ML Major PROJECT

NAME: - Maaz Abdul Munaf

College: - NMAM Institute of technology

Course: - B.E (3rd year, 5th Sem)

Branch: - Information Science

Determination of Autism amoung toddlers

Importing Libraries

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

Importing Classifiers

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```

Importing metrics

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

Supressing warnings

```
import warnings
warnings.filterwarnings("ignore")
```

Loading dataset

```
df=pd.read_csv('/content/Autism_July_18(1).csv')
df.head()
```

Case_No	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Age_Mons	Qchat-10-Score	Sex	Ethnicity

Analyising Dataset and preprocessing

```
df.columns
```

```
Index(['Case_No', 'A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10',
       'Age_Mons', 'Qchat-10-Score', 'Sex', 'Ethnicity', 'Jaundice',
       'Family_mem_with_ASD', 'Who completed the test', 'Class/ASD Traits '],
      dtype='object')
```

```
df.drop(['Case_No', 'Who completed the test','Qchat-10-Score'], axis = 1, inplace = True)
df.columns
```

```
Index(['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'Age_Mons',
       'Sex', 'Ethnicity', 'Jaundice', 'Family_mem_with_ASD',
       'Class/ASD Traits '],
      dtype='object')
```

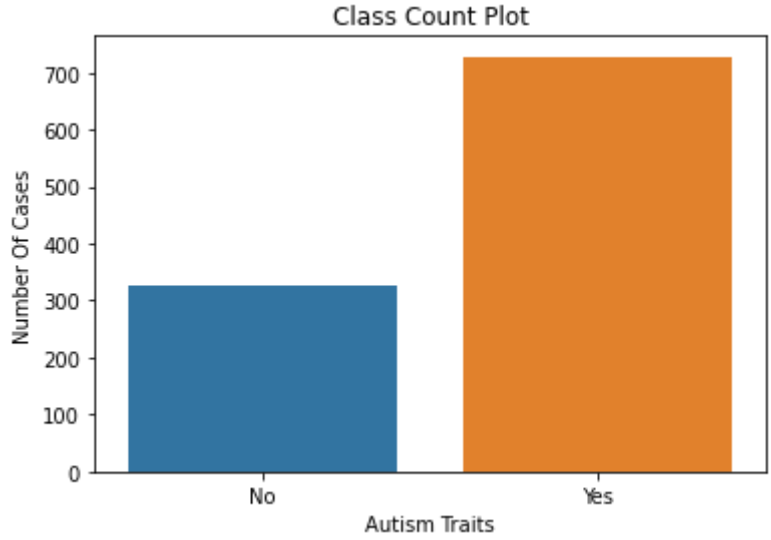
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1054 entries, 0 to 1053
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   A1                   1054 non-null   int64 
 1   A2                   1054 non-null   int64 
 2   A3                   1054 non-null   int64 
 3   A4                   1054 non-null   int64 
 4   A5                   1054 non-null   int64 
 5   A6                   1054 non-null   int64 
 6   A7                   1054 non-null   int64 
 7   A8                   1054 non-null   int64 
 8   A9                   1054 non-null   int64 
 9   A10                  1054 non-null   int64 
 10  Age_Mons             1054 non-null   int64 
 11  Sex                  1054 non-null   object
 12  Ethnicity            1054 non-null   object
 13  Jaundice             1054 non-null   object
 14  Family_mem_with_ASD  1054 non-null   object
 15  Class/ASD Traits     1054 non-null   object
dtypes: int64(11), object(5)
memory usage: 131.9+ KB
```

comparing Autsim Traits and Number of cases from the dataset

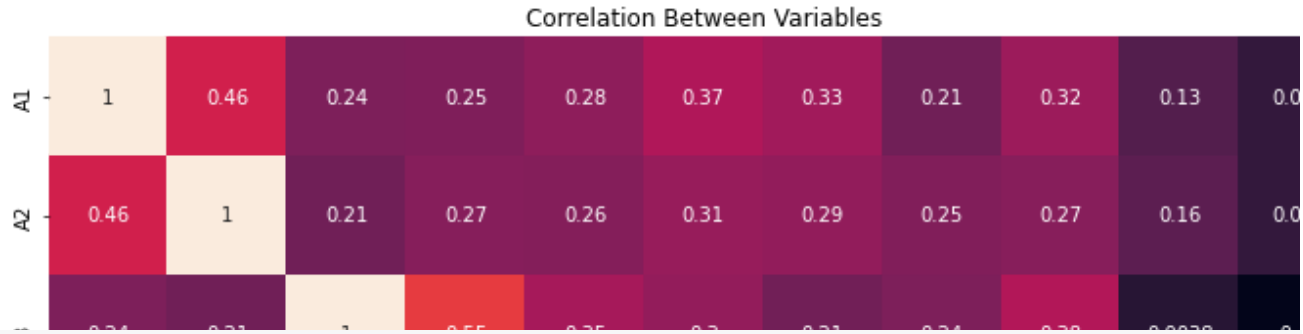
```
ClassCount=sns.countplot(x='Class/ASD Traits ' , data=df)
ClassCount.set(xlabel ='Autism Traits', ylabel = 'Number Of Cases',title='Class Count Plot
ClassCount
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc433435d0>
```



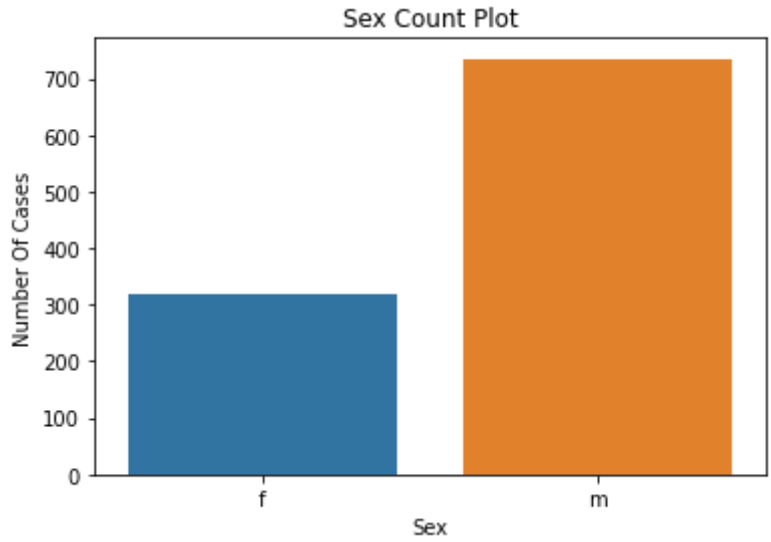

