A PYTHON PROGRAM TO IMPLEMENT KNN MODEL

Name: Meetha Dinesan
Roll no : 241801156
Exp no: 9

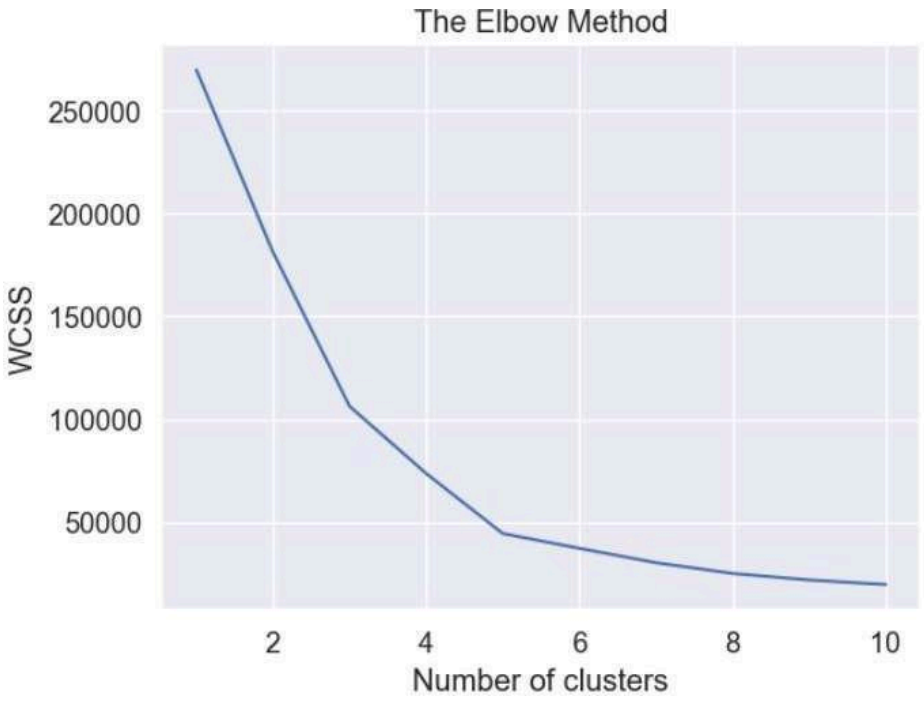
Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans

dataset = pd.read_csv('../input/mallcustomers/
Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
print(dataset)

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_)
```

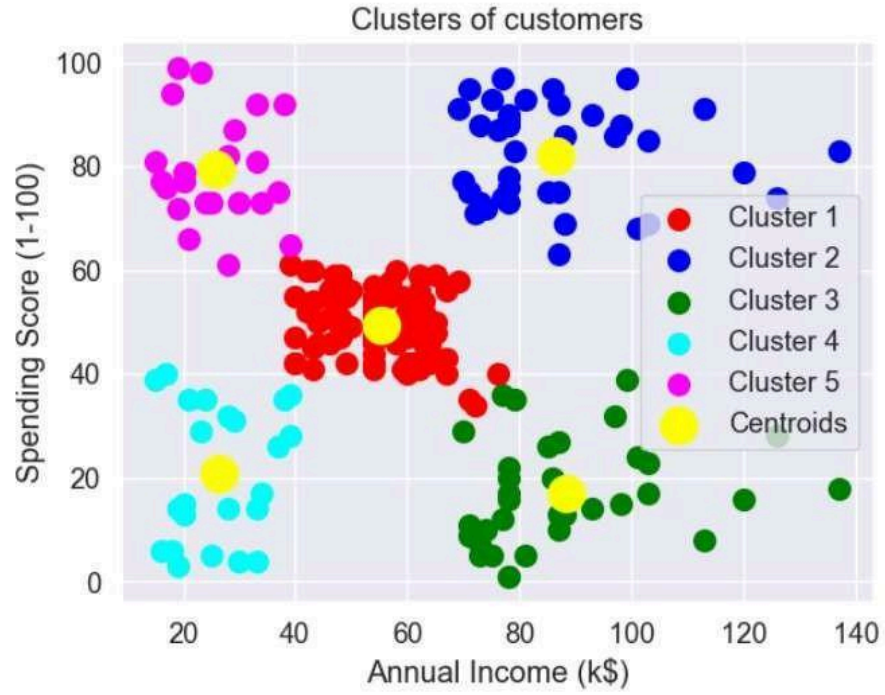
```
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

kmeans = KMeans(n_clusters=5, init='k-means++',
max_iter=300, n_init=10, random_state=0)
y_kmeans = kmeans.fit_predict(X)
print(type(y_kmeans))
print(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()
```

output:



```
<class 'numpy.ndarray'>
[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 3 4 3 0 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 0 1 2 1 2 1 0 1 2 1 2 1 2 1 2 1 0 1 2 1 2 1
 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1]
```



A PYTHON PROGRAM TO IMPLEMENT K-MEANS MODEL

Code:

```
import pandas as pd

import numpy as np

from math import sqrt


data = pd.read_csv('../input/k-means-clustering/KNN (3).csv')

req_data = data.iloc[:, 1:]
```

```
shuffle_index = np.random.permutation(req_data.shape[0])
req_data = req_data.iloc[shuffle_index]

train_size = int(req_data.shape[0] * 0.7)
train_df = req_data.iloc[:train_size, :]
test_df = req_data.iloc[train_size:, :]

train = train_df.values
test = test_df.values
y_true = test[:, -1]

print('Train_Shape:', train_df.shape)
print('Test_Shape:', test_df.shape)

def euclidean_distance(x_test, x_train):
    distance = 0
    for i in range(len(x_test) - 1):
        distance += (x_test[i] - x_train[i]) ** 2
    return sqrt(distance)
```

```
def get_neighbors(x_test, x_train, num_neighbors):
    distances = []
    data = []
    for i in x_train:
        distances.append(euclidean_distance(x_test, i))
        data.append(i)
    distances = np.array(distances)
    data = np.array(data)
    sort_indexes = distances.argsort()
    data = data[sort_indexes]
    return data[:num_neighbors]

def prediction(x_test, x_train, num_neighbors):
    classes = []
    neighbors = get_neighbors(x_test, x_train, num_neighbors)
    for i in neighbors:
        classes.append(i[-1])
    predicted = max(classes, key=classes.count)
    return predicted
```

```
def predict_classifier(x_test):
    classes = []
    neighbors = get_neighbors(x_test, req_data.values, 5)
    for i in neighbors:
        classes.append(i[-1])
    predicted = max(classes, key=classes.count)
    print(predicted)
    return predicted


def accuracy(y_true, y_pred):
    num_correct = 0
    for i in range(len(y_true)):
        if y_true[i] == y_pred[i]:
            num_correct += 1
    accuracy = num_correct / len(y_true)
    return accuracy

y_pred = []
for i in test:
    y_pred.append(prediction(i, train, 5))
```

acc = accuracy(y_true, y_pred)

print('Accuracy:', acc)

output:

```
Train_Shape: (350, 1)
Test_Shape: (150, 1)
Accuracy: 0.0
```