Project Falling Detection: Python + kNN + Colab

Jisen Fang
CS550 - Machine Learning and Business Intelligence
2023 Spring

Table of contents

Manual Process: Determine value of K

- General rule of thumb: K = the closest odd number of the square root of the number of samples
- If no winner, then pick the next odd number greater than K
- The total number of samples is 9, so K=sqrt(9)=3.

Accelerometer Data			Gyroscope Data			Fall (+), Not (-)
x	y	z	x	y	z	+
1	2	3	2	1	3	-
2	1	3	3	1	2	-
1	1	2	3	2	2	-
2	2	3	3	2	1	-
6	5	7	5	6	7	+
5	6	6	6	5	7	+
5	6	7	5	7	6	+
7	6	7	6	5	6	+
7	6	5	5	6	7	??

Manual Process: Find the nearest neighbors

- Calculate the distance between the query-instance and all the training samples. $d_{tq}^{\ 2} = (X_1^{\ t} - X_1^{\ q})^2 + (X_2^{\ t} - X_2^{\ q})^2$

- Find the K-nearest neighbors. Since K=3, the 3 nearest neighbors are [6,5,7,5,6,7], [5,6,6,6,5,7] and [7,6,7,6,5,6] that are highlighted in left, which is + + +, so the result of 765567 is + “Fall”

Accelerometer Data			Gyroscope Data			Distance to each neighbor	Fall (+), Not (-)
x	y	z	x	y	z		
1	2	3	2	1	3	106	-
2	1	3	3	1	2	108	-
1	1	2	3	2	2	115	-
2	2	3	3	2	1	101	-
6	5	7	5	6	7	6	+
5	6	6	6	5	7	7	+
5	6	7	5	7	6	10	+
7	6	7	6	5	6	7	+
7	6	5	5	6	7		+

Python Code - Data Set

```
#Tranining data set and the test data in row 0
dataset = [[7,6,5,5,6,7,1],
    [1,2,3,2,1,3,0],
    [2,1,3,3,1,2,0],
    [1,1,2,3,2,2,0],
    [2,2,3,3,2,1,0],
    [6,5,7,5,6,7,0],
    [5,6,6,6,5,7,1],
    [5,6,7,5,7,6,1],
    [7,6,7,6,5,6,1]]
```

Accelerometer Data			Gyroscope Data			Fall (+), Not (-)
x	y	z	x	y	z	
7	6	5	5	6	7	Expected +
1	2	3	2	1	3	-
2	1	3	3	1	2	-
1	1	2	3	2	2	-
2	2	3	3	2	1	-
6	5	7	5	6	7	+
5	6	6	6	5	7	+
5	6	7	5	7	6	+
7	6	7	6	5	6	+

Python Code - Get euclidean distance

Original code:
https://hc.labnet.sfbu.edu/~henry/sfbu/course/data_science/algorithm/slide/knn_from_scratch.html

```
from math import sqrt

#Get euclidean distance between two vectors
#Euclidean Distance = sqrt(sum i to N (x1_i - x2_i)^2)
def euclidean_distance(row1, row2):
 distance = 0.0
 for i in range(len(row1)-1):
   distance += (row1[i] - row2[i])**2
 return sqrt(distance)
```

x	y	z	x	y	z	distance
1	2	3	2	1	3	106
2	1	3	3	1	2	108
1	1	2	3	2	2	115
2	2	3	3	2	1	101
6	5	7	5	6	7	6
5	6	6	6	5	7	7
5	6	7	5	7	6	10
7	6	7	6	5	6	7

Python Code - Locate three nearest neighbors

```
#Locate three nearest neighbors
def get_neighbors(train, test_row, num_neighbors):
 distances = list()
 for train_row in train:
   dist = euclidean_distance(test_row, train_row)
   distances.append((train_row, dist))
 distances.sort(key=lambda tup: tup[1])
 neighbors = list()
 for i in range(num_neighbors):
   neighbors.append(distances[i][0])
 return neighbors
```

x	y	z	x	y	z	distance
6	5	7	5	6	7	6
5	6	6	6	5	7	7
7	6	7	6	5	6	7

Python Code - Prediction

```
# Make a classification prediction with
neighbors
# - test_row is row 0
# - num_neighbors K is 3
def predict_classification(train, test_row,
num_neighbors):
 neighbors = get_neighbors(train, test_row,
num_neighbors)
 output_values = [row[-1] for row in neighbors]
 prediction = max(set(output_values),
key=output_values.count)
 return prediction
```

```
24 # Make a classification prediction with neighbors
25 # - test_row is row 0
26 # - num_neighbors is 3
27 def predict_classification(train, test_row, num_neighbors):
28   neighbors = get_neighbors(train, test_row, num_neighbors)
29   output_values = [row[-1] for row in neighbors]
30   prediction = max(set(output_values), key=output_values.count)
31   return prediction
32
33 #Tranining data set and the test data in row 0
34 dataset = [[7,6,5,5,6,7,1],
35     [1,2,3,2,1,3,0],
36     [2,1,3,3,1,2,0],
37     [1,1,2,3,2,2,0],
38     [2,2,3,3,2,1,0],
39     [6,5,7,5,6,7,0],
40     [5,6,6,6,5,7,1],
41     [5,6,7,5,7,6,1],
42     [7,6,7,6,5,6,1]]
43
44 prediction = predict_classification(dataset, dataset[0], 3)
45
46 # - Display
47 print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
48
```

```
Expected 1, Got 1.
```

Full Code

Github Link:

https://github.com/JFang2023/JF/tree/main/Machine%20Learning/Supervised%20Learning/Falling%20Prediction%20using%20KNN

Github Code Link :

https://github.com/JFang2023/JF/blob/main/Machine%20Learning/Supervised%20Learning/Falling%20Prediction%20using%20KNN/W3H1_Project_Falling_Detection.ipynb