Importing Libraries

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
```

```
1 data = pd.read_csv('Admission_Predict.csv')
```

```
1 data.head()
```

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

Dropping column 'Serial No.'

```
1 data = data.drop(columns = ['Serial No.'])
2 data.head()
```

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

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE Score          400 non-null    int64
 1   TOEFL Score        400 non-null    int64
```

```
 2   University Rating  400 non-null    int64
 3   SOP                400 non-null    float64
 4   LOR                400 non-null    float64
 5   CGPA               400 non-null    float64
 6   Research           400 non-null    int64
 7   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 25.1 KB
```

Checking for null values

```
data.isnull().sum()
```

```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

```
li = [ i for i in data['Chance of Admit ']]
```

Function for conversion

- add a column callled as admission
- adminsssion have value 1 or 0 based on chance of admit value is greater than or equal to 0.9

```
def conversion(l):
    adlist = []
    for i in l:
        if (i >= 0.9):
            adlist.append(1)
        else:
            adlist.append(0)
    return adlist

admissionlist = conversion(li)
```

```
data['Admission'] = admissionlist
```

```
data.head()
```

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

dropping column 'Chance of Admit'

```
data = data.drop(columns = ['Chance of Admit '])
data.head()
```

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

X - Features, Y - Target Variable

```
X = data.iloc[:,0:7]
X.head()
```

	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

```
Y = data['Admission']
Y.head()
```

```
0    1
1    0
2    0
3    0
```

```
4    0
Name: Admission, dtype: int64
```

train_test_split

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25,
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((300, 7), (300,), (100, 7), (100,))
```

Decision Tree classifier

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion = 'entropy').fit(X,Y)
```

```
x_train_predicted = clf.predict(x_train)
```

```
y_predict = clf.predict(x_test)
```

Training accuracy and confusion matrix

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

```
training_accuracy = accuracy_score(y_train, x_train_predicted)
training_accuracy
```

```
1.0
```

```
training_confusion_matrix = confusion_matrix(y_train, x_train_predicted)
training_confusion_matrix
```

```
array([[255,   0],
       [  0,  45]])
```

Testing accuracy and confusion matrix

```
1 testing_accuracy = accuracy_score(y_test, y_predict)
2 testing_accuracy
```

```
1.0
```

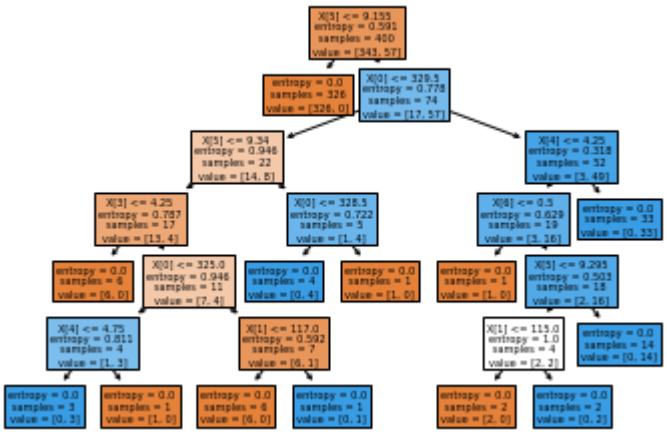
```
1 testing_confusion_matrix = confusion_matrix(y_test, y_predict)
2 testing_confusion_matrix
```

