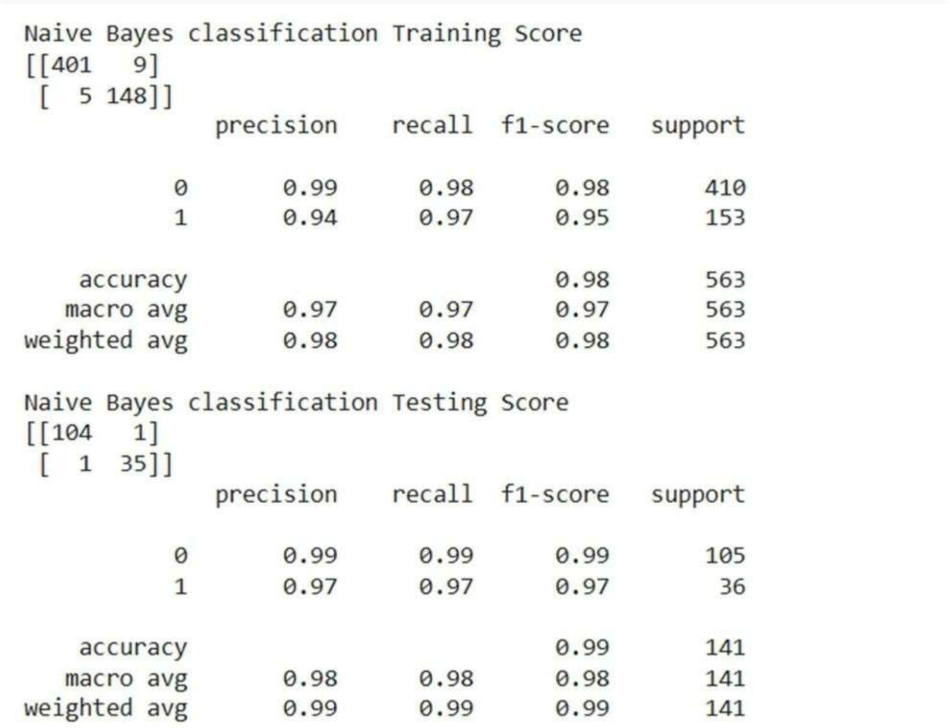
Practical 1: Implement Naïve Bayes algorithm using sample data.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.impute import SimpleImputer
df=pd.read_csv("/content/autism_data.csv")
df.head()
le=LabelEncoder()
for col in df.columns:
  if df[col].dtype=='object':
    df[col]=le.fit_transform(df[col])
df.head()
#df['Class/ASD']=le.fit_transform(df['Class/ASD'])
#df.head()
X=df.drop('Class/ASD',axis=1)
y=df['Class/ASD']
imputer=SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
nb_model=GaussianNB()
nb_model.fit(X_train,y_train)
print("Naive Bayes classification Training Score")
y_pred_train=nb_model.predict(X_train)
```

```
cm_NB_Train = confusion_matrix(y_train, y_pred_train)

print(cm_NB_Train)

print(classification_report(y_train,  y_pred_train))

print("Naive Bayes classification Testing Score")

y_pred_test=nb_model.predict(X_test)

cm_NB_Test = confusion_matrix(y_test, y_pred_test)

print(cm_NB_Test)

print(classification_report(y_test, y_pred_test))
```

Output: Naïve Bayes



```
Naive Bayes classification Training Score
[[401   9]
 [  5 148]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       410
           1       0.94      0.97      0.95       153

    accuracy                           0.98       563
   macro avg       0.97      0.97      0.97       563
weighted avg       0.98      0.98      0.98       563

Naive Bayes classification Testing Score
[[104   1]
 [  1  35]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       105
           1       0.97      0.97      0.97        36

