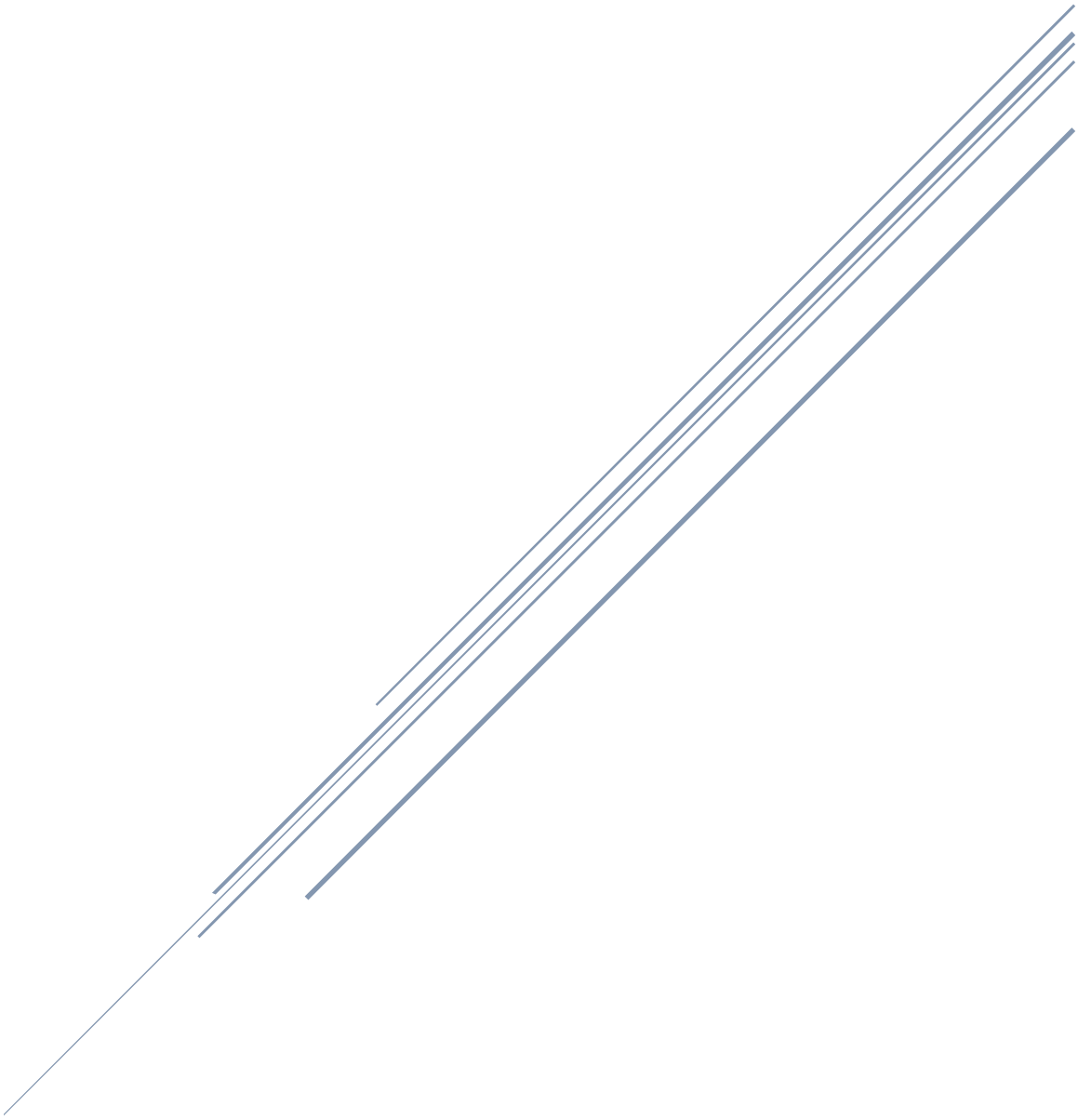
USING THE K-NEAREST NEIGHBOR ALGORITHM

What is k-NN? Where it's Used and How k-NN is Used in Predictive Analytics Classification



Jim Adams, Systems Engineer
April 8, 2019

Contents

Narrative

This paper describes the k-Nearest Neighbor (k-NN) algorithm used in Predictive Analytics (PA). The objective is to simplify the description of the methods used for k-NN and to explain what k-NN is and where it is used. Going further, we will show how to use k-NN with some popular tools such as MS Excel™, Python and R. Additionally, this paper will very briefly describe the four predictive models and show where k-NN fits into that space. The topics are fully referenced and some excellent reading materials are described towards the end.

Predictive Analytics Models and Algorithms

Predictive Analytics (PA) uses many different methods and algorithms. It is still an expanding field of research. Even though this paper is focused on k-NN, it is instructive to at least briefly describe some of the other predictive models that are used. This will ideally put into perspective the relationship among the various models.

