K-Nearest Neighbours Algorithm

PROBLEM STATEMENT:
Attempting to predict weight without using any library.

IMPORTANT FORMULA: Squareroot($(x_1 - x_2)^2 + (y_1 - y_2)^2$)

THEORY

K-nearest neighbours (KNN) algorithm is a supervised Machine Learning algorithm. It is used for classification and regression predictive problems. In industry, it is mainly used for classification predictive problems.

It uses ‘feature similarity’ to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

ALGORITHM

Step 1: Load the train dataset and test dataset required for the algorithm.

Step 2: Choose the K-value. (There is no particular way to determine the best k value so we need to try some values to find the best out of them. The most preferred value for K is 5. A very low value for K such as K=1 or K=2, can be noisy. Large values for K are good.)

Step 3: For each point in the test data do till Step 9

Step 4: Calculate the distance between test data and each row of training data with the help of any method. (Commonly used is Euclidean.)

Step 5: Now, based on the distance value, sort them in ascending order.

Step 6: Next, it will choose the K rows from the sorted array.

Step 7: Calculate the average of sum of previous rows for each row.

Step 8: Calculate the percentage error for each of these values with the actual value in the test data

Step 9: The value with the least percentage error is the required value to be predicted.

Advantages

- It is simple to understand, implement and interpret.
- It can be more effective if the training data is large.

Disadvanatges

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Applications

- Banking System: To predict weather an individual is fit for loan approval? Does that individual have the characteristics similar to the defaulters one?

- Politics: To classify a potential voter into various classes like “Will Vote”, “Will not Vote”
- Other areas include Speech Recognition, Handwriting Detection, Image Recognition and Video Recognition.

CODE

author: dawn elza Zachariah

date started: 7 January 2020

date last edited: 12 January 2020

python version: python 3.7

```
train=[[1,2,3],[4,5,6],[7,8,9]]#train data

test=[[9,10,11],[12,13,14],[15,16,17]]#test
data


n1=len(train)

n2=len(train[0])-1


for testrow in test:#itrating through each of
the test value

    x=[]#1Dlist for storing differences

    y1=[]#2Dlist for storing differences

    y2=[]#list for storing distance
    #finding the differences of each of the train values with the test value

    for i in range (n1):

        x=[]

        for j in range (n2):

            x.append(testrow[j]-train[i][j])

        y1.append(x)

    print("\nDifference Matrix ",y1)


    #finding the euclidean distance

    for i in range (len(y1)):

        s=0

        for j in range(len(y1[0])):

            s=s+y1[i][j]**2

        y2.append(s**0.5)

    print("\nDistance matrix ",y2)


    y3={}#dictionary for storing distance values with train dataset

    for i in range (len(y2)):

        y3[y2[i]]=train[i]#mapping each distance[key] to corresponding value in train dataset[values]

    print("\nCombined matrix ",y3)
    y3=sorted(y3.items())

    print("\nSorted  matrix  ",y3)#sorting  the
'Combined matrix' by the values of'distance'



    predict=[]#list to store predicted values

    avgsum=0

    for i in range (len(y2)):

        avgsum=avgsum+y3[i][1][2]#adding

        avg=avgsum/(i+1)#finding the average of
the sum

        predict.append(avg)

    print("\nPredicted values ",predict)



    percent=[]

    for i in range (len(predict)):

        percent.append((testrow[n2]-
predict[i])*100/(testrow[n2]))#finding
percenatge error

    print("\nPercentage error ",percent)

    print("\nFor the test value ",testrow)

    #findng  the  value  to  be  predicted  on  the
basis of least percenatge error

    print("\nMost          accurate          value
",predict[percent.index(min(percent))])
```

OUTPUT

```
Difference Matrix  [[8, 8], [5, 5], [2, 2]]

Distance matrix  [11.313708498984761,
7.0710678118654755, 2.8284271247461903]

Combined matrix  {11.313708498984761: [1, 2, 3],
7.0710678118654755: [4, 5, 6], 2.8284271247461903:
[7, 8, 9]}

Sorted matrix  [(2.8284271247461903, [7, 8, 9]),
(7.0710678118654755, [4, 5, 6]), (11.313708498984761,
[1, 2, 3])]

Predicted values  [9.0, 7.5, 6.0]

Percentage error  [18.181818181818183,
31.818181818181817, 45.45454545454545]

For the test value  [9, 10, 11]

Most accurate value  9.0


Difference Matrix  [[11, 11], [8, 8], [5, 5]]

Distance matrix  [15.556349186104045,
11.313708498984761, 7.0710678118654755]

Combined matrix  {15.556349186104045: [1, 2, 3],
11.313708498984761: [4, 5, 6], 7.0710678118654755:
[7, 8, 9]}

Sorted matrix  [(7.0710678118654755, [7, 8, 9]),
(11.313708498984761, [4, 5, 6]), (15.556349186104045,
[1, 2, 3])]

Predicted values  [9.0, 7.5, 6.0]

Percentage error  [35.714285714285715,
46.42857142857143, 57.142857142857146]

For the test value  [12, 13, 14]

Most accurate value  9.0
```

```
Difference Matrix  [[14, 14], [11, 11], [8, 8]]

Distance matrix  [19.79898987322333,
15.556349186104045, 11.313708498984761]

Combined matrix  {19.79898987322333: [1, 2, 3],
15.556349186104045: [4, 5, 6], 11.313708498984761:
[7, 8, 9]}

Sorted matrix  [(11.313708498984761, [7, 8, 9]),
(15.556349186104045, [4, 5, 6]), (19.79898987322333,
[1, 2, 3])]

Predicted values  [9.0, 7.5, 6.0]

Percentage error  [47.05882352941177,
55.88235294117647, 64.70588235294117]

For the test value  [15, 16, 17]

Most accurate value  9.0
```