```
corr = df.corr()
plt.figure(figsize = (15,15))
CorrelationHM=sns.heatmap(data = corr, annot = True, square = True, cbar = True)
CorrelationHM.set(title='Correlation Between Variables')
```

```
[Text(0.5, 1.0, 'Correlation Between Variables')]
```



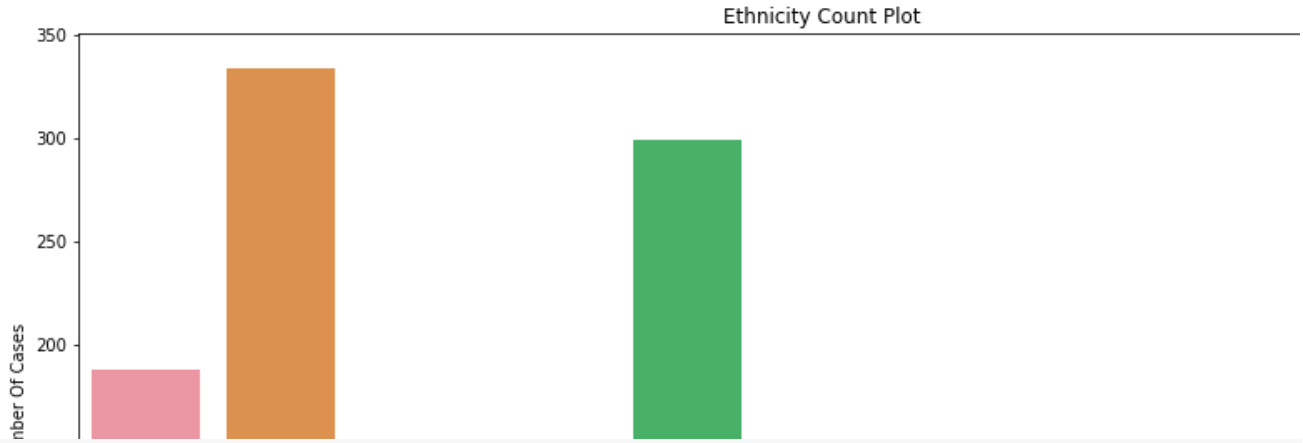