Predictive Analytics models are broken down into four models as follows:

1. Predictive models
2. Clustering and Classification models
3. Decision models
4. Association models.

k-NN belongs to the Clustering and Classification model whereas Regression Analysis falls into the Predictive model. Naïve Bayes is a Decision model. Within each of these PA models are algorithms each with its own use cases, challenges and complexities. Let's briefly look at each model.

Clustering and Classification

Clustering and Classification models attempt to determine a data class from a bunch of numerical data points. This class of algorithms attempts to predict intent of contact. There are lots of algorithms in this space such as k-Means, Support Vector Machines and k-Nearest Neighbor. These are very fuzzy algorithms and there are many. We will be digging into k-NN in this paper.

Regression Analysis

One of the most popular prediction algorithms is Linear Regression. Linear Regression is a statistical method that uses a least squares approach by plotting a line between all the data points. Once the line has been established, one can use the equation of the line to predict future values along that line. There is some uncertainly here given there is variability in the data set. We can extend linear regression to nonlinear regression and multiple regression where we use curves rather than straight lines and more than two variables in the training data set. The regression analysis algorithms are used to predict a future numerical value from a set of past numerical data values.

Naïve Bayes

Naïve Bayes is a long-time favorite that uses probability combinations in series to branch off and narrow down to a decision point. Naïve Bayes is a decision algorithm. In statistical notation it looks like this: P (X|Y) which says, "the probability of X happening given that Y has already occurred". There is certainly some mathematics behind this algorithm and that will be covered in a future paper.

Artificial Neural Networks

Artificial Neural Networks (ANN) is a newer set of algorithms in the cluster and classification model. Artificial neural networks (ANN) or connectionist systems are computing systems that are inspired by biological neural networks. ANN itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together to process complex data sets. ANN systems are generally known for their ability to learn. One example is

image recognition. The algorithm might learn to identify images that contain dogs by analyzing images that have been labeled as "dog" or "no dog" and using the results to identify dogs in other images. An entire text book can be written on this topic.

There are dozens of algorithms in this predictive analytics space such as supervised learning and anomaly detection. Let's move on to k-Nearest Neighbors.

Examples of k-NN Usage

K Nearest Neighbor is a machine learning algorithm that is arguably simple to understand and works incredibly well in practice. Also, it is surprisingly versatile with its applications ranging from vision to protein detection to computational geometry to graphs and so on.

First, let's look at a few examples of where we can use k-NN. These examples are by no means the only examples and they are very simple ones, as well. They are used here to describe the general idea behind the algorithm. In the wild, use cases for k-NN are far more complicated and have huge data sets. Let's look at a few examples of where k-NN is used.

Let's say we are given a bunch of objects and each object has its own unique attribute. For instance, let's say we have 5 chairs, 10 beds and 15 tables, and for each we know the length, width and height. Now, if someone gives us a new object and asks us to predict which category that new object belongs, given we only have the length, width and height, we can predict whether the object is a chair, table or bed using the k-Nearest Neighbor algorithm[1].

Another example is we have the weight, height and gender of a lot of people. If we are given a new weight and height combo, we can predict the gender using k-NN. Stretching this example even further to give an example of where this is used would be using age, height, weight and gender to predict heart size. This is an excellent medical example where we can't see the heart physically, but we can predict with some certainty the size of a heart using other attributes and k-NN[2]. Of course, multiple linear regression analysis could be used in this example, as well.

A very popular college learning example is the Iris example where we have a large table of data on the Iris such as petal width and petal length. We sample a new Iris and with the petal width and length, we can infer which type of flower we are likely to have; Iris Virginica, Iris Setosa, etc.

And, arguably, the biggest use of k-NN would be for Recommender Systems. If we know that a user likes a specific item, then we can recommend similar items for them using k-NN[3]. Recommender Systems are used in the recommendation of news or videos for media, product recommendation or personalization in travel, retail, video on demand, and music streaming[4].

The KNN algorithm is one of the most popular algorithms for text categorization or text mining. Another area where k-NN is used is in agriculture for simulating daily precipitation and other weather variables. Other interesting applications of k-NN are forecasting the stock market, predicting the price of a stock based on company performance measures and economic data, predicting currency exchange rates and bankruptcies, understanding and managing

[1] K-nearest Neighbors Algorithm with Examples in R. https://towardsdatascience.com/k-nearest-neighbors-algorithm-with-examples-in-r-simply-explained-knn-1f2c88da405c

[2] Size Matters! Impact of Age, Sex, Height, and Weight on the Normal Heart Size. Circulation: cardiovascular Imaging. 6 Sep 2013.

[3] What are industry applications of the K-nearest neighbor algorithm? https://www.quora.com/What-are-industry-applications-of-the-K-nearest-neighbor-algorithm

[4] Machine Learning for Recommender systems — Part 1. Pavel Kordík. Jun 3, 2018

financial risk, trading futures, credit ratings and money laundering analyses[5]. All very cool and popular areas of prediction.

