Assignment-3

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.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dataset = pd.read_csv('Admission_Predict.csv')
dataset
```

```
     Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR  
CGPA  \
0             1        337          118                  4  4.5   4.5  
9.65   
1             2        324          107                  4  4.0   4.5  
8.87   
2             3        316          104                  3  3.0   3.5  
8.00   
3             4        322          110                  3  3.5   2.5  
8.67   
4             5        314          103                  2  2.0   3.0  
8.21   
..          ...        ...          ...                ...  ...   ...  
...   
395         396        324          110                  3  3.5   3.5  
9.04   
396         397        325          107                  3  3.0   3.5  
9.11   
397         398        330          116                  4  5.0   4.5  
9.45   
398         399        312          103                  3  3.5   4.0  
8.78   
399         400        333          117                  4  5.0   4.0  
9.66   

     Research  Chance of Admit   
0           1              0.92  
1           1              0.76  
2           1              0.72  
```

```
3          1              0.80
4          0              0.65
..       ...               ...
395        1              0.82
396        1              0.84
397        1              0.91
398        0              0.67
399        1              0.95

[400 rows x 9 columns]
```

```
dataset.describe()
```

```
       Serial No.   GRE Score  TOEFL Score  University Rating
SOP  \
count  400.000000  400.000000   400.000000         400.000000
400.000000
mean   200.500000  316.807500   107.410000           3.087500
3.400000
std    115.614301   11.473646     6.069514           1.143728
1.006869
min      1.000000  290.000000    92.000000           1.000000
1.000000
25%    100.750000  308.000000   103.000000           2.000000
2.500000
50%    200.500000  317.000000   107.000000           3.000000
3.500000
75%    300.250000  325.000000   112.000000           4.000000
4.000000
max    400.000000  340.000000   120.000000           5.000000
5.000000

             LOR         CGPA    Research  Chance of Admit 
count  400.000000  400.000000  400.000000        400.000000
mean     3.452500    8.598925    0.547500          0.724350
std      0.898478    0.596317    0.498362          0.142609
min      1.000000    6.800000    0.000000          0.340000
25%      3.000000    8.170000    0.000000          0.640000
50%      3.500000    8.610000    1.000000          0.730000
75%      4.000000    9.062500    1.000000          0.830000
max      5.000000    9.920000    1.000000          0.970000
```

```
dataset.columns
```

```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating',
'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

Drop Serial No column

```
dataset.drop('Serial No.', axis=1, inplace=True)
dataset
```

```
     GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA
Research  \
0          337          118                  4  4.5   4.5  9.65
1
1          324          107                  4  4.0   4.5  8.87
1
2          316          104                  3  3.0   3.5  8.00
1
3          322          110                  3  3.5   2.5  8.67
1
4          314          103                  2  2.0   3.0  8.21
0
..         ...          ...                ...  ...   ...   ...
...
395        324          110                  3  3.5   3.5  9.04
1
396        325          107                  3  3.0   3.5  9.11
1
397        330          116                  4  5.0   4.5  9.45
1
398        312          103                  3  3.5   4.0  8.78
0
399        333          117                  4  5.0   4.0  9.66
1

     Chance of Admit
0                0.92
1                0.76
2                0.72
3                0.80
4                0.65
..                ...
395              0.82
396              0.84
397              0.91
398              0.67
399              0.95

[400 rows x 8 columns]
```

Splitting dataset into training and testing set

```
from sklearn.model_selection import train_test_split

X = dataset.drop('Chance of Admit ', axis=1)
y = dataset['Chance of Admit ']
```

Converting the 'Chance of Admit ' to 1 and 0

```
for i in range(0, len(y)):
    if (y[i] > 0.85):
        y[i] = 1
    else:
        y[i] = 0

y.value_counts()
```

```
Chance of Admit
0.0    317
1.0     83
Name: count, dtype: int64
```

```
from sklearn import tree

tree_model = tree.DecisionTreeClassifier(criterion='gini',
splitter='best')

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=0)

len(X_train), len(X_test), len(y_train), len(y_test)
```

```
(300, 100, 300, 100)
```