```
SexCount=sns.countplot(x='Sex',data=df)
SexCount.set(xlabel ='Sex', ylabel = 'Number Of Cases',title='Sex Count Plot')
SexCount
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc3ebb5c90>
```




```
plt.figure(figsize = (16,8))
EthCount=sns.countplot(x = 'Ethnicity', data = df)
EthCount.set(xlabel ='Ethnicity', ylabel = 'Number Of Cases',title='Ethnicity Count Plot')
EthCount
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc3eb730d0>
```




```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
catCols = [col for col in df.columns if df[col].dtype=='O']
for col in catCols:
    df[col] = le.fit_transform(df[col])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1054 entries, 0 to 1053
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   A1                   1054 non-null   int64
 1   A2                   1054 non-null   int64
 2   A3                   1054 non-null   int64
 3   A4                   1054 non-null   int64
 4   A5                   1054 non-null   int64
 5   A6                   1054 non-null   int64
 6   A7                   1054 non-null   int64
 7   A8                   1054 non-null   int64
 8   A9                   1054 non-null   int64
 9   A10                  1054 non-null   int64
 10  Age_Mons             1054 non-null   int64
 11  Sex                  1054 non-null   int64
 12  Ethnicity            1054 non-null   int64
 13  Jaundice             1054 non-null   int64
 14  Family_mem_with_ASD  1054 non-null   int64
 15  Class/ASD Traits     1054 non-null   int64
dtypes: int64(16)
memory usage: 131.9 KB
```

Preparing train and test data

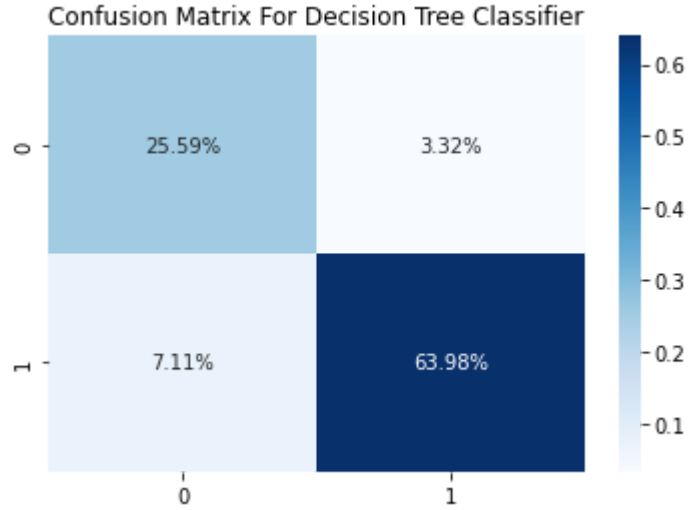
```
from sklearn.model_selection import train_test_split
x = df.drop(['Class/ASD Traits '], axis = 1)
y = df['Class/ASD Traits ']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