The Engine Behind k-NN

The mathematics behind the k-NN algorithm is not magical at all. In fact, you probably learned the basic technique in a high school or college math class, specifically a geometry class. The engine behind k-NN is the formula for distance between two points which is:

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

The k-NN approach leverages this distance formula as its basis for finding the closest neighbor in a cartesian coordinate system. This distance between two points comes from Euclidean Geometry and can have many dimensions. We are only going to use two dimensions here to help with our explanation, but many dimensions can be used. In our case we use x1, y1 for the height and weight of a data point and x2, y2 for the height and weight of a test subject.

High-Level Discussion

Let's start with a high-level and generic example to see how this works. In Figure-1 on the right, we have 16 points located on a scatterplot, plus the test subject noted with the red dot. There is a cluster noted as A and a cluster noted as B, separated by the dashed line. The circle covers the three closest points to the test subject, C. The numbers on the points represent the distance from the red dot to the points. Our objective is to compute all those distances from the red dot to all of the points in the scatterplot. Then, we need to find the closest points to the red dot, which are called the Nearest Neighbors. Figure-1 shows the three closest points. This is a very visual example to illustrate what we will be attempting to do with MS Excel™ and then software.

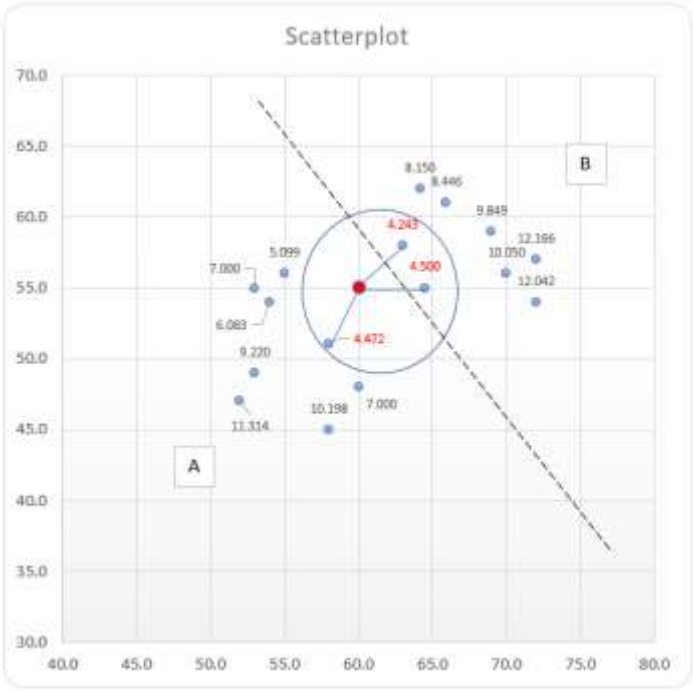


Figure 1 - Generic Example

The intent here is to see what neighborhood our test subject is in. This is never exact but statistically in the "neighborhood".

Keep in mind that this is a simple two-dimensional example. Most real-life examples have thousands of points and many dimensions. But the principle is the same – use Euclidean Geometry to calculate the distances of all points to the test point. That is how we do it.

This specific generic example and associated scatterplot was designed specifically for this demonstration. I scaled the x-axis and the y-axis to both have 40 points. This scaling is critical to normalize the distances on many axes. Let's say we are comparing salaries with credit scores. Salaries are usually in the tens of thousands of dollars whereas credit scores are less than a thousand.

[5] Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background. http://www.ijera.com/papers/Vol3_issue5/DI35605610.pdf

k-NN Predictive Example

Well, let's start with a more specific data set that has some meaning. Each data point is a height, weight and gender combination, in this example. Each height is our x component and each weight represents our y component. Each x, y point is mapped using a 2-dimensional cartesian coordinate system using the height and weight. The height, weight and gender are called *features* and they are all considered to be *classified*. In other words, each height and weight correspond to a gender, the gender being the *class*. Figure-2 shows our training data set and Figure-3 shows a scatterplot of the height and weight for this specific data set.

Height (in)	Weight (lbs)	Gender
67.5	156.5	M
66.1	132.3	F
64.4	121.3	F
66.1	166.8	F
68.3	199.1	M
66.8	132.3	F
68.7	172.0	M
66.5	130.1	F
68.3	187.4	M
65.0	125.7	F
64.2	123.5	F
65.9	155.8	F
68.0	198.4	M
66.7	180.3	M
69.3	200.6	M
66.9	143.3	F

Figure 2 - Data for Height and Weight

Now, let's say we have a new *unclassified* data point that we wish to test. The new unclassified data point is used as a base point to calculate a distance to every other point in the training data. The point with the shortest distance is generally the answer we want – but not always. It depends on how many other data points are close to the point being tested. Our objective is to predict a gender based on the height and weight we are given.

