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.
2. 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().
3. 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.
4. 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.
5. Make Predictions

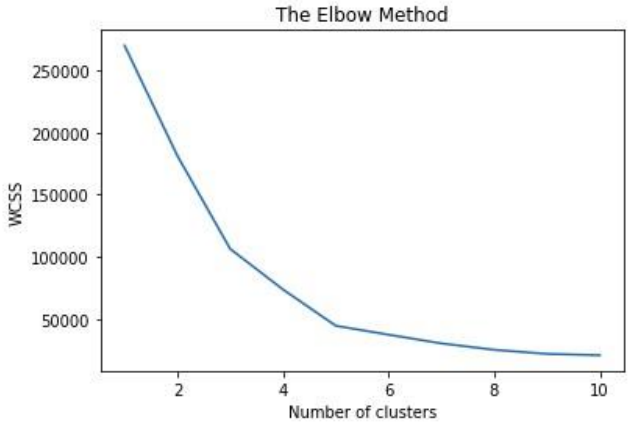
- Use the trained model to make predictions on the test data.
- Evaluate the Model
- Generate the confusion matrix and classification report using the actual and predicted values.
- Print the confusion matrix and classification report.

```
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)
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
..	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

[200 rows x 5 columns]

```
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

```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2], dtype=int32)
```

type(y_kmeans)

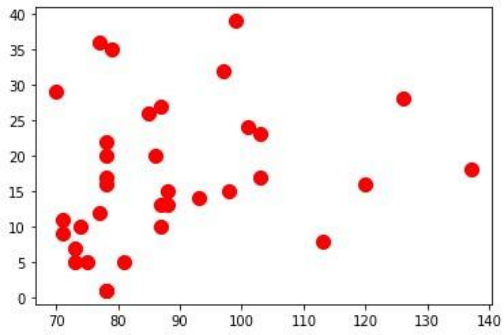
```
numpy.ndarray
```

y_kmeans

```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2], dtype=int32)
```

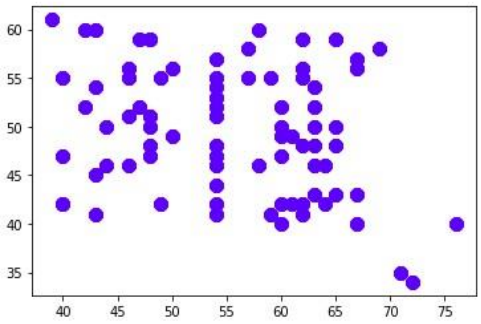
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

```
<matplotlib.collections.PathCollection at 0x7f2c79858c90>
```



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

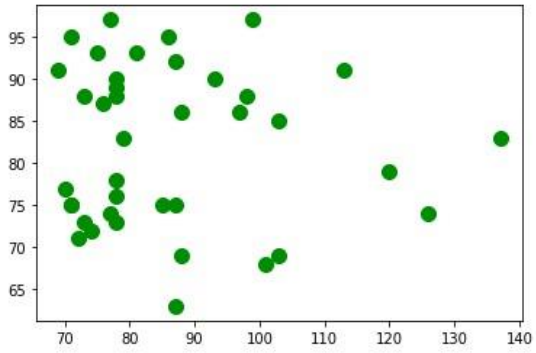
```
<matplotlib.collections.PathCollection at 0x7f2c95155bd0>
```



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

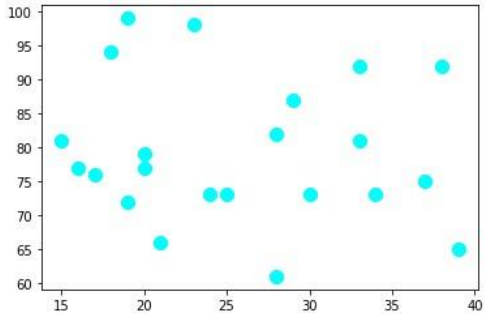
3')

```
<matplotlib.collections.PathCollection at 0x7f2c95063490>
```



