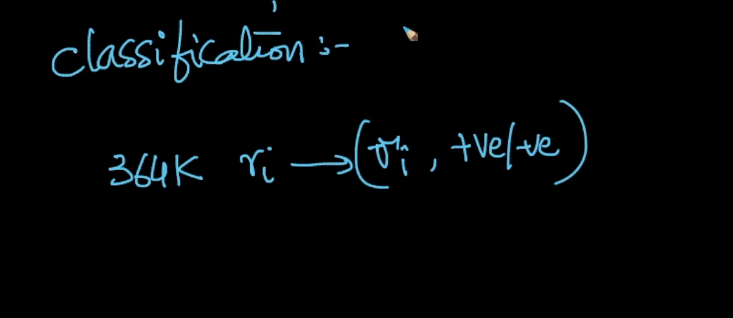
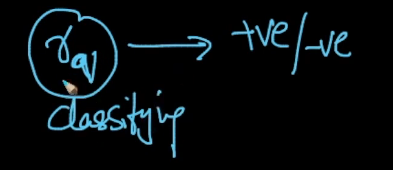
**How classification works?**

If we take Amazon food review dataset as an example, we came across that for each reviews, we convert that to a vector. Each vector will be having either a positive value or negative value.

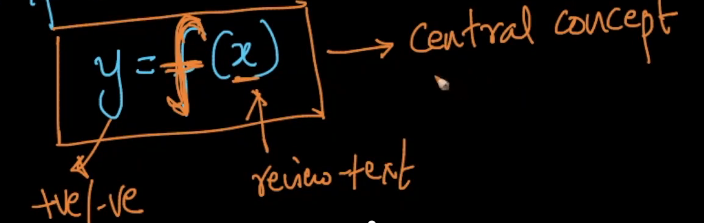


So classification is all about, given a new review, determine whether the review is positive or negative.

This is main task of classification.

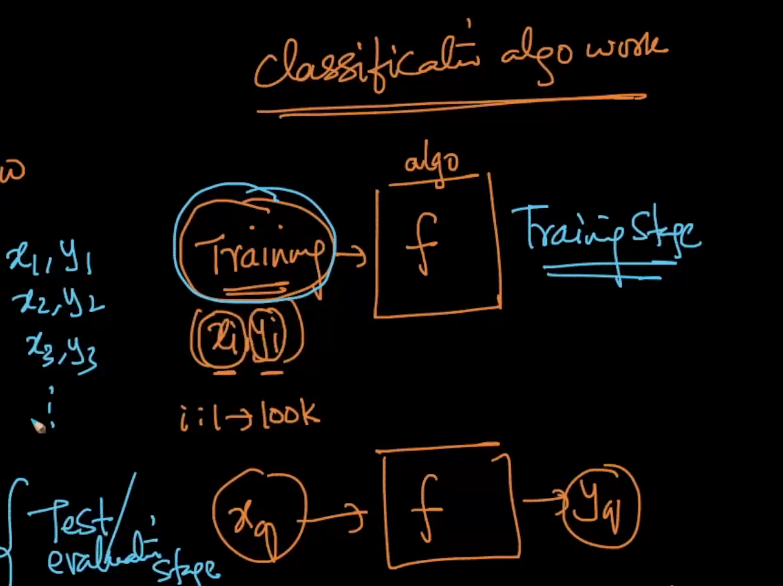


So basically we are classifying into positive or negative.



So Crux of the classification, finding a function f for a given x and storing that in y. Y will say whether it is positive or negative for the given new review x.

How does classification algorithm works?



As show in the above figure, for a classification model,

We can divide the phase into two,

1. Training stage: In the training stage, model learns with n number of values that we feed. As shown xi and yi are the values that been fed to the model.
2. Testing/Evaluation phase: This is the phase to test the model using the new values and getting the output almost close to the real world solution.

Any given model may not be learning the things perfects and predicting the output with 100% accuracy.

**Notations in Classification Algorithm:**

Every vector by default is a column vector



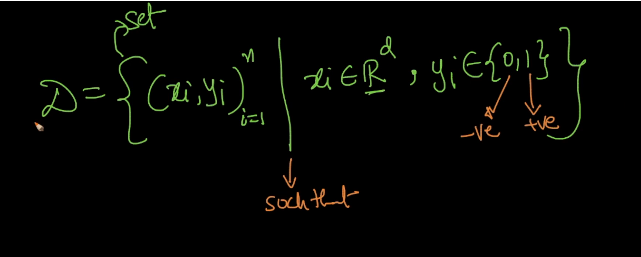
d- dimension, xi – input vector, ri -> review, yi-> is the output

As shown in the above diagram, for each reviews we are converting it to text and text to a vector.

For each of xi and yi , we are

Ri 🡪 xi 🡪 {0,1}

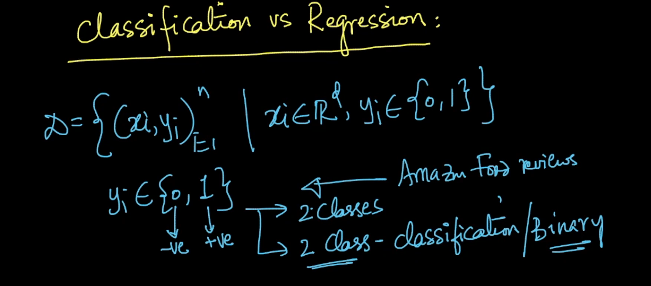
Whole task of Classification / purpose of classification is that “takes the xi and yi pairs as a training data and learns a function f which should output yi when you input xi”.. The better we learn the function, the better the system is.



Data is dataset of pairs of numbers (xi and yi) and I have n such numbers, such that xi is the D dimensional vector and yi could be a 0 or 1. Yi belongs to a set. 1 belongs to positive review and 0 means negative review.

This is a representation of a dataset.

**Classification vs Regression with Example:**



Classification problems is also called as 2 class classification problem or binary classification problem.

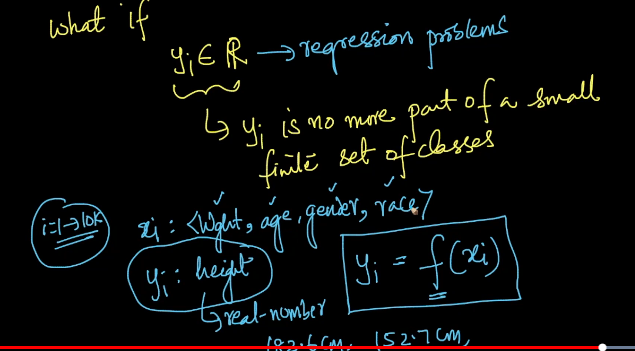
This is when we have outputs in the form of 0 and 1.

