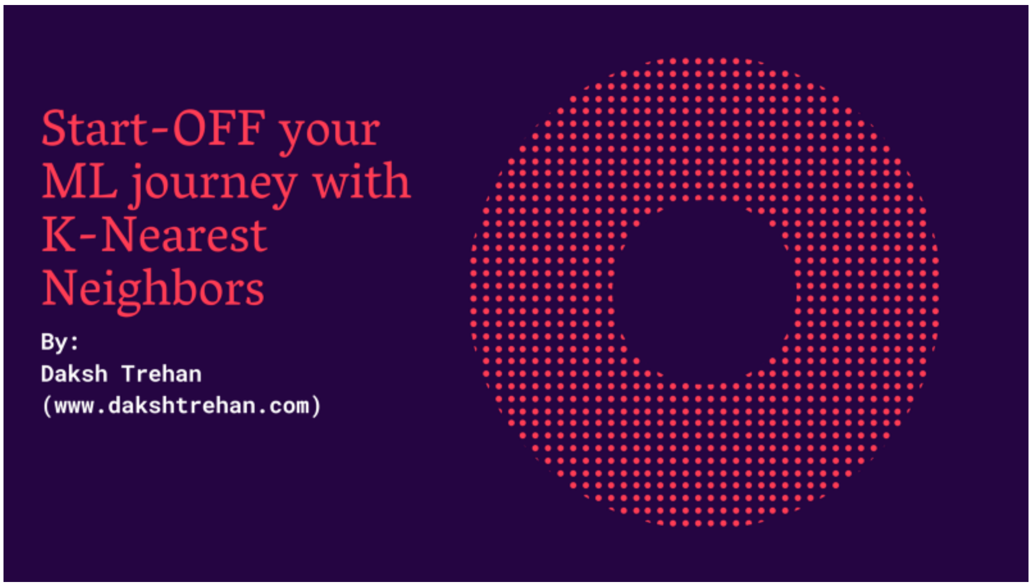
Machine Learning

Start-off your ML journey with K-Nearest Neighbors!

Detailed theoretical explanation and scikit-learn implementation with an example!



K-Nearest Neighbors(KNN) is one of the elementary methods in machine learning and is a great way to introduce yourself in the world of Machine Learning.

Table of Content:

Introduction to K-NN

KNN is the **supervised learning algorithm** that relies upon input data for its training to produce pertinent output when it is fed with new unlabeled data.

For an instance, assume yourself to be a guardian of a 5-year-old child, and you want him to grasp what a "*dog*" looks like, you will show him several pictures of *dog* and rest could be a picture of any other animal.

Whenever you'll encounter a "*dog*" you'll tell your child *"it's a dog"* and whenever other animals occur you'll tell your child *"no it's not a dog"*. Iterating this process several times will help your child to understand what exactly a dog is and he can distinguish and identify dogs from the rest of the animals. This is called a ***supervised learning algorithm.***

“it’s a dog”

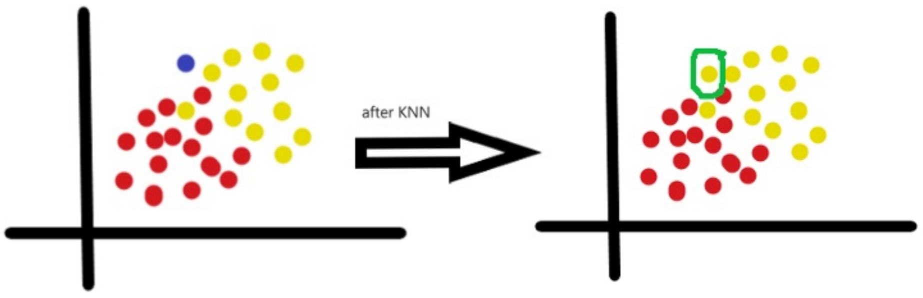
Supervised machine learning algorithms are used to solve *classification* or *regression* problems. And generally, an unsupervised learning algorithm is used to cater *to clustering* problems.

