```
#liner regression SSE,SST,R2,adjusted R2

import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

# Input data
x = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.array([3, 4, 5, 6])

model = LinearRegression()    # Create a linear regression model

model.fit(x, y)    # Fit the model to the data

y_pred = model.predict(x)  # Predict the output
m_se =  mean_squared_error(y,y_pred)#mean square error

sse = np.sum((y_pred - y) ** 2)    # Calculate SSE (Sum of Squared Errors)

sst = np.sum((y - np.mean(y)) ** 2)  # Calculate SST (Total Sum of Squares)

r2 = r2_score(y, y_pred)  # Calculate R2 score

# Calculate adjusted R2
n = x.shape[0]  # Number of samples
p = x.shape[1]  # Number of predictors
adjusted_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)

# Print the results
print("mean square error",m_se)
print("Sum of Squared Errors(SSE):- ", sse)
print("Total Sum of Squares(SST):- ", sst)
print("R Square(R2):- ", r2)
print("Adjusted Square(R2):- ", adjusted_r2 )
```

#liner regertion with graph

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.read_csv('placement.csv')
X = df.iloc[:,0:1]
y = df.iloc[:,-1]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2)
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
plt.scatter(df['cgpa'],df['package'])
plt.plot(X_train,lr.predict(X_train),color='red')
plt.xlabel('CGPA')
plt.ylabel('Package(in lpa)')
plt.show()
```

**WRITE A PROGRAM TO IMPLEMENT DECISION TREE USING PYTHON/R/PROGRAMMING LANGUAGE OF YOUR CHOICE (**load_iris()**)**

```
import matplotlib.pyplot as plt
import pandas as pd
import sklearn.datasets
data_b = sklearn.datasets.load_iris()
df=pd.DataFrame(data_b.data,columns=data_b.feature_names)
df['target'] = data_b.target
#df['target']
print(df)
#print(data_b)
print("Dataset Labels=",data_b.target_names)
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import tree
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df[data_b.feature_names], df['target'])
print(x_train)
print(x_test)
print(y_train)
print(y_test)
clf = DecisionTreeClassifier(max_depth = 5,random_state=1, criterion='gini') #'gini'
clf = clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
fn=['sepal length (cm)','sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
cn=['setosa', 'versicolor', 'virginica']

fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (4, 4), dpi = 300)
tree.plot_tree(clf, feature_names = fn, class_names = cn,filled = True); fig.savefig('dstimq.png')
```

**Write Python Code to demonstrate implementation of Decision Trees Using Python.Use BREAST CANCER Dataset**

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Load the Breast Cancer dataset
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train the Decision Tree classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Visualize the Decision Tree
plt.figure(figsize=(15, 10))
plot_tree(clf, filled=True, feature_names=data.feature_names,
class names=data.target names, rounded=True)
plt.show()
```

**Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the decision tree model.**

```
from pandas import DataFrame
from sklearn.datasets import load_iris
data_b = load_iris()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names],
df['target'], random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

**Write Python/R Programming Code to demonstrate implementation K Nearest Neighbour (KNN) Machine Learning Classifier, using BREAST CANCER Dataset**

```
from pandas import DataFrame
# from sklearn.datasets import load_iris
from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
data_b = load_breast_cancer()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names],
df['target'], random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

**find-S**

```
import pandas as pd
import numpy as np
data = pd.read_csv('FIND-S2.CSV')
concept = np.array(data)[:,:-1]
target = np.array(data)[:,-1]
def train(con,tar):
    for i,val in enumerate(tar):
        if val =='yes':
            sp_h=con[i].copy()
            break
    for i,val in enumerate(con):
        if tar[i] =='yes':
            for x in range(len(sp_h)):
                if val[x] != sp_h[x]:
                    sp_h[x] = '?'
                else:
                    pass
    return sp_h
print(train(concept,target))
```

**Candidate-Elimination**

```
import numpy as np
import pandas as pd
data = pd.read_csv('C:/Users/sarvadnya/Desktop/Sheet01.csv')
concepts = np.array(data)[:,:-1]
print("\nInstances are:\n",concepts)
target = np.array(data)[:,-1]
print("\nTarget Values are: ",target)
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("\nInitialization of specific_h and genearal_h")
    print("\nSpecific Boundary: ", specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print("\nGeneric Boundary: ",general_h)
    for i, h in enumerate(concepts):
        print("\nInstance", i+1 , "is ", h)
        if target[i] == "Yes":
            print("Instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'
                else:
                    print("Instance is Negative ")
    for x in range(len(specific_h)):
        if h[x] != specific_h[x] and specific_h[x] != '?':
            general_h[x][x] = specific_h[x]
        else:
            general_h[x][x] = '?'
        print("Specific Bundary after ", i+1, "Instance is ", specific_h)
        print("Generic Boundary after ", i+1, "Instance is ", general_h)
        print("\n")
        indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
        return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("Final Specific_h: ", s_final, sep="\n")
print("Final General_h: ", g_final, sep="\n")
```

