Assignment 5- Decision Trees

Problem Statement-

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. A.Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if necessary. B.Perform data-preparation (Train-Test Split) C. Apply Machine Learning Algorithm D. Evaluate Model.

Importing python libraries

In [1]:

```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import *
```

loading the csv file into a dataframe

In [2]:

```python
A=pd.read_csv(r"C:\Users\HP\Downloads\Admission_Predict.csv")
A
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

head() function used to access the first n rows of a dataframe

In [3]:

```
A.head(5)
```

Out[3]:

	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

describe() function returns the description of data in dataframe

In [4]:

```
A.describe()
```

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Rese
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.00
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.54
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.49
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.00
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.00
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.00
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.00
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.00

In [5]:

```
A.dtypes
```

Out[5]:

```
Serial No.             int64
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
```

counting the total number of null values in each column

In [6]:

```
A.isnull().sum()
```

Out[6]:

```
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
```

Dropping the column "Serial No"

In [7]:

```
A.drop('Serial No.',axis=1)
```

Out[7]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1	0.82
396	325	107	3	3.0	3.5	9.11	1	0.84
397	330	116	4	5.0	4.5	9.45	1	0.91
398	312	103	3	3.5	4.0	8.78	0	0.67
399	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 8 columns

Assigning the independent variable

In [8]:

```
x=A[["GRE Score","TOEFL Score", "University Rating" ,"SOP","LOR","CGPA","Research"
```

Changing the values of column "Chance of Admit" to 0 and 1

In [9]:

```
A["Chance of Admit"]=[1 if each > 0.8 else 0 for each in A["Chance of Admit"]]
A["Chance of Admit"]
```

Out[9]:

```
0      1
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Chance of Admit, Length: 400, dtype: int64
```

In [10]:

```
y=A["Chance of Admit"]
y
```

Out[10]:

```
0      1
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Chance of Admit, Length: 400, dtype: int64
```

Dividing the dataset into training and testing data

In [11]:

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

creating the object using DecisionTreeClassifier class and fitting the model

In [12]:

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(random_state=1,max_depth=2,criterion='gini').fit(x_tra
```

In [13]:

```
features = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research']
```

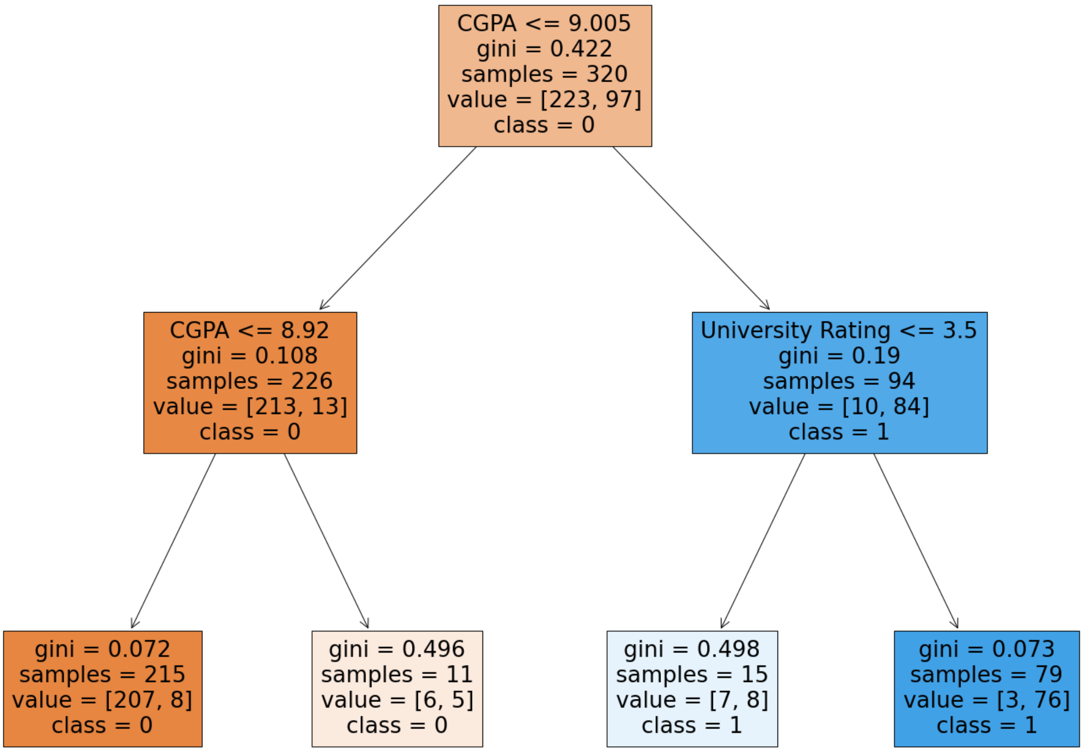
creating decision tree

In [19]:

```
fig = plt.figure(figsize=(25,20))
from sklearn import tree
tree.plot_tree(model,
                   feature_names=features,
                   class_names=['0','1'],
                   filled=True)
```

Out[19]:

```
[Text(0.5, 0.8333333333333334, 'CGPA <= 9.005\ngini = 0.422\nsamples = 320\nvalue
= [223, 97]\nclass = 0'),
 Text(0.25, 0.5, 'CGPA <= 8.92\ngini = 0.108\nsamples = 226\nvalue = [213, 13]\ncl
ass = 0'),
 Text(0.125, 0.16666666666666666, 'gini = 0.072\nsamples = 215\nvalue = [207, 8]\n
class = 0'),
 Text(0.375, 0.16666666666666666, 'gini = 0.496\nsamples = 11\nvalue = [6, 5]\ncla
ss = 0'),
 Text(0.75, 0.5, 'University Rating <= 3.5\ngini = 0.19\nsamples = 94\nvalue = [1
0, 84]\nclass = 1'),
 Text(0.625, 0.16666666666666666, 'gini = 0.498\nsamples = 15\nvalue = [7, 8]\ncla
ss = 1'),
 Text(0.875, 0.16666666666666666, 'gini = 0.073\nsamples = 79\nvalue = [3, 76]\ncl
ass = 1')]
```



predicting the labels of data values

In [15]:

```
x_pred=model.predict(x_test)
print(x_pred)
```

```
[0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 1 0
 1 0 0 0 1 0]
```

Creating confusion matrix

In [16]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,x_pred)
```

Out[16]:

```
array([[58,  2],
       [ 3, 17]], dtype=int64)
```

Calculating accuracy metrics

In [17]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test,x_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        60
           1       0.89      0.85      0.87        20

    accuracy                           0.94        80
   macro avg       0.92      0.91      0.92        80
weighted avg       0.94      0.94      0.94        80
```

Calculating accuracy for given model

In [18]:

```
print("Accuracy: ",metrics.accuracy_score(y_test, x_pred))
```

```
Accuracy:  0.9375
```