Introduction to Machine Learning Using Python

K-Nearest Neighbors:

K-Nearest Neighbors

KNN has been around for a long time and has been very well studied. KNN works by seeking to minimize the distance between the test and training observations, so as to achieve a high classification accuracy.

The KNN algorithm is robust and versatile classifier that is often used as a benchmark for more complex classifiers such as Artificial Neural Networks (ANN) and Support Vector Machines

Despite Its simplicity, KNN can outperform more powerful classifiers and is used in a variety of applications.

K-Nearest Neighbors

K-Nearest Neighbors (KNN) is one of the simplest Machine Learning algorithm based on Supervised technique.

KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

KNN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can easily be classified into a well suite category by using KNN algorithm

The simple version of the KNN algorithm is to predict the target label by finding the nearest neighbor class. The closest class will be identified using the distance measures like Euclidean distance

KNN algorithm can be used for regression as well as classification but mostly it is used for the classification problems.

K-Nearest Neighbors

The following two properties would define KNN well –

Lazy learning algorithm – KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification. While this does mean that we can immediately begin classifying once we have our data, there are some inherent problems with this type of algorithm. We must be able to keep the entire training set in memory unless we apply some type of reduction to the data-set, and performing classifications can be computationally expensive as the algorithm parse through all data points for each classification. For these reasons, KNN tends to work best on smaller data-sets that do not have many features.

Non-parametric learning algorithm – KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

K-Nearest Neighbors

KNN is non-parametric, instance-based and used in a supervised learning setting

It is worth noting that the minimal training phase of KNN comes both at a memory cost, since we must store a potentially huge data set, as well as a computational cost during test time since classifying a given observation requires a run down of the whole data set. Practically speaking, this is undesirable since we want fast responses.

Minimal training but expensive testing

K-Nearest Neighbors

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

K-Nearest Neighbors

Let’s assume a money lending company “XYZ”. Money lending XYZ company is interested in making the money lending system comfortable & safe for lenders as well as for borrowers. The company holds a database of customer’s details.

Using customer’s detailed information from the database, it will calculate a credit score(discrete value) for each customer. The calculated credit score helps the company and lenders to understand the credibility of a customer clearly. So they can simply take a decision whether they should lend money to a particular customer or not.

K-Nearest Neighbors

The customer's details could be:

- Educational background details.
  - Highest Degree
  - Whether to take the education loan or not.
  - Cleared education loan dues.

- Employment details.
  - Salary.
  - Year of experience.
  - Average job change duration.

K-Nearest Neighbors

The company(XYZ) use's these kinds of details to calculate credit score of a customer. The process of calculating the credit score from the customer's details is expensive. To reduce the cost of predicting credit score, they realized that the customers with similar background details are getting a similar credit score.

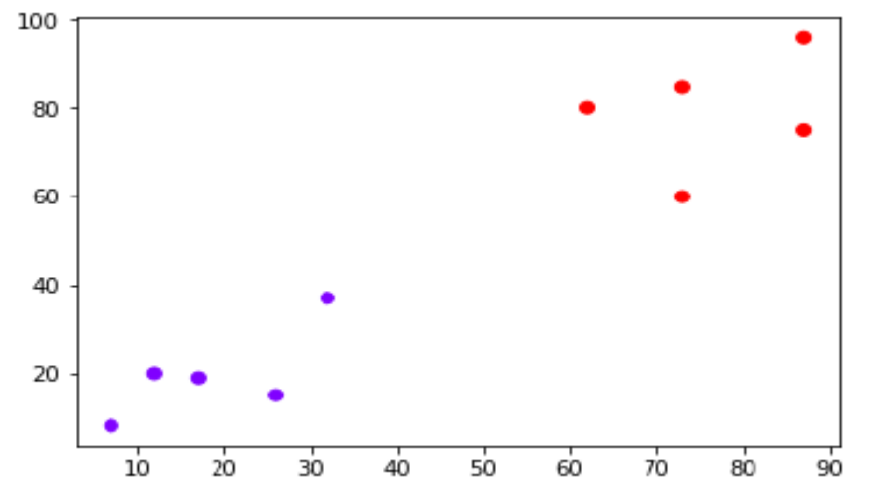
So, they decided to use already available data of customers and predict the credit score using it by comparing it with similar data. These kinds of problems are handled by the K-nearest neighbor classifier for finding the similar kind of customers.