**Write Python Code to demonstrate Precision, Recall, F1-Score of the decision tree model.**

```
from sklearn.datasets import load_iris, load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

# Load the Irish dataset
iris = load_iris()
X_iris = iris.data
y_iris = iris.target

# Split the Irish dataset into training and testing sets
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(X_iris, y_iris, test_size=0.2,
random_state=42)

# Train the KNN classifier on the Irish d3ataset
knn_iris = KNeighborsClassifier()
knn_iris.fit(X_train_iris, y_train_iris)

# Make predictions on the Irish testing set
y_pred_iris = knn_iris.predict(X_test_iris)

# Calculate the confusion matrix for Irish dataset
cm_iris = confusion_matrix(y_test_iris, y_pred_iris)
print("Confusion Matrix (Irish Dataset):")
print(cm_iris)

# Calculate precision, recall, and F-measure for Irish dataset
precision_iris = precision_score(y_test_iris, y_pred_iris, average='macro')
recall_iris = recall_score(y_test_iris, y_pred_iris, average='macro')
f1_iris = f1_score(y_test_iris, y_pred_iris, average='macro')

print("Precision (Irish Dataset):", precision_iris)
print("Recall (Irish Dataset):", recall_iris)
print("F-measure (Irish Dataset):", f1_iris)
```

**Write Python/R Programming Code to demonstrate calculate popular attribute selection measures (ASM) like Information Gain, Gain Ratio, and Gini Index etc.**

**Write Python/R Programming Code to demonstrate implementation K Nearest Neighbour (KNN) Machine Learning Classifier, using IRIS Dataset**

```
from pandas import DataFrame
from sklearn.datasets import load_iris
data_b = load_iris()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
cor=0
for i in range(len(data_b.target_names)):
   cor=cor+cm[i,i]
wrg=len(y_test)-cor
print("number of correct prediction:",cor)
print("number of worng prediction:",wrg)
```

**Write Python/R Programming Code to demonstrate implementation K Nearest Neighbour (KNN) Machine Learning Classifier, using breast cancer Dataset**

```
from pandas import DataFrame
from sklearn.datasets import load_breast_cancer
data_b = load_breast_cancer()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
cor=0
for i in range(len(data_b.target_names)):
   cor=cor+cm[i,i]
wrg=len(y_test)-cor
print("number of correct prediction:",cor)
print("number of worng prediction:",wrg)
```

**Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the KNN Model using IRIS Dataset**

```
from pandas import DataFrame
from sklearn.datasets import load_iris
data_b = load_iris()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

**Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the KNN Model using brest Dataset**

```
from pandas import DataFrame
from sklearn.datasets import load_breast_cancer
data_b = load_breast_cancer()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

**Write Python/R Programming Code to implement the K-Nearest Neighbour (KNN) algorithm to classify the IRIS dataset.**

```
from pandas import DataFrame
from sklearn.datasets import load_iris
data_b = load_iris()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
cor=0
for i in range(len(data_b.target_names)):
   cor=cor+cm[i,i]
wrg=len(y_test)-cor
print("number of correct prediction:",cor)
print("number of worng prediction:",wrg)
```

**Write Python/R Programming Code to implement the K-Nearest Neighbour (KNN) algorithm to classify the Brest cancer dataset.**

```
from pandas import DataFrame
from sklearn.datasets import load_breast_cancer
data_b = load_breast_cancer()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
cor=0
for i in range(len(data_b.target_names)):
   cor=cor+cm[i,i]
wrg=len(y_test)-cor
print("number of correct prediction:",cor)
print("number of worng prediction:",wrg)
```

**Write Python/R Programming Code to demonstrate Precision, Recall, F1- Score of the KNN model.**

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

# Load the Irish dataset
iris = load_iris()
X_iris = iris.data
y_iris = iris.target

# Split the Irish dataset into training and testing sets
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(X_iris, y_iris, test_size=0.2,
random_state=42)

# Train the KNN classifier on the Irish d3ataset
knn_iris = KNeighborsClassifier()
knn_iris.fit(X_train_iris, y_train_iris)

# Make predictions on the Irish testing set
y_pred_iris = knn_iris.predict(X_test_iris)

# Calculate the confusion matrix for Irish dataset
cm_iris = confusion_matrix(y_test_iris, y_pred_iris)
print("Confusion Matrix (Irish Dataset):")
print(cm_iris)