K-NN is a **non-parametric**, which corresponds that it doesn’t make any assumption on underlying data. It is also known as a “**lazy learning algorithm**” as it doesn’t learn from input data immediately but rather the classification happens at query time. It saves all values from the dataset, making its **training blazing fast** and is unlike its contemporary SVM where **we can discard non-support vectors**. Its lazy learning trait leads to **enormous space and time complexity**. But still, the algorithm works fine for **small datasets.**

KNN is the **least accurate algorithm,** whenever we develop a new algorithm we cross-check its accuracy with that of *KNN*. Its accuracy is used as a **minimum threshold** for a new algorithm, and this is both advantage and disadvantage of *KNN*.

The output expected from K-NN is **class membership**.

The feature that makes it exquisite from other algorithms is it’s a dual nature i.e. it can be used both for **classification** as well as **regression** problems.



● New point that is needed to be labelled

● Cluster 1

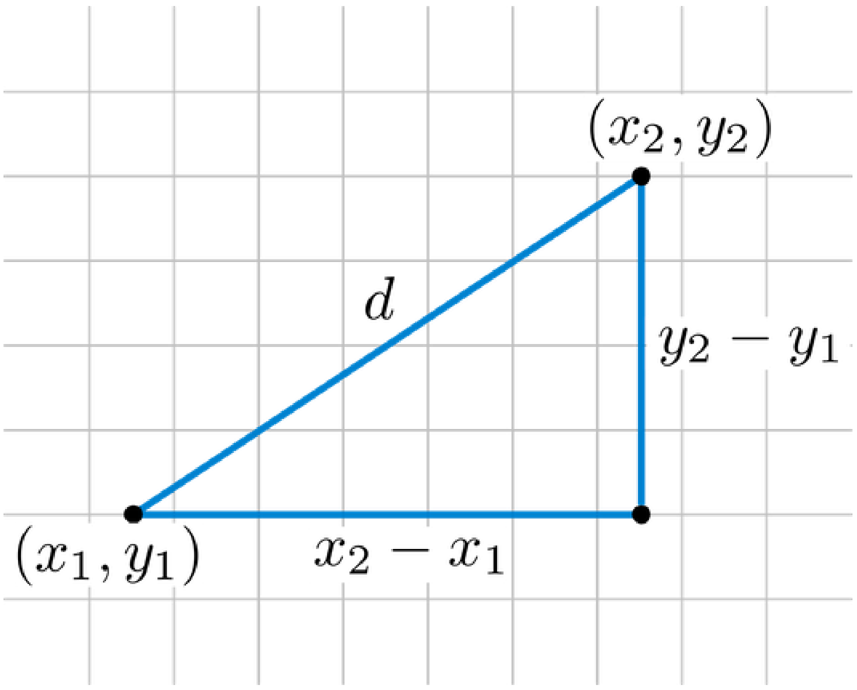
● Cluster 2

KNN works on the **principle of majority votes**, that is, an object is classified by majority votes of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

The votes are decided using **distance metrics** and it is assumed data is in **metric space.**