```
array([[88,  0],
       [ 0, 12]])
```

Visualizing Tree

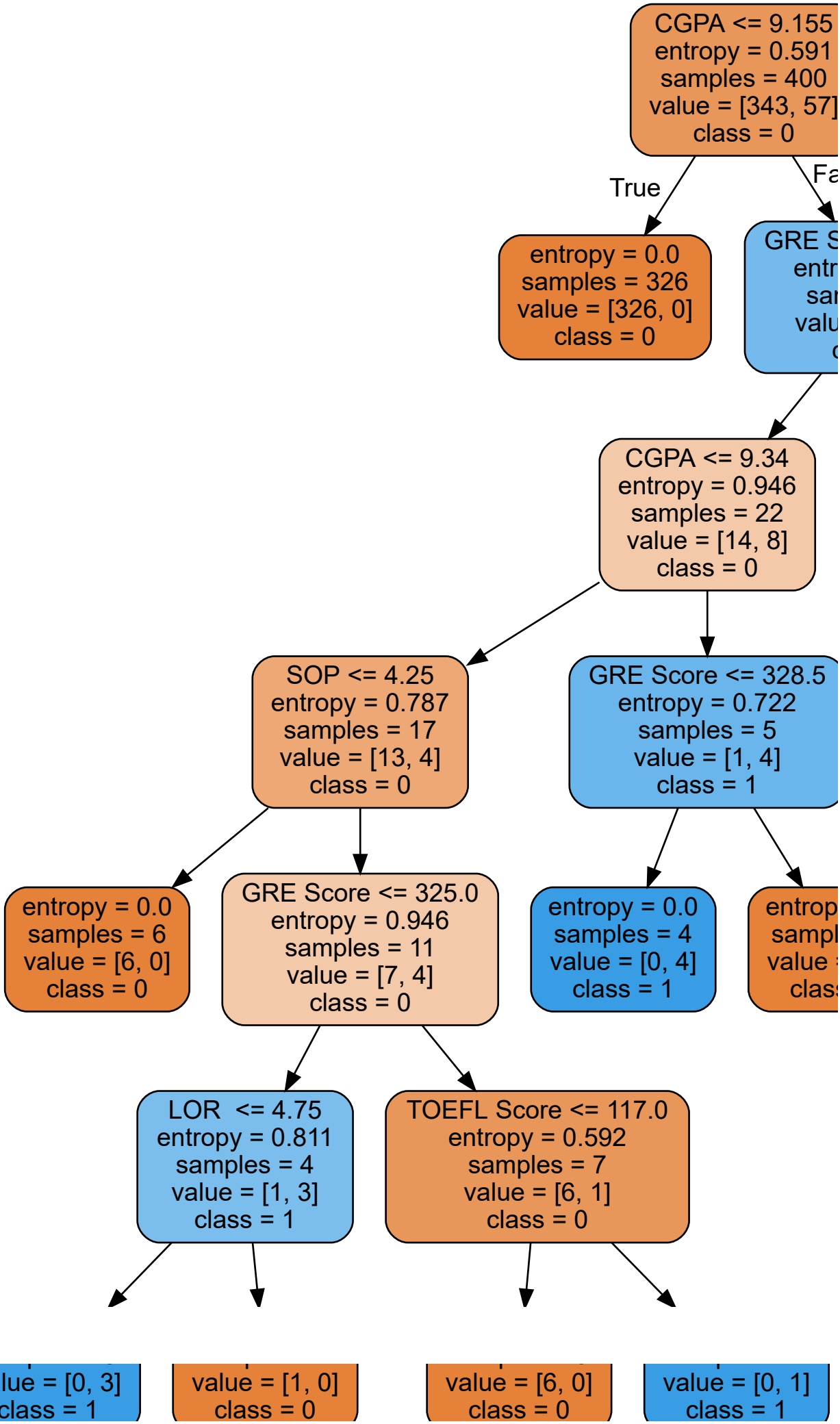
```
1 from sklearn.tree import plot_tree
2 from sklearn.tree import export_graphviz
```

```
1 Tree = plot_tree(clf,filled = True)
2 plt.show(Tree)
```



```
1 features = ['GRE Score','TOEFL Score','University Rating','SOP','LOR ','CGPA
2 classes = ['0','1']
```

```
1 import graphviz
2 dot_data = export_graphviz(clf, feature_names = features, class_names = clas
3 graph = graphviz.Source(dot_data)
4 plt.figure(figsize = (12,12))
5 graph
```


CGPA <= 9.155
entropy = 0.591
samples = 400
value = [343, 57]
class = 0
True
entropy = 0.0
samples = 326
value = [326, 0]
class = 0
CGPA <= 9.34
entropy = 0.946
samples = 22
value = [14, 8]
class = 0
SOP <= 4.25
entropy = 0.787
samples = 17
value = [13, 4]
class = 0
GRE Score <= 328.5
entropy = 0.722
samples = 5
value = [1, 4]
class = 1
entropy = 0.0
samples = 6
value = [6, 0]
class = 0
GRE Score <= 325.0
entropy = 0.946
samples = 11
value = [7, 4]
class = 0
entropy = 0.0
samples = 4
value = [0, 4]
class = 1
LOR <= 4.75
entropy = 0.811
samples = 4
value = [1, 3]
class = 1
TOEFL Score <= 117.0
entropy = 0.592
samples = 7
value = [6, 1]
class = 0
value = [0, 3]
class = 1
value = [1, 0]
class = 0
value = [6, 0]
class = 0
value = [0, 1]
class = 1

1

✓ 0s completed at 8:43 AM