Assignment on Classification technique

Every year many students give the GRE exam to get admission in foreign Universities. The data set contains GRE Scores (out of 340), TOEFL Scores (out of 120), University Rating (out of 5), Statement of Purpose strength (out of 5), Letter of Recommendation strength (out of 5), Undergraduate GPA (out of 10), Research Experience (0=no, 1=yes), Admitted (0=no, 1=yes). Admitted is the target variable. Data Set Available on kaggle (The last column of the dataset needs to be changed to 0 or 1)Data Set : https://www.kaggle.com/mohansacharya/graduate-admissions The counselor of the firm is supposed check whether the student will get an admission or not based on his/her GRE score and Academic Score. So to help the counselor to take appropriate decisions build a machine learning model classifier using Decision tree to predict whether a student will get admission or not. Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if necessary. Perform data-preparation (Train-Test Split) C. Apply Machine Learning Algorithm D. Evaluate Model.

In [ ]:
```
#Loading the essential libraries
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.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report, confusion_m
```

In [ ]:
```
# Loading the dataset
df = pd.read_csv('Admission_Pred.csv')
df.head()
```

Out[ ]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

In [ ]:
```
#Understanding the data
df.info()
df.describe()
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

Out[ ]:

	0
Serial No.	0
GRE Score	0
TOEFL Score	0
University Rating	0
SOP	0
LOR	0
CGPA	0
Research	0
Chance of Admit	0

dtype: int64

In [ ]:

```
# Convert target variable to binary (classification)
df['Admitted'] = df['Chance of Admit '].apply(lambda x: 1 if x >= 0.75 else 0)
df.head()


