UNIT-II Supervised Machine Learning Algorithms

Machine Learning?

Two definitions of Machine Learning are offered.

Arthur Samuel described it as:	**Tom Mitchell provides a more modern definition**
"The field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.	: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Types of Machine Learning (ML)

Machine Learning Algorithms helps computer system learn without being explicitly programmed. These algorithms are categorized into supervised or unsupervised. Let us now see a few algorithms −

Supervised machine learning algorithms

This is the most commonly used machine learning algorithm. It is called supervised because the process of algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. In this kind of ML algorithm, the possible outcomes are

already known and training data is also labeled with correct answers. It can be understood as follows −

Suppose we have input variables **x** and an output variable **y** and we applied an algorithm to learn the mapping function from the input to output such as −

$$Y = f(x)$$

Now, the main goal is to approximate the mapping function so well that when we have new input data (x), we can predict the output variable (Y) for that data.

Mainly supervised leaning problems can be divided into the following two kinds of problems −

- **Classification** − A problem is called classification problem when we have the categorized output such as “black”, “teaching”, “non-teaching”, etc.
- **Regression** − A problem is called regression problem when we have the real value output such as “distance”, “kilogram”, etc.

Decision tree, random forest, knn, logistic regression are the examples of supervised machine learning algorithms.

Unsupervised machine learning algorithms

As the name suggests, these kinds of machine learning algorithms do not have any supervisor to provide any sort of guidance. That is why unsupervised machine learning algorithms are closely aligned with what some call true artificial intelligence. It can be understood as follows −

Suppose we have input variable x, then there will be no corresponding output variables as there is in supervised learning algorithms.

In simple words, we can say that in unsupervised learning there will be no correct answer and no teacher for the guidance. Algorithms help to discover interesting patterns in data.

Unsupervised learning problems can be divided into the following two kinds of problem −

- **Clustering** − In clustering problems, we need to discover the inherent groupings in the data. For example, grouping customers by their purchasing behavior.

- **Association** − A problem is called association problem because such kinds of problem require discovering the rules that describe large portions of our data. For example, finding the customers who buy both **x** and **y**.

K-means for clustering, Apriori algorithm for association are the examples of unsupervised machine learning algorithms.

Reinforcement machine learning algorithms

These kinds of machine learning algorithms are used very less. These algorithms train the systems to make specific decisions. Basically, the machine is exposed to an environment where it trains itself continually using the trial and error method. These algorithms learn from past experience and try to capture the best possible knowledge to make accurate decisions. Markov Decision Process is an example of reinforcement machine learning algorithms.

Example of Machine Learning Algorithms:

1. Linear Regression
2. Logistic Regression
3. Decision Tree
4. Support Vector Machine (SVM)
5. Naïve Bayes
6. K-nearest neighbor(KNN)
7. K-mean clustering
8. Random Forest

1. **Linear Regression**

It is one of the most well-known algorithms in statistics and machine learning.

Basic concept − mainly linear regression is a linear model that assumes a linear relationship between the input variables say x and the single output variable say y. In other words, we can say that y can be calculated from a linear combination of the input variables x. The relationship between variables can be established by fitting a best line.

Types of Linear Regression

Linear regression is of the following two types −

- **Simple linear regression** − A linear regression algorithm is called simple linear regression if it is having only one independent variable.
- **Multiple linear regression** − A linear regression algorithm is called multiple linear regression if it is having more than one independent variable.

Linear regression is mainly used to estimate the real values based on continuous variable(s). For example, the total sale of a shop in a day, based on real values, can be estimated by linear regression.

2. **Logistic Regression**

It is a classification algorithm and also known as **logit** regression.

Mainly logistic regression is a classification algorithm that is used to estimate the discrete values like 0 or 1, true or false, yes or no based on a given set of independent variable. Basically, it predicts the probability hence its output lies in between 0 and 1.

3. NAIVE BAYES CLASSIFIER

1. Introduction

Naive Bayes is a probabilistic machine learning algorithm that can be used in a wide variety of classification tasks. Typical applications include filtering spam, classifying documents, sentiment prediction etc

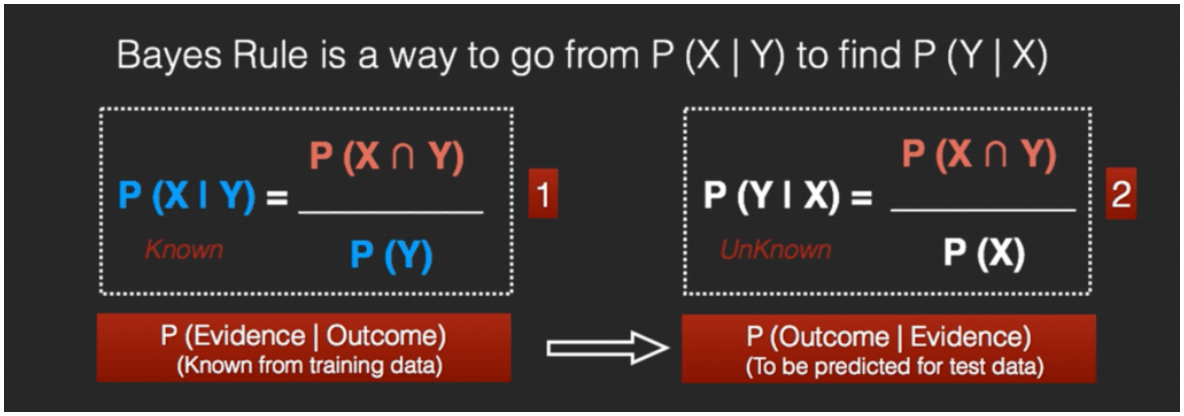
The name **naive** is used because it assumes the features that go into the model are independent of each other. That is changing the value of one feature, does not directly influence or change the value of any of the other features used in the algorithm.

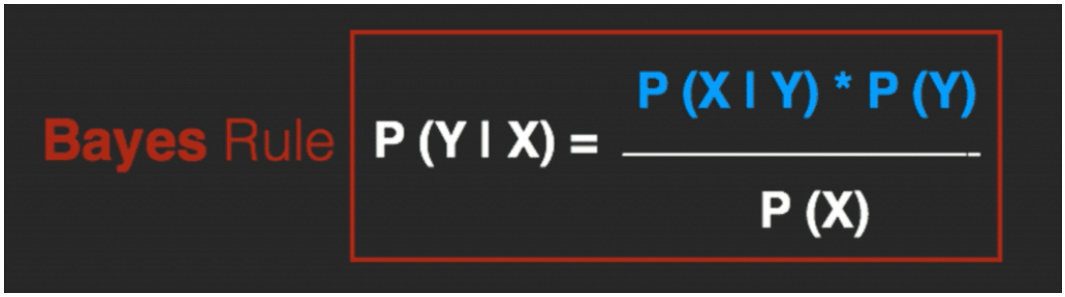
2. The Bayes Rule

The Bayes Rule is a way of going from P(X|Y), known from the training dataset, to find P(Y|X).

