3. Using the "bank-full" dataset, perform the following tasks with detailed analysis and appropriate visualizations: [In Python & R]

i. Load the dataset and examine its structure using basic commands.

```
import pandas as pd
import seaborn as sns
from sklearn import tree

df = pd.read_csv("D:/PYTHON/DATA SCIENCE/DATA/bank-full.csv")

df

       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
...    ...           ...       ...        ...     ...      ...     ...
...
45206   51    technician   married   tertiary      no      825      no
no
45207   71       retired  divorced    primary      no     1729      no
no
45208   72       retired   married  secondary      no     5715      no
no
45209   57   blue-collar   married  secondary      no      668      no
no
45210   37  entrepreneur   married  secondary      no     2971      no
no

         contact  day month  duration  campaign  pdays  previous
poutcome  \
0        unknown    5   may       261         1     -1         0
unknown
1        unknown    5   may       151         1     -1         0
unknown
2        unknown    5   may        76         1     -1         0
unknown
3        unknown    5   may        92         1     -1         0
unknown
4        unknown    5   may       198         1     -1         0
```

```
unknown
...          ...  ...   ...       ...       ...    ...       ...
...
45206   cellular   17   nov       977         3     -1         0
unknown
45207   cellular   17   nov       456         2     -1         0
unknown
45208   cellular   17   nov      1127         5    184         3
success
45209  telephone   17   nov       508         4     -1         0
unknown
45210   cellular   17   nov       361         2    188        11
other

      Target
0         no
1         no
2         no
3         no
4         no
...      ...
45206    yes
45207    yes
45208    yes
45209     no
45210     no

[45211 rows x 17 columns]
```

```
df.info()
```

```
<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   Target      45211 non-null   object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

Displays the data types and number of non-null values, helping identify missing data and column types.

```
df.head()
```

```
   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 Target
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
```

Displays the first few rows of the dataset, allowing you to quickly preview the data and check its structure.

```
df.describe(include='object')
```

```
                job  marital  education default housing   loan  contact  \
count         45211    45211      45211   45211   45211  45211    45211
unique           12        3          4       2       2      2        3
top     blue-collar  married  secondary      no     yes     no cellular
```

```
freq           9732    27214      23202   44396   25130  37967
29285

       month poutcome Target
count   45211    45211  45211
unique     12        4      2
top       may  unknown     no
freq    13766    36959  39922
```

```
df.describe()
```

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

              pdays      previous
count  45211.000000  45211.000000
mean      40.197828      0.580323
std      100.128746      2.303441
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       -1.000000      0.000000
max      871.000000    275.000000
```

Provides statistical summaries of numerical columns, showing their distribution and key metrics.

```
df.shape
```

```
(45211, 17)
```

Provides the number of rows and columns in the dataset, helping to understand its size.

```
df
```

```
       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
...    ...           ...       ...        ...     ...      ...     ...
...
45206   51    technician   married   tertiary      no      825      no
no
45207   71       retired  divorced    primary      no     1729      no
no
45208   72       retired   married  secondary      no     5715      no
no
45209   57   blue-collar   married  secondary      no      668      no
no
45210   37  entrepreneur   married  secondary      no     2971      no
no

         contact  day month  duration  campaign  pdays  previous
poutcome  \
0        unknown    5   may       261         1     -1         0
unknown
1        unknown    5   may       151         1     -1         0
unknown
2        unknown    5   may        76         1     -1         0
unknown
3        unknown    5   may        92         1     -1         0
unknown
4        unknown    5   may       198         1     -1         0
unknown
...          ...  ...   ...       ...       ...    ...       ...
...
45206   cellular   17   nov       977         3     -1         0
unknown
45207   cellular   17   nov       456         2     -1         0
unknown
45208   cellular   17   nov      1127         5    184         3
success
45209  telephone   17   nov       508         4     -1         0
unknown
45210   cellular   17   nov       361         2    188        11
other
```

```
      Target
0         no
1         no
2         no
3         no
4         no
...      ...
45206    yes
45207    yes
45208    yes
45209     no
45210     no

[45211 rows x 17 columns]
```