#We can remove the unnecessary column of chance of admit
#df = df.drop(columns=['Chance of Admit ', 'Serial No.'])
```

Out[ ]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	Admitt
0	1	337	118	4	4.5	4.5	9.65	1	0.92	
1	2	324	107	4	4.0	4.5	8.87	1	0.76	
2	3	316	104	3	3.0	3.5	8.00	1	0.72	
3	4	322	110	3	3.5	2.5	8.67	1	0.80	
4	5	314	103	2	2.0	3.0	8.21	0	0.65	

In [ ]:
```
#Selecting features here x is GPA, TOEFL and CGPA and also considering the adm
X = df[['GRE Score', 'CGPA']]
y = df['Admitted']
```

In [ ]:
```
#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, rand
```

In [ ]:
```
# Create the model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
```

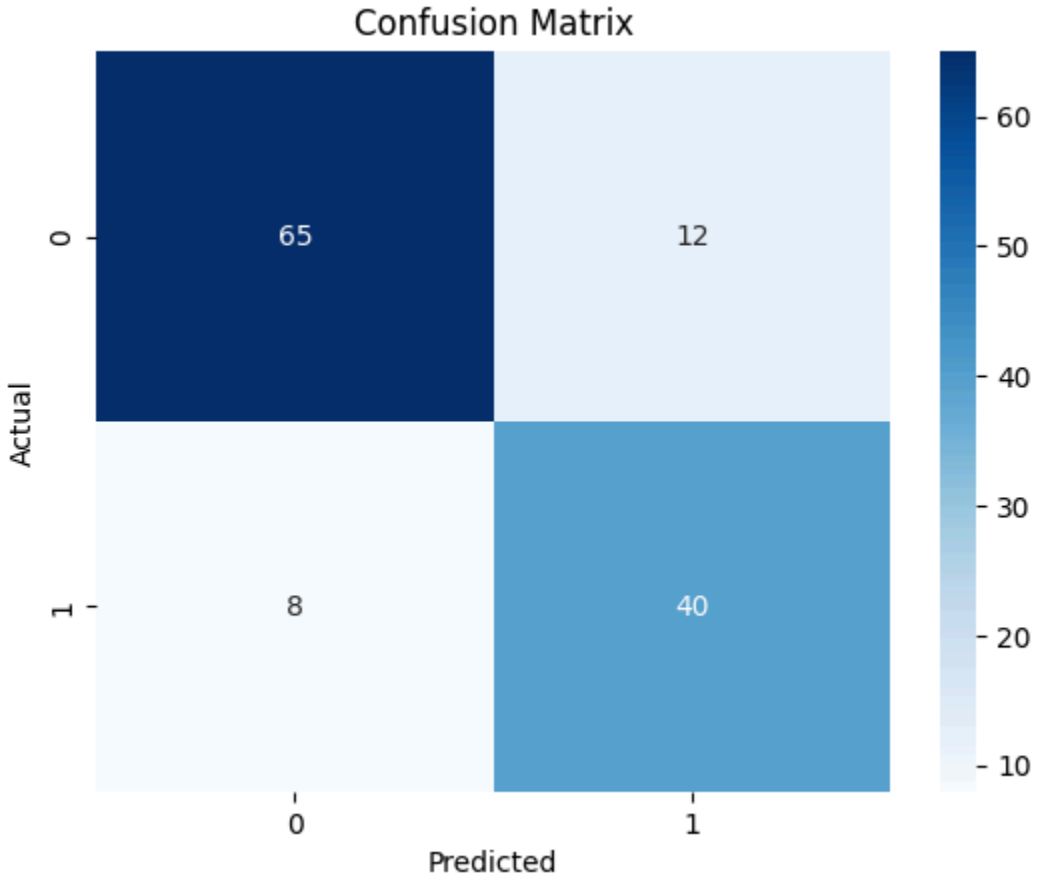
Out[ ]:

▾ DecisionTreeClassifier

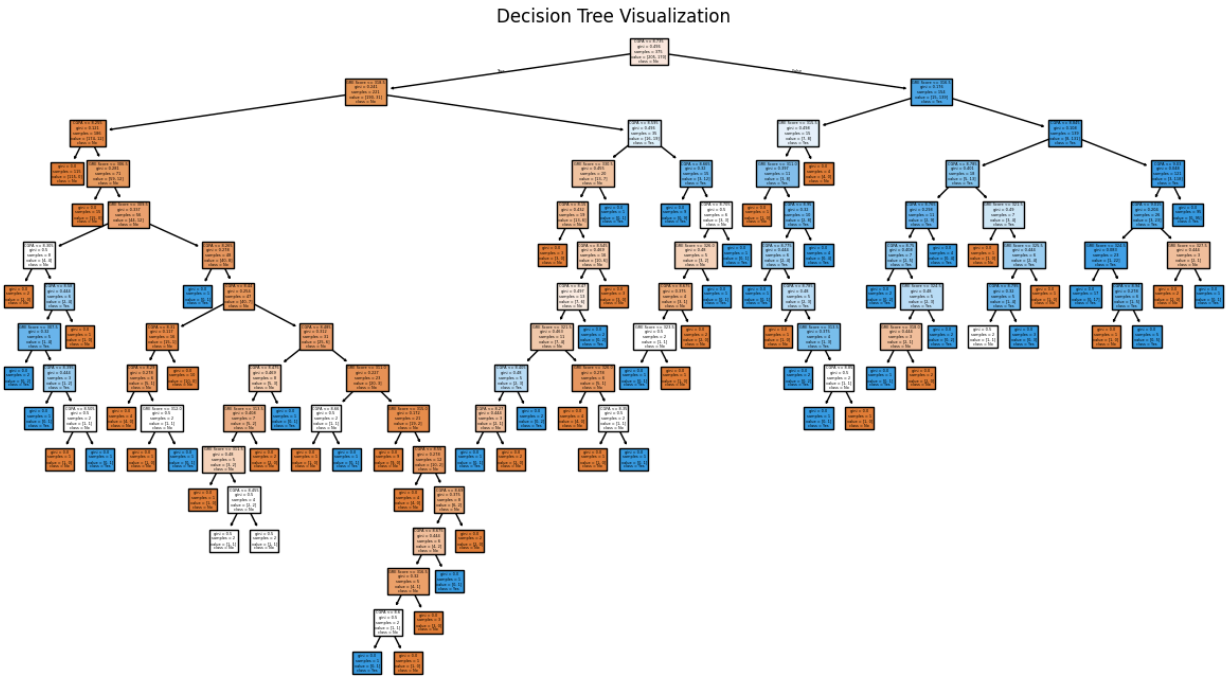
```
DecisionTreeClassifier(random_state=42)
```

In [ ]:
```
# Make predictions
y_pred = model.predict(X_test)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))

# Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.84

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.84      0.87        77
           1       0.77      0.83      0.80        48

    accuracy                           0.84       125
   macro avg       0.83      0.84      0.83       125
weighted avg       0.84      0.84      0.84       125
```



In [ ]:
```
plt.figure(figsize=(15, 8))
plot_tree(model, filled=True, feature_names=['GRE Score', 'CGPA'], class_names
plt.title("Decision Tree Visualization")
plt.show()
```



In [ ]: