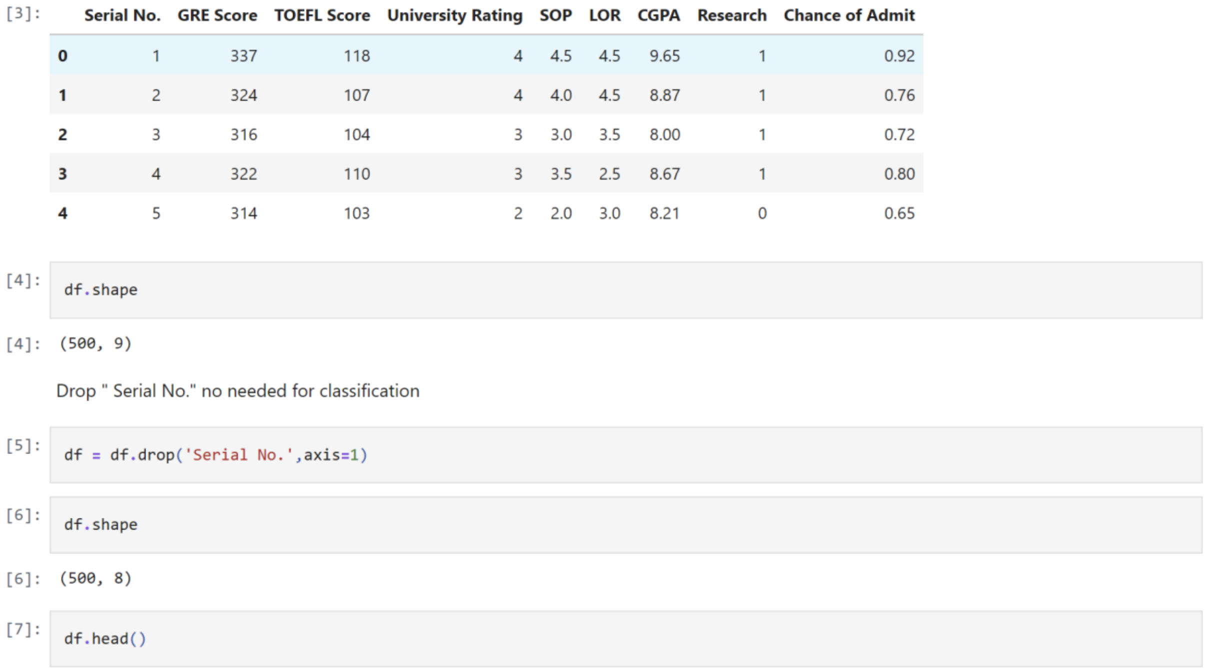
Assignment No. 02

Name : Tejas Shyam Fulumbarkar

Subject : Machine Learning

Class : TE-IT (B)



[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

```
[4]: df.shape
```

```
[4]: (500, 9)
```

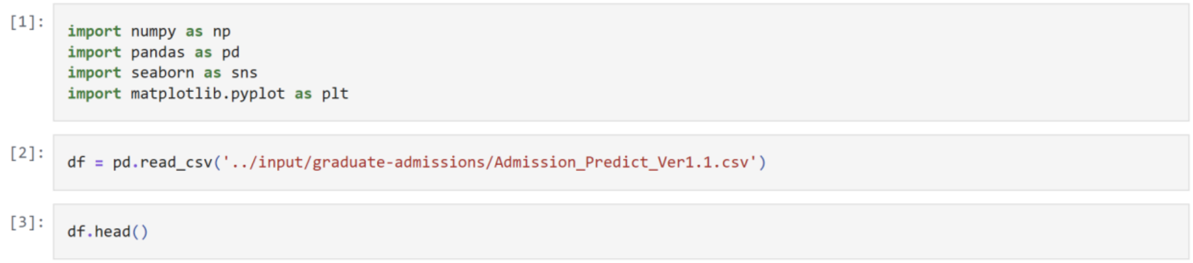
Drop " Serial No." no needed for classification

```
[5]: df = df.drop('Serial No.',axis=1)
```

```
[6]: df.shape
```

```
[6]: (500, 8)
```

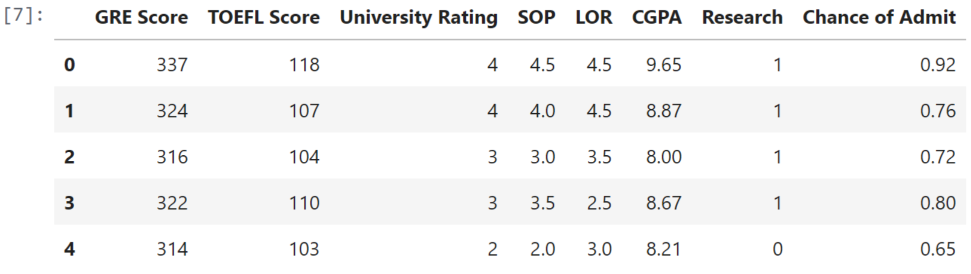
```
[7]: df.head()
```



```
[1]: import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
```

```
[2]: df = pd.read_csv('../input/graduate-admissions/Admission_Predict_Ver1.1.csv')
```

```
[3]: df.head()
```