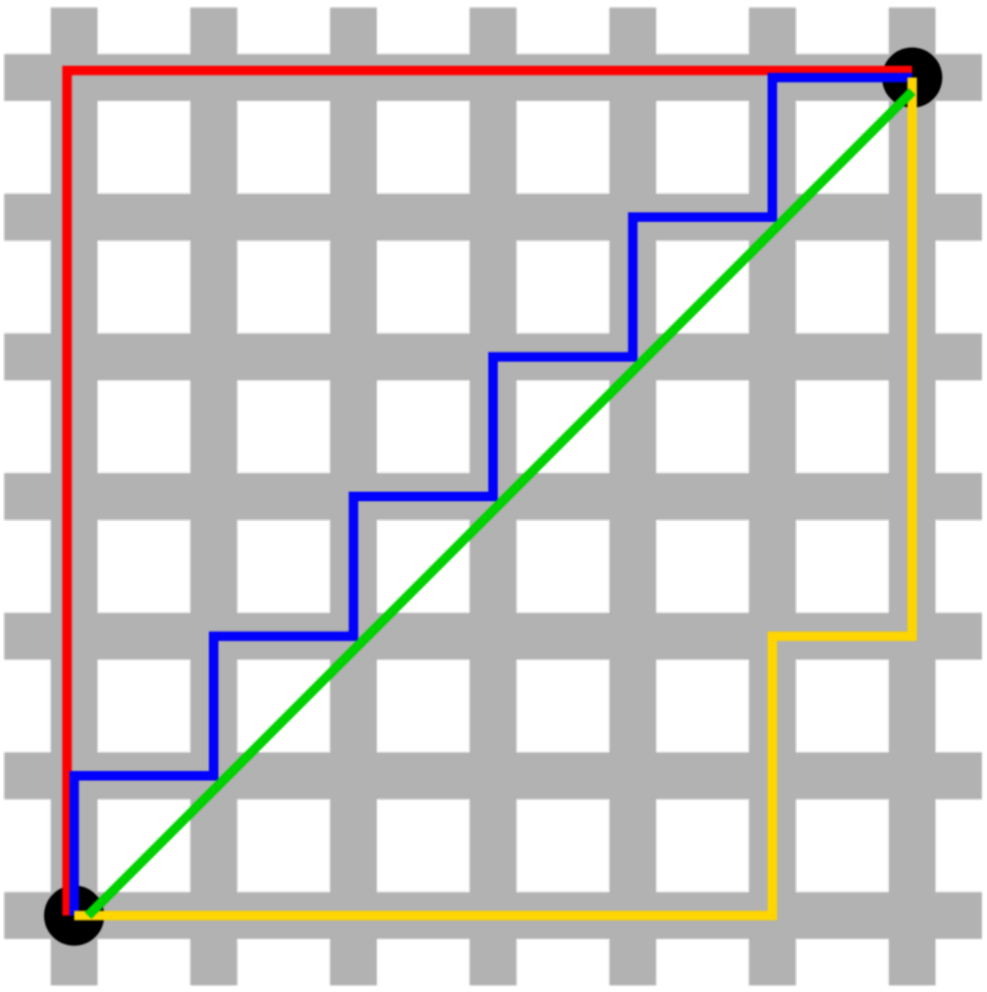
The most common distance metrics are :

1. **Euclidean Distance**: It's defined as the square root of the sum of the squared differences between two points.



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

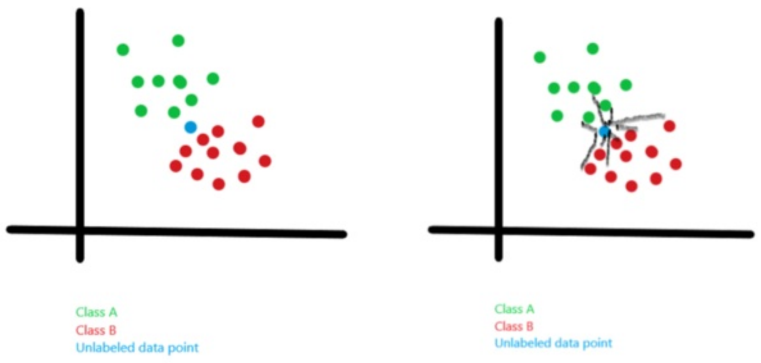
2. Manhattan Distance: This is the distance between real vectors using the sum of their absolute difference.



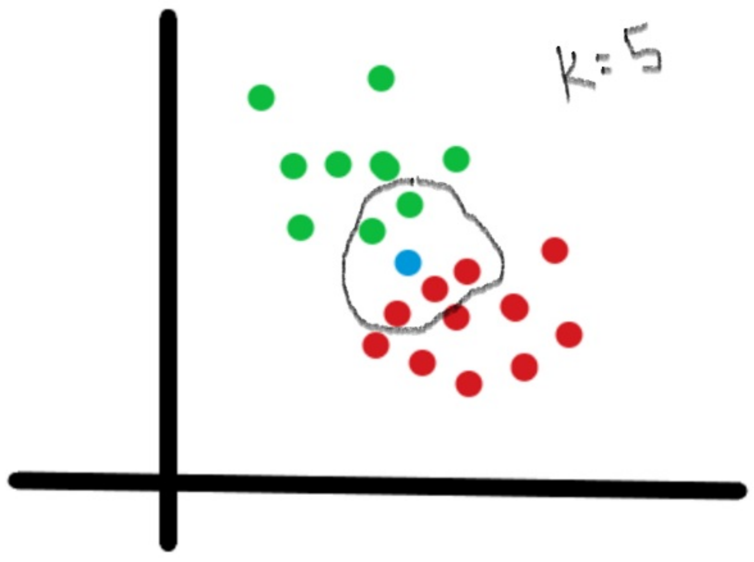
$$\left|(x_2 - x_1)\right| + \left|(y_2 - y_1)\right|$$

