Programs

Kmeans

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

#make\_blobs is a function provided by the scikit-learn library in Python for generating synthetic datasets.

#It's commonly used for clustering and classification tasks.

#The function creates clusters of points, where each cluster is characterized by a Gaussian distribution of points.

X,y=make\_blobs(n\_samples=500,centers=4,cluster\_std=.8,random\_state=42)

print("Shape of X:", X.shape)

print("Shape of y:", y.shape)

kmeans = KMeans(n\_clusters = 4,  random\_state = 42)

kmeans.fit(X)

labels=kmeans.labels\_

labels

plt.figure(figsize=(8,6))

plt.scatter(X[:,0],X[:,1],c=labels,cmap='viridis')

plt.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],s=100,c='red',label='centroids')

plt.title('K means Clustering')

plt.xlabel('X')

plt.ylabel('Y')

plt.show()

**Implement Bagging**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

iris=load\_iris()

X=iris.data

y=iris.target

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

from sklearn.ensemble import RandomForestClassifier

dt= RandomForestClassifier(n\_estimators= 10, criterion="entropy")

dt.fit(X\_train,y\_train)

y\_pred=dt.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix,accuracy\_score

result1 = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(result1)

result2 = classification\_report(y\_test, y\_pred)

print("Classification Report:",)

print (result2)

result3 = accuracy\_score(y\_test,y\_pred)

print("Accuracy:",result3)

**Implement Boosting**

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Load Iris dataset

iris=load\_iris()

X=iris.data

y=iris.target

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

from sklearn.ensemble import GradientBoostingClassifier

gb=GradientBoostingClassifier()

gb.fit(x\_train,y\_train)

y\_pred=gb.predict(x\_test)

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

result = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(result)

result1 = classification\_report(y\_test, y\_pred)

print("Classification Report:",)

print (result1)

**Implement Stacking**

#Stacking

from sklearn.datasets import load\_iris

from sklearn.ensemble import StackingClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

iris=load\_iris()

X=iris.data

y=iris.target

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Define base learners

base\_learners = [

('decision\_tree', DecisionTreeClassifier(max\_depth=1)),

('lr', LogisticRegression()) ]

# Define the meta-learner

meta\_learner = SVC(probability=True, random\_state=42)

# Initialize the Stacking Classifier with the base learners and the meta-learner

stack\_clf = StackingClassifier(estimators=base\_learners, final\_estimator=meta\_learner)

# Train the stacking classifier

stack\_clf.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = stack\_clf.predict(X\_test)

# Evaluate and print the accuracy of the model

print("Stacking Model Accuracy:", accuracy\_score(y\_test, y\_pred))

from sklearn.metrics import classification\_report, confusion\_matrix,

result2 = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(result12)

result11 = classification\_report(y\_test, y\_pred)

print("Classification Report:",)

print (result11)