

```
import pandas as pd

url = "https://raw.githubusercontent.com/Prodigy-InfoTech/data-science-datasets/main/Task%203/bank/bank.csv"

df = pd.read_csv(url, sep=';')

df.head()
```

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

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         4521 non-null   int64
1   job         4521 non-null   object
2   marital     4521 non-null   object
3   education   4521 non-null   object
4   default     4521 non-null   object
5   balance     4521 non-null   int64
6   housing     4521 non-null   object
7   loan        4521 non-null   object
8   contact     4521 non-null   object
9   day         4521 non-null   int64
10  month       4521 non-null   object
11  duration    4521 non-null   int64
12  campaign    4521 non-null   int64
13  pdays       4521 non-null   int64
14  previous    4521 non-null   int64
15  poutcome    4521 non-null   object
16  y           4521 non-null   object
dtypes: int64(7), object(10)
memory usage: 600.6+ KB
```

```
df.isnull().sum()
```

```

      0
age    0
job    0

```

```
df['y'].value_counts()
```

```

default 0
count
balance 0
no      4000
yes     521
loan    0
contact 0
dtype: int64

```

```

# Convert categorical columns to numeric using One-Hot Encoding
df_encoded = pd.get_dummies(df, drop_first=True)

df_encoded.head()

```

```

previous 0
age balance day duration campaign pdays previous job_blue-collar job_entrepreneur job_housemaid ... month_jun month_
outcome 0
0 30 1787 19 79 1 -1 0 False False False ... False Fa
1 33 4789 11 220 1 339 4 False False False ... False Fa
2 35 1350 16 185 1 330 1 False False False ... False Fa
3 30 1476 3 199 4 -1 0 False False False ... True Fa
4 59 0 5 226 1 -1 0 True False False ... False Fa
5 rows x 43 columns

```

```

X = df_encoded.drop('y_yes', axis=1)
y = df_encoded['y_yes']

```

```

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
)

```

```

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(max_depth=5, random_state=42)
model.fit(X_train, y_train)

```

DecisionTreeClassifier ⓘ ?

DecisionTreeClassifier(max_depth=5, random_state=42)

```

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = model.predict(X_test)

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: 0.9060773480662984

Confusion Matrix:

```

[[785  22]
 [ 63  35]]

```

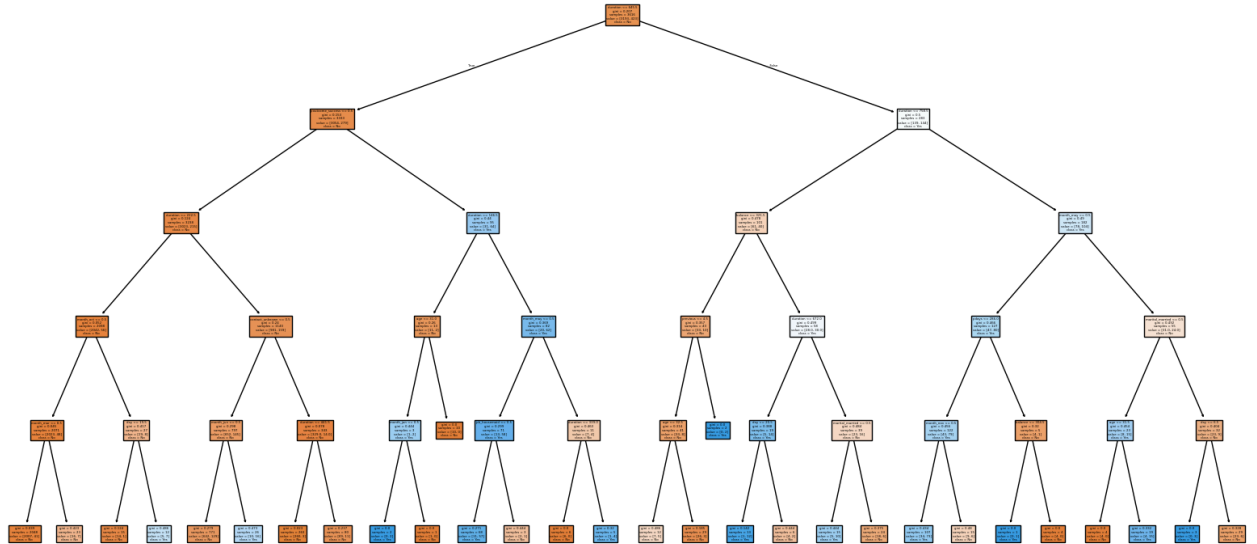
Classification Report:

	precision	recall	f1-score	support
False	0.93	0.97	0.95	807
True	0.61	0.36	0.45	98

accuracy			0.91	905
macro avg	0.77	0.66	0.70	905
weighted avg	0.89	0.91	0.89	905

```
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 pickle

with open("bank_marketing_dt_model.pkl", "wb") as f:
    pickle.dump(model, f)
```

```
sample = X_test.iloc[0]    # Example data

model.predict([sample])
```