Similarly, consider MNSIT dataset in which we have images and yi could be {0,1,2,3,4,5,6,7,8,9,10}

This is called, multi-class classification problem.

So we identified that we have multiple classes



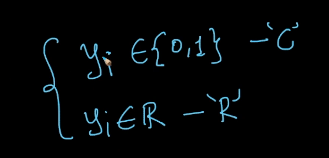


The only difference between Classification and Regression problem is

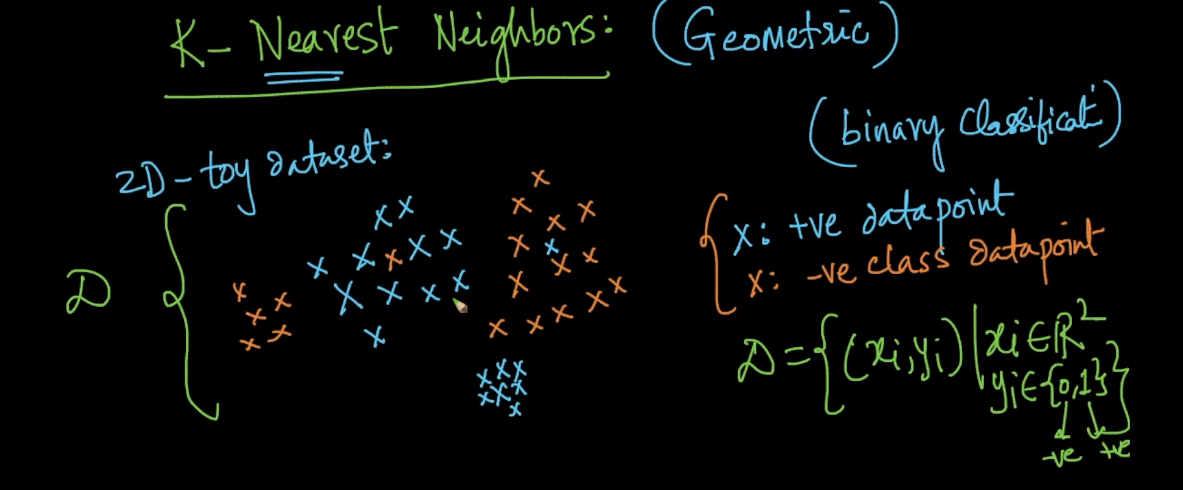
Classification 🡪 output belongs to one of the class. Example in binary classification problem, output will be either 1 or 0. Similarly, in multi-class classification problem output will be one of them in 10 class.

Regression🡪 yi Is no more a small part of a small set of finite classes.

Below equation shows the difference between them with respect to mathematical equation.



**KNN Geometric with Geometrical intuition with practical example (toy dataset)**

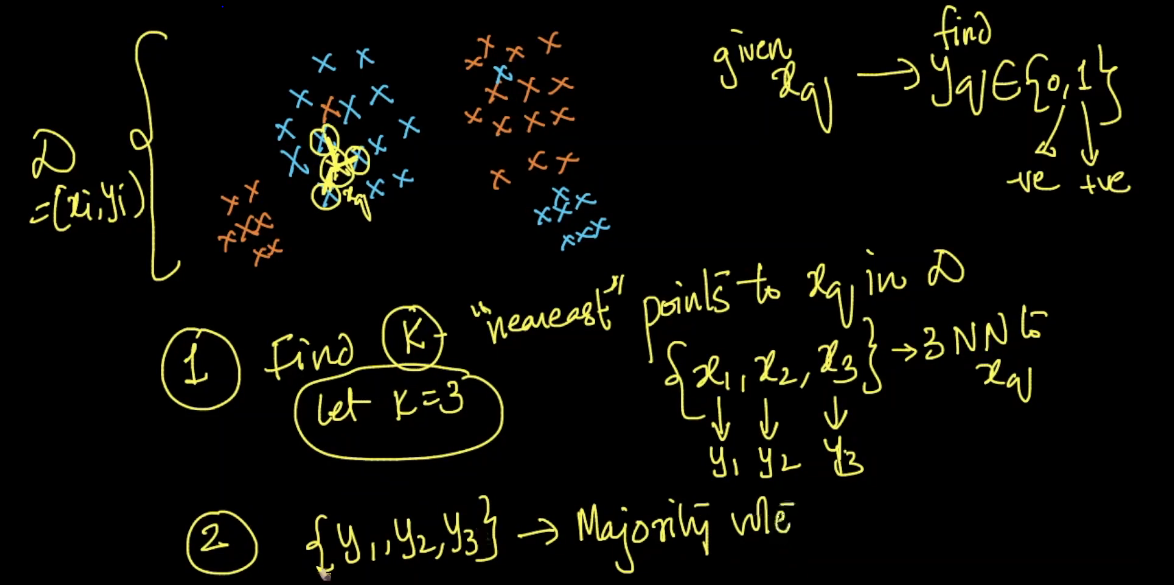


The above screenshot explains that,

For a given data set D, we have two dimensional data set.. x in blue with positive data point and x in red with negative data points.

As we have two classes here, we call it as binary classification problem.

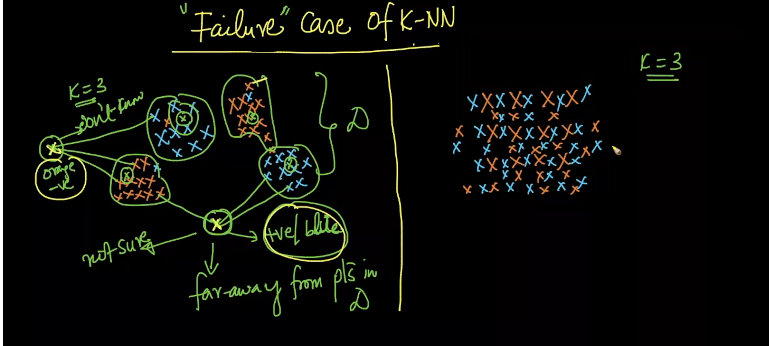
D is a dataset having pairs (xi and yi) data points where xi belongs to R2 (because it is 2 dim dataset) and yi belongs to 2 classes (0 and 1). 1 🡪 positive 0—negative



