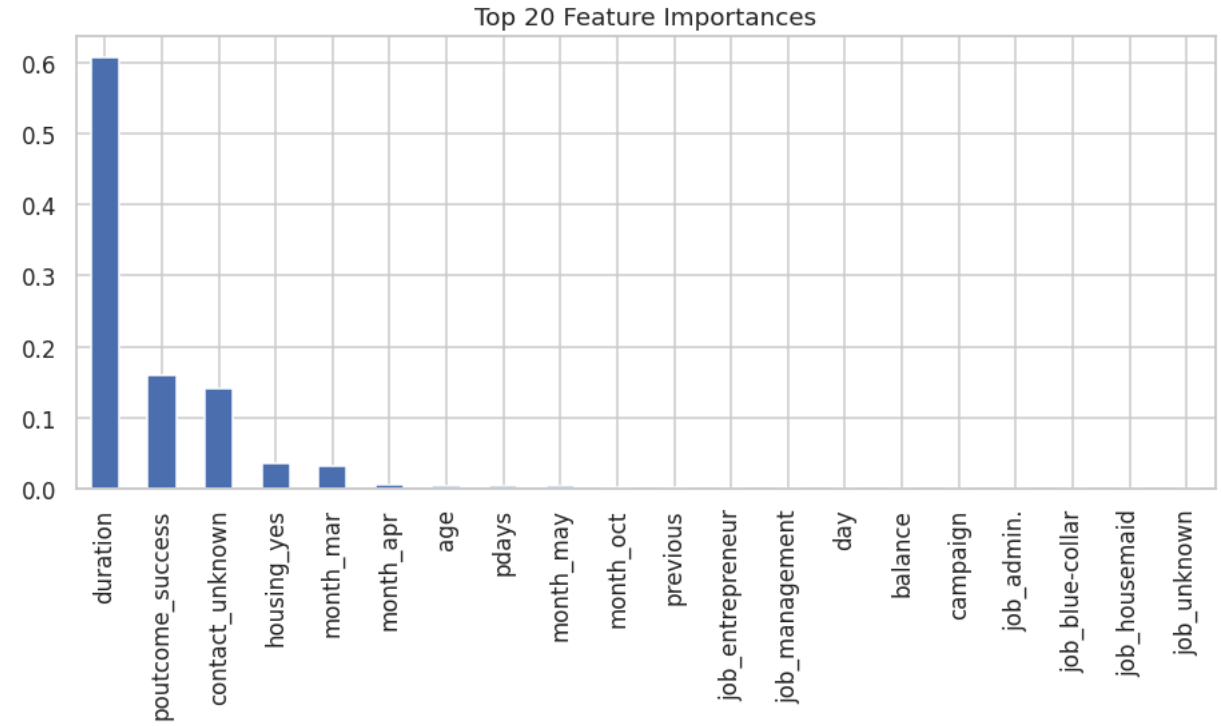



```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

y_pred = clf.predict(X_test)

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

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Blues")
plt.title("Confusion Matrix")
plt.show()
```

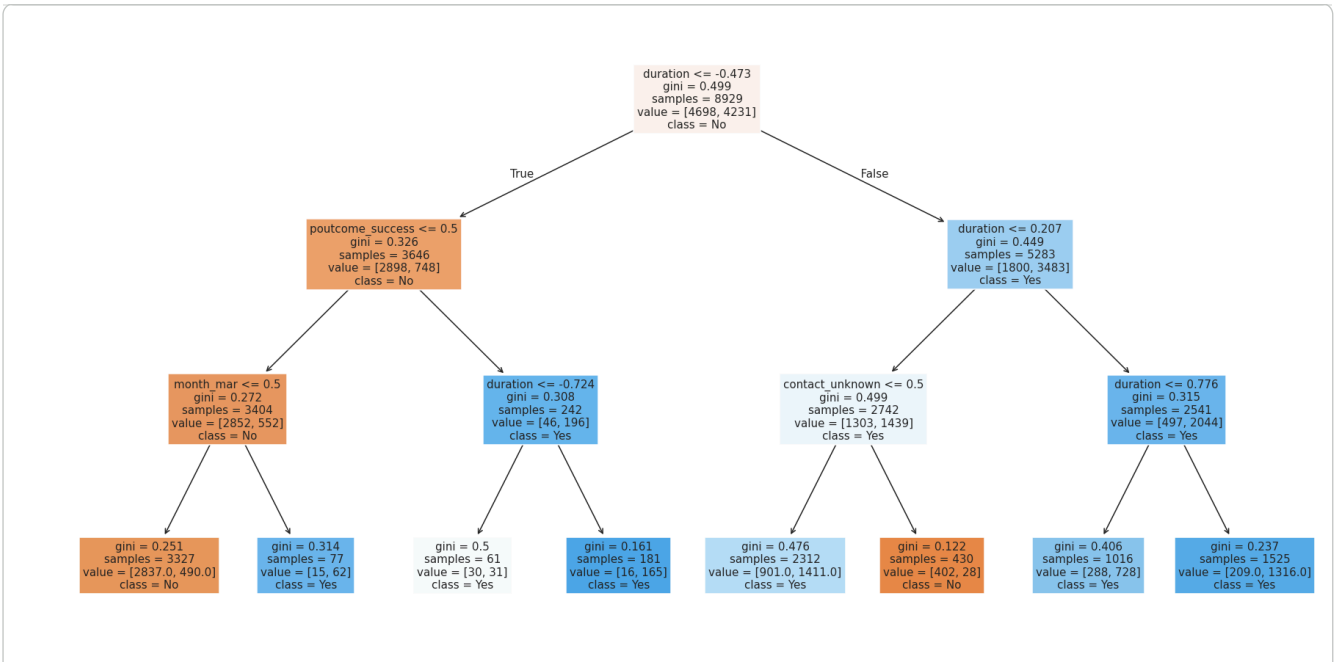
```
Accuracy: 0.8051948051948052

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.80      0.81      1175
           1       0.79      0.81      0.80      1058

    accuracy                           0.81      2233
   macro avg       0.80      0.81      0.80      2233
weighted avg       0.81      0.81      0.81      2233
```




```
# Extract OHE feature names
ohe = clf.named_steps["preprocess"].named_transformers_["cat"]
ohe_cols = list(ohe.get_feature_names_out(cat_cols))

all_features = list(num_cols) + ohe_cols

importances = clf.named_steps["model"].feature_importances_

feat_imp = pd.Series(importances, index=all_features)
feat_imp.sort_values(ascending=False).head(20).plot(kind='bar', figsize=(10,4))
plt.title("Top 20 Feature Importances")
plt.show()
```




```
from sklearn import tree

clf_small = Pipeline([
    ("preprocess", preprocess),
    ("model", DecisionTreeClassifier(max_depth=3, random_state=42))
])

