Hello this is @Ranjeet_Kumbhar, Enjoy the Notebook

GitHub: https://github.com/RanjeetKumbhar01/TE_IT_ML_ASSIGNMENTS_SPPU

Question

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

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df =
pd.read_csv('../input/graduate-admissions/Admission_Predict_Ver1.1.csv
')

df.head()
```

```
   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

   Research  Chance of Admit
0         1             0.92
```

```
1         1             0.76
2         1             0.72
3         1             0.80
4         0             0.65
```

```
df.shape
```

```
(500, 9)
```

Drop " Serial No." no needed for classification

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

```
df.shape
```

```
(500, 8)
```

```
df.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

   Chance of Admit 
0              0.92
1              0.76
2              0.72
3              0.80
4              0.65
```

```
df['Chance of Admit '] = [1 if each > 0.75 else 0 for each in 
df['Chance of Admit ']]
df.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

   Chance of Admit
0                1
1                1
2                0
3                1
4                0
```

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

y = df['Chance of Admit ']

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,random_state=1)

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,)
```

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

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

model_dt.fit(x_train,y_train)
```

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

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

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

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

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

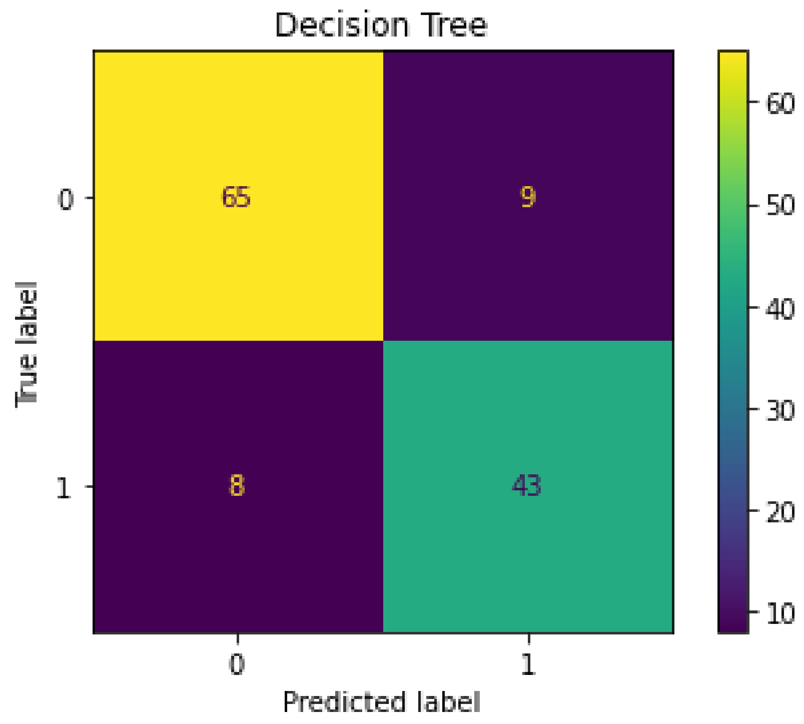
```
y_pred_dt = model_dt.predict(x_test) #int
y_pred_rf = model_rf.predict(x_test) #float
y_pred_lr = model_lr.predict(x_test) #
```

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

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

Decision Tree

```
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))
```



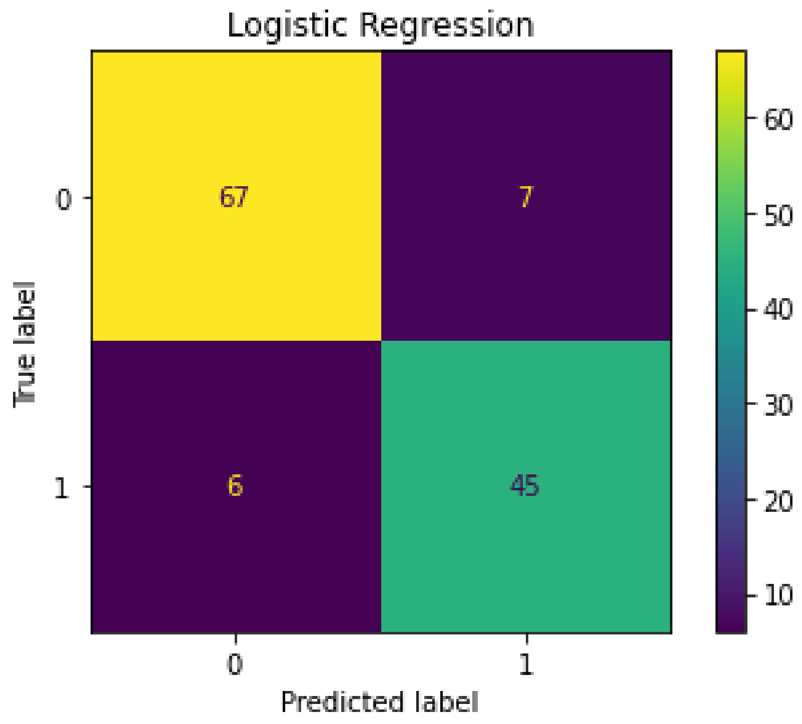
```
 Accuracy is 0.864
              precision    recall  f1-score   support

           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
```

Logistic Regression

```
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))
```



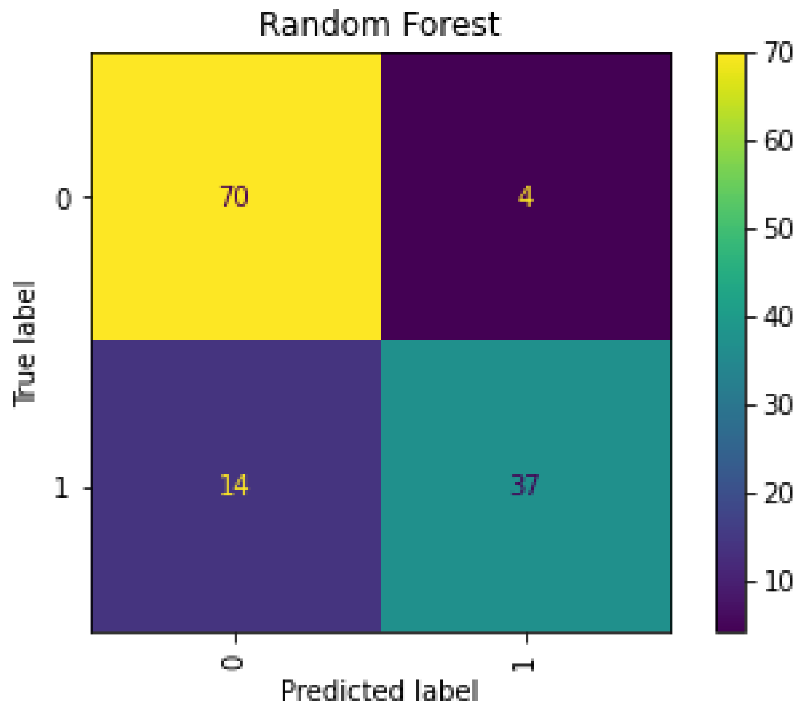
```
 Accuracy is 0.896
              precision    recall  f1-score   support

           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
```

Random Forest

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_rf,xticks_rotati
on='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
```