**Steps for KNN:**

1. **For a given Xq , find the nearest points to Xq in Dataset D.**
2. **Let’s say , k=3 , let’s take {x1,x2,x3} are the 3 nearest points**
3. **For those 3 nearest points, Y1,Y2,Y3 are the outputs.**
4. **Now take the majority vote among three outputs {Y1,Y2,Y3} and estimate what is the Xq is..**
5. **At the end , we will come to know that whether Y is positive or negative.**

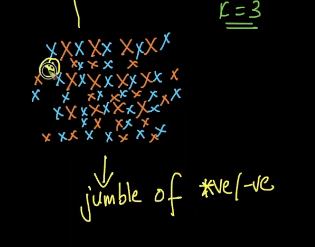
**Failure Cases of KNN:**



**As shown in the above snapshot, we can observe that the x in question is little far away from all the other points**

**In that case, x is far away from the data set. Even if it identities the 3 near x values, we would never know that it’s a correct prediction or not. So in this case it is better to say I don’t know so that model can reject these values for considering.**

**2nd Failure case:**



**Case when the data is jumbled up randomly here. In this case the data is completely randomly spread.**

**In this example we can see that the blue and orange points are jumbled. It is simply a kind of noise.**

**And KNN would miserably would fail.**

**From Comments:**

What is probabilistic estimates:

Answer:

Let us assume we are working on a 3-class classification problem. Let the classes be 'A', 'B' and 'C'. If K = 9 and for a query point ′x′q′xq′, if there are 5 points belonging to class 'A', 3 points belonging to class 'B' and 1 point belonging to class 'C', then the probabilities for the query point to belong to the classes 'A', 'B', 'C' are 5/9,1/3,1/9 respectively.   
You'll learn about probability estimates in the video of 'multi-class classification'

Question:

What we usually see in a data set there exist data points and for each data points, there is a class label defined. For given new data points we have to find its class label. But, as you have said that if a data point is very far from out existed pattern we would declare it does not belong category. But we have to classify it at anyway? How it is going to happen with KNN where it fails?

Answer:

In such cases, KNN predicts the class labels for such data points also but it may or may not be correct because we have a pattern of points at one side and only one point on the other side that too very far away from this pattern. In this case, the K nearest neighbors are computed from this existing pattern of points only. As this individual query point is far away from the other existing points, though the class label is predicted using the majority vote, it is not necessary that this point exhibits the properties of the points in the pattern/cluster. Hence we say KNN fails in such situations.

In this kind of scenarios, algorithms like Logistic Regression, SVM, etc will work better as these algorithms just construct a hyper-plane that separates the classes and making a decision based on the geometric location of a point from the hyper-plane gives better & a confident result when compared to predicting the label from K nearest neighbors.

We generally have very large datasets in the real word, If K-NN isn't useful such cases why, KNN is one of the top most algorithms being used in ML.

Answer:

Yes, KNN algorithm is one of the simplest classification algorithm and it is one of the most used learning algorithms because of its simplicity, high accuracy, no assumptions about data and also versatilt(useful for both classification or regression)  
  
but since the algorithm stores all of the training data So in practice, the computational complexity of KNN increases for many applications, particularly for large data sets. It is at the extreme end of the tradeoff between computational efficiency and model flexibility.  
  
Kindly, check out the lecture

<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/testevaluation-time-and-space-complexity/>

My doubt is regarding Classification and Regression.In my point of view Classification and Regression both come under predicting Model.In Classification we can predict the class label from discrete set where as in Regression it continuous set.  
  
1. any chances of classification produces continuous set of outputs? else Regression produces discrete set of outputs?  
if yes please explain me with an example of real world problem

Answer:

classification may produce a continuous value, but the continuous value is in the form of a probability for a class label. for example, an email can be assigned probabilty 0.1 as being "spam" and probability 0.9 as being "not spam". We can convert these probabilities to a class label by selecting the “not spam” label as it has the highest probability.  
regression may predict a discrete value, but the discrete value in the form of an integer quantity.  
refer this:  
<https://machinelearningmastery.com/classification-versus-regression-in-machine-learning>..

1. if we have 1000 data points, how K will be derived ?, is this the right way of deriving it ( need 5 points distance so K will be 1000/5 which is 200).  
   2. if we would find such a high distance value, does capping works.  
   3. if there any way that how we can check data is jumbled. I got the concept but can't visualize how this data will look like?  
   4. once K is identified, let's suppose K =5 in 1000 data point, how K will help to predict the xq? will it check the distance for each 100 data point with the distance of 5 like it will check 1st data point then 6th then 11th then 16th and so on ...

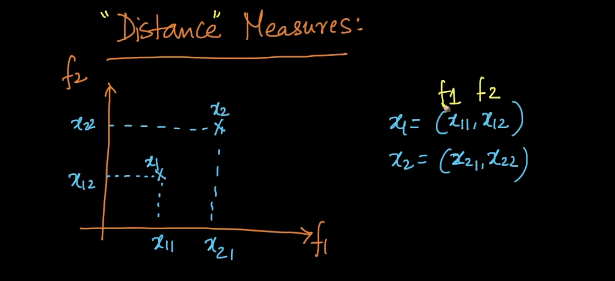
Answer:

1. Optimal 'K'  value has to be obtained only through hyper-parameter tuning irrespective of the dataset size and irrespective of how the points are present in the geometric space.  
     
   2) Could please let us know what 'capping' mean?  
     
   3) The general way to check if the data is jumbled or not is through visualization. If the data is of more than 2 dimensions, then you can reduce the dimensionality and apply visualization and check if there is any possibility to separate the classes using a linear surface.  
     
   4) If the optimal 'K' value obtained is 5, then for a query point, it computes it's distance to all the 1000 points and those nearest 5 points are selected and the majority vote is applied on these 5 points.

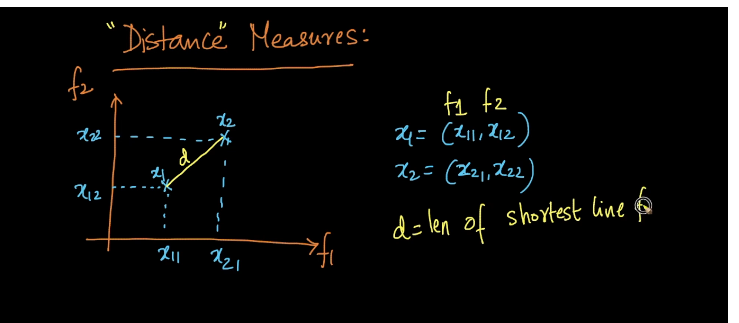
Is KNN good for large data set?

