Assignment No 5 - Assignment on Classification technique (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. 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 required libraries

In [55]:
```
import numpy as np
import pandas as pd
```

Reading data from csv file

In [56]:
```
data = pd.read_csv("C:\\Users\\Durgesh Mahajan\\Downloads\\Admission_Predict.csv")
```

In [57]:
```
data
```

Out[57]:

	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

Converting the data in "Chance of Admit" column in the form of 1's and 0's

In [58]:
```
data.loc[data["Chance of Admit"]>=0.5, "Chance of Admit"] = 1
data.loc[data["Chance of Admit"]<0.5, "Chance of Admit"] = 0
```

Converting the data in "Chance of Admit" column to int datatype

In [59]:
```
data["Chance of Admit"] = data["Chance of Admit"].astype(int)
```

Importing the train_test_split function and declaring dependent and independent variables

In [60]:
```
from sklearn.model_selection import train_test_split
```

In [61]:
```
columns = ["GRE Score", "TOEFL Score", "University Rating", "SOP", "LOR", "CGPA", "Research"]
X = data[columns]
Y = data["Chance of Admit"]
```

Splitting the model into training and testing part

In [62]:
```
xTrain, xTest, yTrain, yTest = train_test_split(X, Y, test_size=0.3)
```

In [63]:
```
print(xTrain)
print(xTest)
print(yTrain)
print(yTest)
```

```
     GRE Score  TOEFL Score  University Rating  SOP  LOR  CGPA  Research
258        326          102                  4  5.0  5.0  8.76         1
369        301           98                  1  2.0  3.0  8.03         1
274        315          100                  1  2.0  2.5  7.95         0
233        304          100                  2  2.5  3.5  8.07         0
332        308          106                  3  3.5  2.5  8.21         1
..         ...          ...                ...  ...  ...   ...       ...
279        304          102                  2  3.0  4.0  8.73         0
75         329          114                  2  2.0  4.0  8.56         1
155        312          109                  3  3.0  3.0  8.69         0
180        300          104                  3  3.5  3.0  8.16         0
50         313           98                  3  2.5  4.5  8.30         1

[280 rows x 7 columns]
     GRE Score  TOEFL Score  University Rating  SOP  LOR  CGPA  Research
246        316          105                  3  3.0  3.5  8.73         0
387        307          105                  2  2.0  3.5  8.10         0
329        297           96                  2  2.5  1.5  7.89         0
152        321          112                  5  5.0  5.0  9.06         1
346        304           97                  2  1.5  2.0  7.64         0
..         ...          ...                ...  ...  ...   ...       ...
192        322          114                  5  4.5  4.0  8.94         1
97         331          120                  3  4.0  4.0  8.96         1
127        319          112                  3  2.5  2.0  8.71         1
147        326          114                  3  3.0  3.0  9.11         1
43         332          117                  4  4.5  4.0  9.10         0

[120 rows x 7 columns]
258    1
369    1
274    1
233    1
332    1
      ..
279    1
75     1
155    1
180    1
50     1
Name: Chance of Admit, Length: 280, dtype: int32
246    1
387    1
329    0
152    1
346    0
      ..
192    1
97     1
127    1
147    1
43     1
Name: Chance of Admit, Length: 120, dtype: int32
```

Importing and creating a decision tree model

In [64]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [65]:
```
dTree = DecisionTreeClassifier()
dTree.fit(xTrain, yTrain)
```

Out[65]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

Predicting the data

In [66]:
```
yPredict = dTree.predict(xTest)
```

In [71]:
```
yPredict
```

Out[71]:
```
array([1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Finding the accuracy score

In [67]:
```
from sklearn.metrics import accuracy_score
```

In [68]:
```
print("Accuracy score of the model is: ", accuracy_score(yTest, yPredict))
```

```
Accuracy score of the model is:  0.9166666666666666
```

Plotting the predicted data

In [69]:
```
from sklearn import tree
import matplotlib.pyplot as plt
```

In [70]:
```
plt.subplots(figsize=(15, 10))
tree.plot_tree(model, feature_names = columns, filled=True, max_depth = 2, fontsize = 10)
```

Out[70]:
```
[Text(0.4230769230769231, 0.875, 'CGPA <= 7.85\ngini = 0.163\nsamples = 280\nvalue = [25, 255]'),
 Text(0.15384615384615385, 0.625, 'TOEFL Score <= 97.5\ngini = 0.475\nsamples = 31\nvalue = [19, 12]'),
 Text(0.07692307692307693, 0.375, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]'),
 Text(0.23076923076923078, 0.375, 'SOP <= 2.75\ngini = 0.499\nsamples = 23\nvalue = [11, 12]'),
 Text(0.15384615384615385, 0.125, '\n  (...)  \n'),
 Text(0.3076923076923077, 0.125, '\n  (...)  \n'),
 Text(0.6923076923076923, 0.625, 'TOEFL Score <= 96.5\ngini = 0.047\nsamples = 249\nvalue = [6, 243]'),
 Text(0.5384615384615384, 0.375, 'LOR <= 1.75\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.46153846153846156, 0.125, '\n  (...)  \n'),
 Text(0.6153846153846154, 0.125, '\n  (...)  \n'),
 Text(0.8461538461538461, 0.375, 'CGPA <= 8.31\ngini = 0.04\nsamples = 247\nvalue = [5, 242]'),
 Text(0.7692307692307693, 0.125, '\n  (...)  \n'),
 Text(0.9230769230769231, 0.125, '\n  (...)  \n')]
```