Supervised: Data which is labelled , ie we have a forecasting attribute

Feature-> Input (Independent variable) ,

Label-> column used as predicted output(dependent variable)

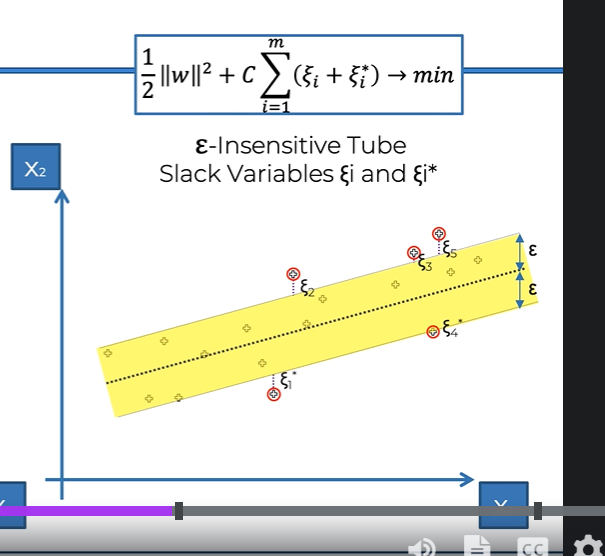
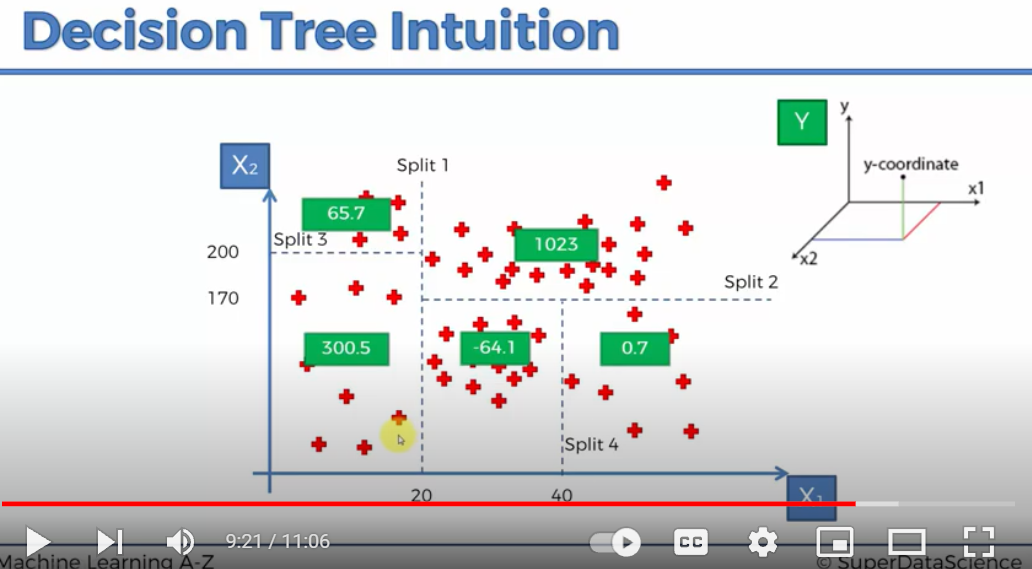
Steps in **preprocessing**->

* Import libraries,
* Import the dataset,
* take care of missing data: Mean, median, dropping data, linear interpolation , most frequent, constant
* Encoding Categorial Data: Label encoding if there is an order (0 for aa and 1 for bb,2 for cc ) and OneHotEncoding if there is no order ( 0 0 1 for A 010 for B and 001 for C)
* Feature Scaling: Standardizing or normalizing the data so different columns are “in the same scale” Standardization: x=x-mean(x)/SD(x) Normalization =x-min(x)/(max(x)-min(x)), Standardization always works

When to apply feature scaling?Ans: When there is no explicit equation between y and x

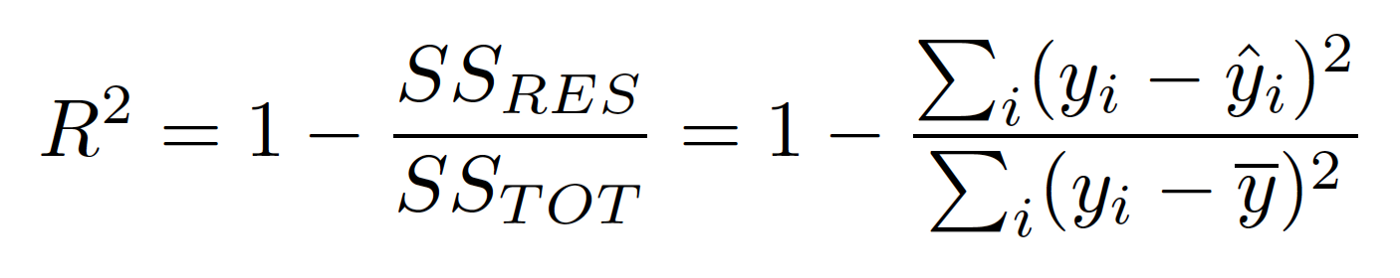
**Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.**

Regression

1. Simple Linear Regression- > y=mx+b where x is the is the feature and y the output.Single feature used to fit a line through the output
2. Multiple Regression-> y=mx1+m2x2+m3x3, where x1, x2, x3 are the 3 features to predict single output y
3. Polynomial Regression-> y=mx1+m2x1^2+m3x1^3….. with multiple powers of same feature
4. SVR-> A tube of width epsilon ε within which if the points lie, the error is not considered. If points lie outside (slack variables) or support vectors (vectors that dictate how the tube is made). Error is calculated using distance between support vectors and the tube
5. Decision Trees: Split is made in the dependent variables say X1 and X2 based on several conditions like X1>20, X2>50 based on information entropy or when there is a certain minimum number of leaves in that split. A yes no based (if this… then or if that… then) decision tree is made based on these conditions. Once that group of leaves found (terminal leaves), take average of y for all of the points in the terminal leaves to find the output. 
6. Random forest Regressor: Ensemble learning-> Performing the same algorithm multiple times to make much more powerful model. Choose at random K datapoints -> Make a decision tree based on the k datapoints -> Repeat first 2 steps for N number of trees-> Take the average of the output of each tree

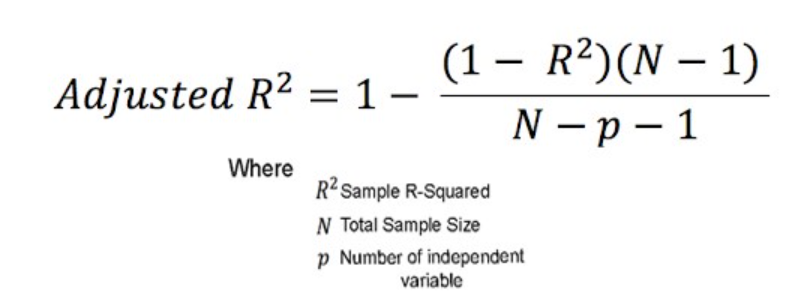
R squared

How good the Residual Sum of squares or the fitted data (datapoint-fitted) is compared to the Total sum of squared (datapoint – average) ( Average line). Higher the SSres, more the R squareds

How much better is the fitted data to the average of data, 

Adjusted R squared:

If new variables added , at the worst, R squared will remain the same or increase. Because there is always some correlation between the extra variable and the output variable, ie R2 is always biased.

R squared-> penalization factor for variables that do not help the E squared to increased.\ 

So if an extra independent variable is added, There is a penalty ass N-p-1 decreases and adj R^2 decreases. But 1-R^2 decreases so adjust R^2 increases. Hence the playoff

Classification:

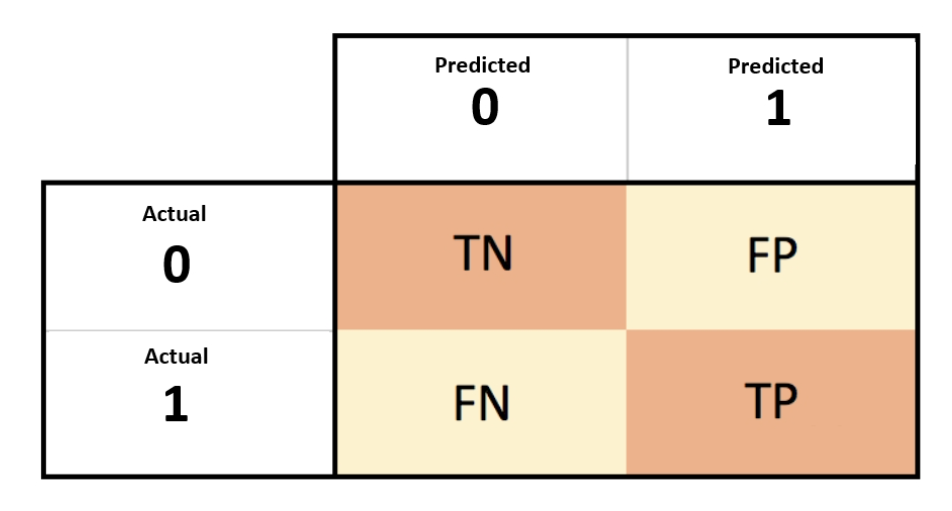
Confusion Matrix: Metrics used to judge the performance of a classification algorithm

False positive: Type 1

False negative: Type 2

Accuracy=(TN+TP)/(TN+TP+FN+FP)

Accuracy Paradox: Suppose we predict all as 0, could be that accuracy increases

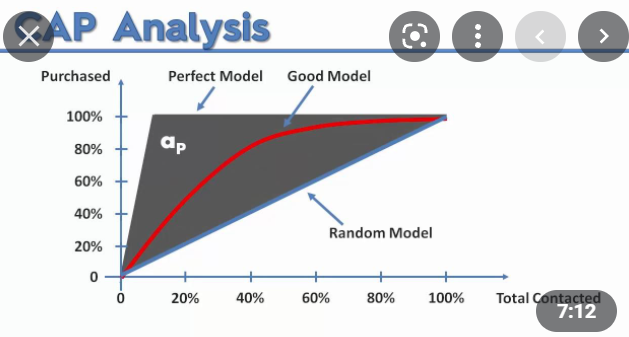
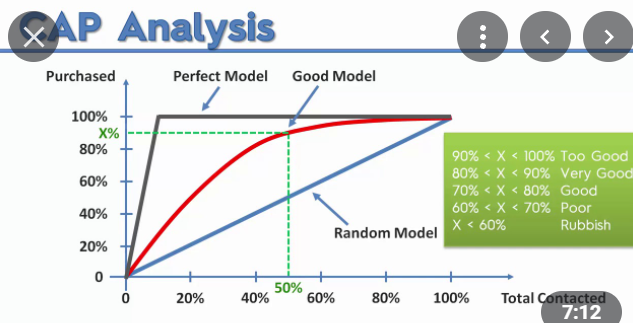


Accuracy=(TN+TP)/(TN+TP+FN+FP)

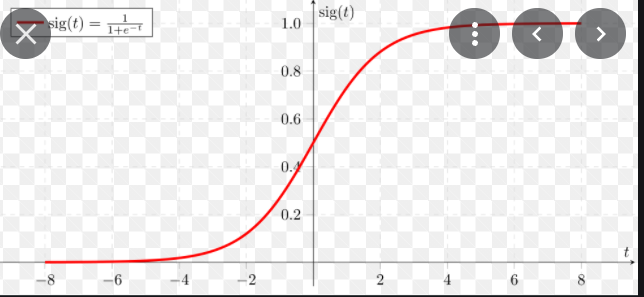
CAP analysis: Cumulative Accuracy profile

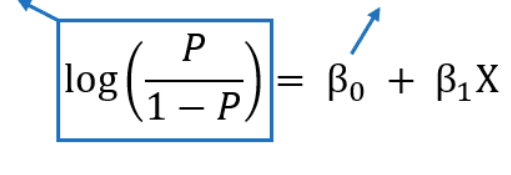
Purchased capped at 100% (10% of total contacted). If a good model is used, we contact people based on a certain algorithm : classifier if they should be contacted or not, then we get a differrnt curve. Greater the area between random model line and model, better the model.

Perfect model is when we contact first 10% of people and all of them purchase



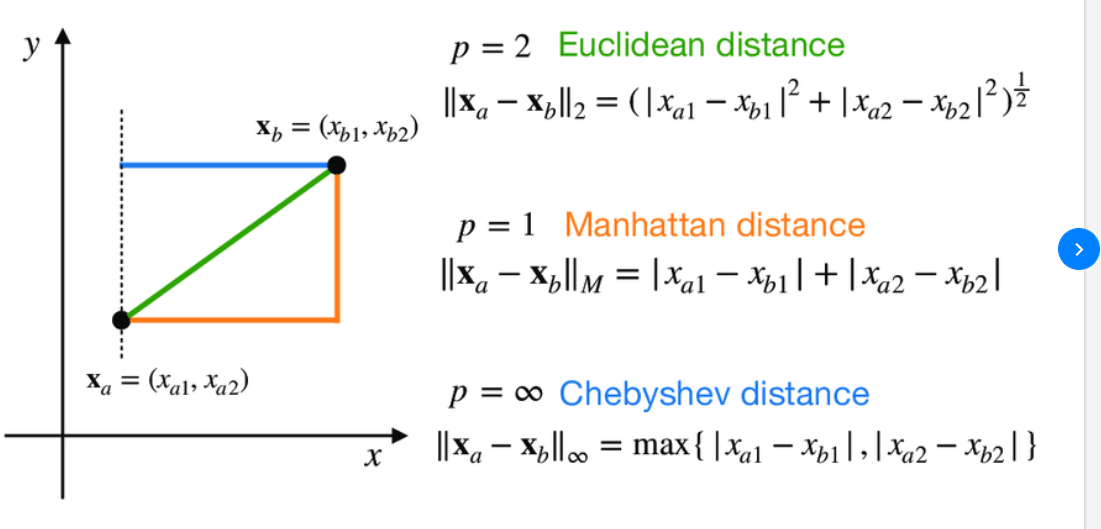
1)Logistic Regression: S shaped function (Sigmoid function) used to predict probability of event happening based on independent variable x. Probability higher than 0.5 is said to be yes and lesser is said to be no. Cuts it linearly





2. KNN (K nearest neighbors): cuts it non linearly

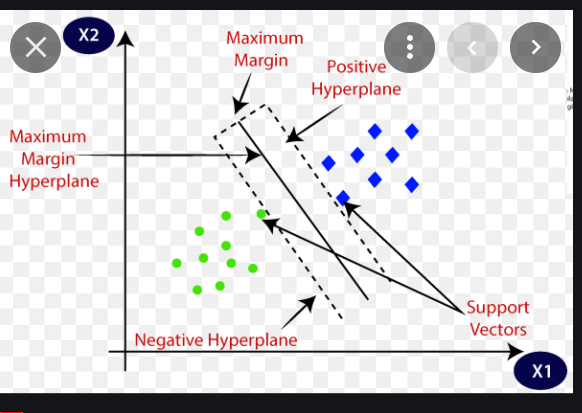
* Choose K neighbors to be chosen
* Take the K nearest points based on Euclidian or Manhattan distances
* Among K neighbors, count the number of points in each category
* Choose the category with most number of points

Minowski: Generalized distance in normed vector space (N dimensional real space) and is generalized for both Euclidian and Manhattan distance

3. SVM: cuts it linearly

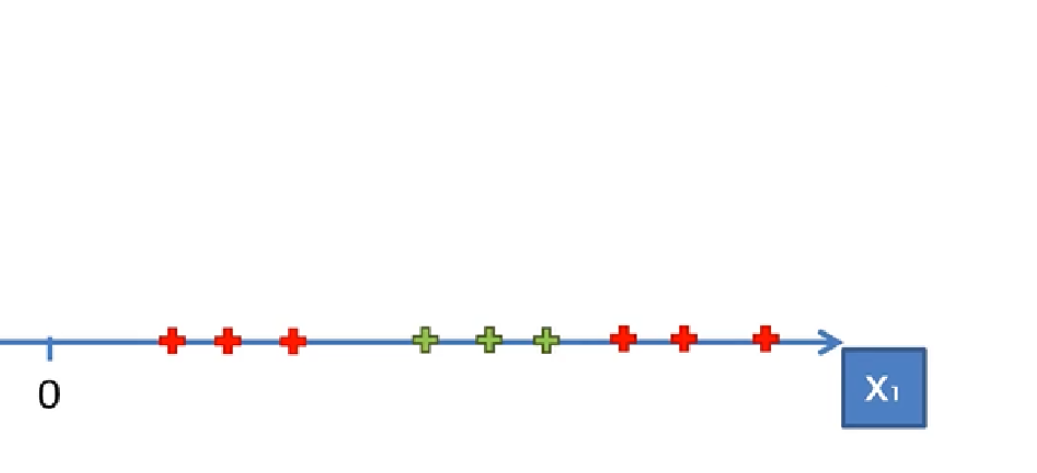
Line (Max Margin Hyperplane) that separates the two classes with the maximum margin ( maximum distance between 2 extreme points , (Support vectors: support the decision boundary)

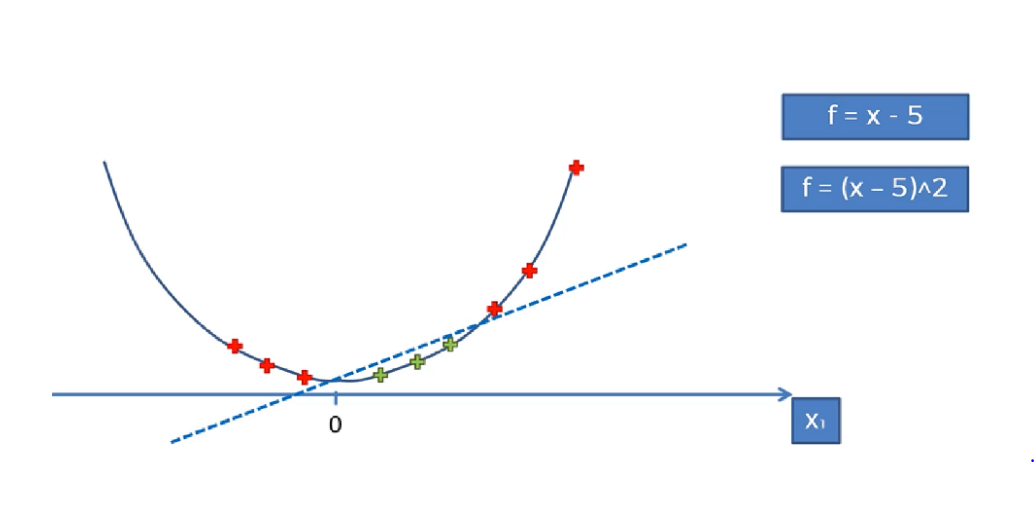
Benefit: look at point in class 1 most looking like class 2 and vice versa : looks at extreme boundary case( for example apples and oranges)



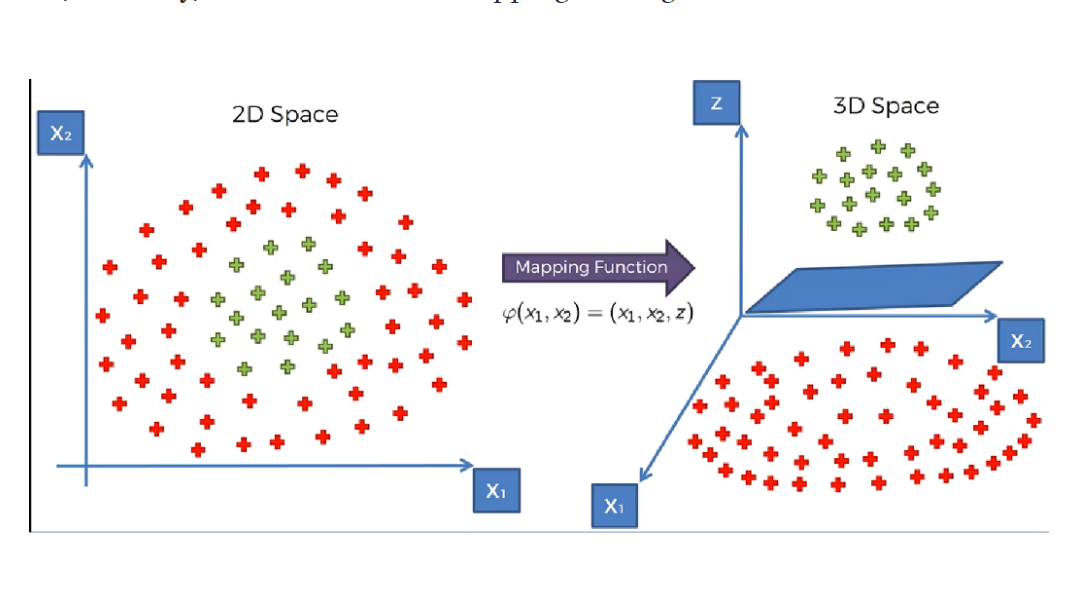
4. SVM Kernel: cuts it non linearly

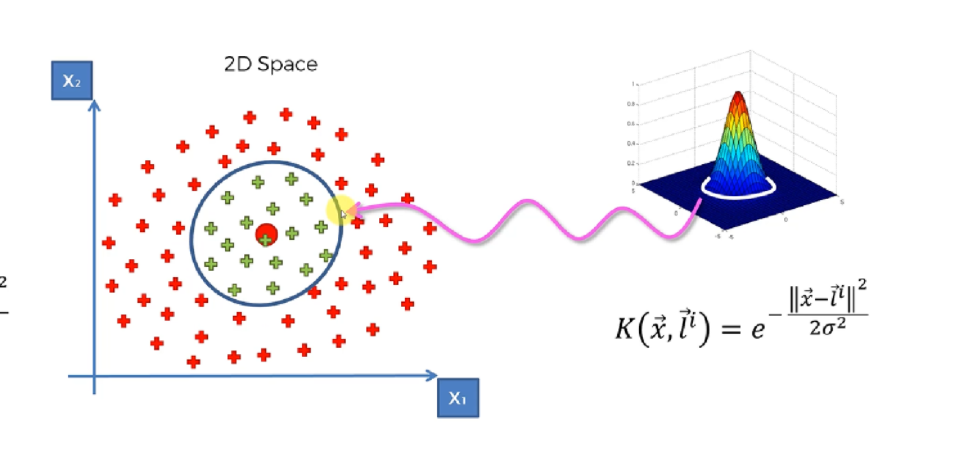
For non linearly separable data:

Example Straight line points, do x-5 and then (x-5)^2 to have linearly separable data



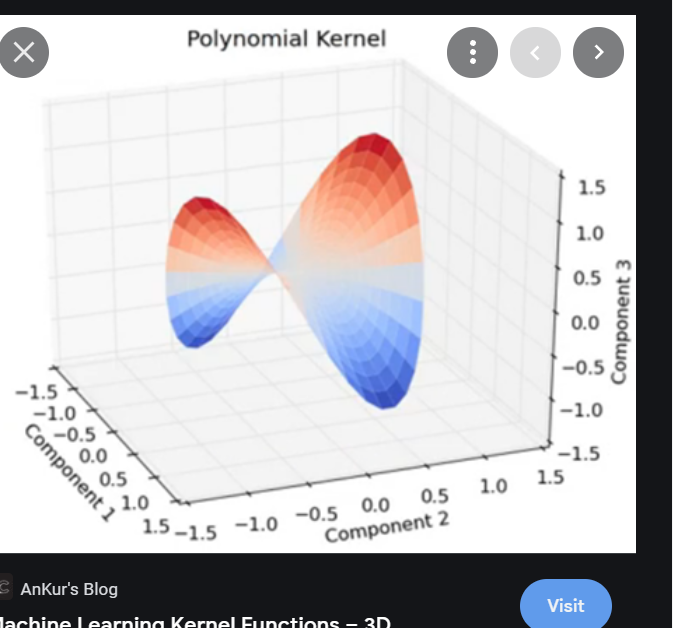
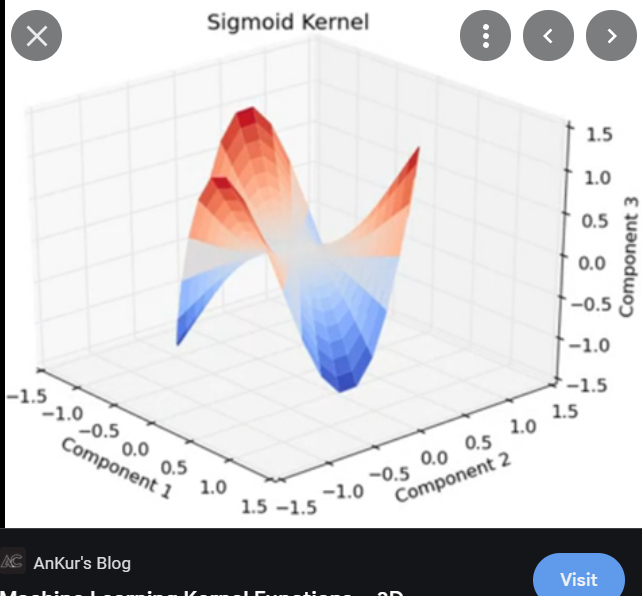
For 2D, fit a radial bound function and have hyperplane separating the data (circle in 2d)



Radial basis/Bound function: gaussian function:Kernel trick

Here, li is defined to be the center and as we move away from the center, (x is far away from li), we get e^-large number, which is 0. Contrarily close to landmark, we have x-li to be close to 0 and e^-small number is ~1. Sigma is directly proportional to diameter of the radial bound part.

Other Kernels:

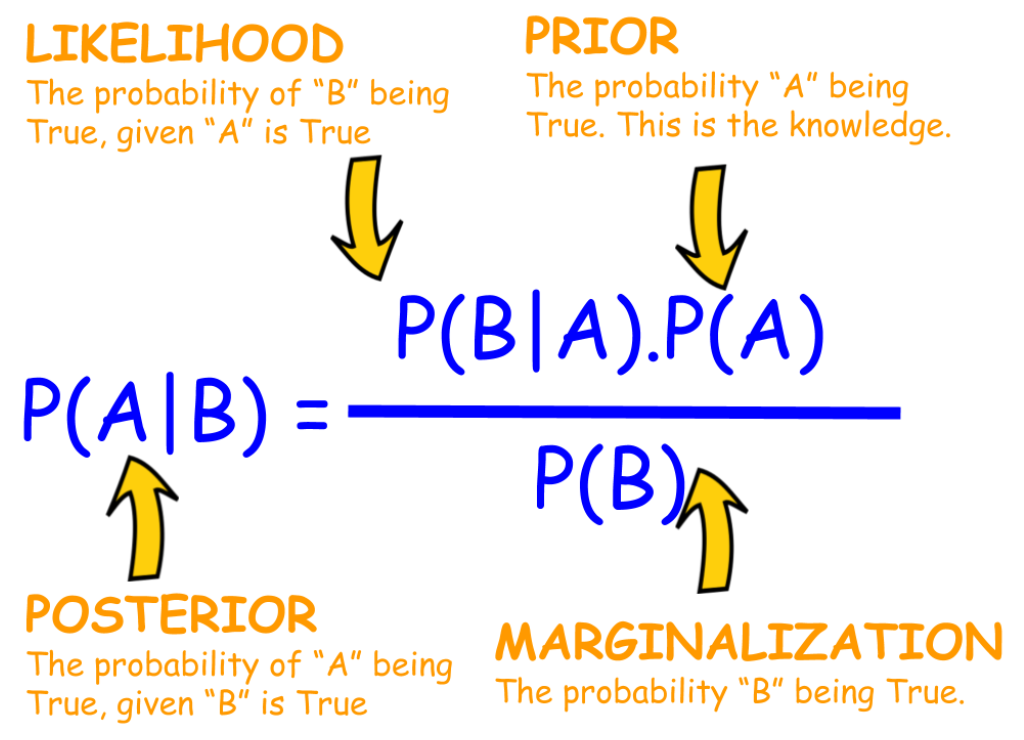
Polynomial Kernel and Sigmoid kernel 

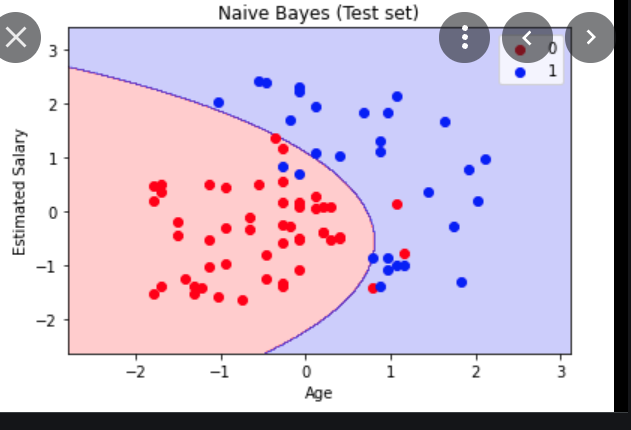
Non Linear SVR

Fit RBF to data, draw hyperplane to fit the data with a tube above and below, then backtrace to 2dimensions to get “middle finger / U shaped” type segregation, but it is a tube

5. Naïve Bayes: non linear split

Bayes Theorem:





Prob Red label|feature x and prob label blue|feature x, compare them. Here, x refers to a point with with features x

Prob Red= Red/(Red+Blue)

Prob x= number of points in small Circle around x/ (Red+Blue)

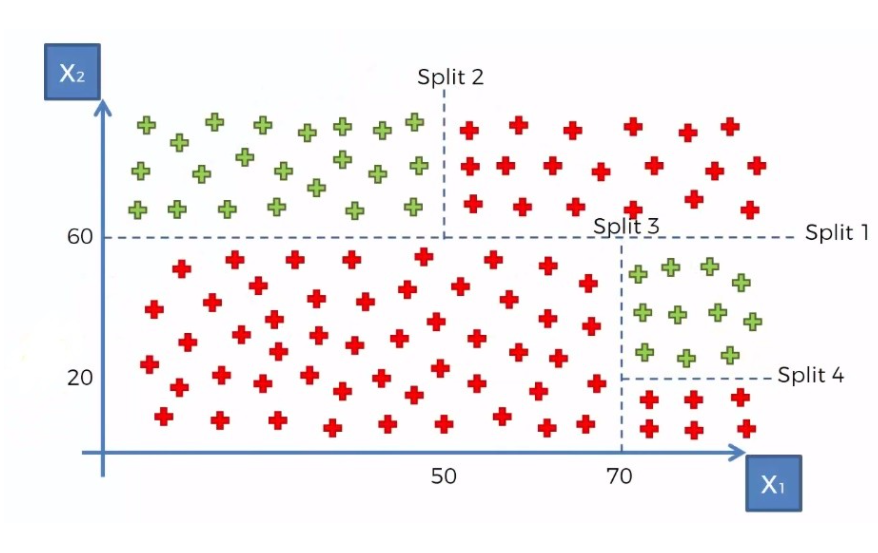
Prob x|Red= number of red points in small circle around x/ Red

Same with Blue and then compare

Why naïve?

Assumes that features are independent of each other whereas in reality it is not.

1. Decision Tree Classification: many boxes like classification non linear split

Makes splits like X1 greater than A or X2 smaller than B etc in order to minimize information entropy that is tries to get maximum number of a certain label in that split. Then classifies data based on those splits by making a decision tree for the same. Terminal leaves lead to the classification. If at the end there is a box for classification with a mix, a probabilistic choice is made

1. Random Forest Classifier: many boxes like classification non linear split
2. Pick at random K data points
3. Built a Decision tree based on K points
4. Choose N trees and repeat steps 1 and 2
5. For a new datapoint, make each tree predict category and choose category based on majority vote

Unlabelled data: no output : Unsupervised learning

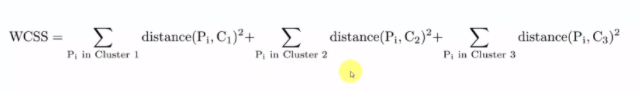
Clustering:

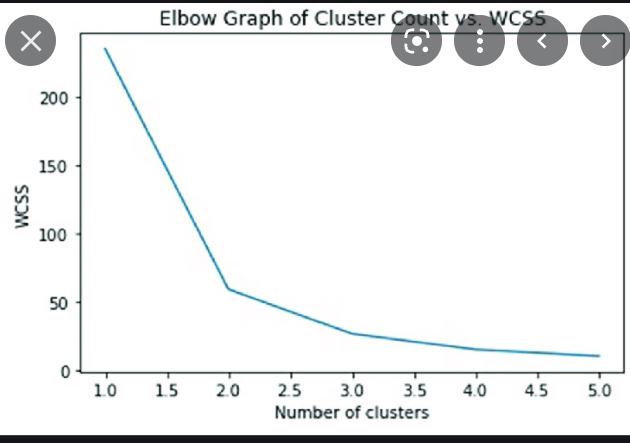
1. K means clustering

* Choose K clusters
* Choose K centroids randomly
* Assign each data point to the closest centroid , giving k clusters
* Find the new centroid of each of the k centroids based on center of gravity of data points
* Reassign points to the correct closest centroid. If a reassignment is made, move back to step 4. Otherwise we have correct K clusters

To deal with initialization trap, we use K means++ algorithm

How to find the number of clusters we should choose?

1. Start increasing K (number of clusters) from 1 to 2 to 3…..
2. Find WCSS which is within cluster sum of squares which is the sum of squares of the distance between each data point to the centroid of each cluster
3. Use elbow method ( ie point where the WCSS elbows and doesn’t decrease substantially after that specific number of clusters). Choose the elbow point as number of clusters to be chosen.

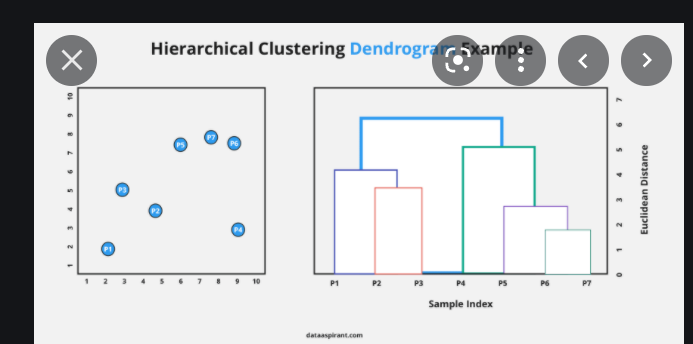


1. Heirarchial clustering

Agglomerative: bottom up

Divisive: top down

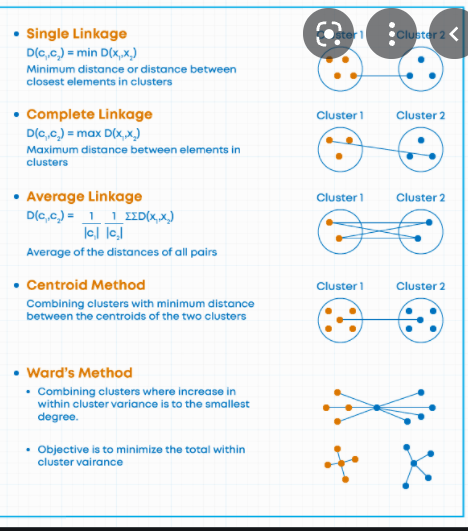
Steps:

1. Consider all datapoints as clusters, making n (n datapoints) clusters
2. Find the closest 2 datapoints and make them one cluster, forming n-1 clusters
3. Find closest 2 clusters and make them one cluster, forming n-2 clusters
4. Repeat step 3 till there is only 1 big cluster
5. For the steps above draw a dendogram:

Dendogram combines the two combined clusters and the height refers to the distance or dissimilarity between clusters

1. Then take the largest vertical line which doesn’t get crossed by a horizontal line. Cross the vertical line (half of it) with a horizontal line and how many ever vertical lines meet that horizontal line, those many clusters are formed. The combinations can be found by the combinations in the dendograms. This is to set a maximum distance beyond which we wont form clusters

Different Linkages:



Association Rule learning ( Recommendation type: If this then that OR association between 2 objects from a big list based on confidence and lift)

1. Apriori

I=items

Support(I)= containing item I/Total items

Confidence I1->I2 (That person who likes item 1 also likes item 2)= containing I1 and I2/containing I1

Lift I1->I2 (That person who likes item 1 also likes item 2) = Confidence(I1->I2)/Support(I2)

1. Choose minimum support that transactions higher than that minimum support will be considered
2. Choose minimum Confidence that transactions higher than that minimum support will be considered
3. Arrange in decreasing order of lift for association
4. Eclat

I = items

Support (I1 and I2)= containing items I1 and I2/total items

Choose minimum support that transactions higher than that minimum support will be considered and Arrange in Decreasing order of support