It is not recommended to use KNN for large dataset because, as the training phase of KNN deals only with storing all the train data points in the RAM whereas the testing phase takes huge time. Moreover if there are millions of data points in the train dataset, then the runtime is very huge as the distances from the query point to every data point in the train dataset have to be computed and then the nearest neighbors have to picked up from them.   
In case of large datasets, if you need to make predictions on unseen data on the basis of the class labels of the nearest neighbors, it is recommended to go with either KDTree or LSH. You'll learn about KDTree & LSH in the later set of videos in the same chapter.

Distance Measures: Euclidean (L2) , Manhattan(L1), Minkowski, Hamming



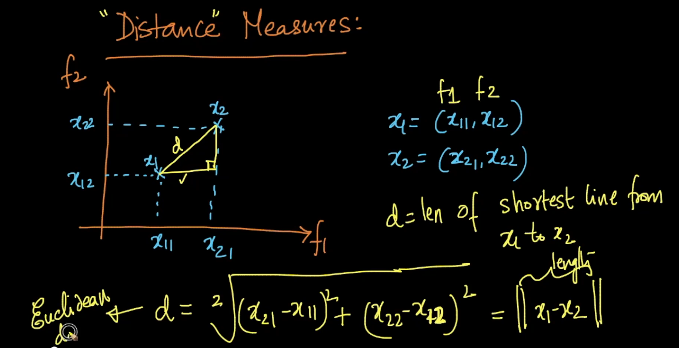
Let us assume that we have two points (x1 and x2) which is in the form of tuple.



Here d= distance of shortest line between x1 an dx2.

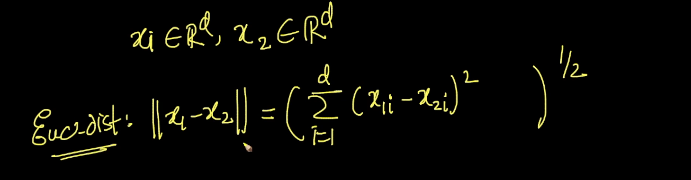
In order to find the distance between two points, we have several techniques.

1. Euclidean distance



Formula for Euclidean distance is given as above for 2 dimensional points.

And formula for N dimension is given below:



In the bellow snapshot we can see the formula for Euclidean distance of two points x1 and x2..which is also known as L2 form.

Below that is a formula for Euclidean distance of x1.. which is a measure of distance from origin to distance.

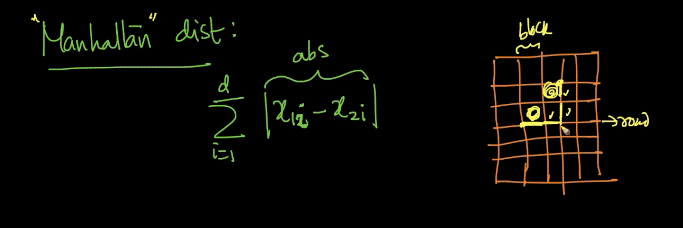


**Manhattan Distance:**

Similarly, distance d can be find using the Manhattan distance as well and here is the formula for the same. This is often called/written as L1 norm.

The formula for the Manhattan distance can be read as , absolute value of x1 –x2.

Where do we prefer manhattan over Euclidean can be understood in the later part of the course.



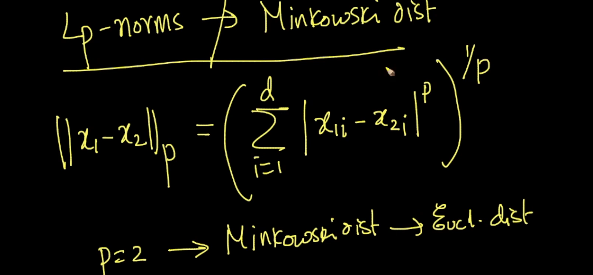
The generalization of L1 and L2 norm is Lp norm.

Lp norm is also called as minkowski distance.

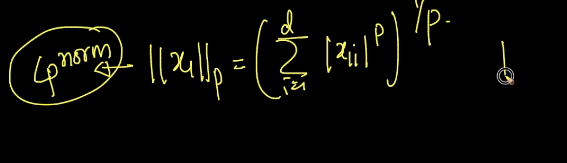
Below is the formula for the same.

If p=2 , we can quickly notice that the minkowski distance is equal to Euclidean distance.

if p=1 , then the minkowski distance is nothing but manhattan distance.



Below is the formula for Lp norm of a vector

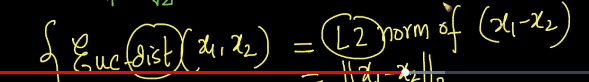


Where p<>0.

We shall discuss the applications of each in later in the course.

The difference between Distance and norms.

Distances are always for 2 points whereas norms are always for a vector.



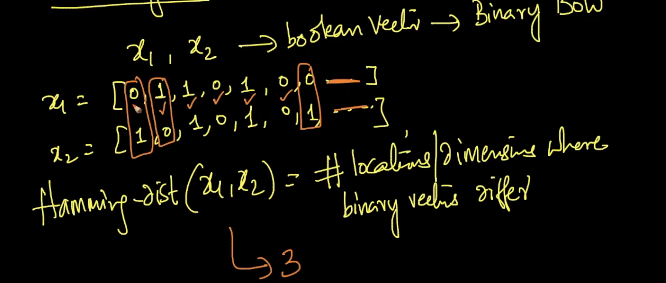
As per the above snapshot, the Euclidean distance between two points is equal to L2 norm of vector x1 and x2.

Distances are between two points and norms are for vector.

There are many other types of distances apart from Euclidean and Manhattan.

Like Hamming distance is one good example.

Example:



X1 and x2 are the Boolean vector where we use in bag of words.

Hamming distance is nothing but the no. of locations/ dimensions where binary vector differs.

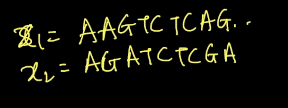
X1={0,1,1,0,1,0,0}

X2={1,0,1,0,1,0,1}

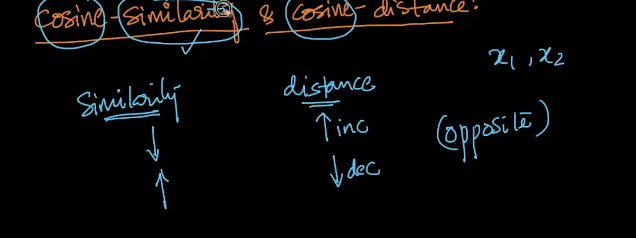
Hamming distance (x1,x2)= # of locations/dimensions where binary vector differs.