Not to complicate matters, but more than two dimensions can be used here with the same methodology. In fact, in most real-life examples there are many features being considered and an n-dimensional array is used. Even though we can't visualize an n-dimensional array of more than three, we can bang on it with software all day long. The principles used in a two-dimensional array work equally as well on a multi-dimensional array. Two-dimensions are used here for explanatory reasons and they are visually pleasing to look at in a scatterplot.

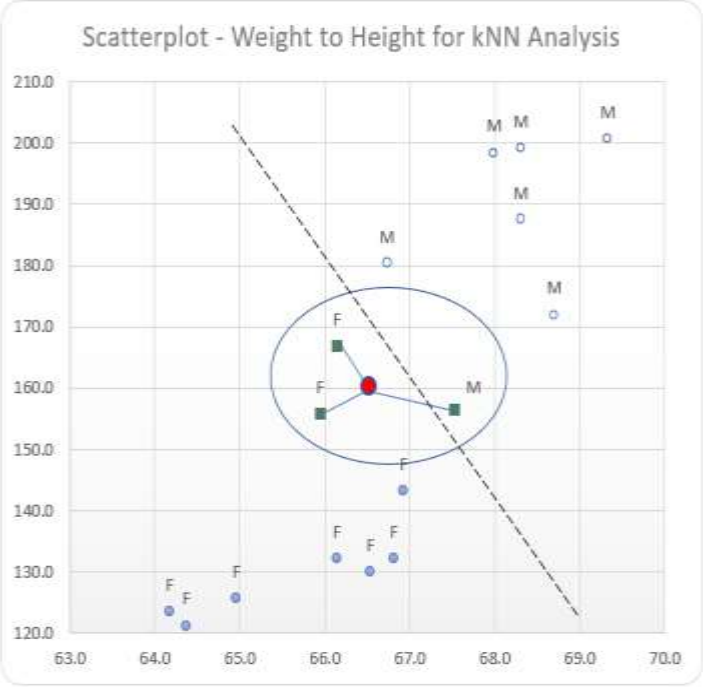


Figure 3 - Scatterplot of Height and Weight

In Figure-3 to the right, the classes are plotted showing M for male and F for female with each cluster separated by a dashed line. The three closest points are noted with tiny green squares.

Please keep in mind the vertical scale on Figure-3 is visually intentionally misleading as there are seven points in the horizontal axis and 90 points on the vertical axis. This gives a very deceiving view of the data as the scatterplot should be stretched way up and down to make the scale look correct but would be ridiculous for a research paper. The horizontal and vertical scale should be much closer for a visually correct graph. I illustrated this intentionally to show why mathematical scaling is necessary.

This is where the mathematics comes in and will give us an exact numerical view of the data rather than a visually pleasing display. There is a Z-score that is used in statistics to normalize the scale. The Z-score uses the mean and standard deviation of the data sets to place the x and y axes. The formula for normalizing the data is,

$$z = \frac{x - \mu}{\sigma}$$

The Z-score is the initial value minus the mean all divided by the standard deviation. We will be using this Z-score in our calculations below.

Going Deeper

Let's dig further into this notion of using k-NN as a class predictor. Just to be clear on this, Regression Analysis uses numbers to predict a number. k-NN Analysis uses numbers to predict a classification such as a gender, race, color, car model, country, state, region, shirt size, etc. – a non-numeric outcome called a classification.

Our objective is to predict the gender of a test subject based upon a height and weight. For instance, say we are given 66.5 inches as a height and 160 pounds as a weight of a test subject. What would be the most probable gender for this height and weight combo? We could try and deduce the answer from the table but what fun is that when we can use mathematics. What?

Earlier in Figure-2, the example that I used had only 16 entries each with a Height, Weight and Gender. It is shown here again as Figure-4 but in an expanded format with normalized values. These normalized values are needed for an accurate prediction.

66.79	157.84						
1.50442	29.06750					Sorted(z-s)	
Height (in)	Weight (lbs)	Distance	Gender	Norm Ht	Norm Wt	Norm Dist	K
66.1	166.8	6.812	F	-0.461	0.308	0.354	1
65.9	155.8	4.243	F	-0.594	-0.070	0.424	2
66.9	143.3	16.705	F	0.071	-0.500	0.633	3
67.5	156.5	3.640	M	0.469	-0.046	0.676	4
66.7	180.3	20.301	M	-0.062	0.773	0.711	5
66.8	132.3	27.702	F	0.004	-0.879	0.974	6
66.1	132.3	27.703	F	-0.461	-0.879	0.989	7
66.5	130.1	29.900	F	-0.195	-0.954	1.029	8
68.7	172.0	12.200	M	1.267	0.487	1.520	9
68.3	187.4	27.459	M	1.001	1.017	1.523	10
65.0	125.7	34.333	F	-1.192	-1.106	1.545	11
68.0	198.4	38.429	M	0.802	1.395	1.655	12
68.3	199.1	39.141	M	1.001	1.420	1.800	13
64.4	121.3	38.757	F	-1.591	-1.257	1.929	14
64.2	123.5	36.572	F	-1.724	-1.181	1.978	15
69.3	200.6	40.696	M	1.666	1.471	2.327	16