To do this, we replace A and B in the above formula, with the feature X and response Y.

For observations in test or scoring data, the X would be known while Y is unknown. And for each row of the test dataset, compute the probability of Y given the X has already happened.



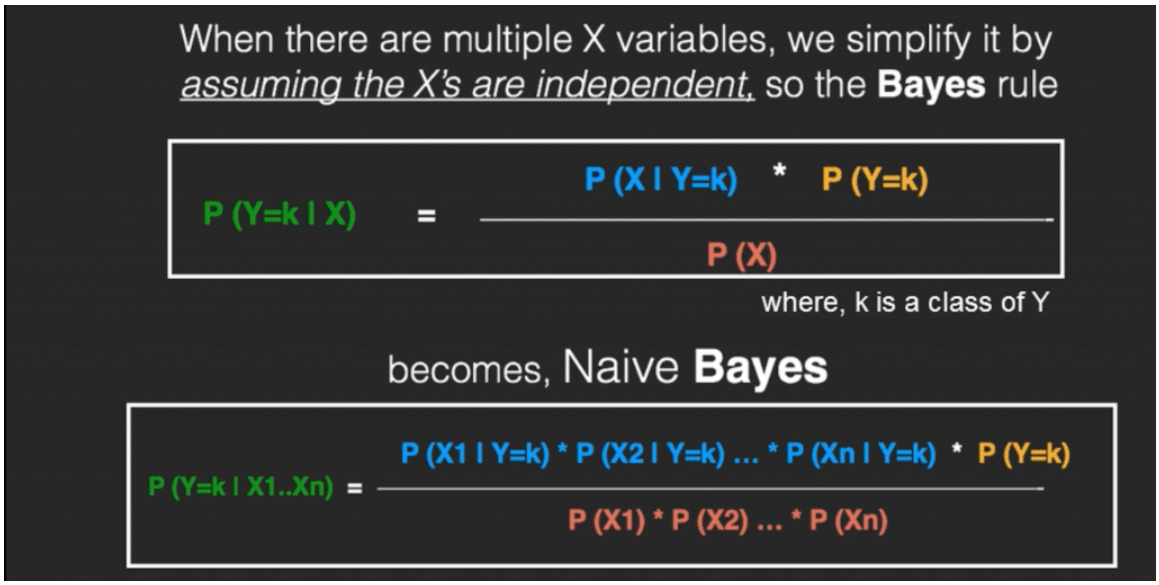


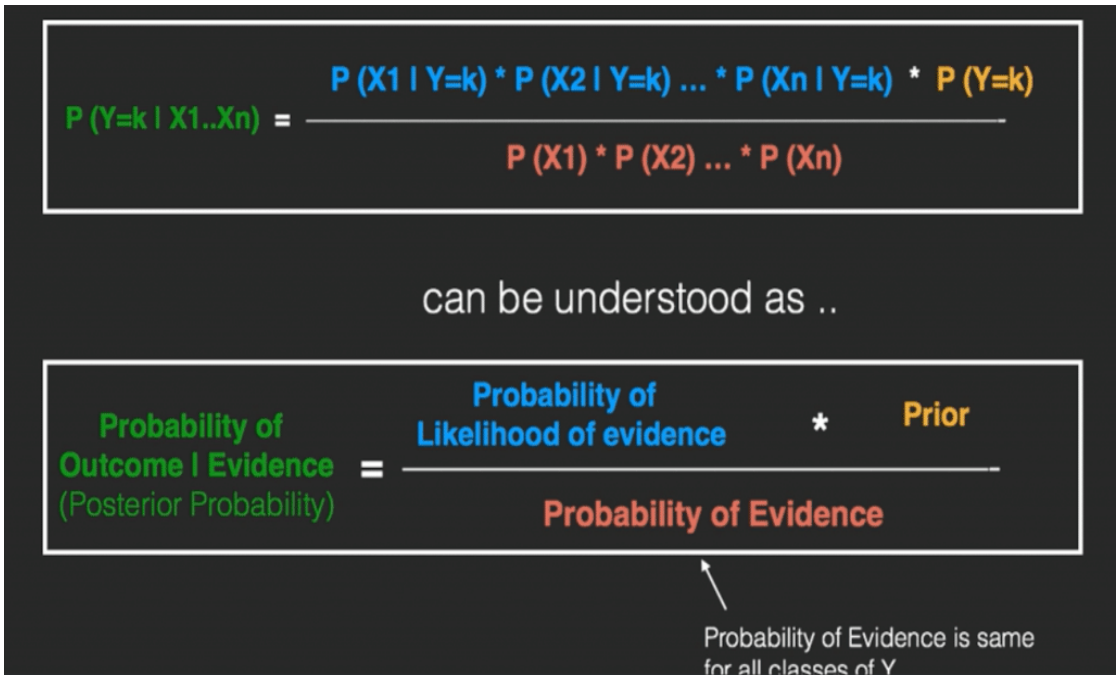
3. The Naive Bayes

The Bayes Rule provides the formula for the probability of Y given X. But, in real-world problems, we typically have multiple X variables.

When the features are independent, we can extend the Bayes Rule to what is called Naive Bayes.

It is called **'Naive'** because of the naive assumption that the X's are independent of each other. Regardless of its name, it's a powerful formula.





In technical jargon, the left-hand-side (LHS) of the equation is understood as the posterior probability or simply the posterior

The RHS has 2 terms in the numerator.

The first term is called the **'Likelihood of Evidence'**. It is nothing but the conditional probability of each X's given Y is of particular class 'c'.

Since the X's entire are assumed to be independent of each other, we can just multiply the 'likelihoods' of all the X's and called it the 'Probability of likelihood of evidence'. This is known from the training dataset by filtering records where Y=c.

The second term is called the prior which is the overall probability of Y=c, where c is a class of Y. In simpler terms, Prior = count(Y=c) / n_Records.

Example

Let's say we have data on 1000 pieces of fruit. The fruit being a Banana, Orange or some other fruit and imagine we know 3 features of each fruit, whether it's long or not, sweet or not and yellow or not, as displayed in the table below.

Fruit	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	**500**	**650**	**800**	**1000**

So from the table we already know?

- 50% of the fruits are bananas
- 30% are oranges
- 20% are other fruits

Based on our training set we can also say the following:

- From 500 bananas 400 (0.8) are Long, 350 (0.7) are Sweet and 450 (0.9) are Yellow
- Out of 300 oranges, 0 are Long, 150 (0.5) are Sweet and 300 (1) are Yellow
- From the remaining 200 fruits, 100 (0.5) are Long, 150 (0.75) are Sweet and 50 (0.25) are Yellow

Which should provide enough evidence to predict the class of another fruit as it's introduced.