As per the above snapshot example, the distance is 3 as we see that at three places the number differs.

This type of distance measurement is useful when finding a gene sequence.



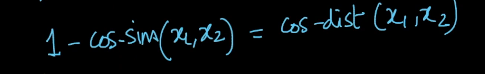
**Cosine Distance and Cosine Similarity:**



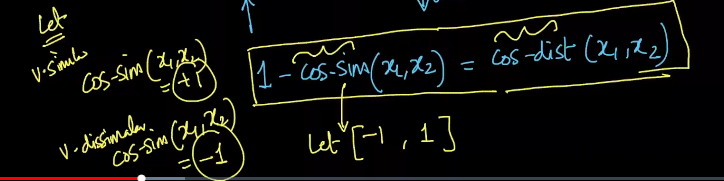
The relationship between cosine similarity and difference is as given below

As shown in the above snapshot similarity and cosine are inversely proportional to each other. As the similarity increases distance increase and vice versa.

Formula relationship between the two:



Another important point of cosine similarity and distance is



Say when we have two points x1 and x2, ,

1-cos similarity (x1,x2)= cos distance(x1,x2)

Let just say , cosine similarity lies between -1 and +1..

If my points x1 and x2 are very similar, then cos sim= +1

If my points x1 and x2 are very dissimilar then cos-sim=-1.

So now let us apply the above values to the below formula

1-cos similarity (x1,x2)= cos distance(x1,x2)

Case when value cos-sim= 1\

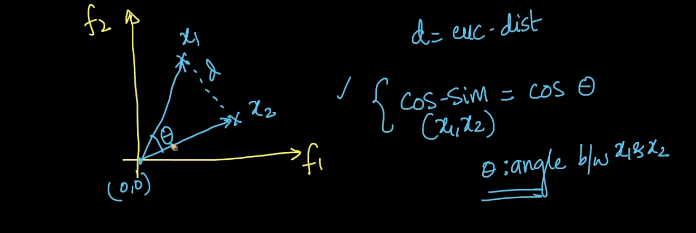
Then cos distance will be 0 because the points are very similar and distance between them will be 0.

Case when value cos-sim= -1

Then cos distance will be 2 because the two points are very dissimilar and distance between them increases.

Let us take an simple example to understand the same:

In the below given snapshot , consider the points x1 and x2 and d is the distance between the two points. As per the geometrical distance ‘d’ is nothing but the distance between the two points. But cos-similarity is nothing but measuring the angle between the two points so in this case we take cos theta to the angle between two points,

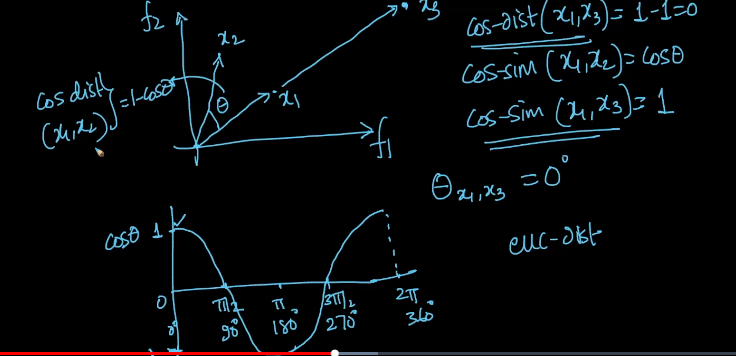


X1 and x2 are the vectors from origin.

Cos sim (x1,x2)= Cos theta

Theta: angle between x1 and x2.

Let us take another example to understand better



As shows in the above example, now we have considered three points names x1,x2 and x3.

**Cos-sim(x1,x2)=Cos theta ----eq.1**

**Cos-sim(x1,x3)=1 ( because angle between x1 and x2 = 0 and cos 0=1)-----eq.2**

And also we can see how Cos behaves in the diagram.

**So now, we know that Cos-dist (x1,x2)= 1- Cos-sim(x1,x2) ---eq 3**

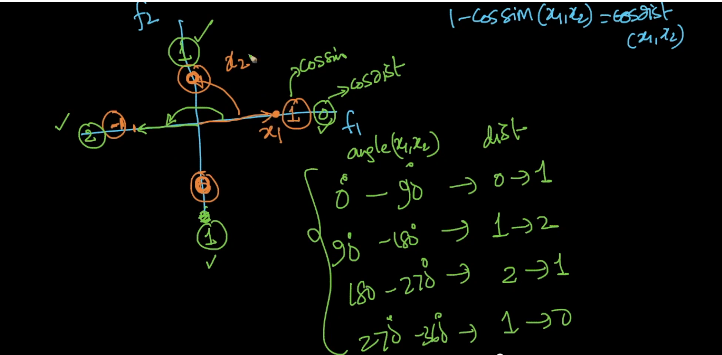
**Cos-dist(x1,x3)= 1 – Cos-sim(x1,x2)**

**Cos-dist(x1,x3)=1-1 = 0**

**Similarly, Cos-dist(x1,x2)= 1- Cos-sim(x1,x2)**

**=1-Cos(theta)**

Here what we can observe is, though the distance between is x1 and x3 is geometrically is greater than 0 but angular distance is 0. Hence cos-distance and cos-similarity is inversely proportional.



As shown in the above figure, orange points shows the Cos-sim and green shows the Cos-distance.

X1 and x2 are at same points then cos-distance(x1,x2)=0 as explained above.

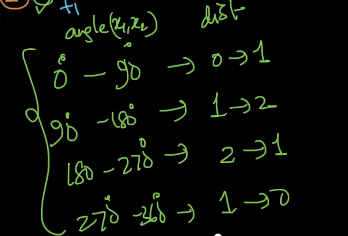
Similarly, as x2 moves farther away from x1, the distance increases.

