```
#import dataset
import pandas as pd
df = pd.read_csv('/content/bank-full.csv', sep=';')
df
```

```
       age           job   marital  education  default  balance  housing  loan  con
    0   58    management   married   tertiary       no     2143      yes    no  unk
    1   44    technician    single  secondary       no       29      yes    no  unk
    2   33  entrepreneur   married  secondary       no        2      yes   yes  unk
    3   47   blue-collar   married    unknown       no     1506      yes    no  unk
    4   33       unknown    single    unknown       no        1       no    no  unk
  ...  ...           ...       ...        ...      ...      ...      ...   ...
45206   51    technician   married   tertiary       no      825       no    no   c
45207   71       retired  divorced    primary       no     1729       no    no   c
45208   72       retired   married  secondary       no     5715       no    no   c
45209   57   blue-collar   married  secondary       no      668       no    no  telep
45210   37  entrepreneur   married  secondary       no     2971       no    no   c

45211 rows × 17 columns
```

```
#Checking for null values
df.isnull().sum()
```

```
                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
```

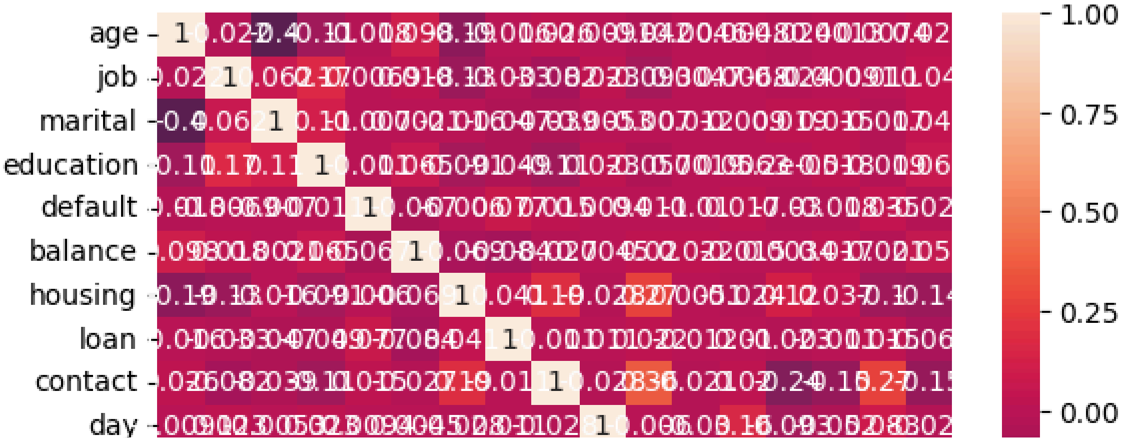
```
df.describe()
```

```
                age         balance           day      duration      campaign          pd
count  45211.000000    45211.000000  45211.000000  45211.000000  45211.000000  45211.000
 mean     40.936210     1362.272058     15.806419    258.163080      2.763841     40.197
  std     10.618762     3044.765829      8.322476    257.527812      3.098021    100.128
  min     18.000000    -8019.000000      1.000000      0.000000      1.000000     -1.000
  25%     33.000000       72.000000      8.000000    103.000000      1.000000     -1.000
  50%     39.000000      448.000000     16.000000    180.000000      2.000000     -1.000
  75%     48.000000     1428.000000     21.000000    319.000000      3.000000     -1.000
  max     95.000000   102127.000000     31.000000   4918.000000     63.000000    871.000
```

```
#Encode categorical variables
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
label_mappings = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_mappings[col] = dict(zip(le.classes_, le.transform(le.classes_)))
```

```
#heatmap
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(df.corr(), annot=True)
plt.show()
```



```
#Split features and target
x = df.drop('y', axis=1)
y = df['y']
```

```
#Split into training and testing sets
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
```

```
#create decisiontree model
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
```

```
▾ DecisionTreeClassifier  ⓘ ?
DecisionTreeClassifier()
```

```
#make prediction
y_pred = model.predict(X_test)
```

```
#accuracy
from sklearn.metrics import accuracy_score, classification_report
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
print('\nClassification Report: ',classification_report(y_test, y_pred))
```

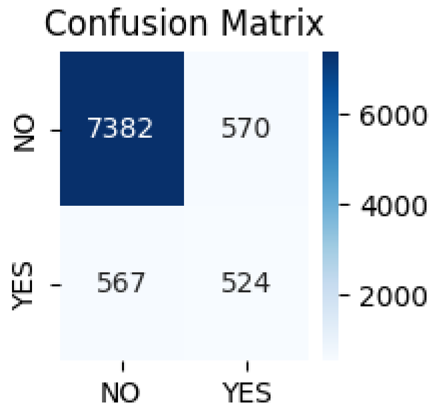
```
Accuracy: 0.8736038925135464

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

```
           0       0.93      0.93      0.93      7952
           1       0.48      0.48      0.48      1091

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

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(2,2))
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,fmt="d",cmap="Blues",xticklabels=["NO","YES"],yticklabels=["NO'
plt.title("Confusion Matrix")
plt.show()
```



```
from sklearn.tree import plot_tree
plt.figure(figsize=(5,5))
plot_tree(model, feature_names=x.columns, class_names=["No", "Yes"], filled=True, max_de
plt.show()
```