K-NN(NEAREST NEIGHBORS):A MACHINE LEARNING ALGORITHM WITH IMPLEMENTATION IN PYTHON

KNN is an *non-parametric lazy learning* algorithm. KNN comes in to the picture when most of the practical data does not obey the typical theoretical assumptions made (eg:gaussian mixtures, linearly separable etc).

DONE BY:Sahiti Emani
ID: 19556
DIRECTED BY:Prof.Henry

TABLE OF CONTENTS:

INTRODUCTION:

- KNN is a supervised machine learning algorithm and a non- parametric method.
- K-NN derives to **K Nearest Neighbor** is one of those algorithms which is very simple to understand and works well in practice.
- It is also called as a **lazy algorithm** because it does not use the training data points to do any ***generalization*** or we can say there is ***no explicit training phase***.
- This clarifies that the training phase is pretty fast and Lazy learning means the algorithm takes almost zero time to learn.
- Lack of generalization means that KNN keeps all the training data and all the training data is needed during the testing phase.
- KNN – makes decision based on the entire training data set in the best case a subset of them and more memory is needed to store all the training data.

WHERE TO USE KNN ALGORITHM?

- KNN can be used in both regression and classification predictive problems.
- when it comes to industrial problems, it's mostly used in classification since it fairs across all parameters evaluated when determining the usability of a techniques like:

  Prediction Power

  Calculation Time

  Ease to Interpret the Output

- KNN algorithm fairs across all parameters of considerations. But mostly, it is used due to its ease of interpretation and low calculation time.

HOW DOES K-NN WORK?

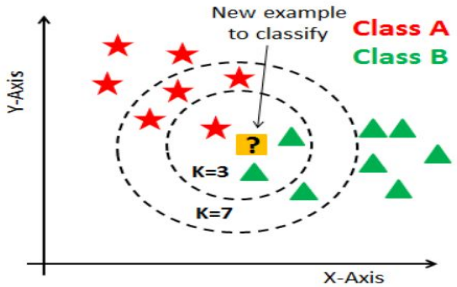
The k-nearest neighbor algorithm stores all the available data and classifies a new data point based on the similarity measure (e.g., distance functions). This means when new data appears.

Suppose there are two classes, i.e., **Class A** and **Class B** and also, we have a new unknown data point "**?**"so,this data point will lie in which of these classes.

K-NN, we can easily identify the class of a particular dataset. The data point is classified by a majority vote of its neighbors, with the data point being assigned to the class most common amongst its K nearest neighbors measured by a distance function.

EXPLAINING THE DIAGRAM:

Here, we can see that if **k = 3**, then based on the distance function used, the nearest three neighbors of the data point is found and based on the majority votes of its neighbors, the data point is classified into a class. In the case of **k = 3,** it's **Class B.** Similarly, when **k = 7,** for the diagram below, based on the majority votes of its neighbors, the data point is classified to **Class A**.



EXAMPLE ON K-NN ALGORITHM CALCULATING THE DISTANCE AND PREDICTING THE RESULT:

USING K-NN ALGORITHM:

- This is the tranining data and the test data:

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	??

CONT'D:

825) Project Falling Detection: Python + KNN + Colab.

→ Using KNN to manually calculate the distance and predict the result.

• This is the training data and the test data:-

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	??

Step-1: Input data Table

• Many devices are technologied in different version.

They are provided with different kind of sensors.

• We have the data sent from Gyroscope and Accelerometer sensors to categorise any motion:-

* 3-(x, y, z) data of Accelerometer.
* 3-(x, y, z) data of Gyroscope.

Step 2:- We need to determine the value of K and find the euclidean distance.

The total no. of observations are 9, but we need to exclude the recently entered data.

So, we have N = 8.

Determine: $K = \sqrt{N}$

$K = \sqrt{N} = \sqrt{8} = 3$

The Euclidean distance formula for '3' different observations:-

$$d_{iq} \equiv (x_1^t - x_1^q)^2 + (x_2^t - x_2^q)^2$$

Step 3:- Determine the distance from Step-2 & use K-value to find it.

Accelerometer			Gyroscope			Fall +/-	Dist. Acc	Dist. Gyr
X	Y	Z	X	Y	Z	+/-		
1	2	3	2	1	3	-	56	50
2	1	3	3	1	2	-	54	54
1	1	2	3	2	2	-	40	45
2	2	3	3	2	1	-	45	56
6	5	7	5	6	7	+	6	0
5	6	6	6	5	7	+	5	2
5	6	7	5	7	6	+	8	2
7	6	7	6	5	6	+	4	3
7	6	5	5	6	7	+		

CONT'D

Explanation on distance calculation:-

For Accelerometer:-

$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$

$d^2 = (7-1)^2 + (6-2)^2 + (5-3)^2$

$d^2 = 6^2 + 4^2 + 2^2$

$d^2 = 36 + 16 + 4$

$d = 56$ ✓

1st observation distance of an Accelerometer.

