**Ex. No.: 9 a.**

**Date: 6/11/24**

**A PYTHON PROGRAM TO IMPLEMENT KNN MODEL**

**Aim:**

To implement a python program using a KNN Algorithm in a model.

**Algorithm:**

1. Import Necessary Libraries

 Import necessary libraries: pandas, numpy, train\_test\_split from sklearn.model\_selection, StandardScaler from sklearn.preprocessing, KNeighborsClassifier from sklearn.neighbors, and classification\_report and confusion\_matrix from sklearn.metrics.

1. Load and Explore the Dataset
   * + Load the dataset using pandas.
     + Display the first few rows of the dataset using df.head().
     + Display the dimensions of the dataset using df.shape().
     + Display the descriptive statistics of the dataset using df.describe().
2. Preprocess the Data
   * + Separate the features (X) and the target variable (y).
     + Split the data into training and testing sets using train\_test\_split.
     + Standardize the features using StandardScaler.
3. Train the KNN Model
   * + Create an instance of KNeighborsClassifier with a specified number of neighbors

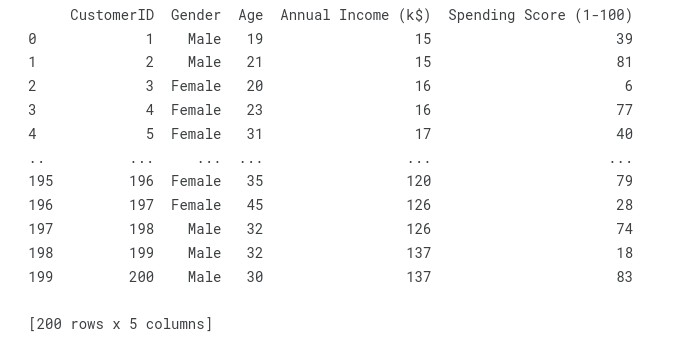
(k).

* + - For each data point, calculate the Euclidean distance to all other data points.
    - Select the K nearest neighbors based on the calculated Euclidean distances.
    - Among the K nearest neighbors, count the number of data points in each category.
    - Assign the new data point to the category for which the number of neighbors is maximum.

1. Make Predictions
   * Use the trained model to make predictions on the test data.
   * Evaluate the Model
   * Generate the confusion matrix and classification report using the actual and predicted values.
   * Print the confusion matrix and classification report.

import numpy as np import matplotlib.pyplot as plt import pandas as pd

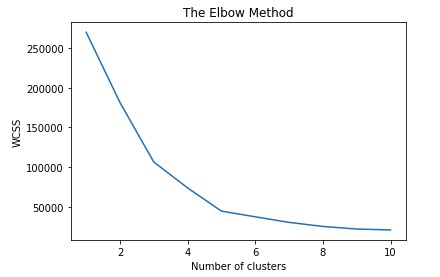
dataset = pd.read\_csv('../input/mall-customers/Mall\_Customers.csv') X = dataset.iloc[:,[3,4]].values print(dataset)



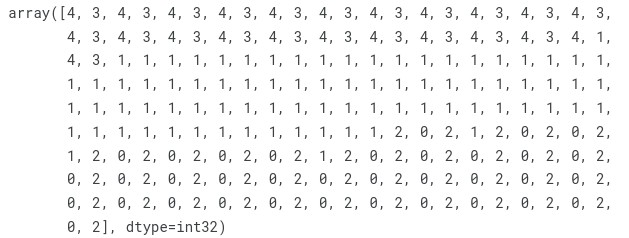
from sklearn.cluster import KMeans wcss =[] for i in range (1,11):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', max\_iter =300, n\_init = 10, random\_state = 0) kmeans.fit(X) wcss.append(kmeans.inertia\_)

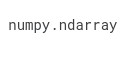
# Plot the graph to visualize the Elbow Method to find the optimal number of cluster plt.plot(range(1,11),wcss) plt.title('The Elbow Method') plt.xlabel('Number of clusters') plt.ylabel('WCSS') plt.show()



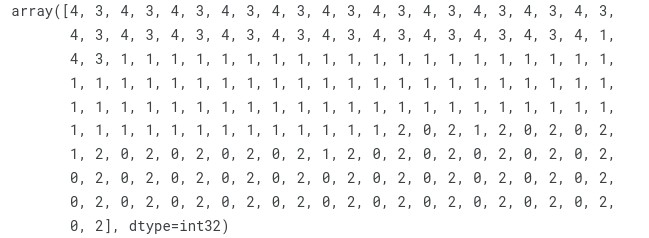
kmeans=KMeans(n\_clusters= 5, init = 'k-means++', max\_iter = 300, n\_init = 10, random\_state = 0) y\_kmeans = kmeans.fit\_predict(X) y\_kmeans