How K-NN works?

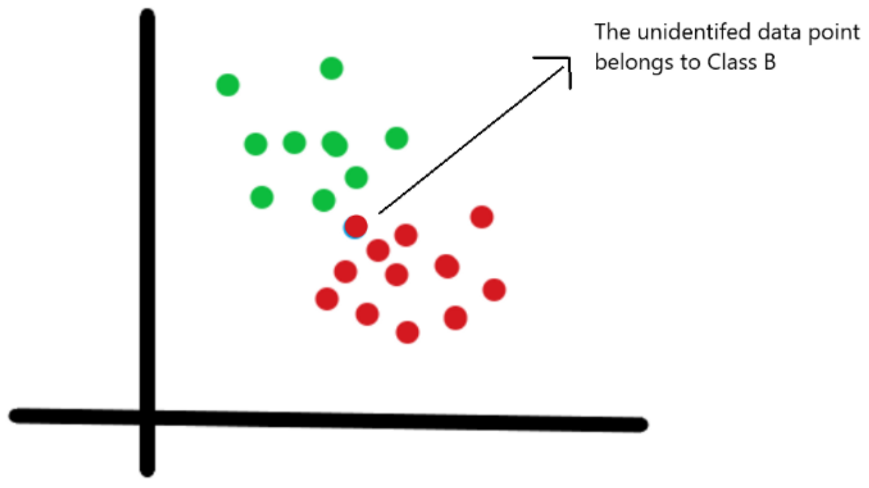
1. Divide data into test and train set.
2. Assume a value for “k”.
3. Assume a distance metric and compute the distance to its n-training samples.
4. Sort distance calculated and take k-nearest samples.
5. According to majority points, assign the class to an unlabeled data point.



We are trying to compute the distance of various points from a new unidentified data point.

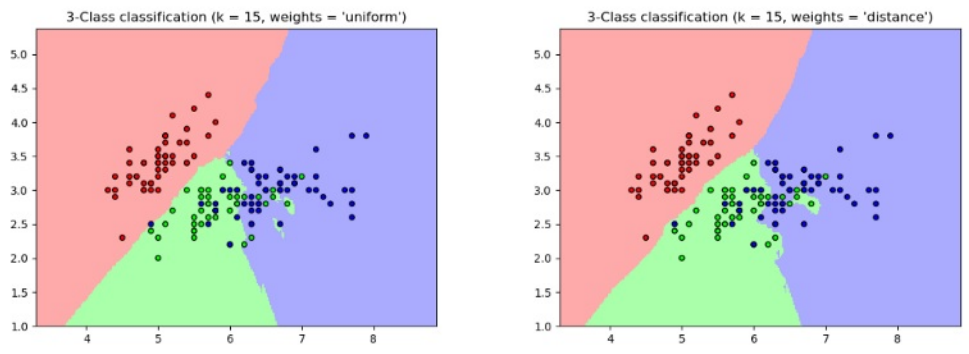


Once the distance is calculated, we take the k-nearest sample and according to majority votes, assign a class to an unidentified data point.



How do we choose “K”?

A concise answer to this is, there is **no optimal value** for “*K*”. It is **hyperparameter** and thus it is chosen by you, but remember it is the decision boundary and therefore must be chosen smartly.



K=1 vs K=15; [Source]

As we can see, when **K=1** the curve was too sharp but as we increase K, the sharpness decreases, and smoothness comes into play. And if we increase K to infinity, everything will be either red, green or blue based on majority votes.