So let's say we're given the features of a piece of fruit and we need to predict the class. If we're told that the additional fruit is Long, Sweet and yellow, we can classify it using the following formula

The one with the highest probability (score) being the winner.

Banana:

$$P(\frac{Banana}{Long, Sweet, Yellow}) = \frac{P(\frac{Long}{Banana}) \times P(\frac{Sweet}{Banana}) \times P(\frac{Yellow}{Banana}) \times P(Banana)}{P(Long)\, P(Sweet)\, P(Yellow)}$$

$$P(\frac{Banana}{Long, Sweet, Yellow}) = \frac{(0.8) \times (0.7) \times (0.9) \times (0.5)}{0.25 \times 0.33 \times 0.41}$$

$$P(\frac{Banana}{Long, Sweet, Yellow}) = 0.252$$

Orange:

$$P(\frac{Orange}{Long, Sweet, Yellow}) = 0$$

Other Fruit:

$$P(\frac{Other}{Long, Sweet, Yellow}) = \frac{P(\frac{Long}{Other}) \times P(\frac{Sweet}{Other}) \times P(\frac{Yellow}{Other}) \times P(Other)}{P(Long)\,P(Sweet)\,P(Yellow)}$$

$$P(\frac{Other}{Long, Sweet, Yellow}) = \frac{(0.5) \times (0.75) \times (0.25) \times (0.2)}{0.25 \times 0.33 \times 0.41}$$

$$P(\frac{Other}{Long, Sweet, Yellow}) = 0.01875$$

In this case, based on the higher score (0.252 for banana) we can assume this Long, Sweet and Yellow fruit is in fact, a Banana.

4. Decision Tree

Decision Trees are a class of very powerful Machine Learning model cable of achieving high accuracy in many tasks while being highly interpretable. What makes decision trees special in the realm of ML models is really their clarity of information representation. The “knowledge” learned by a decision tree through training is directly formulated into a hierarchical structure. This structure holds and displays the knowledge in such a way that it can easily be understood, even by non-experts.

Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

We can represent any boolean function on discrete attributes using the decision tree.

The most notable types of decision tree algorithms are:-

1. Iterative Dichotomiser 3 (ID3): This algorithm uses Information Gain to decide which attribute is to be used classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.

2. C4.5: This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.

3. **Classification and Regression Tree (CART):** It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

ID3 Algorithm:

ID3 is a greedy algorithm that grows the tree top-down, at each node selecting the attribute that best classifies the local training examples. This process continues until the tree perfectly classifies the training examples or until all attributes have been used.

1. **Entropy**

   It is a fundamental theorem which commonly used in information theory to measure important of information relative to its *size*. Let x is our training set contains positive and negative examples, then the entropy of x relative to this classification is:

$$H(x) = - p_+ \ log_2 \ p_+ - p_- \ log_2 \ p_-$$

2. **Information Gain**

   For training set x and its attribute y, the formula of Information Gain is:

$$G(x, y) = H(x) - \sum_{i \in value(y)} \frac{|\Delta y_i|}{|\Delta y|} H(y_i)$$

a. Entropy is 0 if all the members of S belong to the same class.
b. Entropy is 1 when the sample contains an equal number of positive and negative examples.
c. If the sample contains unequal number of positive and negative examples, entropy is between 0 and 1.

Example 1:

Data Set

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

$$H(S) = \sum_{c \in C} -p(c)\log_2 p(c)$$

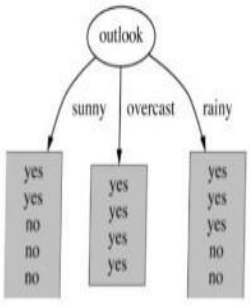
C ={yes,no}

Out of 14 instances, 9 are classified as yes, and 5 as no

pyes = −(9/14)*log2(9/14) = 0.41

pno = −(5/14)*log2(5/14) = 0.53

H (S) = pyes + pno = 0.94



E (Outlook=sunny) = $-\frac{2}{5}\log\left(\frac{2}{5}\right)-\frac{3}{5}\log\left(\frac{3}{5}\right) = 0.971$

E (Outlook=overcast) = $-1\log(1) - 0\log(0) = 0$

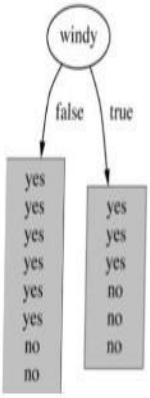
E (Outlook=rainy) = $-\frac{3}{5}\log\left(\frac{3}{5}\right)-\frac{2}{5}\log\left(\frac{2}{5}\right) = 0.971$

} H(S,Outlook)

Average Entropy information for Outlook

I (Outlook) = $\frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 = 0.693$ } $\sum_{t \in T} p(t)H(t)$

Gain (Outlook) = E(S) – I (outlook) = 0.94-.693 = 0.247 ⇨ $IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t)$



E (Windy=false) = $-\frac{6}{8}\log\left(\frac{6}{8}\right)-\frac{2}{8}\log\left(\frac{2}{8}\right) = 0.811$

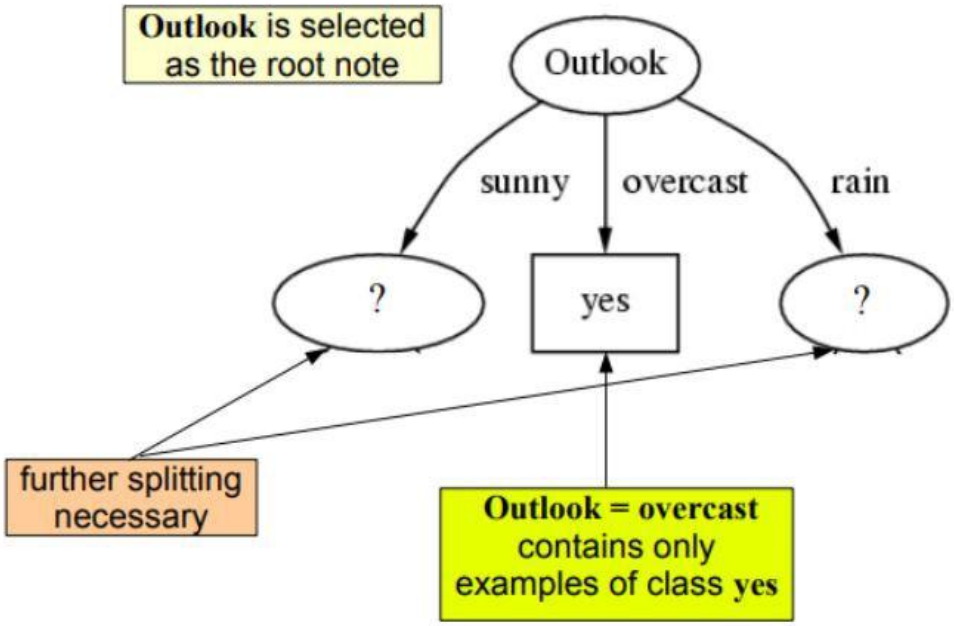
E (Windy=true) = $-\frac{3}{6}\log\left(\frac{3}{6}\right)-\frac{3}{6}\log\left(\frac{3}{6}\right) = 1$

