skillcraft-task-3

July 5, 2025

[4]:
```
# Unzipping the second uploaded file
zip_path_2 = "/mnt/data/unzipped_bank_marketing/bank-additional.zip" #␣
 ↪Corrected path to the second zip file
extract_dir_2 = "/mnt/data/unzipped_bank_marketing_2"

# Create the directory if it doesn't exist
os.makedirs(extract_dir_2, exist_ok=True)

# Extract the contents of the second zip file
with zipfile.ZipFile(zip_path_2, 'r') as zip_ref:
    zip_ref.extractall(extract_dir_2)

# List the extracted files
extracted_files_2 = os.listdir(extract_dir_2)
extracted_files_2
```

[4]:
```
['__MACOSX', 'bank-additional']
```

[6]:
```
# Path to the inner bank.zip file
inner_zip_path = os.path.join("/mnt/data/unzipped_bank_marketing", 'bank.zip')
inner_extract_dir = "/mnt/data/bank_data"

# Create directory and unzip
os.makedirs(inner_extract_dir, exist_ok=True)
with zipfile.ZipFile(inner_zip_path, 'r') as zip_ref:
    zip_ref.extractall(inner_extract_dir)

# List the extracted files
inner_extracted_files = os.listdir(inner_extract_dir)
inner_extracted_files
```

[6]:
```
['bank-full.csv', 'bank.csv', 'bank-names.txt']
```

[7]:
```
import pandas as pd

# Load the dataset
data_path = os.path.join(inner_extract_dir, 'bank-full.csv')
```

```
df = pd.read_csv(data_path, sep=';')

# Display basic information and the first few rows
df.info(), df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

[7]:
```
(None,
    age           job  marital  education default  balance housing loan  \
 0   58    management  married   tertiary      no     2143     yes   no
 1   44    technician   single  secondary      no       29     yes   no
 2   33  entrepreneur  married  secondary      no        2     yes  yes
 3   47   blue-collar  married    unknown      no     1506     yes   no
 4   33       unknown   single    unknown      no        1      no   no

    contact  day month  duration  campaign  pdays  previous poutcome   y
 0  unknown    5   may       261         1     -1         0  unknown  no
 1  unknown    5   may       151         1     -1         0  unknown  no
 2  unknown    5   may        76         1     -1         0  unknown  no
 3  unknown    5   may        92         1     -1         0  unknown  no
 4  unknown    5   may       198         1     -1         0  unknown  no  )
```

[9]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
import os # Import the os module

# Load the dataset
data_path = os.path.join("/mnt/data/bank_data", 'bank-full.csv') # Use the␣
 ↪correct path
df = pd.read_csv(data_path, sep=';')

# Encode categorical variables
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

# Define features and target
X = df.drop('y', axis=1)  # features
y = df['y']               # target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Train Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

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

```
Accuracy: 0.8737835446770864
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93     11966
           1       0.46      0.47      0.47      1598

    accuracy                           0.87     13564
   macro avg       0.70      0.70      0.70     13564
weighted avg       0.87      0.87      0.87     13564
```