Classifying using various classifiers

1. Descision Tree Classifier

```
dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)
dtcpred = dtc.predict(x_test)
print("Classification Report For Decision Tree Classifier :\n\n",classification_report(y_t
dtcConfMatrix=confusion_matrix(y_test,dtcpred)
dtcConfMatrixGraph=sns.heatmap(dtcConfMatrix/np.sum(dtcConfMatrix), annot=True, fmt='.2%',
dtcConfMatrixGraph.set(title='Confusion Matrix For Decision Tree Classifier')
dtcConfMatrixGraph
dtcScore=accuracy_score(y_test,dtcpred)
```

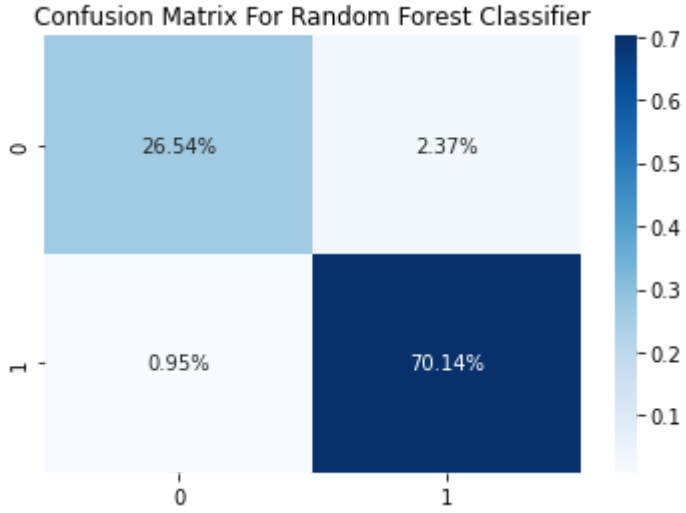
```
Classification Report For Decision Tree Classifier :

              precision    recall  f1-score   support

           0       0.78      0.89      0.83        61
           1       0.95      0.90      0.92       150

    accuracy                           0.90       211
   macro avg       0.87      0.89      0.88       211
weighted avg       0.90      0.90      0.90       211
```



2. Random Forest Classifier

```
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
rfcpred = rfc.predict(x_test)
print("Classification Report For Random Forest Classifier :\n\n",classification_report(y_t
rfcConfMatrix=confusion_matrix(y_test,rfcpred)
rfcConfMatrixGraph=sns.heatmap(rfcConfMatrix/np.sum(rfcConfMatrix), annot=True, fmt='.2%',
rfcConfMatrixGraph.set(title='Confusion Matrix For Random Forest Classifier')
rfcConfMatrixGraph
rfcScore=accuracy_score(y_test,rfcpred)
```

```
Classification Report For Random Forest Classifier :

              precision    recall  f1-score   support

           0       0.97      0.92      0.94        61
           1       0.97      0.99      0.98       150

    accuracy                           0.97       211
   macro avg       0.97      0.95      0.96       211
weighted avg       0.97      0.97      0.97       211
```



