Problem statement:Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioral data.

Dataset:Bank Marketing About the dataset: The dataset contains demographic and behavioral information of individuals contacted during a marketing campaign. It includes features such as age, job type, marital status, education level, credit default status, housing loan status, and contact details. The target variable 'y' indicates whether each individual subscribed to a product or service ('yes' or 'no').

```
In [ ]: import pandas as pd
        from sklearn.model selection import train test split
        from sklearn.preprocessing import LabelEncoder, StandardScaler
        from sklearn.tree import DecisionTreeClassifier
        # Load the data
        data path = 'c:\\Users\\Admin\\Downloads\\bank.csv'
        df = pd.read csv(data path, sep=';')
        # Display the first few rows of the dataframe
        print(df.head())
                                     education
          age
                     job marital
                                                default housing loan
                                                                        contact \
               housemaid married
                                      basic.4y
                                                                  no telephone
       0
                                                     no
                                                             no
               services married high.school
       1
           57
                                                unknown
                                                                  no telephone
                                                             no
       2
           37
                services married high.school
                                                                      telephone
                                                     no
                                                            yes
       3
          40
                  admin. married
                                      basic.6y
                                                                      telephone
                                                     no
                                                             no
                                                                  no
       4
           56
               services married high.school
                                                                      telephone
                                                     no
                                                             no yes
                                 campaign pdays previous
         month day_of_week
                                                               poutcome emp.var.rate \
       0
           may
                                        1
                                             999
                                                            nonexistent
                                                                                 1.1
                       mon
                            . . .
                                             999
                                                         0
                                                                                 1.1
       1
          may
                       mon
                                        1
                                                            nonexistent
       2
          may
                       mon
                                        1
                                             999
                                                            nonexistent
                                                                                 1.1
       3
                                             999
                                                            nonexistent
                                                                                 1.1
                                        1
          may
                       mon
                                             999
                                                            nonexistent
                                                                                 1.1
          may
                       mon
          cons.price.idx cons.conf.idx euribor3m nr.employed
                                                                  У
       0
                  93.994
                                  -36.4
                                             4.857
                                                         5191.0
                                                                 no
                  93.994
                                  -36.4
                                             4.857
       1
                                                         5191.0 no
       2
                  93.994
                                  -36.4
                                             4.857
                                                         5191.0 no
       3
                  93.994
                                  -36.4
                                             4.857
                                                         5191.0 no
                  93.994
                                  -36.4
                                             4.857
                                                         5191.0 no
       [5 rows x 21 columns]
In [ ]: # Check the structure of the dataset
        print(df.info())
        # Summary statistics of the dataset
        print(df.describe())
```

> <class 'pandas.core.frame.DataFrame'> RangeIndex: 41188 entries, 0 to 41187 Data columns (total 21 columns):

```
Column
                     Non-Null Count Dtype
     -----
                     -----
0
                     41188 non-null
                                    int64
     age
1
     job
                     41188 non-null object
 2
     marital
                     41188 non-null
                                     object
 3
     education
                     41188 non-null
                                    object
 4
     default
                     41188 non-null
                                    object
 5
     housing
                     41188 non-null
                                    object
 6
     loan
                     41188 non-null
                                    object
 7
     contact
                     41188 non-null
                                    object
 8
     month
                     41188 non-null object
 9
     day of week
                     41188 non-null
                                    object
 10
     duration
                     41188 non-null
                                    int64
 11
     campaign
                     41188 non-null
                                    int64
 12
     pdays
                     41188 non-null
                                    int64
 13
     previous
                     41188 non-null int64
                                    object
 14
     poutcome
                     41188 non-null
 15
     emp.var.rate
                     41188 non-null float64
     cons.price.idx
                    41188 non-null
                                    float64
 16
 17
                     41188 non-null float64
     cons.conf.idx
                     41188 non-null float64
 18
    euribor3m
 19
     nr.emploved
                     41188 non-null
                                    float64
 20
                     41188 non-null
                                    object
    У
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

None

```
duration
                                        campaign
                                                                      previous
                age
                                                          pdays
       41188.00000
                     41188.000000
                                                                  41188.000000
count
                                    41188.000000
                                                   41188.000000
mean
          40.02406
                       258.285010
                                        2.567593
                                                     962.475454
                                                                      0.172963
std
          10.42125
                       259.279249
                                        2.770014
                                                     186.910907
                                                                      0.494901
min
          17.00000
                         0.000000
                                        1.000000
                                                       0.000000
                                                                      0.000000
25%
          32.00000
                       102.000000
                                        1.000000
                                                     999.000000
                                                                      0.000000
50%
          38.00000
                       180.000000
                                                     999.000000
                                        2.000000
                                                                      0.000000
75%
          47.00000
                       319.000000
                                        3.000000
                                                     999.000000
                                                                      0.000000
max
          98.00000
                      4918.000000
                                       56.000000
                                                     999.000000
                                                                      7.000000
```