K-Nearest Neighbors

Example

The following is an example to understand the concept of K and working of KNN algorithm −

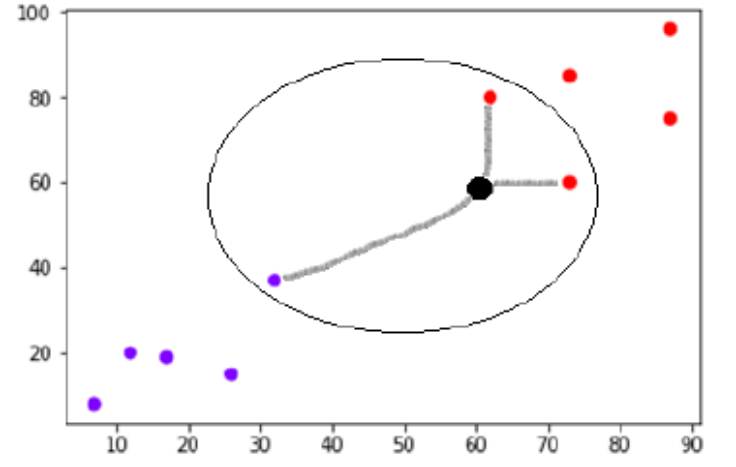
Suppose we have a dataset which can be plotted as follows −