[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

[9]:

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

y = df['Chance of Admit ']
```

[10]:

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

[11]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,random_state=1)
```

[12]:

```
print(f"Size of splitted data")
print(f"x_train {x_train.shape}")
print(f"y_train {y_train.shape}")
print(f"y_train {x_test.shape}")
print(f"y_test {y_test.shape}")
```

```
Size of splitted data
x_train (375, 7)
y_train (375,)
y_train (125, 7)
y_test (125,)
```

[13]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LogisticRegression
```

[14]:

```
model_dt = DecisionTreeRegressor(random_state=1)
model_rf = RandomForestRegressor(random_state=1)
model_lr = LogisticRegression(random_state=1,solver='lbfgs',max_iter=1000)
```

[15]:

```
model_dt.fit(x_train,y_train)
```

t[15]:

```
DecisionTreeRegressor(random_state=1)
```

[16]:

```
model_rf.fit(x_train,y_train)
```

t[16]:

```
RandomForestRegressor(random_state=1)
```

[17]:

```
model_lr.fit(x_train,y_train)
```

t[17]:

```
LogisticRegression(max_iter=1000, random_state=1)
```

[18]:

```
y_pred_dt = model_dt.predict(x_test) #int
y_pred_rf = model_rf.predict(x_test) #float
y_pred_lr = model_lr.predict(x_test) #
```

[19]:

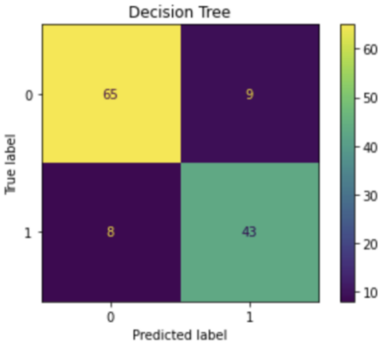
```
y_pred_rf = [1 if each > 0.75 else 0 for each in y_pred_rf]
```

[20]:

```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
```

[21]:

```python
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_dt)
plt.title('Decision Tree')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_dt)}")
print(classification_report(y_test,y_pred_dt))
```



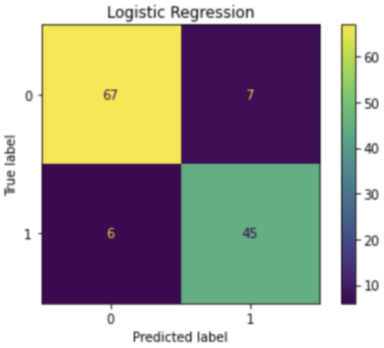
```
 Accuracy is 0.864
              precision    recall  f1-score   support

           0       0.89      0.88      0.88        74
           1       0.83      0.84      0.83        51

    accuracy                           0.86       125
   macro avg       0.86      0.86      0.86       125
weighted avg       0.86      0.86      0.86       125
```

[22]:

```python
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_lr)
plt.title('Logistic Regression')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_lr)}")
print(classification_report(y_test,y_pred_lr))
```

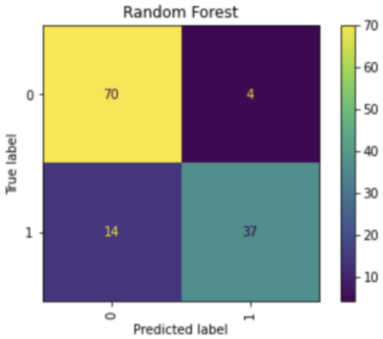


```
 Accuracy is 0.896
              precision    recall  f1-score   support

           0       0.92      0.91      0.91        74
           1       0.87      0.88      0.87        51

    accuracy                           0.90       125
   macro avg       0.89      0.89      0.89       125
weighted avg       0.90      0.90      0.90       125
```

```
[23]: ConfusionMatrixDisplay.from_predictions(y_test,y_pred_rf,xticks_rotation='vertical')
      plt.title('Random Forest')
      plt.show()
      print(f" Accuracy is {accuracy_score(y_test,y_pred_rf)}")
      print(classification_report(y_test,y_pred_rf))
```



```
 Accuracy is 0.856
              precision    recall  f1-score   support

           0       0.83      0.95      0.89        74
           1       0.90      0.73      0.80        51

    accuracy                           0.86       125
   macro avg       0.87      0.84      0.85       125
weighted avg       0.86      0.86      0.85       125
```