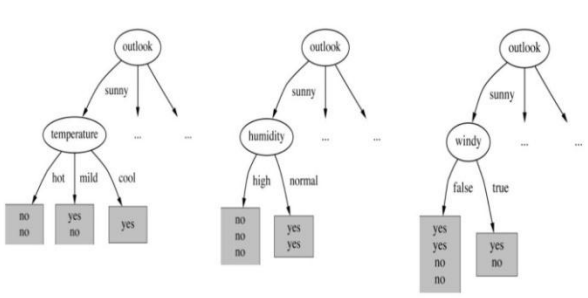
Average entropy information for Windy

I (Windy) = $\frac{8}{14} * 0.811 + \frac{6}{14} * 1 = 0.892$

Gain (Windy) = E(S) – *I* (Windy) = 0.94-0.892=0.048

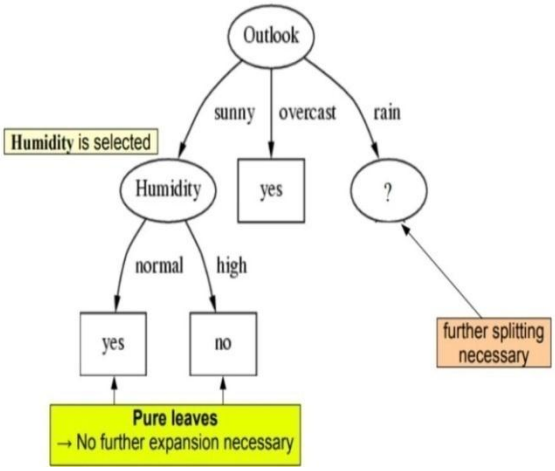
Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048



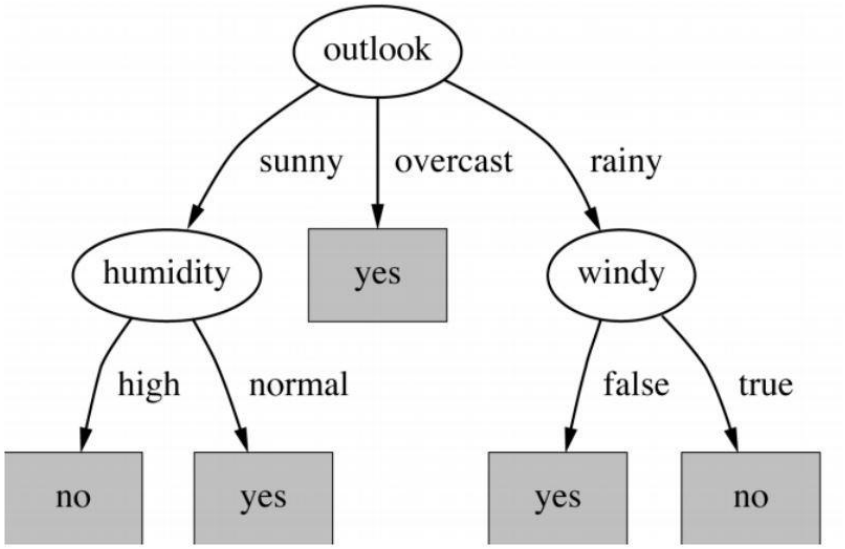


Gain(*Temperature*) = 0.571 bits
Gain(*Humidity*) = 0.971 bits
Gain(*Windy*) = 0.020 bits

Humidity is selected



Final decision tree



5. K NEAREST NEIGHBOR

Introduction to K-nearest neighbor classifier

K-nearest neighbor classifier is one of the introductory **supervised classifier**, which every data science learner should be aware of.

For simplicity, this classifier is called as **Knn Classifier**. To be surprised k-nearest neighbor classifier mostly represented as Knn, even in many research papers too. Knn address the pattern recognition problems and also the best choices for addressing some of the classification related tasks.

The simple version of the K-nearest neighbor classifier algorithms 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.

K-nearest neighbor classification step by step procedure

Before diving into the k-nearest neighbor, classification process lets' understand the application-oriented example where we can use the knn algorithm.

Knn Algorithm Pseudocode:

1. Calculate "$d(x, x_i)$" 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 $\mathbf{k}_i$ denotes the number of points belonging to the i^{th} class among **k** points i.e. $k \geq 0$
6. If $k_i > k_j \; \forall \; i \neq j$ then put x in class i.

K- Nearest neighbor algorithm example

Let's consider that we have two different target classes' **white and orange** circles. We have total 26 training samples. Now we would like to predict the target class for the **blue circle.** Considering k value as **three**, we need to calculate the similarity distance using similarity measures like Euclidean distance.

If the similarity score is less which means the classes are close. We have calculated distance and placed the less distance circles to blue circle inside the Big circle.

Let's consider a setup with "n" training samples, where x_i is the training data point. The training data points are categorized into "c" classes. Using KNN, we want to predict class for the new data point. So, the first step is to calculate the distance (Euclidean) between the new data point and all the training data points.

Next step is to arrange all the distances in non-decreasing order. Assuming a positive value of "K" and filtering "K" least values from the sorted list. Now, we have K top distances. Let k_i denotes no. of points belonging to the i^{th} class among k points. If $k_i > k_j \; \forall i \neq j$ then put x in class i.

Nearest Neighbor Algorithm:

Nearest neighbor is a special case of k-nearest neighbor class. Where k value is 1 (k = 1). In this case, new data point target class will be assigned to the 1st closest neighbor.

How to choose the value of K?

Selecting the value of **K** in K-nearest neighbor is the most critical problem. A small value of K means that noise will have a higher influence on the result i.e., the probability of over fitting is very high. A large value of K makes it computationally expensive and defeats the basic idea behind KNN (that points that are near might have similar classes). A simple approach to select k is k = n^(1/2).

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.