Figure 4 - Example Showing Calculated Distances and normalized height and weight

The column headed as "Distance" contains the distance calculated between each Height and Weight point to the test subject height and weight of 66.5 inches high and 160 pounds, respectively. The columns with heading of "Norm Ht" and "Norm Wt" are the normalized values of height and weight using the Z-score mentioned earlier. The Column with the heading "Dist" is the distance between the normalized values and the test subject. This normalized distance is the number we want to use. The table is sorted by the "Dist" column from lowest to highest. As we march downward on the "Dist" column, each one represents another K value. In other words, the first line is k=1, the second line k=2 and so on. The heading marked as "K" shows the k-NN value which is the number of Nearest Neighbors.

Note the "Gender" values for the first three lines. They are all "Female". This means that using k=3 for 3-NN we start out with a good guess that the test subject is female. Cool.

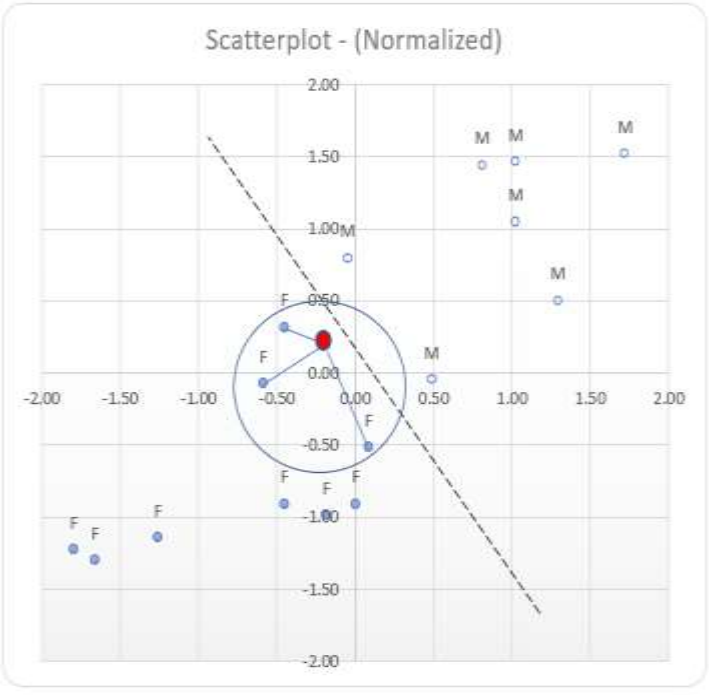


Figure 5 - Normalized Height and Weight Scatterplot

Notice that had we used the unnormalized "Distance" column we would have been close as 2 of the 3 lowest distances are female. The key take-away here is always normalize the data first, before calculating distance.

Sure, sixteen data points is not a big deal. In reality, we could have thousands or even millions of data points.

Results

What is going on here is we plotted all the points and the test subject using a scatterplot. We normalized all the data points with the Z-score. Then we calculated the distance from the test subject to all the points in the data set. Finally, we selected the points with the shortest distance to the test subject. From our k-NN Analysis – in this case our 3-NN Analysis – we deduced that the test subject is a female since 3 of the 3 shortest paths plot to female classifications. We could easily have done k=5 or k=7 but what does that get us?

Had we chosen k=5 we would get 3 females and 2 males or 60% female. Had we chosen k=7, we would get 5 females and 2 males or 71% female. Just to be clear, there is no absolute here. It is an estimation and there are some conditional probability calculations that are beyond the scope of this paper to show how confident we can be on these estimations. But, given that we calculated all the distances and are looking at the top several we can deduce what the outcome is.

Losing weight is a decent example of what I am talking about here. I can walk 4 miles a day and lose 2 pounds a week for a while. I will never get to zero pounds. At some point there is a limit to which the method and math is effective. Another way to look at this is mathematically, if we cut a distance between two points in half, and half again and do that a thousand times we will never ever mathematically get to zero. But we will get very close to where we want to go.

Doing All This with Software

So, let's discuss this a little further. I used MS Excel™ simply because it is easy to use for small and medium data sets. Excel can have many thousands of rows, but it is difficult to scroll and copy formulas into many thousands of cells if you have a large data set. Moreover, what if you have many millions of lines. This is where the super-power of software comes into the picture. For k-NN, you can use literally any language that you are comfortable with. I performed all these k-NN calculations with R and Python, but they could have just as easily been crafted with C++, C#, Java or COBOL for that matter.