```
emp.var.rate
                      cons.price.idx
                                       cons.conf.idx
                                                          euribor3m
                                                                       nr.employed
count
       41188.000000
                        41188.000000
                                        41188.000000
                                                       41188.000000
                                                                      41188.000000
                           93.575664
                                          -40.502600
                                                                       5167.035911
mean
           0.081886
                                                           3.621291
std
                            0.578840
                                                           1.734447
                                                                         72.251528
           1.570960
                                            4.628198
min
          -3.400000
                           92.201000
                                          -50.800000
                                                           0.634000
                                                                       4963.600000
25%
          -1.800000
                           93.075000
                                          -42.700000
                                                           1.344000
                                                                       5099.100000
50%
           1.100000
                           93.749000
                                          -41.800000
                                                           4.857000
                                                                       5191.000000
75%
           1.400000
                           93.994000
                                          -36.400000
                                                                       5228.100000
                                                           4.961000
           1.400000
                           94.767000
                                          -26.900000
                                                           5.045000
                                                                       5228.100000
max
```

```
In [ ]: # Check for missing values
        missing values = df.isnull().sum()
        print(missing_values)
```

age 0 0 job 0 marital education 0 0 default housing 0 0 loan 0 contact 0 month day\_of\_week 0 duration 0 0 campaign pdays 0 0 previous 0 poutcome 0 emp.var.rate cons.price.idx 0 cons.conf.idx 0 euribor3m 0 nr.employed 0 0 У dtype: int64

```
In []: # Analyze categorical variables
    categorical_columns = df.select_dtypes(include=['object']).columns

# Display unique values and their counts for each categorical column
for col in categorical_columns:
    print(f'Column: {col}')
    print(df[col].value_counts())
    print('\n')
```

Column: job job admin. 10422 blue-collar 9254 technician 6743 services 3969 management 2924 retired 1720 entrepreneur 1456 self-employed 1421 housemaid 1060 unemployed 1014 student 875 unknown 330

Name: count, dtype: int64

Column: marital

marital

married 24928 single 11568 divorced 4612 unknown 80

Name: count, dtype: int64

Column: education

education

university.degree 12168 high.school 9515 basic.9y 6045 professional.course 5243 basic.4y 4176 basic.6y 2292 unknown 1731 illiterate 18

Name: count, dtype: int64

Column: default

default

no 32588 unknown 8597 yes 3

Name: count, dtype: int64

Column: housing

housing

yes 21576 no 18622 unknown 990

Name: count, dtype: int64

Column: loan

loan

no 33950 yes 6248 unknown 990

Name: count, dtype: int64

Column: contact

contact

cellular 26144 telephone 15044

Name: count, dtype: int64

Column: month

month

dec

may 13769 7174 jul 6178 aug jun 5318 4101 nov apr 2632 oct 718 570 sep mar 546

Name: count, dtype: int64

182

Column: day\_of\_week

day\_of\_week
thu 8623
mon 8514
wed 8134
tue 8090
fri 7827

Name: count, dtype: int64

Column: poutcome

poutcome

nonexistent 35563 failure 4252 success 1373

Name: count, dtype: int64

Column: y

У

no 36548 yes 4640

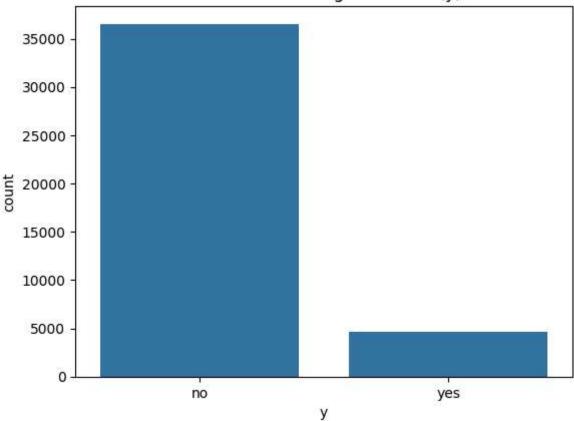
Name: count, dtype: int64