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

Most of the classification techniques can be classified into the following three groups:

1. Parametric
2. Semi parametric
3. Non-Parametric

Parametric & Semi parametric classifiers need specific information about the structure of data in training set. It is difficult to fulfil this requirement in many cases. So, **non-parametric classifier like KNN was considered.**

Advantages of K-nearest neighbors algorithm

- Knn is simple to implement.
- Knn executes quickly for small training data sets.
- Performance asymptotically approaches the performance of the Bayes Classifier.
- Don't need any prior knowledge about the structure of data in the training set.
- No retraining is required if the new training pattern is added to the existing training set.

Limitation to K-nearest neighbors algorithm

- When the training set is large, it may take a lot of space.
- For every test data, the distance should be computed between test data and all the training data. Thus a lot of time may be needed for the testing.

6. SUPPORT VECTOR MACHINE

Introduction to Support Vector Machines

A support vector machine (SVM) is a supervised learning technique from the field of machine learning applicable to both classification and regression. Rooted in the Statistical Learning Theory developed by **Vladimir Vapnik** and co-workers at AT&T Bell Laboratories in 1995, SVMs are based on the principle of **Structural Risk Minimization**

Support Vector Machines was worked out for linear two-class classification with margin, where margin means the minimal distance from the separating hyperplane to the closest data points. SVM learning machine seeks for an optimal separating hyperplane, where the margin is maximal. An important and unique feature of this approach is that the solution is based only on those data points, which are at the margin. **These points are called support vectors.**

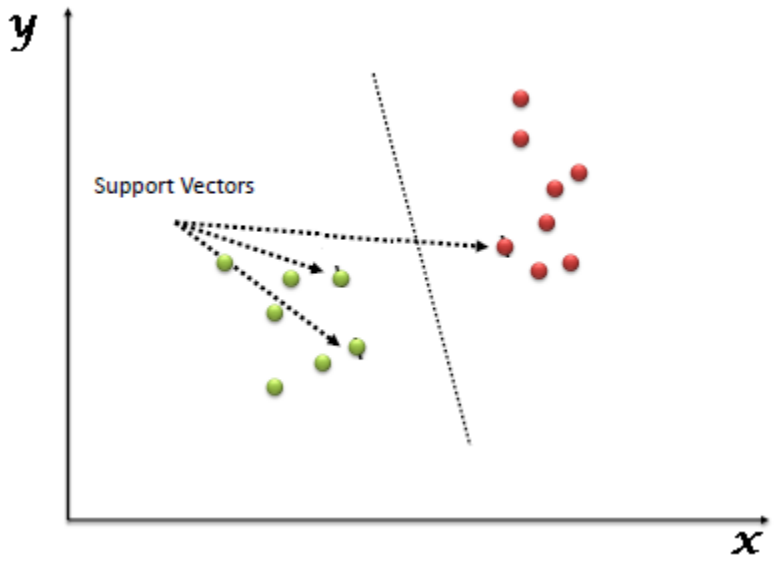


Figure 1

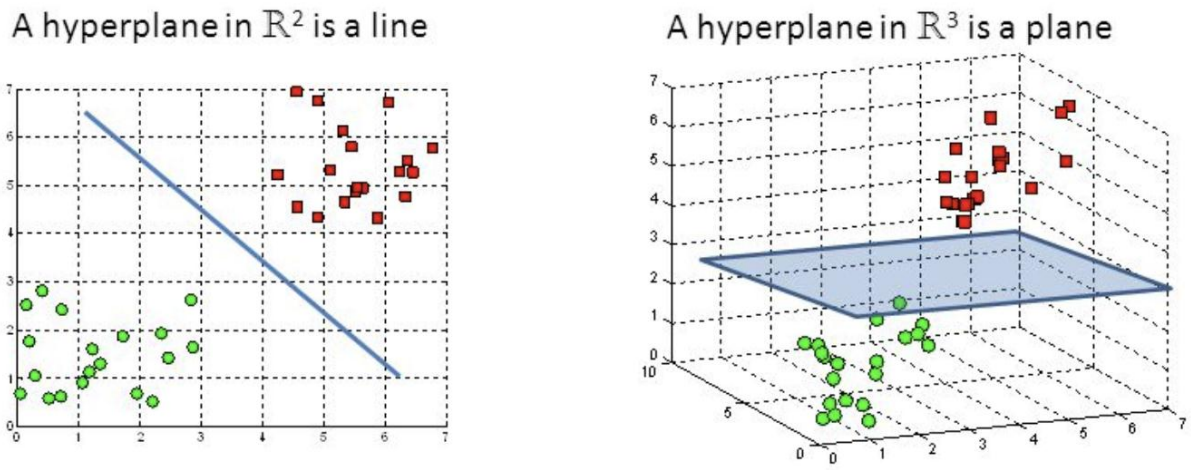
Support Vector machines (SVM) are a new statistical learning technique that can be seen as a new method for training classifiers based on **polynomial functions**, **radial basis functions**, **neural networks, spines or other functions**.

Support Vector machines use a hyper-linear separating plane to create a classifier. For problems that cannot be linearly separated in the input space, this machine offers a possibility to find a solution by making a non-linear transformation of the original input space into a high dimensional feature space, where an optimal separating hyperplane can be found.

SVM differs from the other classification algorithms in the way that it chooses the decision boundary that maximizes the distance from the nearest data points of all the classes. An SVM doesn't merely find a decision boundary; it finds the most optimal decision boundary.

The most optimal decision boundary is the one which has maximum margin from the nearest points of all the classes. The nearest points from the decision boundary that maximize the distance between the decision boundary and the points are called support vectors as seen in Figure 1. The decision boundary in case of support vector machines is called the maximum margin classifier, or the maximum margin hyper plane.

The main objective in SVM is to find the optimal hyperplane to correctly classify between data points of different classes (Figure 2). The hyperplane dimensionality is equal to the number of input features minus one (eg. when working with three feature the hyperplane will be a two-



dimensional plane).

Figure 2

Data points on one side of the hyperplane will be classified to a certain class while data points on the other side of the hyperplane will be classified to a different class (eg. green and red as in Figure 2). The distance between the hyperplane and the first point (for all the different classes) on either side of the hyperplane is a measure of sure the algorithm is about its classification decision. The bigger the distance and the more confident we can be SVM is making the right decision.

The data points closest to the hyperplane are called Support Vectors. Support Vectors determines the orientation and position of the hyperplane, in order to maximise the classifier margin (and therefore the classification score). The number of Support Vectors the SVM algorithm should use can be arbitrarily chosen depending on the applications.

There are two main types of classification SVM algorithms Hard Margin and Soft Margin:

- **Hard Margin:** aims to find the best hyperplane without tolerating any form of misclassification.
- **Soft Margin:** we add a degree of tolerance in SVM. In this way we allow the model to voluntary misclassify a few data points if that can lead to identifying a hyperplane able to generalise better to unseen data.