	sex	hgt	wgt	nht	nwt	dst
1	F	66.1	166.8	-0.461139935	0.30833400	0.3541477
2	M	67.5	156.5	0.469448763	-0.04601358	0.6755241
3	M	66.7	180.3	-0.062316207	0.77277015	0.7109150
4	F	66.5	130.1	-0.195257450	-0.95424427	1.0286401
5	M	68.7	172.0	1.267096218	0.48722792	1.5195093
6	M	68.3	187.4	1.001213733	1.01702916	1.5231878
7	F	65.0	125.7	-1.192316769	-1.10561606	1.5448480
8	M	68.0	198.4	0.801801869	1.39545862	1.6550934
9	F	66.9	143.3	0.070625035	-0.50012893	0.6330657
10	M	68.3	199.1	1.001213733	1.41954049	1.8002660
11	F	64.4	121.3	-1.591140496	-1.25698784	1.9290080
12	F	64.2	123.5	-1.724081739	-1.18130195	1.9784035
13	M	69.3	200.6	1.665919945	1.47114451	2.3269912
14	F	65.9	155.8	-0.594081177	-0.07009546	0.4241911
15	F	66.8	132.3	0.004154414	-0.87855838	0.9735948
16	F	66.1	132.3	-0.461139935	-0.87855838	0.9893509

Figure 6 – Unsorted Data

Using R

See Appendix-A for the R source code. Figure-6 is the unsorted output from my R code after reading it into the program. It shows the normalized values of height and weight and the calculated distance to the test subject.

Figure-7 is the sorted version of Figure-6, sorted on the column named dst.

The output in Figure-6 and Figure-7 is a little ragged since I did not spend any time making it look good. The objective is to find the shortest 3 or 4 distances and we have that as the top 3 rows in Figure-7, wrapped by the blue box. How cool is that. It matches the sorted data depicted in Figure-4.

	sex	hgt	wgt	nht	nwt	dst
1	F	66.1	166.8	-0.461139935	0.30833400	0.3541477
14	F	65.9	155.8	-0.594081177	-0.07009546	0.4241911
9	F	66.9	143.3	0.070625035	-0.50012893	0.6330657
2	M	67.5	156.5	0.469448763	-0.04601358	0.6755241
3	M	66.7	180.3	-0.062316207	0.77277015	0.7109150
15	F	66.8	132.3	0.004154414	-0.87855838	0.9735948
16	F	66.1	132.3	-0.461139935	-0.87855838	0.9893509
4	F	66.5	130.1	-0.195257450	-0.95424427	1.0286401
5	M	68.7	172.0	1.267096218	0.48722792	1.5195093
6	M	68.3	187.4	1.001213733	1.01702916	1.5231878
7	F	65.0	125.7	-1.192316769	-1.10561606	1.5448480
8	M	68.0	198.4	0.801801869	1.39545862	1.6550934
10	M	68.3	199.1	1.001213733	1.41954049	1.8002660
11	F	64.4	121.3	-1.591140496	-1.25698784	1.9290080
12	F	64.2	123.5	-1.724081739	-1.18130195	1.9784035
13	M	69.3	200.6	1.665919945	1.47114451	2.3269912

Figure 7 - Sorted by Distance

Using Python

Not to be throwing shade on R, here is the same example above worked out using Python. Appendix-B has the Python code I created for this example. There is no good answer for which language to use. It comes down to what language you are familiar with. If the language you like can read files, do some basic math, loop and print, then it will work. I chose Python since it is a very popular language today, but I have used C# and Java for this several times in the past. There is nothing magical about the programming language except perhaps the math functions. I had to create the function for the distance between two points myself in both R and Python. It is shown in the source code.

Again, the output in Figure-8 is not as sexy as an Excel spreadsheet but the general idea is that software can be used. Software is recommended if you are processing millions of lines of data.

```
k-NN Python Project
157.8375
29.067504766204706
66.79374999999999
1.5044240315372068
     Ht      Wgt   NormHt  NormWgt G     Dist
-------------------------------------------------
   66.1   166.8  -0.4611   0.3083 F   0.3546
   67.5   156.5   0.4694  -0.0460 M   0.6752
   66.7   180.3  -0.0623   0.7728 M   0.7113
   66.5   130.1  -0.1953  -0.9542 F   1.0282
   68.7   172.0   1.2671   0.4872 M   1.5194
   68.3   187.4   1.0012   1.0170 M   1.5232
   65.0   125.7  -1.1923  -1.1056 F   1.5447
   68.0   198.4   0.8018   1.3955 M   1.6553
   66.9   143.3   0.0706  -0.5001 F   0.6326
   68.3   199.1   1.0012   1.4195 M   1.8004
   64.4   121.3  -1.5911  -1.2570 F   1.9289
   64.2   123.5  -1.7241  -1.1813 F   1.9784
   69.3   200.6   1.6659   1.4711 M   2.3270
   65.9   155.8  -0.5941  -0.0701 F   0.4243
   66.8   132.3   0.0042  -0.8786 F   0.9732
   66.1   132.3  -0.4611  -0.8786 F   0.9890
```

Figure 8 - Python Output

Confidence Intervals

If you have any statistical experience at all, you are probably asking how confident are we of all this k-NN predictive analysis? You would be correct in asking this.