```
import matplotlib.pyplot as plt
          import seaborn as sns
          # Histograms for numerical variables
          numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns
         df[numerical_columns].hist(figsize=(10, 10))
         plt.tight_layout()
         plt.show()
                                                        duration
                        age
                                                                                         campaign
        12500
                                                                          40000
        10000
                                         30000
                                                                          30000
        7500
                                         20000
                                                                          20000
        5000
                                         10000
                                                                          10000
         2500
           0
                                     100
                                                  1000 2000 3000 4000 5000
              20
                          60
                       pdays
                                                        previous
                                                                                       emp.var.rate
                                                                          25000
        40000
                                         30000
                                                                          20000
        30000
                                                                          15000
                                         20000
        20000
                                                                          10000
                                         10000
        10000
                                                                           5000
           0
                                             0
                      400 600
                               800 1000
                 200
                                                                                            -1
                                                                                        euribor3m
                    cons.price.idx
                                                      cons.conf.idx
                                         15000
                                                                          25000
        15000
                                                                          20000
        10000
                                         10000
                                                                          15000
                                                                           10000
        5000
                                          5000
                                                                           5000
           0
                                                             -35
               92.5 93.0 93.5 94.0 94.5
                                                   -45 -40
                     nr.employed
        15000
        10000
         5000
                5000 5050 5100 5150 5200
In [ ]: # Analyze the target variable 'y'
          print(df['y'].value_counts())
         # Plot the distribution of the target variable
          sns.countplot(x='y', data=df)
         plt.title('Distribution of Target Variable (y)')
```

plt.show()

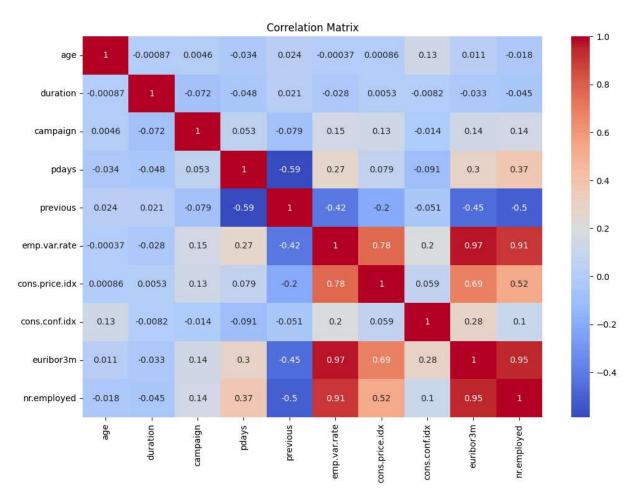
y no 36548 yes 4640 Name: count, dtype: int64

## Distribution of Target Variable (y)



```
In []: # Select only numerical columns for correlation matrix
   numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns
   corr_matrix = df[numerical_columns].corr()

# Plot the heatmap
   plt.figure(figsize=(12, 8))
   sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
   plt.title('Correlation Matrix')
   plt.show()
```



```
In []: # Encode categorical variables using LabelEncoder
from sklearn.preprocessing import LabelEncoder

label_encoders = {}
categorical_columns = df.select_dtypes(include=['object']).columns

for col in categorical_columns:
    label_encoders[col] = LabelEncoder()
    df[col] = label_encoders[col].fit_transform(df[col])