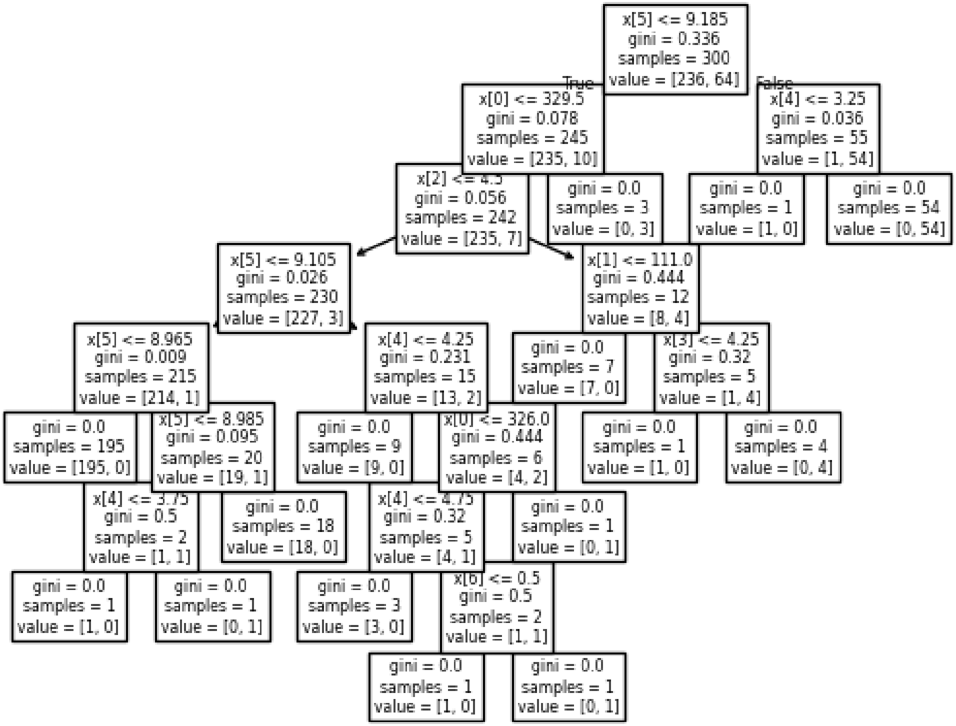
Fitting the Decission Tree on the training set

```
tree_model.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
tree.plot_tree(tree_model)
```

```
[Text(0.7037037037037037, 0.9444444444444444, 'x[5] <= 9.185\ngini =
0.336\nsamples = 300\nvalue = [236, 64]'),
 Text(0.5555555555555556, 0.8333333333333334, 'x[0] <= 329.5\ngini =
0.078\nsamples = 245\nvalue = [235, 10]'),
 Text(0.6296296296296297, 0.8888888888888888, 'True  '),
 Text(0.48148148148148145, 0.7222222222222222, 'x[2] <= 4.5\ngini =
0.056\nsamples = 242\nvalue = [235, 7]'),
 Text(0.2962962962962963, 0.6111111111111112, 'x[5] <= 9.105\ngini =
0.026\nsamples = 230\nvalue = [227, 3]'),
 Text(0.14814814814814814, 0.5, 'x[5] <= 8.965\ngini = 0.009\nsamples
= 215\nvalue = [214, 1]'),
 Text(0.07407407407407407, 0.3888888888888889, 'gini = 0.0\nsamples =
195\nvalue = [195, 0]'),
 Text(0.2222222222222222, 0.3888888888888889, 'x[5] <= 8.985\ngini =
0.095\nsamples = 20\nvalue = [19, 1]'),
 Text(0.14814814814814814, 0.2777777777777778, 'x[4] <= 3.75\ngini =
0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.07407407407407407, 0.16666666666666666, 'gini = 0.0\nsamples =
1\nvalue = [1, 0]'),
 Text(0.2222222222222222, 0.16666666666666666, 'gini = 0.0\nsamples =
1\nvalue = [0, 1]'),
 Text(0.2962962962962963, 0.2777777777777778, 'gini = 0.0\nsamples =
18\nvalue = [18, 0]'),
 Text(0.4444444444444444, 0.5, 'x[4] <= 4.25\ngini = 0.231\nsamples =
15\nvalue = [13, 2]'),
 Text(0.37037037037037035, 0.3888888888888889, 'gini = 0.0\nsamples =
9\nvalue = [9, 0]'),
 Text(0.5185185185185185, 0.3888888888888889, 'x[0] <= 326.0\ngini =
0.444\nsamples = 6\nvalue = [4, 2]'),
 Text(0.4444444444444444, 0.2777777777777778, 'x[4] <= 4.75\ngini =
0.32\nsamples = 5\nvalue = [4, 1]'),
 Text(0.37037037037037035, 0.16666666666666666, 'gini = 0.0\nsamples =
3\nvalue = [3, 0]'),
 Text(0.5185185185185185, 0.16666666666666666, 'x[6] <= 0.5\ngini =
0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.4444444444444444, 0.05555555555555555, 'gini = 0.0\nsamples =
1\nvalue = [1, 0]'),
 Text(0.5925925925925926, 0.05555555555555555, 'gini = 0.0\nsamples =
1\nvalue = [0, 1]'),
 Text(0.5925925925925926, 0.2777777777777778, 'gini = 0.0\nsamples =
1\nvalue = [0, 1]'),
 Text(0.6666666666666666, 0.6111111111111112, 'x[1] <= 111.0\ngini =
0.444\nsamples = 12\nvalue = [8, 4]'),
 Text(0.5925925925925926, 0.5, 'gini = 0.0\nsamples = 7\nvalue = [7,
0]'),
 Text(0.7407407407407407, 0.5, 'x[3] <= 4.25\ngini = 0.32\nsamples =
5\nvalue = [1, 4]'),
 Text(0.6666666666666666, 0.3888888888888889, 'gini = 0.0\nsamples =
1\nvalue = [1, 0]'),
 Text(0.8148148148148148, 0.3888888888888889, 'gini = 0.0\nsamples =
4\nvalue = [0, 4]'),
 Text(0.6296296296296297, 0.7222222222222222, 'gini = 0.0\nsamples =
3\nvalue = [0, 3]'),
 Text(0.8518518518518519, 0.8333333333333334, 'x[4] <= 3.25\ngini =
0.036\nsamples = 55\nvalue = [1, 54]'),
 Text(0.7777777777777778, 0.8888888888888888, '  False'),
 Text(0.7777777777777778, 0.7222222222222222, 'gini = 0.0\nsamples =
1\nvalue = [1, 0]'),
 Text(0.9259259259259259, 0.7222222222222222, 'gini = 0.0\nsamples =
54\nvalue = [0, 54]')]
```