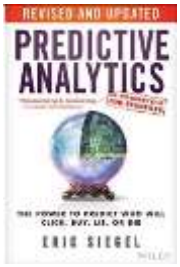
The problem is calculating a confidence interval or better yet, a predictive interval, requires some interesting mathematics that is challenging to describe and understand. There are several ways to get a confidence interval and the easiest I discovered is to use a weighted distance from the point being tested to all the other points or optionally a density calculation. And, there others. I did not include these calculations here simply because the paper would be enormous in size and would take away from the main theme of implementing k-NN. If you choose to use k-NN for your project, you should investigate how to calculate a confidence value.

What Does All This Mean

To summarize everything in this paper, k-NN can be used as a class predictor for machine learning applications. It is part of the predictive analytics classification model. Given a bunch of numerical data we can predict a class value such as a gender, race, color, car model, country, state, region, shirt size, etc. We can have many features with each feature residing in a dimension in an array. These are known collectively as the feature space. K-NN results are not absolute but can get us close to a cluster of points using Euclidean geometry.

K-NN is, arguably, one of the easiest algorithms to learn and implement. It can be implemented with software or a spreadsheet. The results are intuitive. There are hundreds of web sites devoted to k-NN and related machine learning methods. I sincerely hope this paper was helpful.

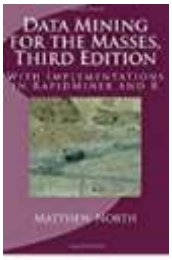
Further Reading and Research



Well, I want to offer up some excellent books on Predictive Analytics. There are hundreds of web sites devoted to the subjects and the ones that I used are listed in the reference section of this paper, but I want to highlight three books that I have studied extensively and give a little push for each.

Predictive Analytics. The Power to Predict Who Will Click, Buy, Lie or Die. Eric Siegel. (2016). This is an excellent book on what PA is and where it is used. It literally has 182 examples of where PA is used. It is an easy read with very little mathematics involved. It is more story-based and covers machine learning, wisdom of crowds and even IBM Watson. It is one of the best books I have ever read.

Predictive Analytics for Dummies. Anasse Bari, Mohamed Chaouchi and Tommy Jung. (2017). Yeah, I know, another Dummies book. However, this one is well worth the effort. This book covers all things under the umbrella of PA. It covers classification models and predictive models including dozens of algorithms. It goes into the mathematics of the algorithms and models and even how to approach and run a data analytics project. I was inspired by this book and used it a lot for verifying many of the areas of this paper.



Predictive Analytics and Data Mining. Vijay Kotu and Bala Deshpande. (2015). This text book is used in the course *Data Mining and Predictive Analytics* offered through ASU here in Phoenix. This is an official graduate level text book that goes deep and wide in to all things PA. It is very mathematical in nature and covers topics such as regression, clustering, classification, anomaly detection and data mining. The chapter on K-Nearest Neighbors is incredible and deep and is the basis for this paper. It is my "go to" book for predictive analytics.

References Used in this Paper

Kshitiz Sirohi. *K-nearest Neighbors Algorithm with Examples in R*. 12/30/2018. https://towardsdatascience.com/k-nearest-neighbors-algorithm-with-examples-in-r-simply-explained-knn-1f2c88da405c

Chris McCormick. *What are industry applications of the K-nearest neighbor algorithm*? 10/03/2016. https://www.quora.com/What-are-industry-applications-of-the-K-nearest-neighbor-algorithm

Sadegh Bafandeh Imandoust and Mohammad Bolandraftar. *Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background*. Journal of Engineering Research and Applications. Vol. 3, Issue 5, Sep-Oct 2013, pp.605-610. http://www.ijera.com/papers/Vol3_issue5/DI35605610.pdf

Size Matters! Impact of Age, Sex, Height, and Weight on the Normal Heart Size. Circulation: Cardiovascular Imaging. AHA/ASA Journals. 6 Sep 2013. https://www.ahajournals.org/

Pavel Kordík. *Machine Learning for Recommender Systems — Part 1*. Jun 3, 2018

Eric Siegal. *Predictive Analytics. The Power to Predict Who Will Click, Buy, Lie or Die*. (2016). Wiley.

Anasse Bari, Mohamed Chaouchi and Tommy Jung. *Predictive Analytics for Dummies*. (2017).

Vijay Kotu and Bala Deshpande. *Predictive Analytics and Data Mining*. (2015).