3. Support vector machine(SVM)

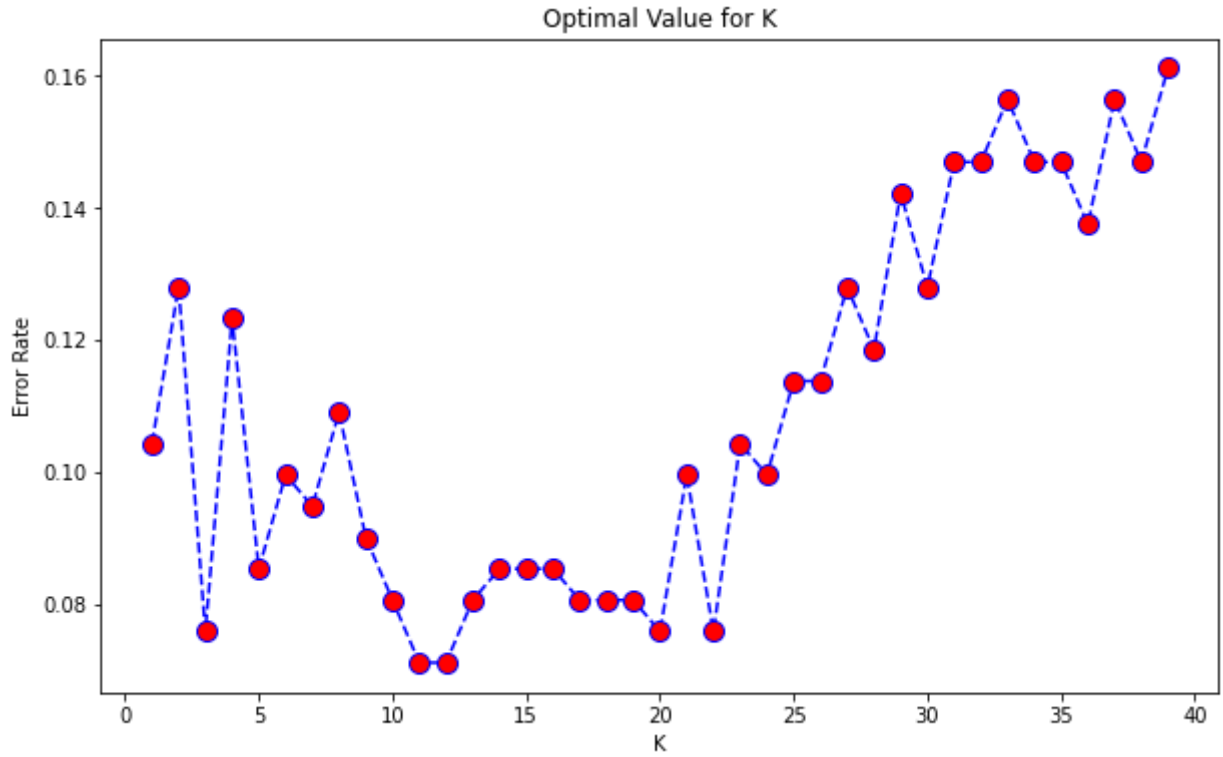
```
svm = SVC()
svm.fit(x_train, y_train)
svmpred = svm.predict(x_test)
print("Classification Report For Support Vector Machine :\n\n",classification_report(y_tes
svmConfMatrix=confusion_matrix(y_test,svmpred)
svmConfMatrixGraph=sns.heatmap(svmConfMatrix/np.sum(svmConfMatrix), annot=True, fmt='.2%',
svmConfMatrixGraph.set(title='Confusion Matrix For Support Vector Machine')
svmConfMatrixGraph
svmScore=accuracy_score(y_test,svmpred)
```

```
Classification Report For Support Vector Machine :

              precision    recall  f1-score   support

           0       0.97      0.51      0.67        61
           1       0.83      0.99      0.91       150

    accuracy                           0.85       211
   macro avg       0.90      0.75      0.79       211
weighted avg       0.87      0.85      0.84       211
```

4. K Nearest neighbours(KNN)

```
error_rate = []
for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train,y_train)
    pred_i = knn.predict(x_test)
    error_rate.append(np.mean(pred_i != y_test))
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed',
         marker='o',markerfacecolor='red', markersize=10)
plt.title('Optimal Value for K')
plt.xlabel('K')
plt.ylabel('Error Rate')
print("\nMinimum error :",min(error_rate),"at K =",error_rate.index(min(error_rate)))
```