ii. Create a new variable called "conversion" by transforming the categorical values in the "Target" column into numerical representations.

```
df['conversion']= df['Target'].apply (lambda x:0 if x == 'no' else 1)

df
```

```
       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
...    ...           ...       ...        ...     ...      ...     ...
...
45206   51    technician   married   tertiary      no      825      no
no
45207   71       retired  divorced    primary      no     1729      no
no
45208   72       retired   married  secondary      no     5715      no
no
45209   57   blue-collar   married  secondary      no      668      no
no
45210   37  entrepreneur   married  secondary      no     2971      no
no
```

```
          contact  day month  duration  campaign  pdays  previous
poutcome  \
0         unknown    5   may       261         1     -1         0
unknown
1         unknown    5   may       151         1     -1         0
unknown
2         unknown    5   may        76         1     -1         0
unknown
3         unknown    5   may        92         1     -1         0
unknown
4         unknown    5   may       198         1     -1         0
unknown
...           ...  ...   ...       ...       ...    ...       ...
...
45206    cellular   17   nov       977         3     -1         0
unknown
45207    cellular   17   nov       456         2     -1         0
unknown
45208    cellular   17   nov      1127         5    184         3
success
45209   telephone   17   nov       508         4     -1         0
unknown
45210    cellular   17   nov       361         2    188        11
other

      Target  conversion
0         no           0
1         no           0
2         no           0
3         no           0
4         no           0
...      ...         ...
45206    yes           1
45207    yes           1
45208    yes           1
45209     no           0
45210     no           0

[45211 rows x 18 columns]
```

```
df['conversion']
```

```
0        0
1        0
2        0
3        0
4        0
        ..
45206    1
45207    1
```

```
45208    1
45209    0
45210    0
Name: conversion, Length: 45211, dtype: int64
```

```
df.columns
```

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

iii. Calculate and interpret the Conversion Rate. How does the code implement this calculation, and what does it reveal about the target variable distribution?

```
# Calculate conversion rate
conversion_rate = df['conversion'].mean()
print(f"Conversion Rate: {conversion_rate:.2%}")
```

```
Conversion Rate: 11.70%
```

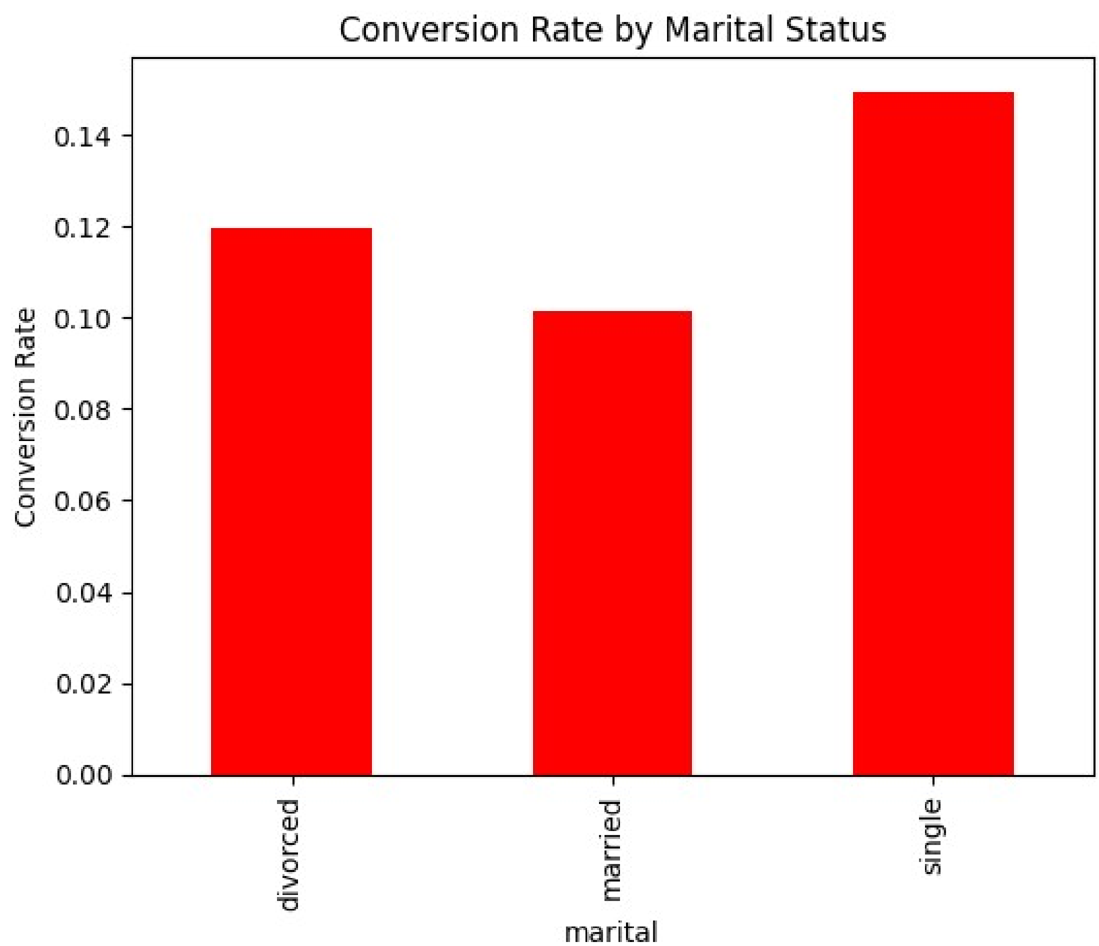
Interpretation:

The conversion rate is 11.70%, meaning about 12 out of every 100 users completed the desired action, such as making a purchase or signing up. This indicates the effectiveness of the campaign or funnel.

iv. Analyze and visualize Conversion Rates by Marital Status: Explain how conversion rates are computed for each marital status. Create a bar chart to display these rates and interpret the visualization.

```
# Group by marital status and calculate conversion rate
marital_conversion = df.groupby('marital')['conversion'].mean()

# Bar chart
import matplotlib.pyplot as plt
marital_conversion.plot(kind='bar', color='red')
plt.title("Conversion Rate by Marital Status")
plt.ylabel("Conversion Rate")
plt.show()
```



Interpretation:

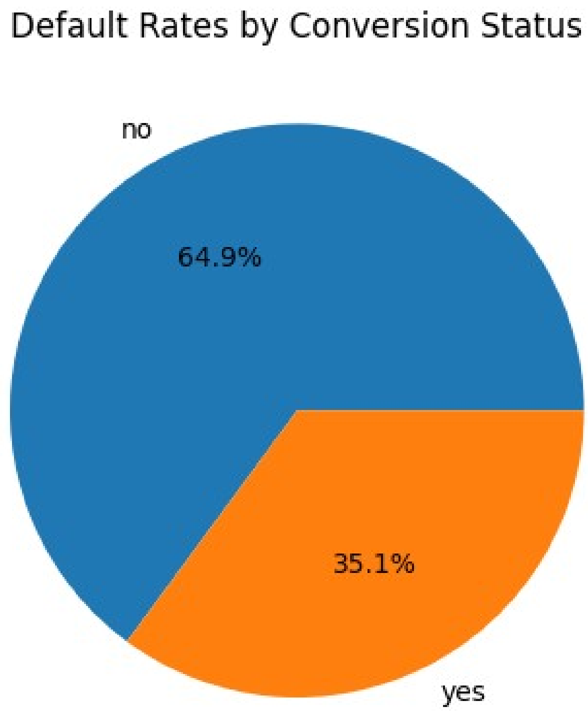
The bar chart visualizes conversion rates by marital status, highlighting differences in conversion performance across groups.

v. Investigate Default Rates by Conversion Status using a pivot table and pie chart visualizations. What insights can you draw from these visual representations?

```
# Pivot table
default_pivot = df.pivot_table(values='conversion', index='default',
aggfunc='mean')
default_pivot
```

	conversion
default	
no	0.117961
yes	0.063804

```
# Pie chart
default_pivot.plot(kind='pie', subplots=True, autopct='%1.1f%%',
legend=False)
plt.title("Default Rates by Conversion Status")
plt.ylabel("")
plt.show()
```