x[5] <= 9.185 gini = 0.336 samples = 300 value = [236, 64]
True
False
x[0] <= 329.5 gini = 0.078 samples = 245 value = [235, 10]
x[4] <= 3.25 gini = 0.036 samples = 55 value = [1, 54]
x[2] <= 4.5 gini = 0.056 samples = 242 value = [235, 7]
gini = 0.0 samples = 3 value = [0, 3]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 54 value = [0, 54]
x[5] <= 9.105 gini = 0.026 samples = 230 value = [227, 3]
x[1] <= 111.0 gini = 0.444 samples = 12 value = [8, 4]
x[5] <= 8.965 gini = 0.009 samples = 215 value = [214, 1]
x[4] <= 4.25 gini = 0.231 samples = 15 value = [13, 2]
gini = 0.0 samples = 7 value = [7, 0]
x[3] <= 4.25 gini = 0.32 samples = 5 value = [1, 4]
gini = 0.0 samples = 195 value = [195, 0]
x[5] <= 8.985 gini = 0.095 samples = 20 value = [19, 1]
gini = 0.0 samples = 9 value = [9, 0]
x[0] <= 326.0 gini = 0.444 samples = 6 value = [4, 2]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 4 value = [0, 4]
x[4] <= 3.75 gini = 0.5 samples = 2 value = [1, 1]
gini = 0.0 samples = 18 value = [18, 0]
x[4] <= 4.75 gini = 0.32 samples = 5 value = [4, 1]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 3 value = [3, 0]
x[6] <= 0.5 gini = 0.5 samples = 2 value = [1, 1]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 1 value = [0, 1]

```
y_preds = tree_model.predict(X_test)
y_preds
```

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

```
from sklearn.metrics import  mean_absolute_error, accuracy_score,
confusion_matrix

mae = mean_absolute_error(y_test, y_preds)
acc = accuracy_score(y_test, y_preds)
```

```
cm = confusion_matrix(y_test, y_preds)
```

```
print(f'MAE: {mae}, ACC: {acc}')
```

```
MAE: 0.1, ACC: 0.9
```

```
print(cm)
```

```
[[78  3]
 [ 7 12]]
```

```
tree_model = 
tree.DecisionTreeClassifier(criterion='entropy',splitter='best')
```

```
tree_model.fit(X_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy')
```

```
tree.plot_tree(tree_model)
```

```
gre = input()
```

```
 34
```

```
message = []
```

```
random=dataset.iloc[:14]
```

```
toel = input()
```

```
 23
```

```
message = []
```

```
message.append(toel)
```

```
dataset
```

```
     GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  
Research  \
0          337          118                  4  4.5   4.5  9.65  
1  
1          324          107                  4  4.0   4.5  8.87  
1  
2          316          104                  3  3.0   3.5  8.00  
1  
3          322          110                  3  3.5   2.5  8.67  
1  
4          314          103                  2  2.0   3.0  8.21  
0  
..         ...          ...                ...  ...   ...   ...  
```

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

     Chance of Admit
0                 1.0
1                 0.0
2                 0.0
3                 0.0
4                 0.0
..                ...
395               0.0
396               0.0
397               1.0
398               0.0
399               1.0

[400 rows x 8 columns]
```