Soft Margin SVM can be implemented in Scikit-Learn by adding a C penalty term in svm.SVC. The bigger C and the more penalty the algorithm gets when making a misclassification.

Kernel Trick

If the data we are working with is not linearly separable (therefore leading to poor linear SVM classification results), it is possible to apply a technique known as the Kernel Trick. This method is able to map our non-linear separable data into a higher dimensional space, making our data linearly separable. Using this new dimensional space SVM can then be

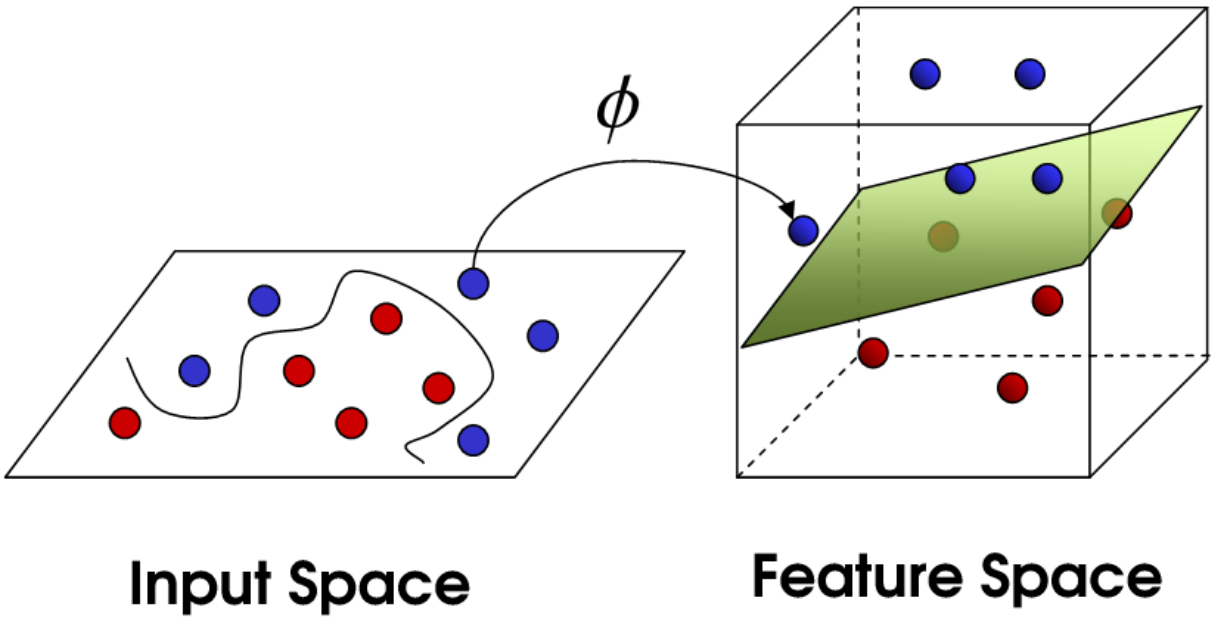


Figure 3

Easily implemented (as shown in Figure 3).

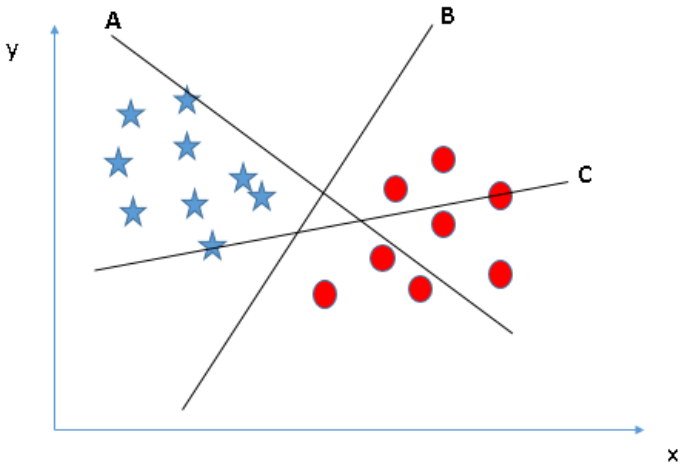
There are many different types of Kernels which can be used to create this higher dimensional space; some examples are linear, polynomial, Sigmoid and Radial Basis Function (RBF). In Scikit-Learn a Kernel function can be specified by adding a kernel parameter in svm.SVC. An additional parameter called gamma can be included to specify the influence of the kernel on the model.

Feature Selection

Reducing the number of features in Machine Learning plays a really important role especially when working with large datasets. This can in fact: speed up training, avoid overfitting and ultimately lead to better classification results thanks to the reduced noise in the data.

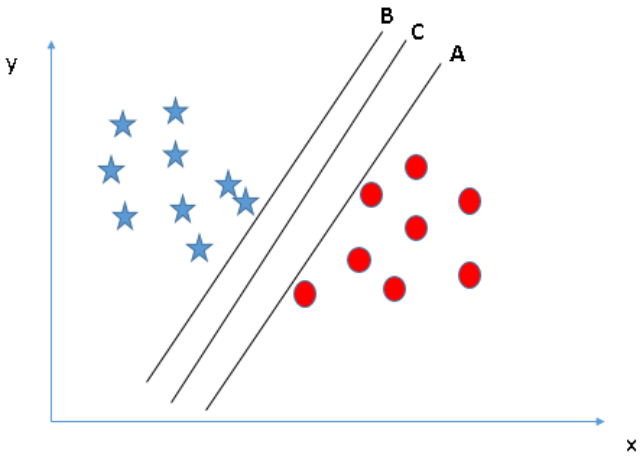
How does it work?