    accuracy                           0.99       141
   macro avg       0.98      0.98      0.98       141
weighted avg       0.99      0.99      0.99       141
```

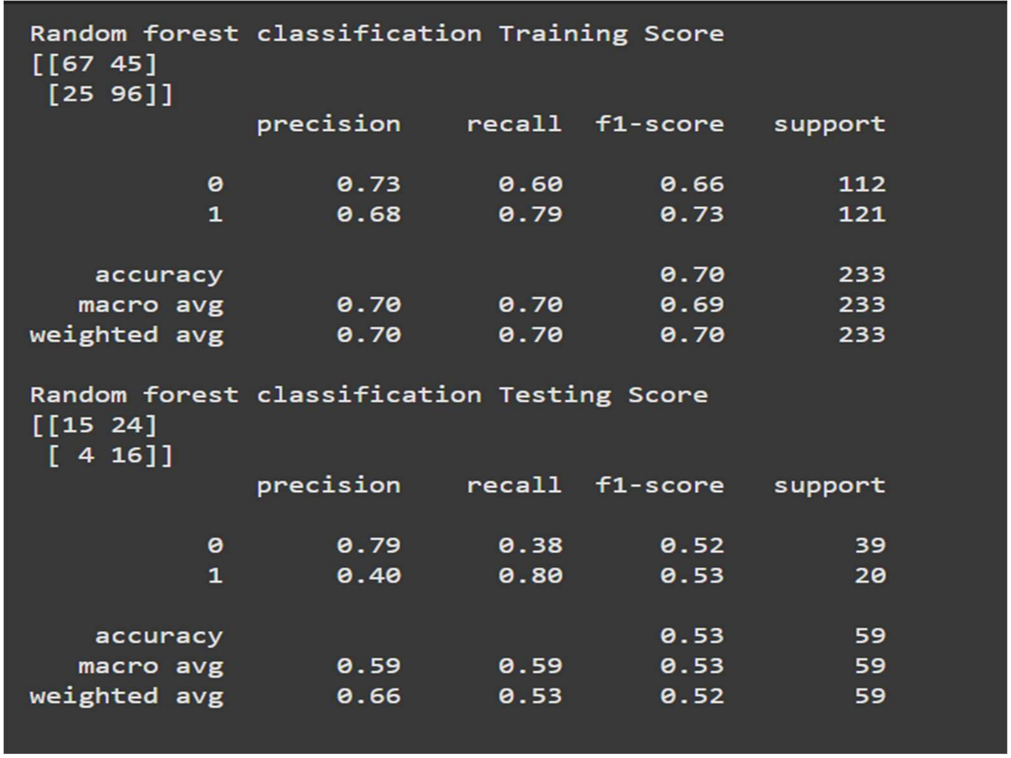
Practical 2: Implement Random Forest and ensemble learning techniques.

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

from sklearn.impute import SimpleImputer

df=pd.read_csv("/content/Autism-Child-Data.csv")

df.head()

le=LabelEncoder()

for col in df.columns:

  if df[col].dtype=='object': df[col]=le.fit_transform(df[col])

df.head() #df['Class/ASD']=le.fit_transform(df['Class/ASD']) #df.head()

df=df.dropna(subset=['Class/ASD'])

X=df[['gender' , 'austim' , 'used_app_before' ,'relation' , 'ethnicity']]

y=df['Class/ASD']

imputer=SimpleImputer(strategy='mean')

X = imputer.fit_transform(X)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

rf=RandomForestClassifier(n_estimators=100 , random_state=42)

rf.fit(X_train,y_train)


print("Random forest classification Training Score")

y_pred_train=rf.predict(X_train)

cm_NB_Train = confusion_matrix(y_train, y_pred_train)
```

```
print(cm_NB_Train)

print(classification_report(y_train, y_pred_train))

print("Random forest classification Testing Score")

y_pred_test=rf.predict(X_test)

cm_NB_Test = confusion_matrix(y_test, y_pred_test)

print(cm_NB_Test)

print(classification_report(y_test,y_pred_test))
```

Output: Random forest



```
Random forest classification Training Score
[[67 45]
 [25 96]]
              precision    recall  f1-score   support

           0       0.73      0.60      0.66       112
           1       0.68      0.79      0.73       121

    accuracy                           0.70       233
   macro avg       0.70      0.70      0.69       233
weighted avg       0.70      0.70      0.70       233

Random forest classification Testing Score
[[15 24]
 [ 4 16]]
              precision    recall  f1-score   support

           0       0.79      0.38      0.52        39
           1       0.40      0.80      0.53        20

    accuracy                           0.53        59
   macro avg       0.59      0.59      0.53        59
weighted avg       0.66      0.53      0.52        59
```

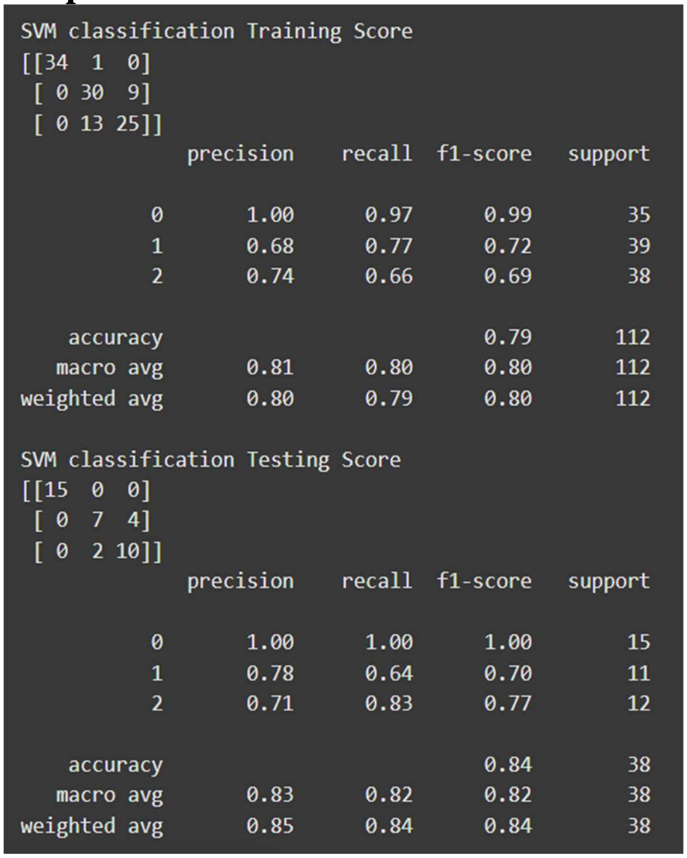
Practical 3 : Using a dataset implement SVM classifier.