If x2 lies on 90 degree then cos-dist will be 1

If x2 lies on 180 degree then distance will be 2 ( 1-cos-sim= 1-(-1)= 2 )

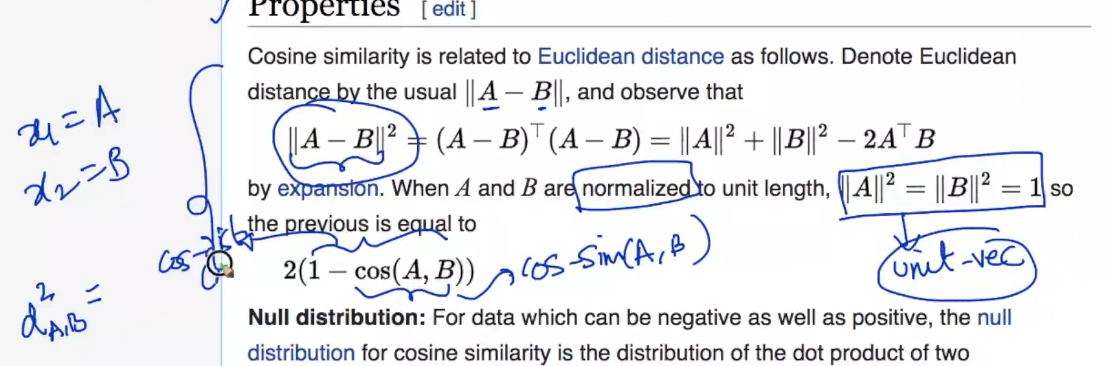
Ix x2 lies on 270 degree then distance be reducing as the angle is approaching back to x1 cos-distance = 1

Below tables shows the relation between cos-sim and cos-distance at various points in a plane.



Relationship between Euclidean distance and Cos-sim or Cos-distance

As shown below , the cos-distance (A,B), if A and B are unit vector then the Euclidean distance = 2 times the Cos-distance.



As shown below the x1 and x2 are unit vectors and theta is the angle between x1 and x2, the square of Euclidean distance between x1 and x2 is nothing but the 2 times the cos-distance.

**[ Euc-dist (x1,x2)]^2** = 2(1-cos(theta))

Where **(Cos(theta)=cos-sim and 1-cos(theta)=cos distance) .**

**Above holds true only IF x1 and x2 are unit vectors**



**Measure How good is KNN:**

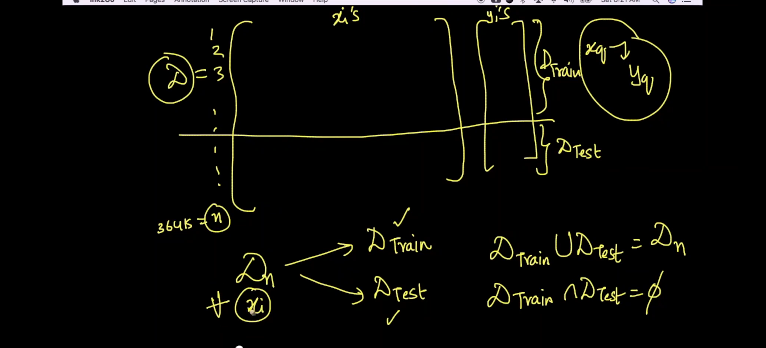
Problem Statements: Our problem is given a new food review, what is its polarity.

step 1:

Given xq, determine what is Yq.. ( is nothing but the polarity)

Measure: How well KNN works

KNN --- Compute the K nearest neighbor and take the majority vote



As given in the above snapshot , D is full data set having train and test dataset.

Given xq, we have to predict yq..

Union of Dtrain and Dtest should be present in Dn..

Where as Dintersection of Dtest should be null.. Because any given xi should not be existing in both the dataset.

Splitting the dataset:

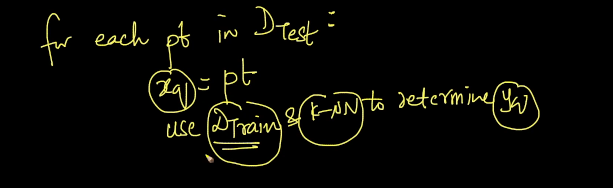
Dtrain and Dtest

Why we have to split the dataset?

Dtrain – is used to train my model

And test the data is to how well the model is doing

How KNN algo works while testing Dtrain and Dtest:



For each point in Dtest m make xq = point

Use Dtrain and KNN to determine Yq..

For xq, using Dtrain only and KNN I determined what is Yq is .

Next compare Yq== Y pt

Cnt +=1

At the end of for loop we get cnt= no. of points for which Dtrain + KNN gave a correct class label

Now I can measure and it is called Accuracy.

Accuracy = cnt/n2

Cnt= No. of points for which we got correct class lables

N2= # of points in the total Dtest

Lets say my accuracy is 0.91 which means 91% of the times , my KNN was accurately predict Yq for a given Xq.

Now we can conclude that KNN on my amazon food review using Dtrain gives me the accuracy of 91%...

To conclude:

We have Dn , split it into two sets Dtrain and Dtest..

KNN Algo facts:

k-NN is said to be a lazy learner which mean there is not training phase. Hence we directly took the query point and measure the k nearest points in the training data and then do the maximum voting of the class labels for all those k points. Hence there is no training phase of the algorithm.

In KNN, the training phase consists of loading all the train datapoints into the RAM. Apart from that there's no learning. The actual task of prediction begins in the test phase.   
As The train phase consists of only storing the datapoints, we say that there's no training in KNN

In K-NN, the training phase consists of only storing all the datapoints in the RAM. You have to find the optimal 'K' value through k-fold cross validation..(Here k in K-fold CV is different from K in KNN)  
  
1) Split the given dataset into train and test datasets(mostly 70:30 or 75:25)

2) Let us take a  random set of values of K(in K-NN) using trial and error method. All of them should be odd numbers. Let them be denoted as {set\_of\_values\_for\_K}

3) Divide the train dataset(both X and Y components) into 'k' folds(parts). Let us name them as fold\_1,fold\_2,fold\_3,.....fold\_k

4)stage 1: Keep fold\_1 as CV dataset and the remaining folds combinedly as training dataset. Train the mode using this train dataset and make predictions on the CV dataset(fold\_1). Compute the train error and CV error.  
stage 2: Keep fold\_2 as CV dataset and the remaining folds combinedly as training dataset. Train the mode using this train dataset and make predictions on the CV dataset(fold\_2). Compute the train error and CV error.  
.  
.  
.  
stage k: Keep fold\_k as CV dataset and the remaining folds combinedly as training dataset. Train the mode using this train dataset and make predictions on the CV dataset(fold\_k). Compute the train error and CV error.  
Now compute the average of all the train errors and let us denote it as **average\_train\_error**.  
Now compute the average of all the CV errors and let us denote it as **average\_cv\_error**.

5) For each value in {set\_of\_values\_for\_K}, repeat step 4 and compute the **average\_train\_error**and **average\_cv\_error.**

6) Build a graph with K values on X-xis and Errors on Y-axis and plot both the **average\_train\_error**and **average\_cv\_error**associated with each value in {set\_of\_values\_for\_K} and whichever value of 'K' yields **low** **average\_train\_error**and **average\_cv\_error**, that is considered as an optimal 'K(hyperparameter in KNN)

7) Now build an optimal model using the train data with the optimal 'K' value obtained and then make predictions on the test data and compute the performance metrics.