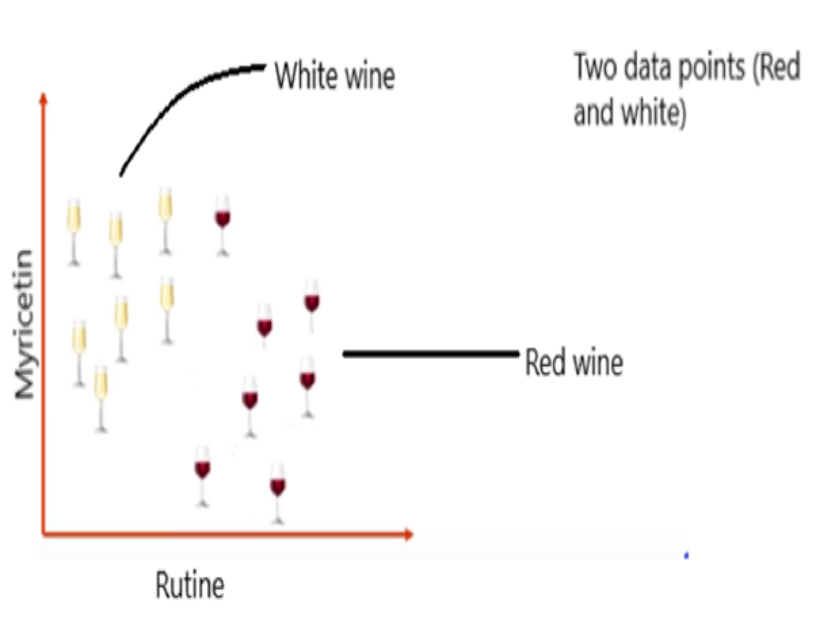
K-Nearest Neighbors

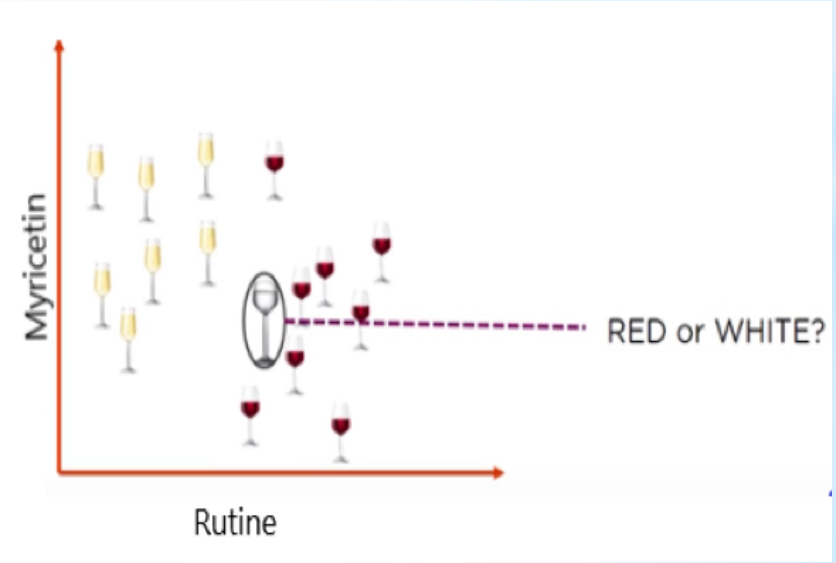
Now, we need to classify new data point with black dot, into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −

We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.



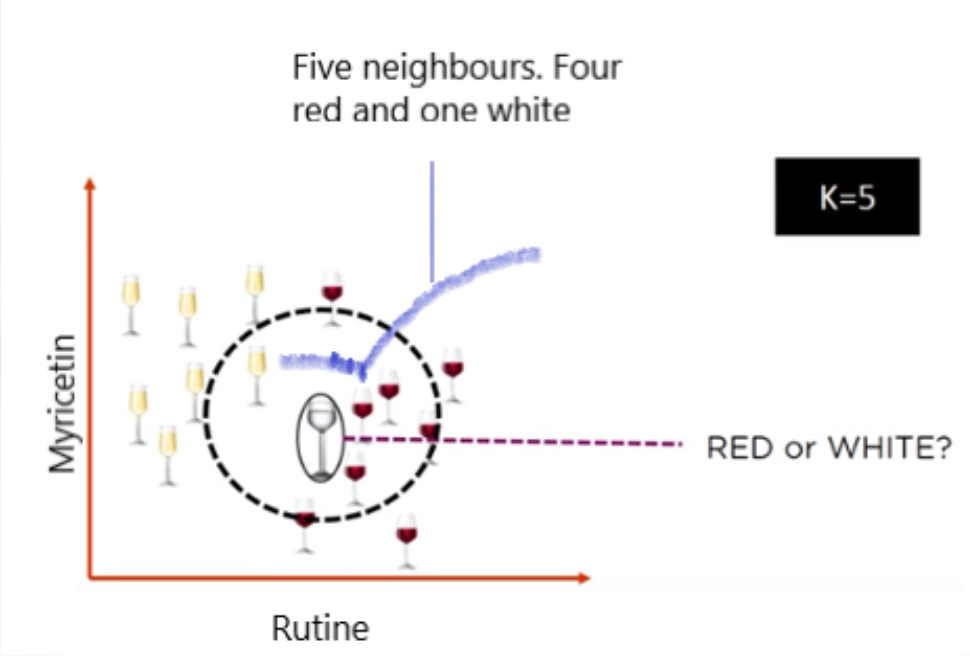
K-Nearest Neighbors





K-Nearest Neighbors

So, we need to find out what the neighbors are in this case. Let's say k = 5 and the new data point is classified by the majority of votes from its five neighbors and the new point would be classified as red since four out of five neighbors are red.



K-Nearest Neighbors

K-nearest neighbor (KNN) algorithm pseudocode:
Let (Xi, Ci) where i = 1, 2……., n be data points. Xi denotes feature values & Ci denotes labels for Xi for each i.
Assuming the number of classes as 'c'
Ci ∈ {1, 2, 3, ……, c} for all values of i

Let x be a point for which label is not known, and we would like to find the label class using k-nearest neighbor algorithms.

KNN Algorithm Pseudocode:
1. Calculate "d(x, xi)" i =1, 2, ….., n; where d denotes the Euclidean distance between the points.
2. Arrange the calculated n Euclidean distances in non-decreasing order.
3. Let k be a +ve integer, take the first k distances from this sorted list.
4. Find those k-points corresponding to these k-distances.
5. Let ki denotes the number of points belonging to the ith class among k points i.e. k ≥ 0
6. If ki >kj ∀ i ≠ j then put x in class i.

