In [ ]:
```
# Author : Amir Shokri
# github link : https://github.com/amirshnll/Online-Shoppers-Purchasing-Intent
ion/
# dataset link : http://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purcha
sing+Intention+Dataset
# email : amirsh.nll@gmail.com
```

In [2]:
```
import pandas as pd
import numpy as np
import math
import operator
```

In [3]:
```
# Importing data
data = pd.read_csv("O_S_I_train.csv")

data.head()
```

Out[3]:

	Administrative	Informational	ProductRelated	ProductRelated_Duration	BounceRates	ExitRate
0	0	0	1	0	20.0	200
1	0	0	2	64	0.0	100
2	0	0	1	0	20.0	200
3	0	0	2	3	5.0	140
4	0	0	10	628	2.0	50

In [14]:
```
# Defining a function which calculates euclidean distance between two data poi
nts
def euclideanDistance(data1, data2, length):
    distance = 0
    for x in range(length):
        distance += np.square(data1[x] - data2[x])
    return np.sqrt(distance)


# Defining our KNN model
def knn(trainingSet, testInstance, k):
    distances = {}
    sort = {}
    length = testInstance.shape[1]
        # Calculating euclidean distance between each row of training data and
test data
    for x in range(len(trainingSet)):
        dist = euclideanDistance(testInstance, trainingSet.iloc[x], length)
        distances[x] = dist[0]
        # Sorting them on the basis of distance
        sorted_d = sorted(distances.items(), key=operator.itemgetter(1))
        neighbors = []
    for x in range(k):
        neighbors.append(sorted_d[x][0])
        classVotes = {}
        # Calculating the most freq class in the neighbors
    for x in range(len(neighbors)):
        response = trainingSet.iloc[neighbors[x]][-1]
        if response in classVotes:
            classVotes[response] += 1
        else:
            classVotes[response] = 1
        sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), r
everse=True)
    return(sortedVotes[0][0], neighbors)
```

In [16]:
```
# Creating a dummy testset
testSet = [[4,0,56,1886,2,382,5,0,11,2,5,3,11,1,0,1]]
test = pd.DataFrame(testSet)
```

In [18]:
```
# Setting number of neighbors = 5
k = 5
# Running KNN model
result,neigh = knn(data, test, k)
# Predicted class
print(result)
```

```
0.0
```