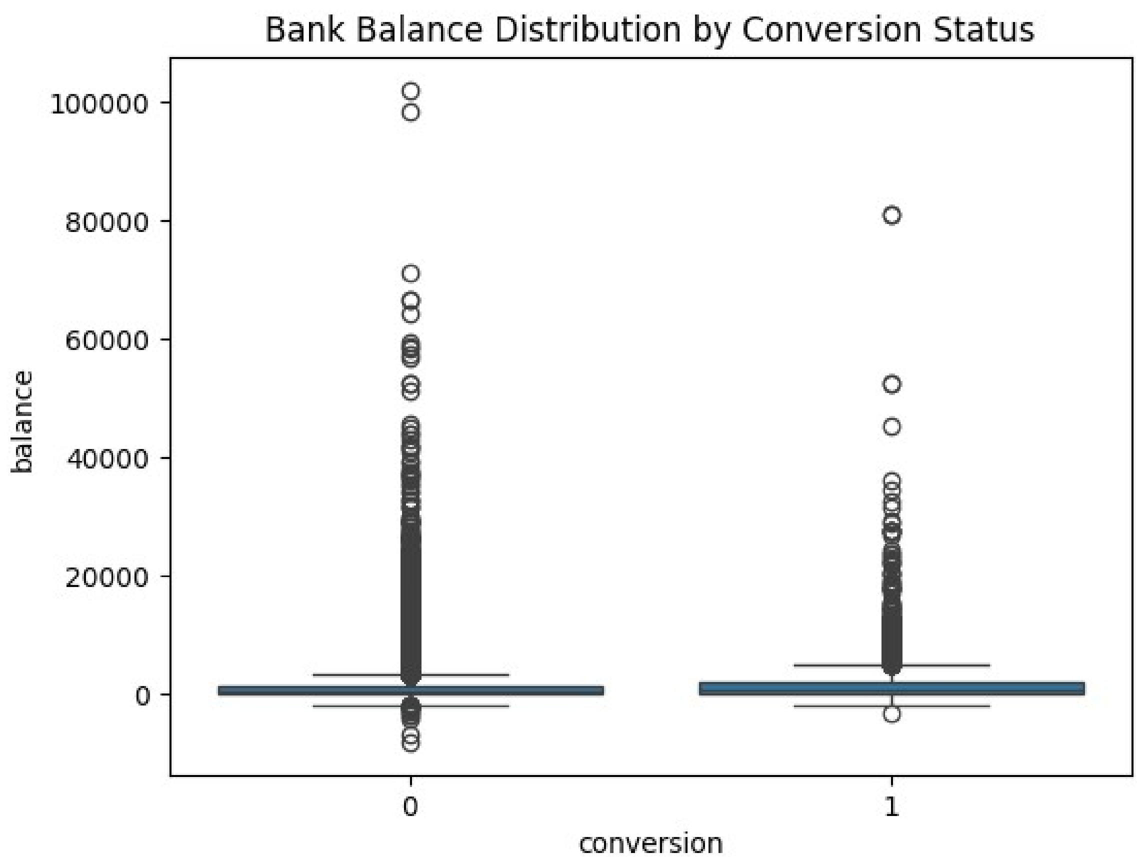


Interpretation:

The pivot table shows the average conversion rate based on default status, while the pie chart highlights which group (default or non-default) converts more effectively

vi. Use a boxplot to analyze the relationship between conversion status and bank balance distributions. Why are outliers excluded, and what does the plot tell you about customer balance patterns?

```
# Boxplot
sns.boxplot(x='conversion', y='balance', data=df)
plt.title("Bank Balance Distribution by Conversion Status")
plt.show()
```