K-Nearest Neighbors

- Given training data D = $\{(x_1,y_1), \ldots(x_n,y_n)\}$ and a test point

- **Prediction Rule**: look at the K most similar Training examples
  - **For classification**: assign the majority class label (majority voting)
  - **For regression**: assign the average response
- The algorithm requires
  - **Parameter K:** number of nearest neighbors
  - **Distance Function**: To compute the similarities between example

Computing the Distance

- The K-NN algorithm requires computing distances of the test example from each of the training examples
- Several ways to compute distances
- The choice depends on the type of the features in the data
- **Euclidean distance**: Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (y)
- **Manhattan Distance**: Calculate the distance between real vectors using the sum of their absolute difference.

Computing the Distance

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

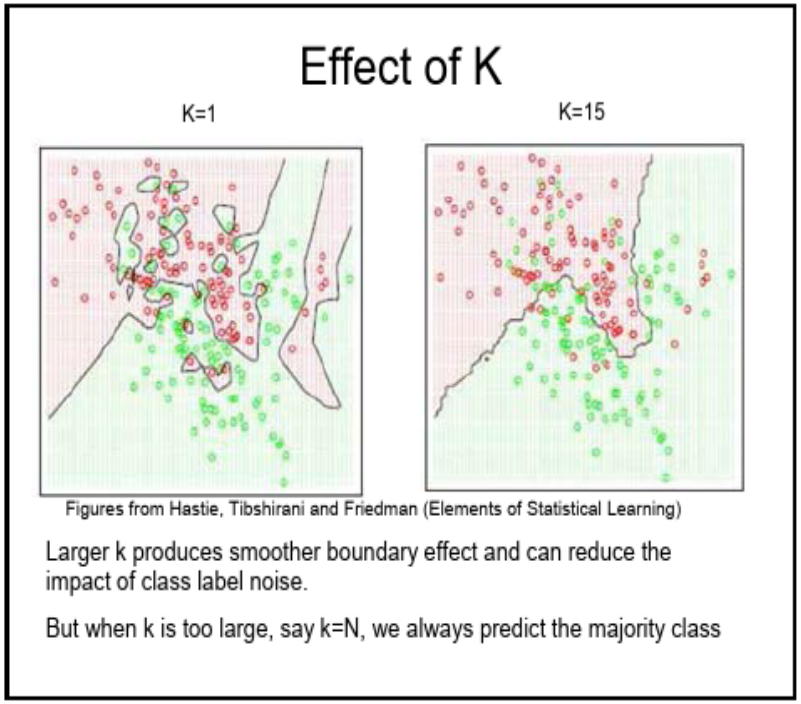
$$\sum_{i=1}^{k}|x_i - y_i|$$

K-Nearest Neighbors

How to choose the value of K?
Selecting the value of K is the most critical problem. Choosing the right value of K is called parameter tuning an dis important for better accuracy. Find the right value of K is not an easy task.

- K too small: we will model the noise, creates a small regions for each class and may lead to non-smooth decision boundaries and overfit

- **K too large**: neighbors include too many points from other classes, creates fewer larger regions and usually leads to smoother decision boundaries (caution: too smooth decision boundary can underfit)



K-Nearest Neighbors

Often data dependent and heuristic based Or using cross-validation (using some held-out data) In general, a K too small or too big is bad!

To optimize the results, we can use Cross Validation. Using the cross-validation technique, we can test KNN algorithm with different values of K. The model which gives good accuracy can be considered to be an optimal choice.
A simple approach to select k is k = n^(1/2). Try and keep the value of K odd in order to avoid confusion between two classes of data.

It depends on individual cases, at times best process is to run through each possible value of k and test our result.