Value of "*K*" plays a really important role in determining the decision boundary;

> K : too large ; everything is classified in given classes(Underfitting)
>
> K : too small ; highly variable dataset and unstable decision boundary(Overfitting)

The value of "K" can be finalized using two methods:

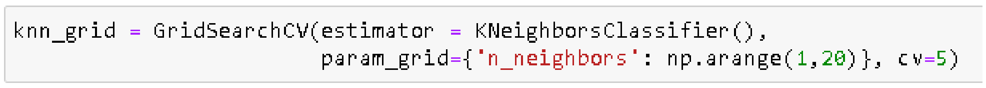
1. ***Manual method***: We need to use the Hit & Trial method; vary the value of K and observe training and validation error.

If the training error is very low but the test error is high, then our model is **overfitting**. And if we experience high training error but low testing error then our model is **underfitting**.

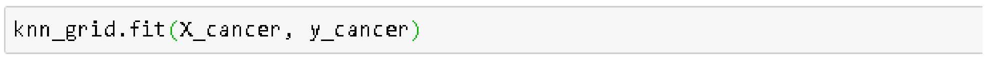
To know more about selecting an optimal fit for your model :

[Determining the perfect fit for your ML model.]
[Teaching Overfitting vs Underfitting vs Perfect fit in easiest way.] [medium.com]

2. ***Grid search***: Scikit-learn provides **GridSearchCV** function that allows us to easily check multiple values for K.



```
knn_grid = GridSearchCV(estimator = KNeighborsClassifier(),
                        param_grid={'n_neighbors': np.arange(1,20)}, cv=5)
```



```
knn_grid.fit(X_cancer, y_cancer)
```

Pseudocode for KNN

```python
def dist(x1,x2):
    return np.sqrt(sum((x1-x2)**2))
```

distance knn hosted with ❤ by GitHub view raw

```python
def knn(X,Y,queryx,k=5):           #X,Y = Distance metrics; queryx = unl
    vals=[]
    m=X.shape[0]
    print(m)
    for i in range(m):
        d=dist(X[i],queryx)
        vals.append((d,Y[i]))

    vals=sorted(vals)                #Sorting distance
    vals=vals[:k]
    vals=np.array(vals)
    new=np.unique(vals[:,1],return_counts=True)
    index=new[1].argmax()
    pred=new[0][index]
    print(pred)
```

knn pseudo code hosted with ❤ by GitHub view raw

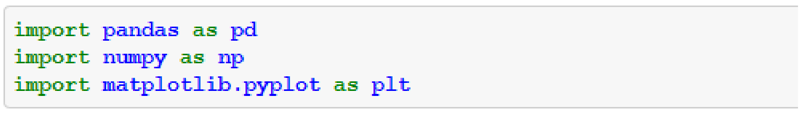
Here, we are accepting the parameters as distance metrics and then we are calculating the distance of unlabeled data point from each labeled point.

Once the distance is computed we are taking unique values and using argmax to find majority votes, later we can classify the point into classes using these votes only.

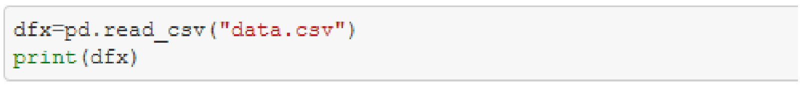
Implementing KNN to classify breast cancer as Malign and Benign

We are going to experiment with our KNN model on one of the most common datasets i.e. Breast cancer detection.

The dataset contains 569 rows & 33 columns. The cancer is segregated in two parts i.e. Benign(B) and Malignant(M). Our aim is to classify the traits of random cancer patients to either B or M.