What is CV Error and Training error:

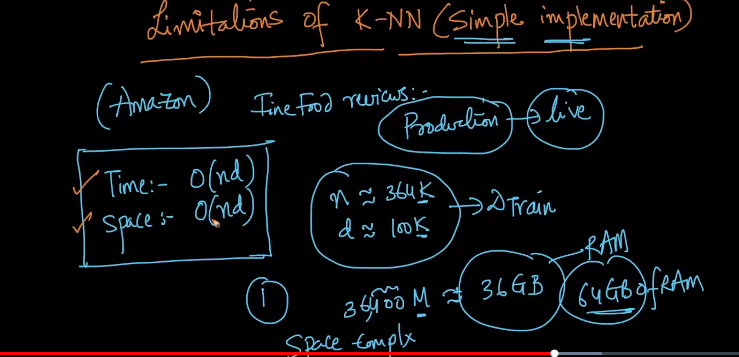
Training Phase in KNN:   
All the data points of the training  dataset are stored in the RAM.  
  
Cross Validation Phase in KNN:  
For different values of 'K' the model is fitted with the training data and the optimal hyperparameter is obtained by checking the performance of those models on CV data. The performance metric whichever is used in cross validation(say accuracy or AUC or Precision or Recall or F1) is the cross validation score. (CV error = 1-cross validation score)  
  
During this phase itself, we try to make predictions with the same models used in CV phase on the training data points and compute the same scores. These are called Training scores. (Train error = 1 - Training scores).  
  
The optimal hyperparameter is the one which gives least train and cv errors.

**Test/Evaluate time and space complexity:**

Limitations of KNN (simple implementation)

1. Implementation of KNN is very difficult and they are given below

As shown in the below image, the time complexity of a KNN algorithm us 0(nd) and space complexity is also 0(nd) which is terrible in real time.



For n~364k and d~100k, we need easily 36GB of RAM and which is costly.

As per the amazon food review dataset, n=364k and dimension is 100k

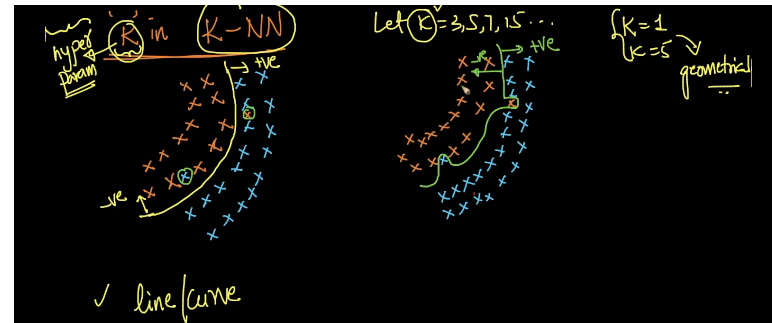
KNN is simple, super and elegant but it’s not used because of terrible time and space complexity.

**Decision surface for K-NN as K changes**

We always have been talking about the value of K and its importance.

But we never discussed about how to decide the value of k and also what value of k rightly fits.

So lets do that below



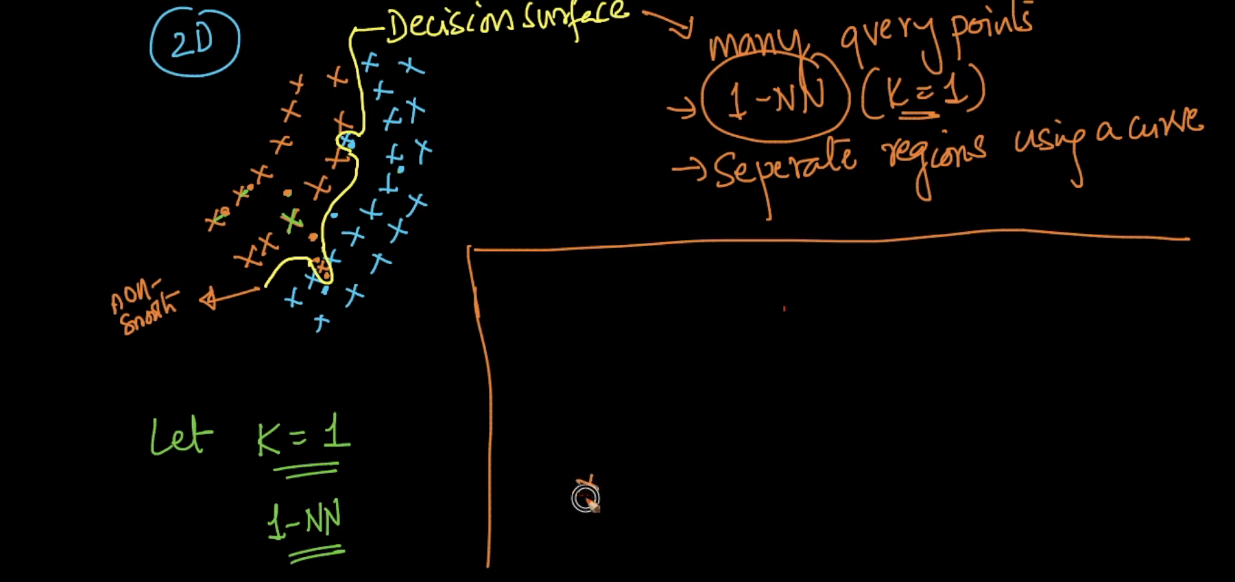
As shown in the above figure , there are two data sets which is almost similar and if we ask someone to draw a line or curve line and fig 1 could be the one possibility and fig 2 could be another possibility.

Let us call fig 1 curve as Yellow curve and fig 2 as Green curve.

Both of them are trying to separate the positive and negative points. But fig 1 curve or Yellow curve is smooth whereas other curve is rough.

These curves which separates the positive points from negative and vice-versa is knows as decision curve surface. Surface we call It because it’s a 3d space.

Now we are clear with the concept of what is decision surfaces



Now let us understand the KNN with different K values

K=1

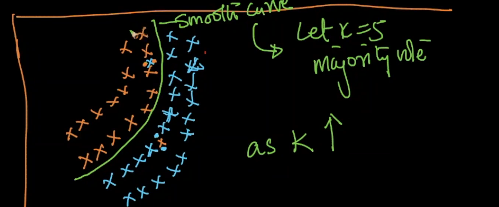
When K=1 , in the figure we can see that the we try to fetch a 1-NN ( one nearest neighbor) for the given the query point in the dataset. Similarly, we have queried for other points and drawn a decision surface as shown above.