Step 4: Find the K-nearest neighbours.

Here, we are including the sample of training data as the nearest neighbours if the distance of this training sample (data) to the query instance $\leq$ to the K-th smallest distance.

If the distance of training sample is below the K-th minimum, then we gather the category Y of this nearest neighbor's training samples.

- highest K=8 choose the arbitrary among the 3rd until the 8th NN is very unstable.

The KNN prediction of the query instance is based on simple majority the category of nearest neighbors.

'+' & '-' signs $\Rightarrow + > -$ we predict the query instance as + & vice-versa.

$+ = -$ ÷ choose arbitray to determine + or -.

CODE EXECUTED IN COLAB:

```
# Example of making predictions
from math import sqrt

# calculate the Euclidean distance between two vectors
#     Euclidean Distance = sqrt(sum i to N (x1_i – x2_i)^2)
# Result:
#    0.0
#    1.3290173915275787
#    1.9494646655653247
#    1.5591439385540549
#    0.5356280721938492
#    4.850940186986411
#    2.592833759950511
#    4.214227042632867
#    6.522409988228337
#    4.985585382449795
def euclidean_distance(row1, row2):
  distance = 0.0
  for i in range(len(row1)-1):
    distance += (row1[i] - row2[i])**2
  return sqrt(distance)

# Locate the most similar neighbors
# Result
#    [2.7810836, 2.550537003, 0]
#    [3.06407232, 3.005305973, 0]
#    [1.465489372, 2.362125076, 0]
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

  # Make a classification prediction with neighbors
# - test_row is row 0
# - num_neighbors 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
```

CODE CONT,D:

```
# Test distance function
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],
[6,5,7,5,6,7,1],
[5,6,6,6,5,7,1],
[5,6,7,5,7,6,1],
[7,6,7,6,5,6,1]]
# row 0 (i.e., dataset[0]) is the one to be predicted
prediction = predict_classification(dataset, dataset[8], 3)
# Caluclate euclidean_distance
print("Euclidean distance between two vectors results are shown accordingly:")
for i in range(1,9):
  print(euclidean_distance(dataset[0],dataset[i]))

# - dataset[0][-1] is the last element of row 0 of dataset
# - Display
#    Expected 0, Got 0.
print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
```

```
Euclidean distance between two vectors results are shown accordingly:
10.295630140987
10.392304845413264
10.723805294763608
10.04987562112089
2.449489742783178
2.6457513110645907
3.1622776601683795
2.6457513110645907
Expected 1, Got 1.
```

K-NN ALGORITHM IN PYTHON:

Supervised Learning :K-NN

It is the learning where the value or result that we want to predict is within the training data (labeled data) and the value which is in data that we want to study is known as Target or Dependent Variable or *Response Variable*.

All the other columns in the dataset are known as the Feature or Predictor Variable or Independent Variable.

Supervised Learning is classified into two categories:

1. **Clarification**: Here our target variable consists of the categories.
2. **Regression**: Here our target variable is continuous and we usually try to find out the line of the curve.

There are various ways to get labeled data:

1. Historical labeled Data
2. Experiment to get data: We can perform experiments to generate labeled data like A/B Testing.
3. Crowd-sourcing

Leveraging the distance metric chosen by Python program, K-Nearest Neighbor algorithm discovers the number of k samples that are similar and closest to the data points. A majority vote is performed to determine class label of each new data point.

ADVANTAGES AND DISADVANTAGES OF K-NN:

Some Advantages of KNN:

- Quick calculation time
- Simple algorithm – to interpret
- Versatile – useful for regression and classification
- High accuracy – you do not need to compare with better-supervised learning models
- No assumptions about data – no need to make additional assumptions, tune several parameters, or build a model. This makes it crucial in nonlinear data case.

Some Disadvantages of KNN:

- Accuracy depends on the quality of the data
- With large data, the prediction stage might be slow
- Sensitive to the scale of the data and irrelevant features
- Require high memory – need to store all of the training data
- Given that it stores all of the training, it can be computationally expensive

WHAT ARE IT'S APPLICATIONS?

K-NN is one of the top 10 data mining applications which has numerous uses as discussed below:

- Protein-protein interaction and 3-D structure prediction: A graph based K-NN is used in protein-protein interaction and its used in structure prediction as well.
- Gene expression: A combination of K-NN and SVM techniques are used here.
- Nearest-neighbor content retrieval: Can be used in computer vision in most of the cases let's consider Handwriting detection is rudimentary nearest neighbor problem.
- Detection of outliers: A possible credit card fraud detection.

BIBLIOGRAPHY:

https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

https://towardsdatascience.com/k-nearest-neighbors-knn-algorithm-23832490e3f4

https://medium.com/@gp_pulipaka/coding-k-nearest-neighbors-machine-learning-algorithm-in-python-9e15a61d9b97