# Compute the correlation matrix including encoded categorical variables
corr_matrix = df.corr()

# heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Including Encoded Categorical Variables')
plt.show()
```

```
Correlation Matrix Including Encoded Categorical Variables
                                                                                                                        1.0
                      100.00120.39-0.12 0.160.00160.074.0070.0250.01800068700460.0340.0240.020.006870086.130.0110.0180.03
                  job e.001; 1 0.028<mark>0.13</mark>-0.0280.007-0.010.0250.03B00084006050069.0280.0210.01<del>2</del>0.0088.0160.0530.00790.020.025
               marital -0.390.028 1 0.11-0.0790.010.005@.056.007600220.010.0072.0380.039.001@.0840.0570.0340.0920.0860.046
                                                                                                                       - 0.8
             education --0.12 0.13 0.11 1 -0.190.0170.00640.110.0830.0180.015000307.0470.0390.0170.0440.0820.0790.0360.0410.058
               default -0.16-0.0280.0790.19 1 0.016.00380.14-0.016.0080.0120.0330.08 -0.1 0.023 0.2 0.170.027 0.2 0.19-0.099
                                                                                                                       - 0.6
              housing -0.0016.007 0.010.0170.016 1 0.0440.0820.018.0036.00770.0130.0130.0120.060.0830.0340.0590.0460.012
                 loan -0.00740.010.005880066.0038.044 1 1 0.00860057.00930009200520003500128.001500148.00249.0112000153039.0049
              contact -0.0070.0250.0550.11 0.140.080.008 1 0.280.0096.0270.0770.12 -0.21 0.12 0.39 0.59 0.25 0.4 0.27 -0.14
                                                                                                                       - 0.4
               month -0.0250.036.0076.0830.0160.0160.00570.28 1 0.0280.00370.0620.048 0.1 -0.0650.180.0040200970.12 -0.250.0061
          day_of_week -0.0308.0006.40230.01-8.0080700308.00908.00906.028 1 0.0220.038.00905.0040.0190.0380.0056.0410.0390.0280.016
              - 0.2
             campaign 6.0046.0069.0000200030.0330.010.00520.0770.0620.0380.072 1 0.0530.0790.0330.15 0.13-0.0140.14 0.14-0.066
                pdays -0.0340.0280.0380.0470.08-0.0101000315.12-0.048.00915.0480.053 1 -0.59-0.48 0.27 0.0790.091 0.3 0.37 -0.32
                                                                                                                       - 0.0
             previous -0.0240.0210.0390.039 -0.1 0.0240.00130.21 0.1-0.0040.0210.079 0.59 1 -0.31 -0.42 -0.2-0.051 0.45 -0.5 0.23
            poutcome - 0.020.012.0019.0170.0230.012.00150.12-0.0650.0190.0330.033-0.48-0.31 1 0.19 0.21 0.18 0.18 0.12 0.13
          emp.var.rate9.0008.008.00840.044 0.2 -0.06.00180.39 -0.180.0330.0280.15 0.27 -0.42 0.19 1 0.78 0.2 0.97 0.91 -0.3
                                                                                                                       - -0.2
         cons.price.idx0,00086.0160.0570.0820.17-0.0810.00240.590.0042005600530.130.079 -0.2 0.21 0.78 1 0.059 0.69 0.52 -0.14
          cons.conf.idx -0.130.0530.0340.0790.0270.0340.0120.250.00970.0470.00870.0140.0930.0510.18 0.2 0.059 1 0.28 0.1 0.055
            euribor3m -0.01-0.007-0.0920.036 0.2 -0.0509000130.4 -0.120.0390.0330.14 0.3 -0.45 0.18 0.97 0.69 0.28 1 0.95 -0.31
                                                                                                                       -0.4
          nr.employed -0.0180.020.0860.0410.19-0.046.00390.27 -0.22 0.0280.0450.14 0.37 -0.5 0.12 0.91 0.52 0.1 0.95 1
                    y-0.030.0250.0460.0580.0990.01<del>2</del>0.00490.140.0060.016<mark>0.41</mark>-0.06€0.32 <mark>0.23 0.13 -</mark>0.3 -0.140.055-0.31-0.35
                                                                                              cons.conf.idx
In [ ]: # Assign X (features) and y (target)
           X = df.drop('y_yes', axis=1)
           y = df['y_yes']
           # Split the data into training and testing sets
           X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
In [ ]: # Initialize the classifier
           dt classifier = DecisionTreeClassifier(random state=42)
           # Train the classifier on the training data
           dt_classifier.fit(X_train, y_train)
Out[ ]: ▼
                         DecisionTreeClassifier
           DecisionTreeClassifier(random state=42)
In [ ]: # Predict on the test data
           y_pred = dt_classifier.predict(X_test)
           # Evaluate the model
           from sklearn.metrics import accuracy score, classification report, confusion matrix
           print("Accuracy:", accuracy_score(y_test, y_pred))
           print("\nClassification Report:\n", classification_report(y_test, y_pred))
           print("\nConfusion Matrix:\n", confusion matrix(y test, y pred))
```

Accuracy: 0.8866229667394999

Classification Report:

	precision	recall	f1-score	support
False	0.94	0.93	0.94	7303
True	0.50	0.52	0.51	935
accuracy			0.89	8238
macro avg	0.72	0.73	0.72	8238
weighted avg	0.89	0.89	0.89	8238

Confusion Matrix:

[[6816 487] [ 447 488]]

The performance metrics of a decision tree classifier trained on demographic and behavioral data to predict whether customers will purchase a product or service. The classifier achieved an accuracy of approximately 88.7%, indicating that it correctly predicted the outcome for nearly 89% of the instances in the test set. The classification report reveals that the model performs well in identifying non-purchasing customers (precision of 94% and recall of 93%), but less effectively for purchasing customers (precision of 50% and recall of 52%). The confusion matrix further illustrates these results, showing 6816 true negatives, 487 false negatives, 447 false positives, and 488 true positives