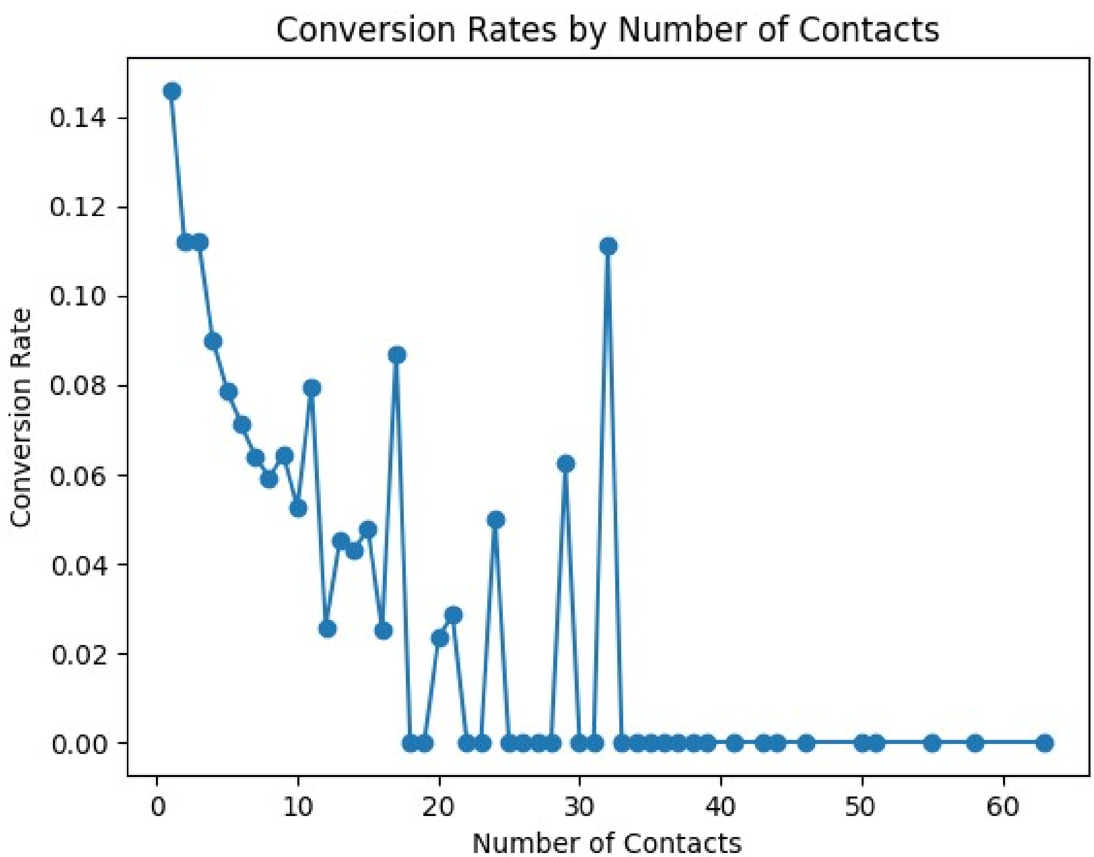
Interpretation:

The boxplot, excluding outliers, clarifies trends by preventing skewed data. It compares bank balances of converted and non-converted customers, highlighting wealth-related conversion patterns.

vii. Explore Conversion Rates by Number of Contacts (campaign): Describe the method used to calculate these rates, and explain why this metric is significant in a marketing campaign.

```
# Calculate conversion rates
contact_conversion = df.groupby('campaign')['conversion'].mean()

# Visualization
contact_conversion.plot(kind='line', marker='o')
plt.title("Conversion Rates by Number of Contacts")
plt.xlabel("Number of Contacts")
plt.ylabel("Conversion Rate")
plt.show()
```



Interpretation:

The conversion rate is averaged for each contact count, revealing how conversion changes with contact frequency. This insight helps optimize outreach for better marketing effectiveness.

viii. Describe how to encode categorical variables, such as job, marital, housing, and loan, for machine learning models.

```
# One-hot encoding
df_encoded = pd.get_dummies(df, columns=['job', 'marital', 'housing',
'loan'], drop_first=True)
```

Interpretation:

One-hot encoding converts categorical variables (**job, marital, housing, loan**) into binary columns, allowing machine learning models to process them without assuming any inherent order.

ix. Build a Decision Tree Model using the provided features:

Explain the selection of features and the target variable. Visualize the decision tree using appropriate plotting techniques. How does this visualization help in understanding the decision-making process of the model?

```
dt_model = tree.DecisionTreeClassifier(max_depth=4)

dt_model

DecisionTreeClassifier(max_depth=4)

encoded_df = pd.get_dummies(df,columns=['job','marital'],dtype=int)
full_df =  pd.concat([df,encoded_df],axis=1)
df=full_df.loc[:,~full_df.columns.duplicated()]
df['housing'] = df['housing'].apply(lambda x:0 if x == 'no'
                                 else 1)

df['loan'] = df['loan'].apply(lambda x:0 if x == 'no'
                                 else 1)

C:\Users\AJITH N\AppData\Local\Temp\ipykernel_2652\3956986206.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['housing'] = df['housing'].apply(lambda x:0 if x == 'no'
C:\Users\AJITH N\AppData\Local\Temp\ipykernel_2652\3956986206.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['loan'] = df['loan'].apply(lambda x:0 if x == 'no'

list(df.columns)

['age',
 'job',
 'marital',
 'education',
 'default',
 'balance',
 'housing',
 'loan',
 'contact',
 'day',
 'month',
 'duration',
 'campaign',
 'pdays',
 'previous',
```

```
 'poutcome',
 'Target',
 'conversion',
 'job_admin.',
 'job_blue-collar',
 'job_entrepreneur',
 'job_housemaid',
 'job_management',
 'job_retired',
 'job_self-employed',
 'job_services',
 'job_student',
 'job_technician',
 'job_unemployed',
 'job_unknown',
 'marital_divorced',
 'marital_married',
 'marital_single']
```

```
response_var = 'conversion'
features = ['age',  'balance', 'housing', 'campaign',
       'previous', 'job_admin.', 'job_blue-collar',
'job_entrepreneur',
       'job_housemaid', 'job_management', 'job_retired', 'job_self-
employed',
       'job_services', 'job_student', 'job_technician',
'job_unemployed',
       'job_unknown', 'marital_divorced', 'marital_married',
'marital_single']


dt_model.fit(df[features],df[response_var])
```

```
DecisionTreeClassifier(max_depth=4)
```

```
dt_model.classes_
```

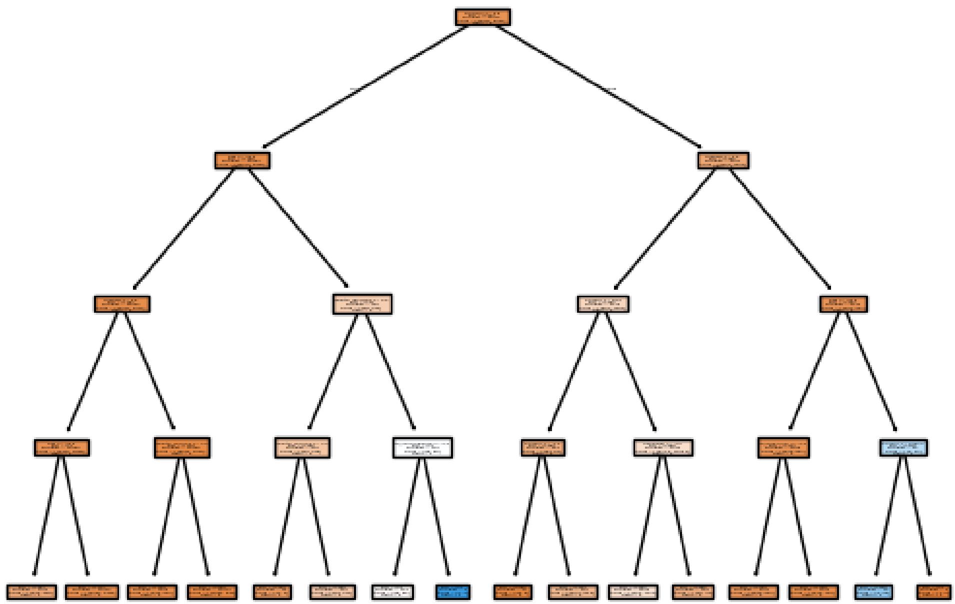
```
array([0, 1])
```

```
dt_model.feature_importances_
```

```
array([0.22958146, 0.03357839, 0.39439427, 0.01104479, 0.30370049,
       0.        , 0.00785975, 0.        , 0.        , 0.        ,
       0.        , 0.00235943, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.0061807 , 0.01130072, 0.        ])
```

```
import matplotlib.pyplot as plt
class_names = [str(label) for label in dt_model.classes_]
#plt.figure(figsize=(15,10))#
tree.plot_tree(dt_model,
               feature_names=features,
```

```
                  class_names=class_names,
                  filled=True)
plt.show()
```



Interpretation:

The decision tree predicts conversion using features like age, income, and job. Visualization reveals how the model splits data at each node, highlighting key decision factors.