2. Write a Python program to extract social_network_ads.csv file. Apply k-Nearest Neighbor technique to identify the users who purchased the item or not.

KNN (K Nearest Neighbors) algorithm is a supervised Machine Learning classification algorithm. It is one of the simplest and widely used classification algorithms in which a new data point is classified based on similarity in the specific group of neighboring data points. This gives a competitive result.

Working:

For a given data point in the set, the algorithms find the distances between this and all other K numbers of data point in the dataset close to the initial point and votes for that category that has the most frequency. Usually, Euclidean distance is taking as a measure of distance. Thus the end resultant model is just the labeled data placed in a space. This algorithm is popularly known for various applications like genetics, forecasting, etc. The algorithm is best when more features are present.

KNN reducing over fitting is a fact. On the other hand, there is a need to choose the best value for K. So now how do we choose K? Generally we use the Square root of the number of samples in the dataset as value for K. An optimal value has to be found out since lower value may lead to overfitting and higher value may require high computational complication in distance. So using an error plot may help. Another method is the elbow method. You can prefer to take root else can also follow the elbow method.

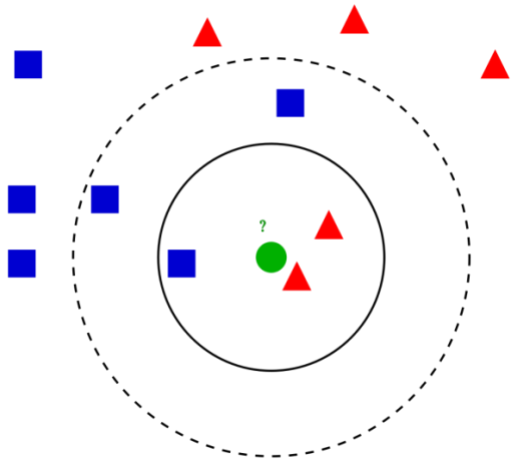
Different steps of K-NN for classifying a new data point:

Step 1: Select the value of K neighbors (say k=5)

Step 2: Find the K (5) nearest data point for our new data point based on Euclidean distance.

Step 3: Among these K data points count the data points in each category.

Step 4: Assign the new data point to the category that has the most neighbors of the new data point.



Example:

Consider an example problem for getting a clear intuition on the K -Nearest Neighbor classification. We are using the Social network ad dataset. The dataset contains the details of users in a social networking site to find whether a user buys a product by clicking the ad on the site based on their salary, age, and gender.

Importing essential libraries:

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import sklearn
```

Importing of the dataset and slicing it into independent and dependent variables:

```
dataset = pd.read_csv('Social_Network_Ads.csv')

X = dataset.iloc[:, [1, 2, 3]].values

y = dataset.iloc[:, -1].values
```

Since the dataset containing character variables, need to encode it using LabelEncoder.

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X[:,0] = le.fit_transform(X[:,0])
```

Split the dataset into train and test set. Providing the test size as 0.20, that means training sample contains 320 training set and test sample contains 80 tests set.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.20, random_state = 0)
```

Next, feature scaling is done to the training and test set of independent variables for reducing the size to smaller values.

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)
```

Build and train the K Nearest Neighbor model with the training set.

```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski',
p = 2)

classifier.fit(X_train, y_train)
```

Three different parameters are used in the model creation. n_neighbors is setting as 5, which means 5 neighborhood points are required for classifying a given point. The distance metric used is Minkowski. Equation for the same is given below.

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

As per the equation, we need to select the p-value.

p = 1 , Manhattan Distance

p = 2 , Euclidean Distance

p = infinity , Cheybchev Distance

In this example, we are choosing the p value as 2. Machine Learning model is created, now we have to predict the output for the test set.

```
y_pred = classifier.predict(X_test)
```

Comparing true and predicted value:

```
y_test
```

>>

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1], dtype=int64)
```

```
y_pred
```

>>

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1], dtype=int64)
```

Evaluating the model using the confusion matrix and accuracy score by comparing the predicted and actual test values.

```
from sklearn.metrics import confusion_matrix,accuracy_score

cm = confusion_matrix(y_test, y_pred)

ac = accuracy_score(y_test,y_pred)
```

Confusion Matrix :

```
cm
```

>>

```
[[64  4]
 [ 3 29]]
```

```
ac
```

>>

```
0.95
```