1. **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

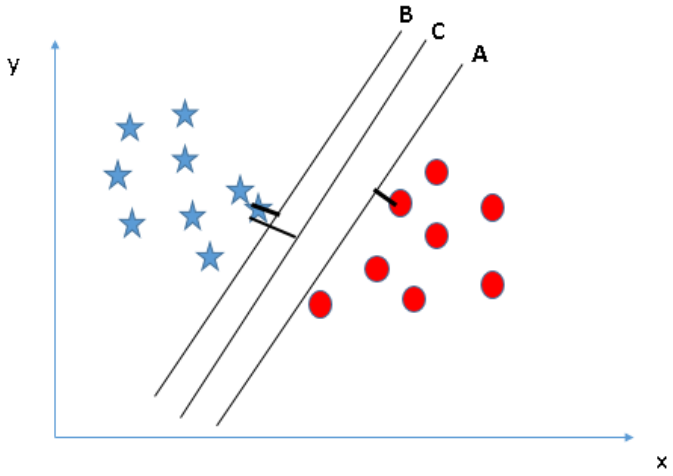


- We need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

2. **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

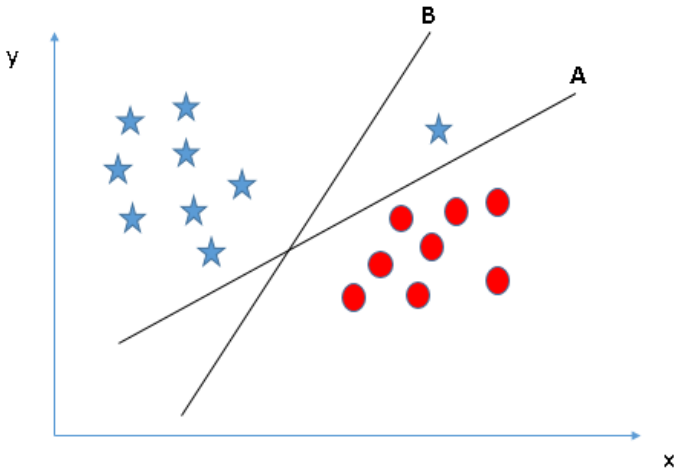


Here, maximizing the distances between nearest data points (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.



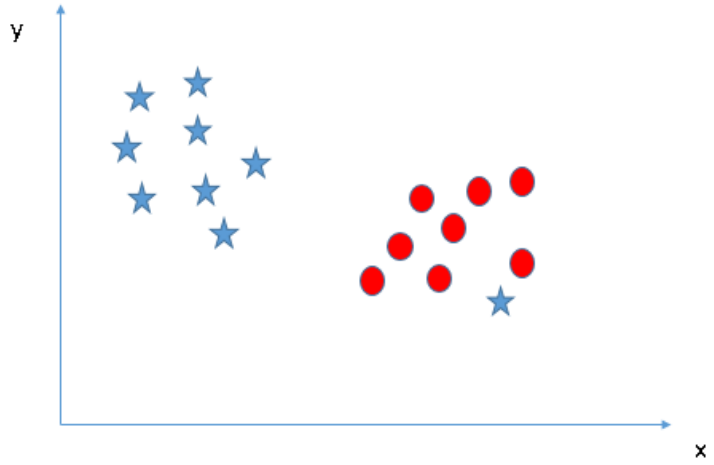
Above, we can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the **right hyper-plane as C**. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

3. Identify the right hyper-plane (Scenario-3):

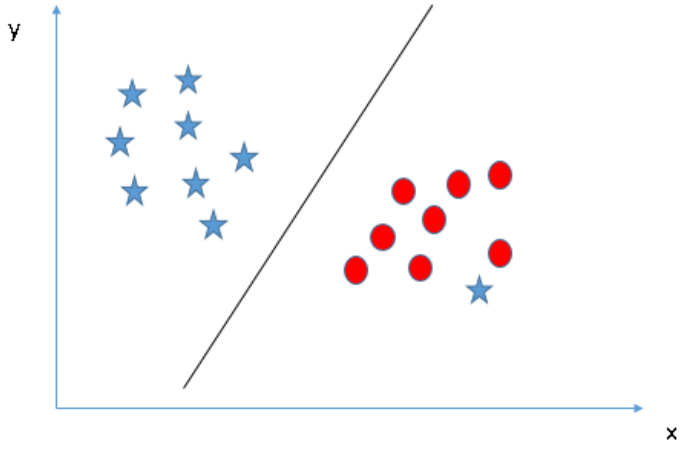


Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch; SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

4. **Can we classify two classes (Scenario-4)?: In the scenario below**

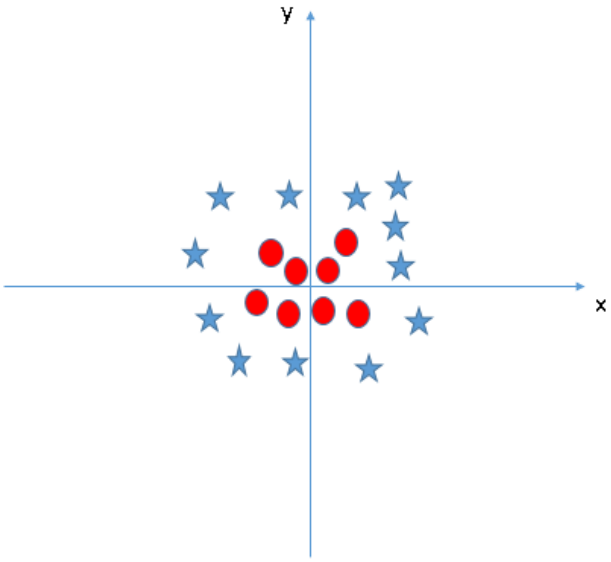


It is mentioned that, one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.

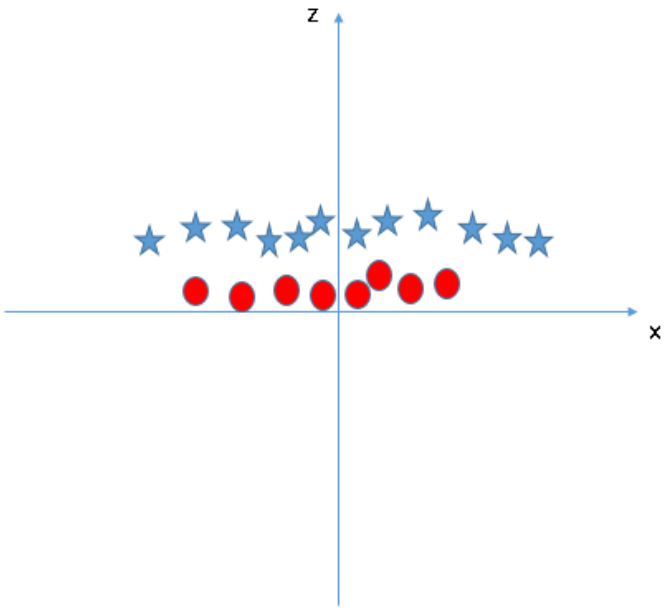


5. **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM

classify these two classes? Till now, we have only looked at the linear hyper-plane.



SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on



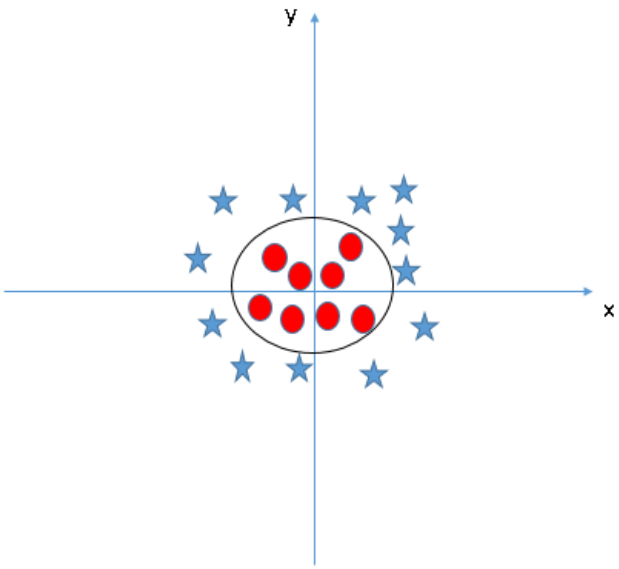
Axis x and z.