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

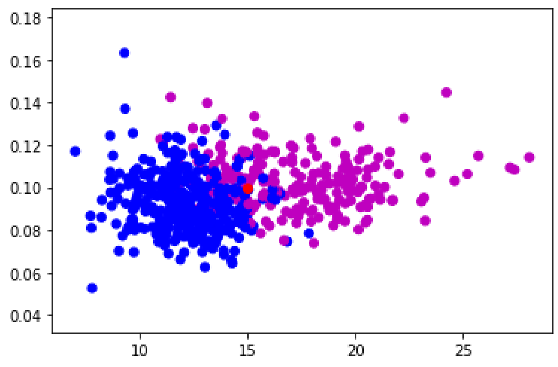


```python
dfx=pd.read_csv("data.csv")
print(dfx)
```

Importing all the required libraries, the dataset can be found at :

[Breast cancer data]()

```
plt.scatter(X[:,0],X[:,1],c=Y)
queryx=np.array([15,0.10])
plt.scatter(queryx[0],queryx[1],color="red")
plt.show()
```



Plotting the data as Benign(Blue) and Malignant(Purple), the unlabeled data point is shown in red color.

```
def dist(x1,x2):
    return np.sqrt(sum((x1-x2)**2))
```

```
def knn(X,Y,queryx,k=5):
    vals=[]
    m=X.shape[0]
    print(m)
    for i in range(m):
        d=dist(X[i],queryx)
        vals.append((d,Y[i]))

    vals=sorted(vals)
    vals=vals[:k]
    vals=np.array(vals)
    new=np.unique(vals[:,1],return_counts=True)
    index=new[1].argmax()
    pred=new[0][index]
    print(pred)
```

```
knn(X,Y,queryx)
```

```
569
B
```

After executing the algorithm, the data point can be regarded as a part of the benign family.

The code can be found at:

dakshtrehan/KNN-on-Breast-Cancer
*You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or…*github.com

Execution of KNN using sci-kit learn

```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifer(n_neighbors=3)
neigh.predict(test)
```

knn scikit hosted with ❤ by **GitHub** **view raw**

Pros and Cons of KNN

Pros:

1. Easy and Simple
2. Training time is low.
3. Robust to noisy training data.

Cons:

1. Determining perfect “K” is tedious.
2. Large storage requirement.
3. Sensitive to outliers.
4. Least accurate algorithm.

Conclusion

Hopefully, this article will help you to understand KNN in the best way and also assist you to its practical usage.

As always, thanks so much for reading, and please share this article if you found it useful!

Feel free to connect:

> *LinkedIN ~ https://www.linkedin.com/in/dakshtrehan/*
>
> *Instagram ~ https://www.instagram.com/_daksh_trehan_/*
>
> *Github ~ https://github.com/dakshtrehan*

Follow for further Machine Learning/ Deep Learning blogs.

> *Medium ~ https://medium.com/@dakshtrehan*

Want to learn more?

The inescapable AI algorithm: TikTok
*Describing a progressive recommendation system used by TikTok to keep its users hooked!*towardsdatascience.com
Detecting COVID-19 using Deep Learning
*A practical approach to help medical practitioners helping us in the battle against COVID-19*towardsdatascience.com
Why are YOU responsible for George Floyd’s murder & Delhi Communal Riots!!
*A ML enthusiast’s approach to change the world.*medium.com
Things you never knew about Naive Bayes!!
*A quick guide to Naive Bayes, that will help you to develop a spam filtering system!*medium.com
Activation Functions Explained
*Step, Sigmoid, Hyperbolic Tangent, Softmax, ReLU, Leaky ReLU Explained*medium.com
Parameters Optimization Explained
*A brief yet descriptive guide to Gradient Descent, ADAM, ADAGRAD, RMSProp*towardsdatascience.com
Gradient Descent Explained
*A comprehensive guide to Gradient Descent*towardsdatascience.com
Logistic Regression Explained
*Explaining Logistic Regression as easy as it could be.*towardsdatascience.com
Linear Regression Explained

*Explaining Linear Regression as easy as it could be.*medium.com
Determining perfect fit for your ML model.
Teaching Overfitting vs Underfitting vs Perfect fit in easiest way. medium.com
Relating Machine Learning Techniques to Real-Life.
*Explaining types of ML model as easy as it could be.*levelup.gitconnected.com
Serving Data Science to a Rookie
*So, last week my team head asked me to interview some of the possible interns for the team, for the role of data…*medium.com

Cheers

By Daksh Trehan on June 27, 2020.

Canonical link