Example

Movie Title	# of kicks	# of kisses	Type of movie
California man	3	104	Romance
He's nt really into Dudes	2	100	Romance
Beautiful woman	1	81	Romance
Kevin Longblade	101	10	Action
Robo Slayer 3000	99	5	Action
Amped II	98	2	Action
?	18	90	Unknown

Example

Movie title	Distance to movie "?"
He's not really into Dudes	18.7
Beautiful woman	19.2
California Man	20.5
Kevin Longblade	115.3
Robo Slayer 3000	117.4
Amped II	118.9

Once we have distances, we need to find the k-nearest movies by sorting the distances. Let's assume k =3. Then the three closest movies are "*He's Not Really into Dudes*, *Beautiful Woman*, and *California Man*". The KNN algorithm says to take the majority votes from these three movies to determine the class of the new movie. Because all three movies are romances, we forecast that the new movie is a romance movie.

Properties K-Nearest Neighbors

What is nice:

- Simple and powerful. No need for tuning complex parameters to build a model as KNN has no model other than storing the entire data
- Easy to implement and has gained good popularity
- Makes no assumption of the underlying data
- No training involved ("lazy"). New training examples can be added easily
- Asymptotically consistent
    - With infinite training data and large enough K, KNN approaches the best possible classifier
- KNN makes predictions using the training dataset directly
- Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) an d summarizing the output for those K instances. For regression this might be the mean output variable, in classification this might be the mode (or most common) class value

Properties of K-Nearest Neighbors

What's not so nice:

- Expensive and slow: O(md) , m = # examples, d = # dimensions
  - To determine the nearest neighbor of a new point x, must compute the distance to all m training examples. Runtime performance is slow, but can be improved
    - Pre-sort training examples into fast data structures
    - Compute only an approximate distance
    - Remove redundant data
- Sensitive to noisy features
- May perform badly in high dimensions (curse of dimensionality)
    - In high dimensions, distance notion can be counter intuitive

K-Nearest Neighbors

- KNN stores the entire training dataset which it uses as its representation.
- KNN does not learn any model.
- KNN makes predictions just-in-time by calculating the similarity between an input sample and each training instance.
- There are many distance measures to choose from to match the structure of your input data.
- That it is a good idea to rescale your data, such as using normalization, when using KNN.

Prepare data for KNN

Rescale Data: KNN performs much better if all of the data has the same scale. Normalizing your data to the range [0, 1] is a good idea. It may also be a good idea to standardize your data if it has a Gaussian distribution.

Address Missing Data: Missing data will mean that the distance between samples can not be calculated. These samples could be excluded or the missing values could be imputed.

Lower Dimensionality: KNN is suited for lower dimensional data. You can try it on high dimensional data (hundreds or thousands of input variables) but be aware that it may not perform as well as other techniques. KNN can benefit from feature selection that reduces the dimensionality of the input feature space.

Summary

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point.

Application

- Handwritten character classification using nearest neighbor in large databases.
- Fast content-based image retrieval based on equal-average K-nearest-neighbor search schemes
- Use of K-Nearest Neighbor classifier for intrusion detection
- Fault Detection Using the k-Nearest Neighbor Rule for Semiconductor Manufacturing Processes
- Computer vision applications, including optical character recognition and facial recognition in both still images and video
- Predicting whether a person enjoys a movie which he/she has been recommended (as in the Netflix challenge)
- Identifying patterns in genetic data, for use in detecting specific proteins or diseases

Conclusion

KNN is a simple yet powerful classification algorithm. It requires no training for making predictions, which is typically one of the most difficult parts of a machine learning algorithm. The KNN algorithm have been widely used to find document similarity and pattern recognition. It has also been employed for developing recommender systems and for dimensionality reduction and pre-processing steps for computer vision, particularly face recognition tasks.

The test problem we will be using in this tutorial is iris classification.

The problem is comprised of 150 observations of iris flowers from three different species. There are 4 measurements of given flowers: sepal length, sepal width, petal length and petal width, all in the same unit of centimeters. The predicted attribute is the species, which is one of setosa, versicolor or virginica.

It is a standard dataset where the species is known for all instances. As such we can split the data into training and test datasets and use the results to evaluate our algorithm implementation. Good classification accuracy on this problem is above 90%.