In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs we have defined.

When we look at the hyper-plane in original input space it looks like a circle:



Pros and Cons associated with SVM

- **Pros:**
  - It works really well with clear margin of separation

- It is effective in high dimensional spaces.
- It is effective in cases where number of dimensions is greater than the number of samples.
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

- **Cons:**
  - It doesn't perform well, when we have large data set because the required training time is higher
  - It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
  - SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

---

7. RANDOM FOREST ALGORITHM

Introduction

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an **ensemble** method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

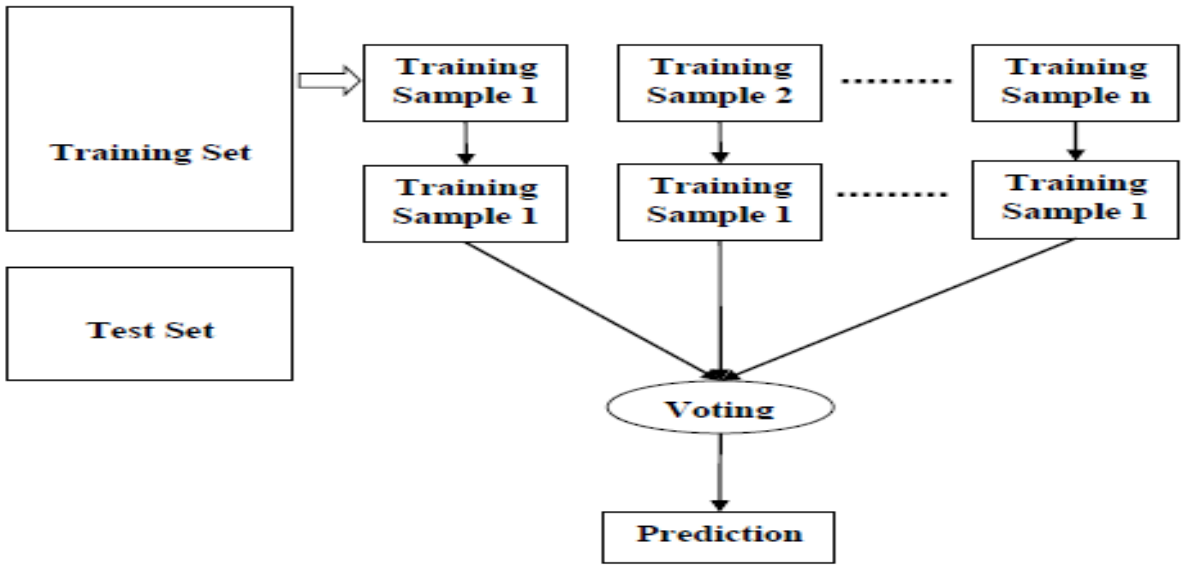
Working of Random Forest Algorithm

We can understand the working of Random Forest algorithm with the help of following steps −

- **Step 1** − First, start with the selection of random samples from a given dataset.

- **Step 2** − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** − In this step, voting will be performed for every predicted result.
- **Step 4** − At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working −



Overfitting

Overfitting is a practical problem while building a decision tree model. The model is having an issue of overfitting is considered when the algorithm continues to go deeper and deeper in the to reduce the training set error but results with an increased test set error i.e, Accuracy of prediction for our model goes down. It generally happens when it builds many branches due to outliers and irregularities in data.

Two approaches which we can use to avoid overfitting are:

- Pre-Pruning
- Post-Pruning

Pre-Pruning

In pre-pruning, it stops the tree construction bit early. It is preferred not to split a node if its goodness measure is below a threshold value. But it's difficult to choose an appropriate stopping point.

Post-Pruning

In post-pruning first, it goes deeper and deeper in the tree to build a complete tree. If the tree shows the overfitting problem then pruning is done as a post-pruning step. We use a cross-validation data to check the effect of our pruning. Using cross-validation data, it tests whether expanding a node will make an improvement or not.

If it shows an improvement, then we can continue by expanding that node. But if it shows a reduction in accuracy then it should not be expanded i.e, the node should be converted to a leaf node.

Decision Tree Algorithm Advantages and Disadvantages

Advantages:

1. Decision Trees are easy to explain. It results in a set of rules.
2. It follows the same approach as humans generally follow while making decisions.
3. Interpretation of a complex Decision Tree model can be simplified by its visualizations. Even a naive person can understand logic.
4. The Number of hyper-parameters to be tuned is almost null.

Disadvantages:

1. There is a high probability of overfitting in Decision Tree.
2. Generally, it gives low prediction accuracy for a dataset as compared to other machine learning algorithms.
3. Information gain in a decision tree with categorical variables gives a biased response for attributes with greater no. of categories.

4. Calculations can become complex when there are many class labels.

How Random Forest algorithm works?

There are two stages in Random Forest algorithm, one is random forest creation, the other is to make a prediction from the random forest classifier created in the first stage. The whole process is shown below, and it's easy to understand using the figure.

Here the author firstly shows the Random Forest creation pseudocode:

1. Randomly select "**K**" features from total "**m**" features where **k << m**
2. Among the "**K**" features, calculate the node "**d**" using the best split point
3. Split the node into **daughter nodes** using the **best split**
4. Repeat the **a to c** steps until "l" number of nodes has been reached
5. Build forest by repeating steps **a to d** for "n" number times to create **"n"**

Random Forest algorithm Application.

- For the application in banking, Random Forest algorithm is used to find loyal customers, who mean customers who can take out plenty of loans and pay interest to the bank properly, and fraud customers, which means customers who have bad records like failure to pay back a loan on time or have dangerous actions.
- For the application in medicine, Random Forest algorithm can be used to both identify the correct combination of components in medicine, and to identify diseases by analyzing the patient's medical records.
- For the application in the stock market, Random Forest algorithm can be used to identify a stock's behaviour and the expected loss or profit.

- For the application in e-commerce, Random Forest algorithm can be used for predicting whether the customer will like the recommend products, based on the experience of similar customers.

Advantages of Random Forest algorithm.

Compared with other classification techniques, there are three advantages as the author mentioned.

1. For applications in classification problems, Random Forest algorithm will avoid the overfitting problem.
2. For classification and regression task, the same random forest algorithm can be used.