K= 5

Now what happens when K=5

As shows in the below snapshot

We see that , for the given dataset , we have majority votes are with Blue or with orange for the query points taken. Though we have one or two other values but the majority values is the correct prediction( as per the taken sample dataset) and hence we can draw the decision surface as show in the figure and also we can mention that as K increases smoothness of the curve increases.



Now let’s say K= N

Because the maximum value can be n

N= total no. of points then what happens

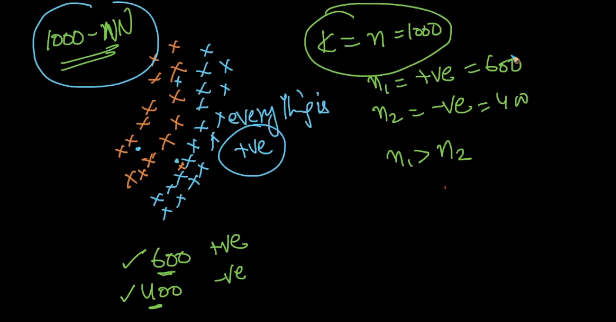
N1= Positive

N2= Negative

N1+N2= N (total points)

Let’s assume N1> N2 ( N=1000, N1=600, N2= 400)

Let’s understand what happens when K=N



As shown in the above figure, here the K=1000 points ..In this case , if my points lies in the positive area, then we take all 10000-NN points ( k=1000) and when taken the majority we classify it as positive which is correct.

Say suppose , if my points lies in the negative area and again we try to pick all the 1000 points near by and taken the majority votes. Since we are taking maximum of N ,we will have N1=600 and N2=400 and classifying it as Positive points which is incorrect.

So to conclude , we understood what happens when k=1,2,3,--n.

When K=1 the curve will be rough and less accuracy too . as in when K increases curve tend to get smoothers with more accuracy in the prediction.

Finally when k becomes maximum , then whatever the majority of the points is it will be taken ( example shown above)

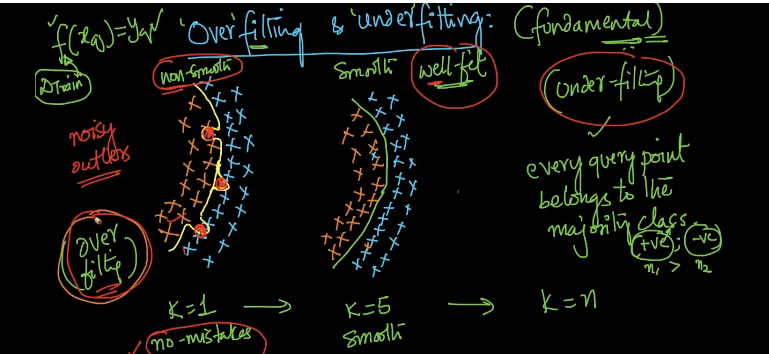
Worst case, if N1=N2 then model will behave randomly and will have no idea what to predict.

Questions on Decision surface:

1. We have used the existing data points to compute the decision surface. Is it possible that the algorithm can become indecisive when given a new data point, it lies exactly on the decision surface?
2. When K=n & n1=n2, what happens in the code implementation? When you say that the algorithm becomes indecisive on what to do, then do we get some error? What happens?

. Yes, it is possible to have a test point on the decision surface. In such a case, it is better to give a [probabilistic class label](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/probabilistic-class-label/) with P(y\_q=1)=0.5 which implies that there is only a 50% chance that this point, x\_q belongs to class 1. 2. When k=n and n1=n2, the model will give have the same number of points belonging to \_+ve class and the same number of points belonging to -ve class for any query point (x\_q). Now, the kNN cannot decide which class to assign to x\_q as we have an equal number of points of both classes as k-nearest neighbors. In such a case, for every query point, the probabilistic class label would 0.5 which means there is an equal chance of each query point belonging to class 1 and 0. This is a boundary case and we don't have k=n in the real world for any reasonable probelm.

# Overfitting and Underfitting



As shown in the above figure, here we have three different types of model fitting.

1. Over fitting 2. Under fitting 3. Smooth fit

**Over fitting:**

Over fitting is all about, if a classifier tries to fit a function or decision surface is extremely non-smooth.

Here we are trying to work extremely hard to pull all the points and make it to cover all the points

This is too jealous and works hard to get all the points.

Why over fitting is not used because there is a high chance that it will consider the noisy data also to be to be in wrong classification.

**Under fitting:**

Here, am not even trying to make any hard work and I don’t care about the decision surface. This is super imperfect.

**Smooth/ well-fit functions:**

In this case, its ok making some mistakes but majority of the points are in the right path so its ok to miss some minor points.

Example :

I am giving you an example to understand underfitting and overfitting in a very intuitive way that will be related to Gandhiji that someone explained in some comment.  
  
let's suppose I am creating a model to detect the image of Gandhiji.  
  
case 1:  
I trained the model of Gandhiji by just two features...  
1. having specs.  
2. having no hair on the head.  
  
let's see, what happened if I am giving the image of other person having no hair on the head and having specs, model treats it as gandhiji. It is called underfitting.

case 2:  
I trained the model of Gandhiji by using lots of features.  
1. having no hair on the head  
2. having specs  
3. having smile  
4. trained by using only old age photos  
5. etc.  
  
let's see, what happened if I am giving the image of Gandhiji having no specs then my "model" is unable to treat it as Gandhiji because the model is trying to satisfy every feature.  
This is called as "Overfitting".

Not very clear though,overfitting is the condition where the train data is implicitly studied by the model with each and every detail thus it perform well on train data but fails to perform well on test data.On the other hand undefitting happens when model assume about the data distribution when we have less data points.

 If it is giving 100 % accuracy on train data and too low accuracy on test data, then it is called overfitting.In example given by rthothad, in case of overfitting, the model will classify Gandhi with out specs as not Gandhi just because he is not wearing specs. You can avoid relating overfitting and underfitting to this example as it is slightly on confusing side.

It is not mandatory that when we take K=1, the nearest neighbor is always an outlier. There are chances for that point to be an outlier sometimes. Hence when we are working with K-NN, it is always recommended to remove the outliers explicitly before building a model.

A model is said to overfit if the train error is low and CV error is high.  
A model is said to underfit if both the train error and CV error are high.  
A model is said to be a best fit if both the train error and CV error are reasonablly low.