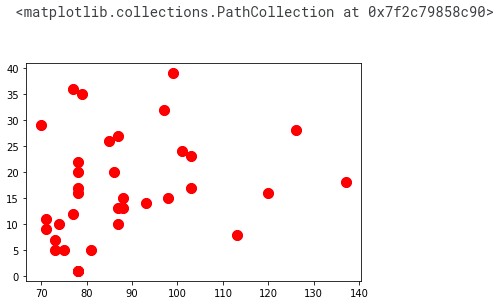
type(y\_kmeans)



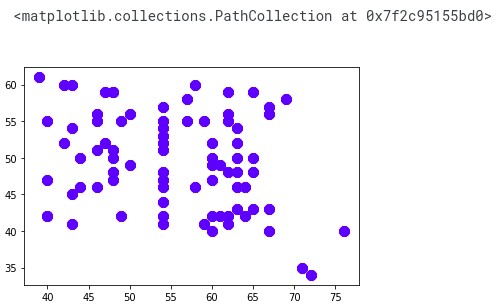
y\_kmeans



plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

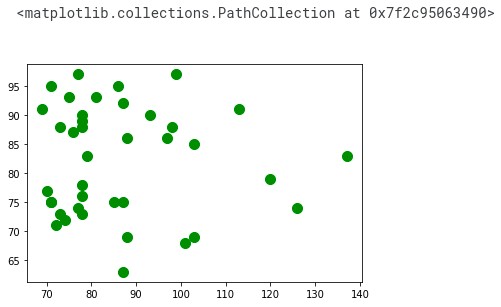


plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

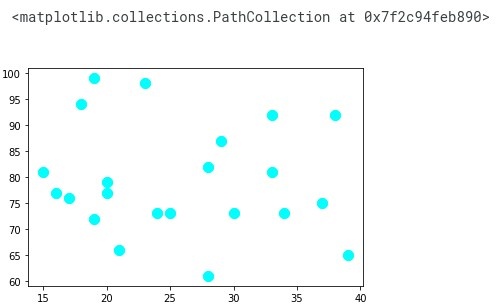


plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster

3')

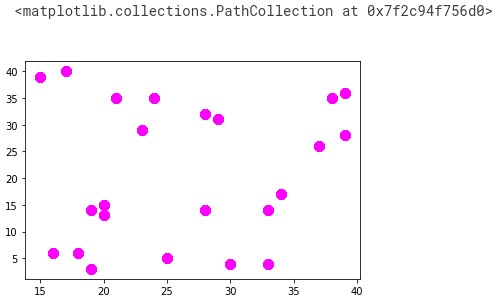


plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

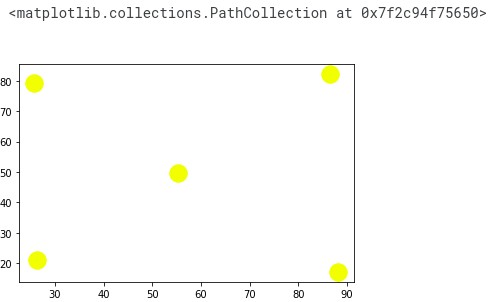


plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label =

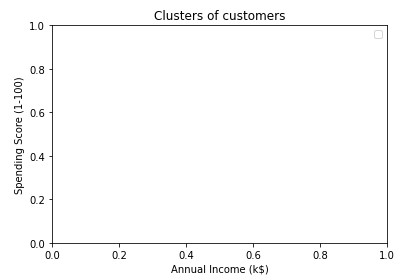
'Cluster 5')



plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroids')



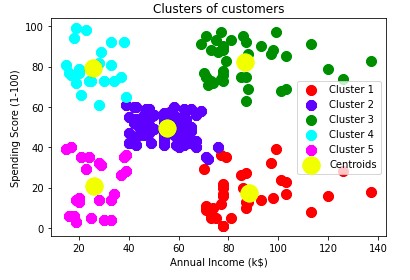
plt.title('Clusters of customers') plt.xlabel('Annual Income (k$)') plt.ylabel('Spending Score (1-100)') plt.legend() plt.show()



plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1') plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2') plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster

3') plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label =

'Cluster 5') plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroids') plt.title('Clusters of customers') plt.xlabel('Annual Income (k$)') plt.ylabel('Spending Score (1-100)') plt.legend() plt.show()



**RESULT:-**

Thus the python program to implement KNN model has been successfully implemented and the results have been verified and analyzed.