# Calculate precision, recall, and F-measure for Irish dataset
precision_iris = precision_score(y_test_iris, y_pred_iris, average='macro')
recall_iris = recall_score(y_test_iris, y_pred_iris, average='macro')
f1_iris = f1_score(y_test_iris, y_pred_iris, average='macro')

print("Precision (Irish Dataset):", precision_iris)
print("Recall (Irish Dataset):", recall_iris)
print("F-measure (Irish Dataset):", f1_iris)
```

**Write Python/R Programming Code Print both correct and wrong predictions and Accuracy of the KNN Model**

```
from pandas import DataFrame
from sklearn.datasets import load_breast_cancer
data_b = load_breast_cancer()
df= DataFrame(data_b.data, columns=data_b.feature_names)
df['target'] = data_b.target
#print(df)
#print(data_b.DESCR)
print("Dataset Labels=",data_b.target_names)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, y_test = train_test_split(df[data_b.feature_names], df['target'],
random_state=1)
print(X_train.head(6))
print(Y_train.head(6))
print(X_test.head())
clf = KNeighborsClassifier(n_neighbors=6)
clf.fit(X_train, Y_train) # model is trained
y_pred=clf.predict(X_test)
#print(y_test, y_pred)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
cor=0
for i in range(len(data_b.target_names)):
   cor=cor+cm[i,i]
wrg=len(y_test)-cor
print("number of correct prediction:",cor)
print("number of worng prediction:",wrg)
```

**Write Python/R Programming Code Print both correct and wrong predictions and Print Accuracy of the Naive Bayes Classifier Model**

```
#naive Basesian Classfier
# for dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
iris=datasets.load_iris()
x=iris.data
y=iris.target
print("Features:",iris['feature_names'])

#Accuracy Confusion Matrix

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_stat
e=0)
NB=GaussianNB()
NB.fit(x_train,y_train)
y_pred=NB.predict(x_test)
cm=confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cm)
```

**Write Python/R Programming Code to implement the implement Naïve Bayes Classifier to classify the IRIS dataset**

```
#naive Basesian Classfier
# for dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
iris=datasets.load_iris()
x=iris.data
y=iris.target
print("Features:",iris['feature_names'])

#Accuracy Confusion Matrix

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
NB=GaussianNB()
NB.fit(x_train,y_train)
y_pred=NB.predict(x_test)
cm=confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cm)
```

**Write Python/R Programming Code Print Precision, Recall, F1-Score of the Naive Bayes Classifier Model.**

```
#naive Basesian Classfier
# for dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, precision_score, recall_score,
f1_score
iris=datasets.load_iris()
x=iris.data
y=iris.target
print("Features:",iris['feature_names'])

#Accuracy Confusion Matrix

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_stat
e=0)
NB=GaussianNB()
NB.fit(x_train,y_train)
y_pred=NB.predict(x_test)
cm=confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cm)

# Calculate precision, recall, and F-measure for Irish dataset
precision_iris = precision_score(y_test, y_pred, average='macro')
recall_iris = recall_score(y_test, y_pred, average='macro')
f1_iris = f1_score(y_test, y_pred, average='macro')

print("Precision (Irish Dataset):", precision iris)
print("Recall (Irish Dataset):", recall_iris)
print("F-measure (Irish Dataset):", f1_iris)
```

**Write Python/R Programming Code to demonstrate Accuracy and Confusion Matrix of the Naive Bayes Classifier Model.**

```
#naive Basesian Classfier
# for dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets, __all__
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import confusion_matrix, precision_score, recall_score,
f1_score
iris=datasets.load_iris()
x=iris.data
y=iris.target
print("Features:",iris['feature_names'])

#Accuracy Confusion Matrix


x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_stat
e=0)
NB=GaussianNB()
NB.fit(x_train,y_train)
y_pred=NB.predict(x_test)
cm=confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cm)
print("Accuracy:",accuracy score(y test, y pred))
```

**Write Python/R Programming Code for Implementing Agglomerative Clustering in Python**

```
from sklearn.cluster import AgglomerativeClustering
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# Generate sample data
X, y = make_blobs(n_samples=200, centers=4, random_state=0)

# Create an instance of AgglomerativeClustering
clustering = AgglomerativeClustering(n_clusters=4)

# Perform clustering
clustering.fit(X)

# Retrieve the cluster labels
labels = clustering.labels_

# Plot the data points with their corresponding cluster labels
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.title("Agglomerative Clustering")
plt.show()
```

**Write a Program for Fuzzy c-means clustering in Python.**

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# Generate some example data
np.random.seed(0)
data = np.random.rand(100, 2)

# Define the number of clusters
n_clusters = 3

# Apply fuzzy c-means clustering
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
data.T, n_clusters, 2, error=0.005, maxiter=1000, init=None

)
```