

## Task3

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn import tree
```

```
df = pd.read_csv("C:\\Users\\hp\\Internship\\bank-additional.csv",
sep=';')
```

```
df.shape
```

```
(4119, 21)
```

```
df.head()
```

	age	job	marital	education	default	housing
loan \						
0	30	blue-collar	married	basic.9y	no	yes
no						
1	39	services	single	high.school	no	no
no						
2	25	services	married	high.school	no	yes
no						
3	38	services	married	basic.9y	no	unknown
unknown						
4	47	admin.	married	university.degree	no	yes
no						

	contact	month	day_of_week	...	campaign	pdays	previous
poutcome \							
0	cellular	may	fri	...	2	999	0
nonexistent							
1	telephone	may	fri	...	4	999	0
nonexistent							
2	telephone	jun	wed	...	1	999	0
nonexistent							
3	telephone	jun	fri	...	3	999	0
nonexistent							
4	cellular	nov	mon	...	1	999	0
nonexistent							

emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
--------------	----------------	---------------	-----------	-------------

y					
0	-1.8	92.893	-46.2	1.313	5099.1
no					
1	1.1	93.994	-36.4	4.855	5191.0
no					
2	1.4	94.465	-41.8	4.962	5228.1
no					
3	1.4	94.465	-41.8	4.959	5228.1
no					
4	-0.1	93.200	-42.0	4.191	5195.8
no					

[5 rows x 21 columns]

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 4119 entries, 0 to 4118

Data columns (total 21 columns):

#	Column	Non-Null Count	Dtype
0	age	4119 non-null	int64
1	job	4119 non-null	object
2	marital	4119 non-null	object
3	education	4119 non-null	object
4	default	4119 non-null	object
5	housing	4119 non-null	object
6	loan	4119 non-null	object
7	contact	4119 non-null	object
8	month	4119 non-null	object
9	day_of_week	4119 non-null	object
10	duration	4119 non-null	int64
11	campaign	4119 non-null	int64
12	pdays	4119 non-null	int64
13	previous	4119 non-null	int64
14	poutcome	4119 non-null	object
15	emp.var.rate	4119 non-null	float64
16	cons.price.idx	4119 non-null	float64
17	cons.conf.idx	4119 non-null	float64
18	euribor3m	4119 non-null	float64
19	nr.employed	4119 non-null	float64
20	y	4119 non-null	object

dtypes: float64(5), int64(5), object(11)

memory usage: 675.9+ KB

df.isnull().sum()

age	0
job	0
marital	0

```

education      0
default        0
housing        0
loan           0
contact        0
month          0
day_of_week    0
duration       0
campaign       0
pdays        0
previous       0
poutcome       0
emp.var.rate   0
cons.price.idx 0
cons.conf.idx  0
euribor3m      0
nr.employed    0
y              0
dtype: int64

label = LabelEncoder()
for column in df.columns:
    if df[column].dtype == 'object':
        df[column] = label.fit_transform(df[column])
X = df.drop('y', axis=1)
y = df['y']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

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

DecisionTreeClassifier(criterion='entropy', max_depth=5,
random_state=42)

#Make Predictions
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))

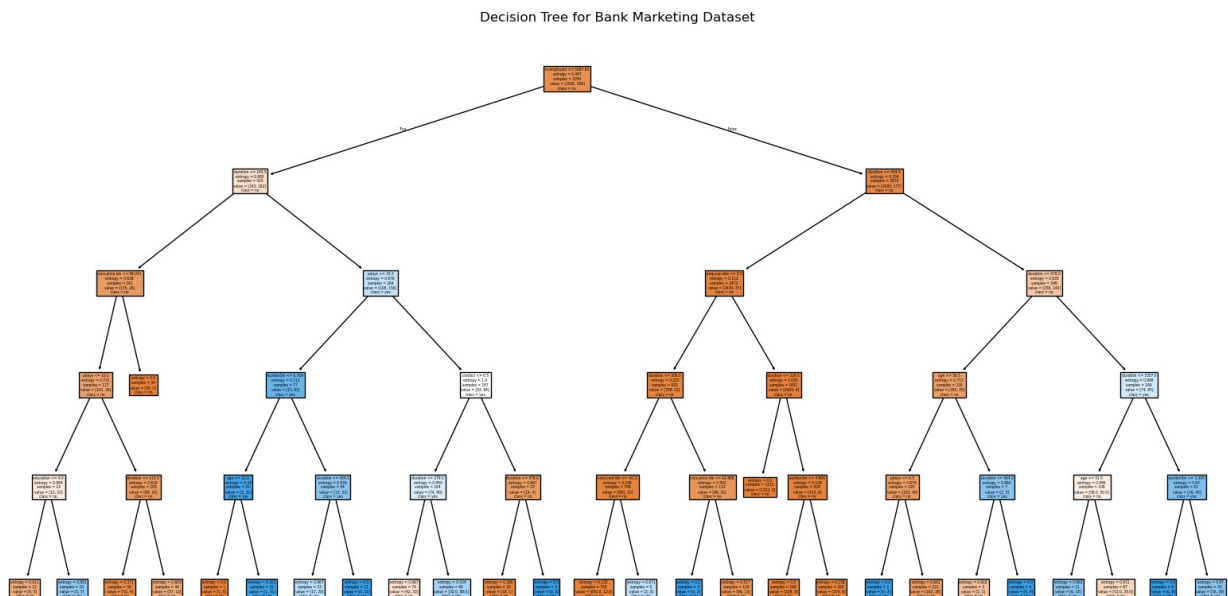
Accuracy: 0.9029126213592233

Confusion Matrix:
[[700  32]
 [ 48  44]]

```

Classification Report:				
	precision	recall	f1-score	support
0	0.94	0.96	0.95	732
1	0.58	0.48	0.52	92
accuracy			0.90	824
macro avg	0.76	0.72	0.73	824
weighted avg	0.90	0.90	0.90	824

```
plt.figure(figsize=(20,10))
tree.plot_tree(model, filled=True, feature_names=X.columns,
class_names=['no','yes'])
plt.title("Decision Tree for Bank Marketing Dataset")
plt.show()
```



```
importances = pd.Series(model.feature_importances_, index=X.columns)
importances = importances.sort_values(ascending=False)
plt.figure(figsize=(10,6))
sns.barplot(x=importances, y=importances.index)
plt.title("Feature Importance")
plt.show()
```