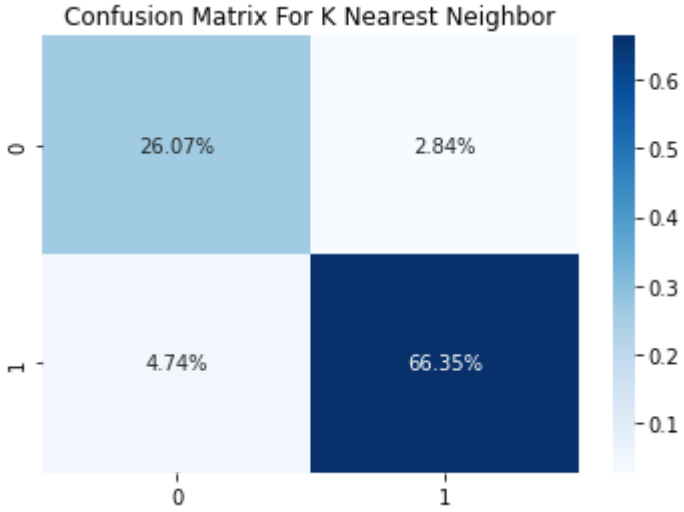
```
Minimum error : 0.07109004739336493 at K = 10
```




```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)
```

```
knnpred = knn.predict(x_test)
print("Classification Report For K Nearest Neighbor :\n\n",classification_report(y_test,kn
knnConfMatrix=confusion_matrix(y_test,knnpred)
knnConfMatrixGraph=sns.heatmap(knnConfMatrix/np.sum(knnConfMatrix), annot=True, fmt='.2%',
knnConfMatrixGraph.set(title='Confusion Matrix For K Nearest Neighbor')
knnConfMatrixGraph
knnScore=accuracy_score(y_test,knnpred)
```

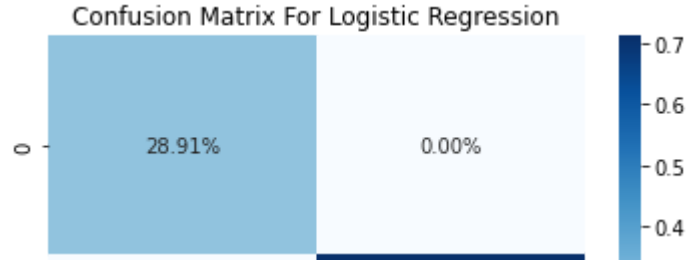
```
Classification Report For K Nearest Neighbor :

              precision    recall  f1-score   support

           0       0.85      0.90      0.87        61
           1       0.96      0.93      0.95       150

    accuracy                           0.92       211
   macro avg       0.90      0.92      0.91       211
weighted avg       0.93      0.92      0.92       211
```



5. Logistic Regression

```
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
logregpred = logreg.predict(x_test)
print("Classification Report For Logistic Regression :\n\n",classification_report(y_test,l
logregConfMatrix=confusion_matrix(y_test,logregpred)
logregConfMatrixGraph=sns.heatmap(logregConfMatrix/np.sum(logregConfMatrix), annot=True, f
logregConfMatrixGraph.set(title='Confusion Matrix For Logistic Regression')
logregConfMatrixGraph
logregScore=logreg.score(x_train, y_train)
```

```
Classification Report For Logistic Regression :

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        61
           1       1.00      1.00      1.00       150

    accuracy                           1.00       211
   macro avg       1.00      1.00      1.00       211
weighted avg       1.00      1.00      1.00       211
```



6 : Naive Bayes Classifier

```
nb = GaussianNB()
nb.fit(x_train, y_train)
nbpred = nb.predict(x_test)
print("Classification Report For Naive Bayes Classifier :\n\n",classification_report(y_tes
nbConfMatrix=confusion_matrix(y_test,nbpred)
nbConfMatrixGraph=sns.heatmap(nbConfMatrix/np.sum(nbConfMatrix), annot=True, fmt='.2%', cm
nbConfMatrixGraph.set(title='Confusion Matrix For Naive Bayes Classifier')
nbConfMatrixGraph
nbScore=accuracy_score(y_test,nbpred)
```

