**Using Keras/any standard dataset write the programs for the following Machine learning tasks:**

**Lab 7. Use the Decision tree classifier to classify the dataset.**

**Program**

from sklearn.datasets import load\_breast\_cancer

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Load breast cancer dataset

data = load\_breast\_cancer()

X = data.data # Features

y = data.target # Target (0: malignant, 1: benign)

# Split 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)

# Initialize Decision Tree Classifier

clf = DecisionTreeClassifier()

# Train the classifier

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

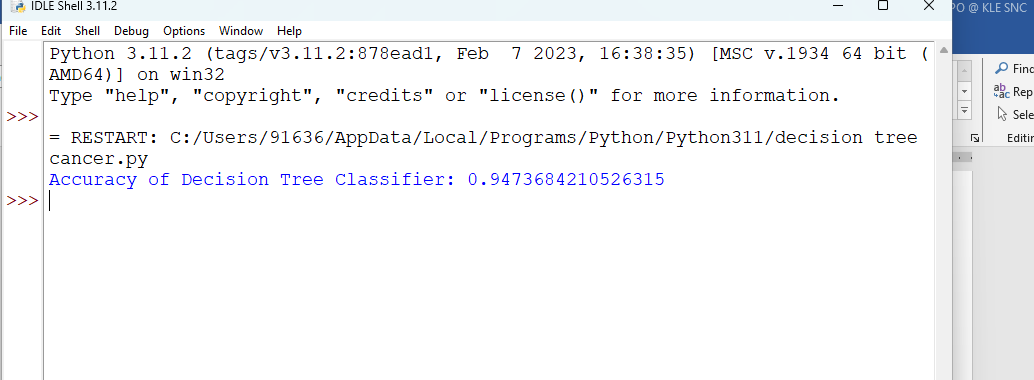
# Predict on the test set

predictions = clf.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, predictions)

print(f"Accuracy of Decision Tree Classifier: {accuracy}")

**output :**

**Lab 8 : Use the Naïve Bayes classifier to classify the dataset.**

from sklearn.datasets import load\_iris

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

# Load dataset (Example with Iris dataset)

iris = load\_iris()

X = iris.data

y = iris.target

# Split 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)

# Initialize Gaussian Naive Bayes Classifier

clf\_nb = GaussianNB()

# Train the classifier

clf\_nb.fit(X\_train, y\_train)

# Predict on the test set

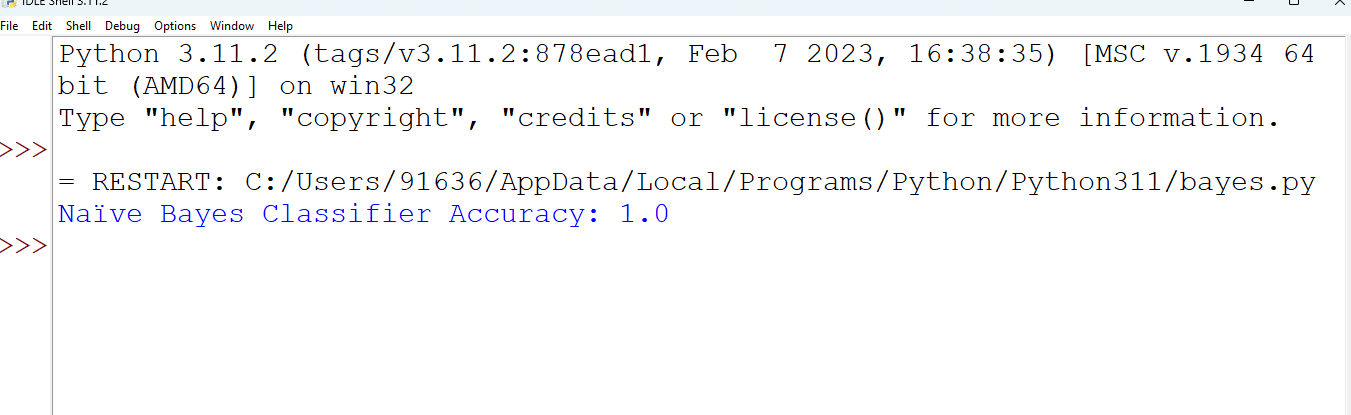
predictions\_nb = clf\_nb.predict(X\_test)

# Calculate accuracy

accuracy\_nb = accuracy\_score(y\_test, predictions\_nb)

print(f"Naïve Bayes Classifier Accuracy: {accuracy\_nb}")

**output:**

****

**Lab 9 :Implement K-Means clustering Algorithm.**

1. Program using iris flower dataset

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

import numpy as np

# Generate sample data

X, \_ = make\_blobs(n\_samples=100, centers=3, cluster\_std=1.0)

# Initialize KMeans

kmeans = KMeans(n\_clusters=3)

# Fit KMeans to data

kmeans.fit(X)

# Get cluster centroids and labels

centroids = kmeans.cluster\_centers\_

labels = kmeans.labels\_

# Print cluster centroids and labels

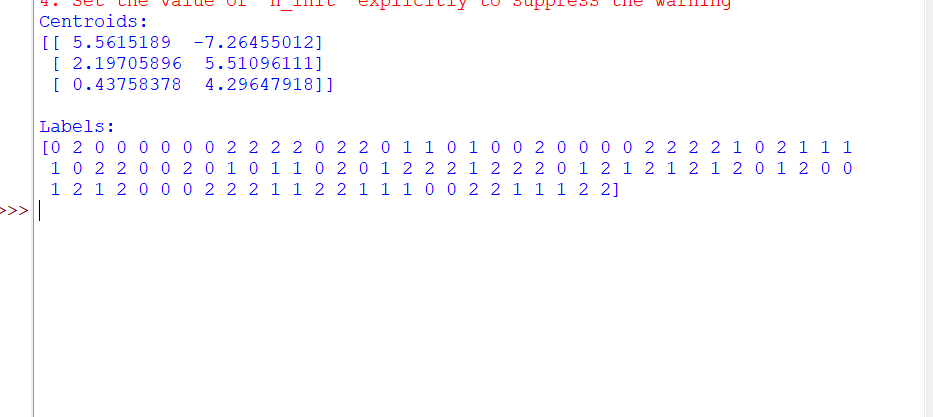
print("Centroids:")

print(centroids)

print("\nLabels:")

print(labels)

**output:**



1. Program using cancer dataset

from sklearn.datasets import load\_breast\_cancer

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

# Load breast cancer dataset

data = load\_breast\_cancer()

X = data.data # Features

# Standardize the features (important for K-Means)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Apply K-Means clustering

kmeans = KMeans(n\_clusters=2) # Assuming 2 clusters for illustration

kmeans.fit(X\_scaled)

# Get cluster labels and centroids

cluster\_labels = kmeans.labels\_

centroids = kmeans.cluster\_centers\_

# Visualize the clusters (for first two features for simplicity)

plt.scatter(X\_scaled[:, 0], X\_scaled[:, 1], c=cluster\_labels, cmap='viridis', marker='o', alpha=0.5)

plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=200, c='red', label='Centroids')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('K-Means Clustering')

plt.legend()

plt.show()

Oytput :