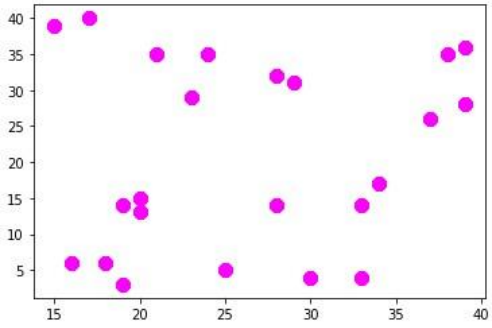
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

```
<matplotlib.collections.PathCollection at 0x7f2c94feb890>
```



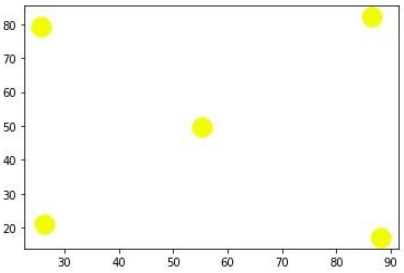
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

```
<matplotlib.collections.PathCollection at 0x7f2c94f756d0>
```

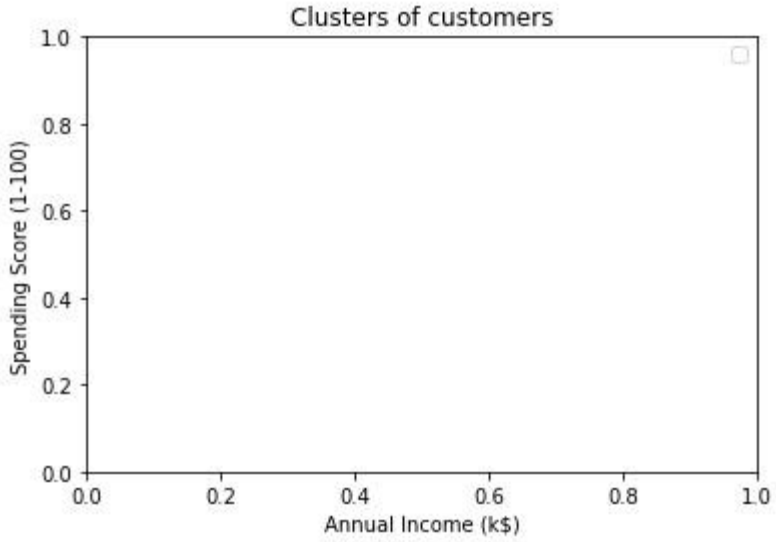


plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')

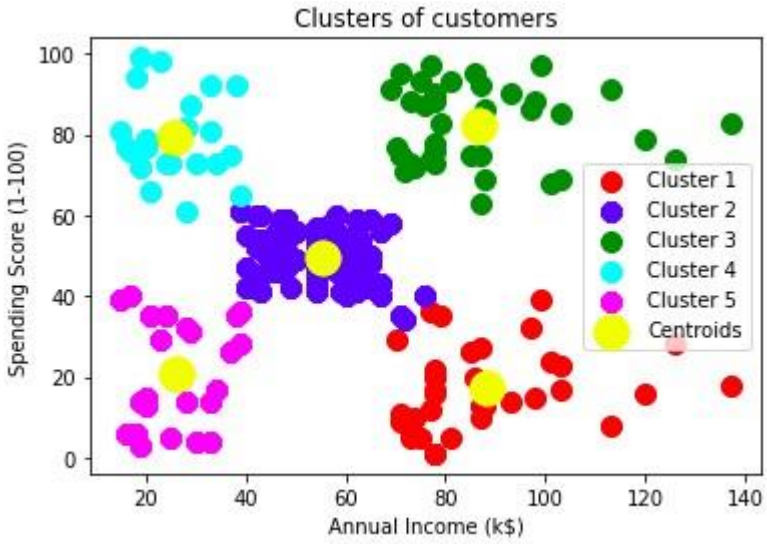
```
<matplotlib.collections.PathCollection at 0x7f2c94f75650>
```



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.