clf_small.fit(X_train, y_train)

plt.figure(figsize=(20,10))
tree.plot_tree(clf_small.named_steps["model"],
               filled=True,
               feature_names=all_features,
               class_names=["No","Yes"])
plt.show()
```


duration <= -0.473
gini = 0.499
samples = 8929
value = [4698, 4231]
class = No
True
False
poutcome_success <= 0.5
gini = 0.326
samples = 3646
value = [2898, 748]
class = No
duration <= 0.207
gini = 0.449
samples = 5283
value = [1800, 3483]
class = Yes
month_mar <= 0.5
gini = 0.272
samples = 3404
value = [2852, 552]
class = No
duration <= -0.724
gini = 0.308
samples = 242
value = [46, 196]
class = Yes
contact_unknown <= 0.5
gini = 0.499
samples = 2742
value = [1303, 1439]
class = Yes
duration <= 0.776
gini = 0.315
samples = 2541
value = [497, 2044]
class = Yes
gini = 0.251
samples = 3327
value = [2837.0, 490.0]
class = No
gini = 0.314
samples = 77
value = [15, 62]
class = Yes
gini = 0.5
samples = 61
value = [30, 31]
class = Yes
gini = 0.161
samples = 181
value = [16, 165]
class = Yes
gini = 0.476
samples = 2312
value = [901.0, 1411.0]
class = Yes
gini = 0.122
samples = 430
value = [402, 28]
class = No
gini = 0.406
samples = 1016
value = [288, 728]
class = Yes
gini = 0.237
samples = 1525
value = [209.0, 1316.0]
class = Yes