Appendix A - R Source Code for k-NN Example

```
# ;-----------------------------------------------------
# ; R code
# ; Written by Jim Adams on 04/09/2019
# ; k-NN Demonstration
# ;-----------------------------------------------------
require(stats)
require(graphics)

# ;-----------------------------------------------------
# ; Read in the training data set
# ;-----------------------------------------------------
kNNdata <- "C:/Users/jaadams/Desktop/Data/_Tools/R/Data/kNN2_MKS.txt"
tryCatch (
  stats <- read.csv(kNNdata, header=TRUE),
  error = function(e)
  {
    message("Whoa Nelly:", e$message)
    stop()
  }
)

# ;-----------------------------------------------------
# ; Create a data frame with all the data
# ; This is a 2-dimensional matrix of all the columns
# ;-----------------------------------------------------
sx    <- stats$gender
ht    <- stats$height
wt    <- stats$weight

htwt <- data.frame(
  sex = sx,
  hgt = ht,
  wgt = wt,
  nht = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
  nwt = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
  dst = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
)
print(sum(htwt$wgt))
print(sum(htwt$hgt))

# ;-----------------------------------------------------
# ; Calculate mean and stdev of the weights and heights
# ;-----------------------------------------------------
meanht = mean(htwt$hgt)
meanwt = mean(htwt$wgt)
sdht = sd(htwt$hgt)
sdwt = sd(htwt$wgt)


# ;-----------------------------------------------------
# ; Set up the test subject weight and height
# ; Normalize the test subject weight and height
# ; Normalized Z = (actual - mean)/ standard deviation
# ;-----------------------------------------------------
testht = 66.5
testwt = 160
zwt = (testwt - meanwt)/sdwt
zht = (testht - meanht)/sdht

print(sdht)
print(sdwt)
```

```
# ;---------------------------------------------------
# ; Loop through the matrix
# ; Normalize the weights and heights
# ; Calculate the distance to the test subject
# ;---------------------------------------------------
for(i in 1:16)
{
  htwt$nht[i] = ((htwt$hgt[i]-meanht)/sdht)
  htwt$nwt[i] = ((htwt$wgt[i]-meanwt)/sdwt)
  htwt$dst[i] = sqrt((zht-htwt$nht[i])^2+(zwt-htwt$nwt[i])^2)
}

print(htwt)
sorted = htwt[order(htwt$dst),]
print(sorted)

plot(nwt, nht, main="Scatterplot of Normalized Height and Weight", xlab="Weight", ylab="Height",
pch=19, labels=row.names(s))
#abline(lm(nht~nwt), col="red") # regression line (y~x)
#lines(lowess(nwt,nht), col="blue") # lowess line (x,y)
```

Appendix-B – Python Code for k-NN Example

```
# ;------------------------------------------------
# ; Python Code
# ; Written by Jim Adams on 04/09/2019
# ; k-NN Demonstration
# ;------------------------------------------------
import numpy as np
import math
import statistics


# ;------------------------------------------------
# ; Python function to calc distance
# ; between two points
# ;------------------------------------------------
def distance(x1, y1, x2, y2):
     dist = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
     return dist


# ;------------------------------------------------
# ; Start of Main logic area (Fall through)
# ;------------------------------------------------
print("k-NN Python Project")


# ;------------------------------------------------
# ; Set up the lists
# ;------------------------------------------------
wt = []     # original weight array
ht = []     # original height array
gn = []     # original gender array
nwt = []    # normalized weight array
nht = []    # normalized height array
dst = []    # distance array


# ;------------------------------------------------
# ; Initialize the lists
# ;------------------------------------------------
for i in range(16):
    nwt.append(0.0)
    nht.append(0.0)
    wt.append(0.0)
    ht.append(0.0)
    dst.append(0.0)
    gn.append(" ")


# ;------------------------------------------------
# ; Set up the Test Subject
# ;------------------------------------------------
testht = -0.195
testwt = 0.074


# ;------------------------------------------------
# ; Read in the training data set, populate arrays
# ; Skip the header line
# ;------------------------------------------------
file = open("C:/Users/jaadams/Desktop/Data/_Tools/R/Data/kNN2_MKS.txt", "r")
x = 0
i = 0
```

```
for line in file:
    if x > 0:
        knn = line.split(',')       # split the input string
        ht[i] = float(knn[0])       # append the height array
        wt[i] = float(knn[1])       # append the weight array
        gn[i] = knn[2][0]           # append the gender array
        i += 1                      # increment the line counter

    x += 1
file.close()


# ;---------------------------------------------------
# ; Calculate means and standard deviations of initial data
# ;---------------------------------------------------
wtmean  = np.mean(wt)
htmean  = np.mean(ht)
wtstd   = statistics.stdev(wt)
htstd   = statistics.stdev(ht)

print(wtmean)
print(wtstd)
print(htmean)
print(htstd)


# ;---------------------------------------------------
# ; Loop and calculate and print everything
# ;---------------------------------------------------
print("%8s %8s %8s %8s %1s %8s" %("Ht","Wgt","NormHt","NormWgt","G","Dist"))
print("-----------------------------------------------")
for i in range(len(ht)):
    nwt[i]  = (wt[i]-wtmean)/wtstd
    nht[i]  = (ht[i]-htmean)/htstd
    dst[i]  = distance(testht, testwt, nht[i], nwt[i])
    print("%8.1f %8.1f %8.4f %8.4f %c %8.4f" %( ht[i], wt[i], nht[i], nwt[i], gn[i], dst[i]))
```