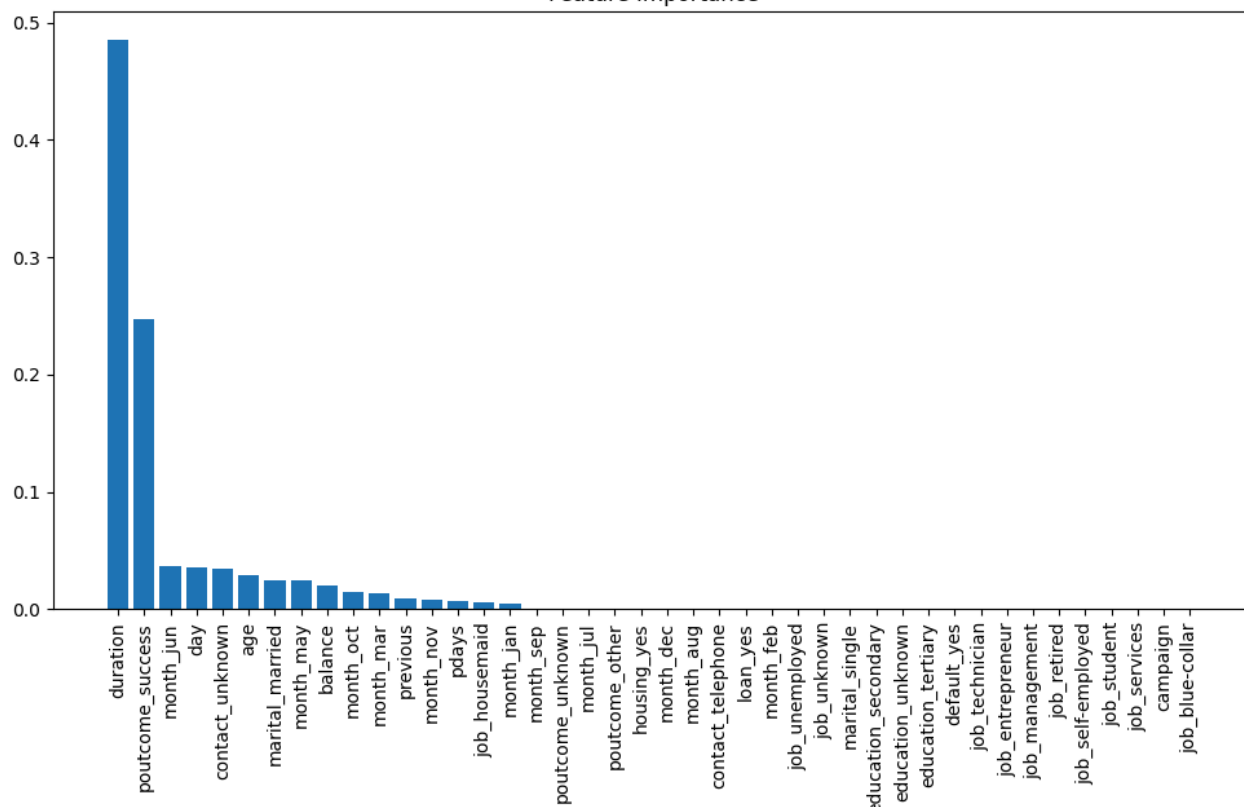
```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names,
warnings.warn(
array([False])
```

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

# Feature importance
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(12,6))
plt.title("Feature Importance")
plt.bar(range(len(importances)), importances[indices])
plt.xticks(range(len(importances)), X.columns[indices], rotation=90)
plt.show()
```

Feature Importance

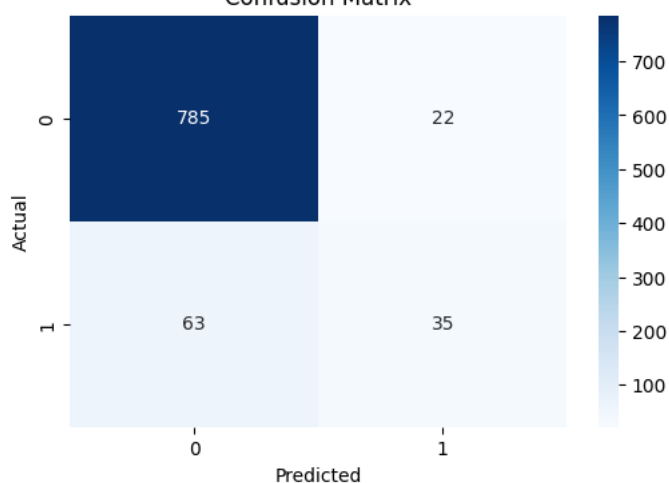


```
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Confusion Matrix



```
import pickle

with open("bank_marketing_dt_model.pkl", "wb") as f:
    pickle.dump(model, f)

print("Model saved successfully!")
```

Model saved successfully!

```

from sklearn.model_selection import GridSearchCV

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 5, 7, 10, None],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4]
}

grid = GridSearchCV(
    DecisionTreeClassifier(random_state=42),
    param_grid,
    cv=5,
    scoring="accuracy"
)

grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)
print("Best Score:", grid.best_score_)

```

```

Best Parameters: {'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 2, 'min_samples_split': 5}
Best Score: 0.8954681613595898

```

```

best_model = grid.best_estimator_

y_pred_best = best_model.predict(X_test)

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

print("Final Accuracy:", accuracy_score(y_test, y_pred_best))
print("\nClassification Report:\n", classification_report(y_test, y_pred_best))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_best))

```

```

Final Accuracy: 0.9049723756906077

```

```

Classification Report:

```

	precision	recall	f1-score	support
False	0.93	0.97	0.95	807
True	0.59	0.40	0.48	98
accuracy			0.90	905
macro avg	0.76	0.68	0.71	905
weighted avg	0.89	0.90	0.90	905

```

Confusion Matrix:
[[780  27]
 [ 59  39]]

```

```

import pickle

with open("bank_marketing_best_model.pkl", "wb") as f:
    pickle.dump(best_model, f)

```