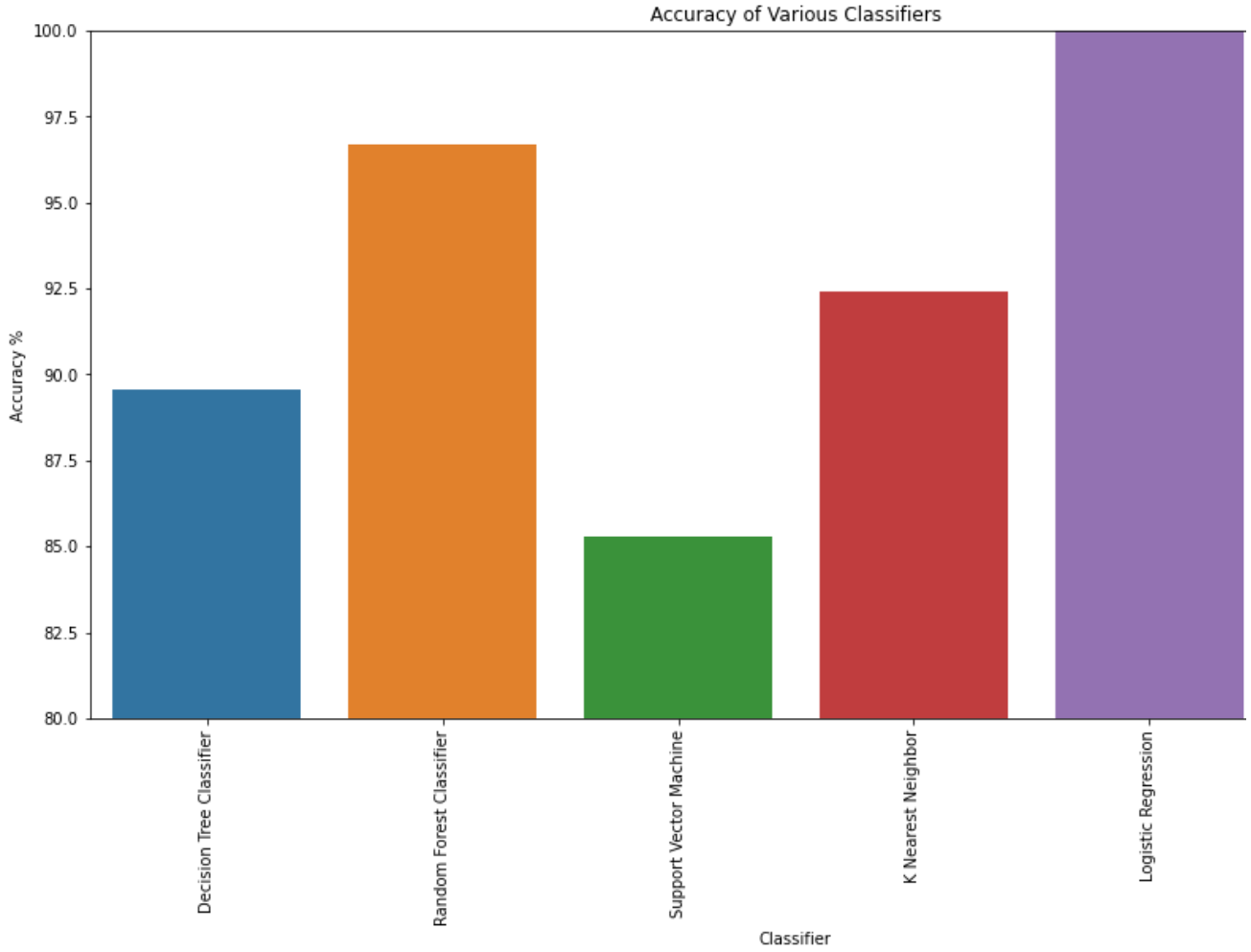
```
Classification Report For Naive Bayes Classifier :

              precision    recall  f1-score   support
```

Comparing Accuracy

```
    accuracy                           0.95       211
```

```
y=[i*100 for i in [dtcScore,rfcScore,svmScore,knnScore,logregScore,nbScore]]
x=['Decision Tree Classifier','Random Forest Classifier','Support Vector Machine','K Neare
plt.figure(figsize = (16,8))
compareAcc=sns.barplot(x,y)
compareAcc.set(title='Accuracy of Various Classifiers',xlabel='Classifier',ylabel='Accurac
plt.xticks(rotation=90)
plt.ylim(80, 100)
compareAcc
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc3db27d50>
```



GitHub link

https://github.com/Maaz0001/Rinex