Code:-

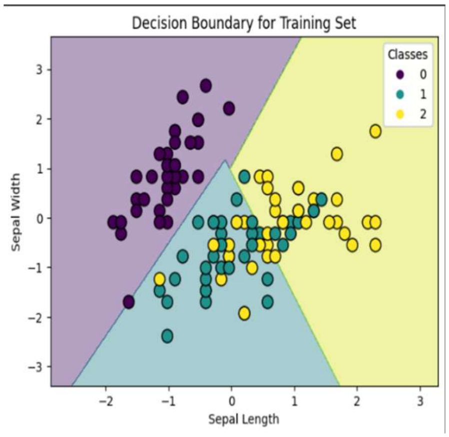
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score


iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
iris_df.head()


x = iris.data[:, :2]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25,
    random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train, y_train)


print("SVM classification Training Score")
y_pred_train = svm.predict(X_train)
cm_train = confusion_matrix(y_train, y_pred_train)
print(cm_train)
print(classification_report(y_train, y_pred_train))


print("SVM classification Testing Score")
y_pred_test = svm.predict(X_test)

cm_test = confusion_matrix(y_test, y_pred_test)
print(cm_test)
print(classification_report(y_test, y_pred_test))
```

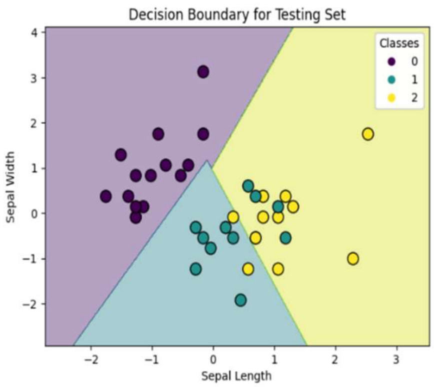
```
def plot_decision_boundary(x, y, model, ax):
  x_min, x_max = x[:, 0].min() - 1, x[:, 0].max() + 1
  y_min, y_max = x[:, 1].min() - 1, x[:, 1].max() + 1
  xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max,
    0.01))
  z = model.predict(np.c_[xx.ravel(), yy.ravel()])
  z = z.reshape(xx.shape)
  ax.contourf(xx, yy, z, alpha=0.4)
  scatter = ax.scatter(x[:, 0], x[:, 1], c=y, cmap='viridis', s=100, edgecolor='k')
  ax.set_xlabel('Sepal Length')
  ax.set_ylabel('Sepal Width')
  ax.set_title('Decision Boundary')
  legend1 = ax.legend(*scatter.legend_elements(), title="Classes")
  ax.add_artist(legend1)
  return ax

fig, ax = plt.subplots()
plot_decision_boundary(X_train, y_train, svm, ax)
ax.set_title('Decision Boundary for Training Set')
plt.show()


fig, ax = plt.subplots()
plot_decision_boundary(X_test, y_test, svm, ax)
ax.set_title('Decision Boundary for Testing Set')
plt.show()
```

Output:- SVM



```
SVM classification Training Score
[[34  1  0]
 [ 0 30  9]
 [ 0 13 25]]
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        35
           1       0.68      0.77      0.72        39
           2       0.74      0.66      0.69        38

    accuracy                           0.79       112
   macro avg       0.81      0.80      0.80       112
weighted avg       0.80      0.79      0.80       112

SVM classification Testing Score
[[15  0  0]
 [ 0  7  4]
 [ 0  2 10]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.78      0.64      0.70        11
           2       0.71      0.83      0.77        12

    accuracy                           0.84        38
   macro avg       0.83      0.82      0.82        38
weighted avg       0.85      0.84      0.84        38
```


Decision Boundary for Training Set
Classes
Sepal Width
Sepal Length


Decision Boundary for Testing Set
Classes
Sepal Width
Sepal Length