

Data Analytics for Better Business Insights

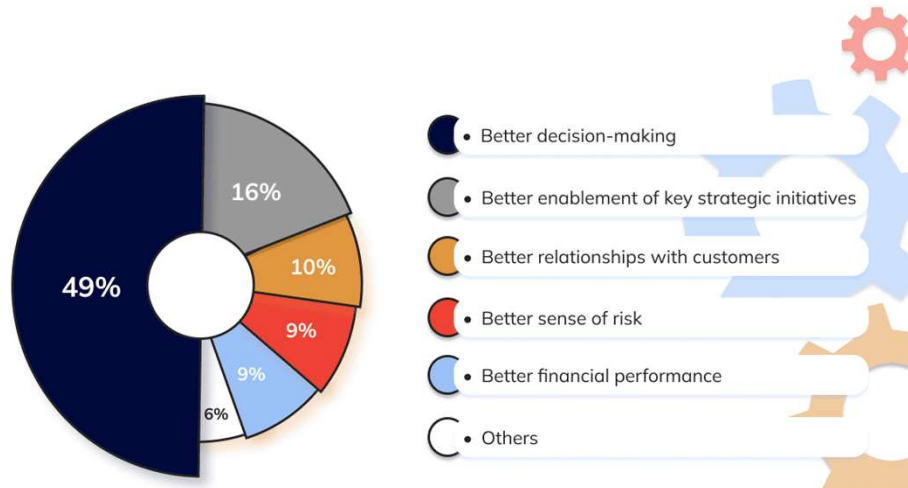


Dr. Foni Agus Setiawan, M.Kom.
masagus@masagus.id

Data Analytics

- Data analytics is the **collection**, **transformation**, and **organization** of data in order to draw conclusions, make predictions, and drive informed decision making.
- Data analytics is a multidisciplinary field that employs a wide range of analysis techniques, including math, statistics, and computer science, to **draw insights from data sets**.
- Data analytics is a broad term that includes everything from simply analyzing data to theorizing ways of collecting data and creating the frameworks needed to store it.
- Data analytics is often confused with data analysis. While these are related terms, they aren't exactly the same. In fact, data analysis is a subcategory of data analytics that deals specifically with extracting meaning from data. Data analytics, as a whole, includes processes beyond analysis, including data science (using data to theorize and forecast) and data engineering (building data systems).

Key Types of Data Analytics for Business



source: intellisoft.io

Why do I need big data analytics for my business?




Empowers management to make better decisions



Helps identify trends to stay competitive



Increase the efficiency and commitment to staff in handling core tasks and issues



Identifies and acts upon opportunities



Promotes low risk data-driven action plans



Validate Decisions



Helps in selecting target Audience



Facilitates sensible recruitment of talent

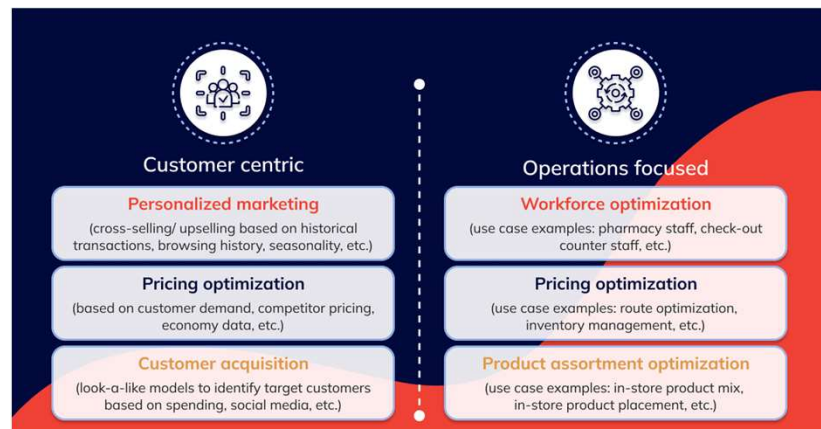
<https://marutitech.com/big-data-analytics-need-business/>

Data Analytics For Business: An Example

Walmart

Walmart collects 2.45 petabytes of unstructured data from one million customers every hour from nearly 20,000 stores serving over 245 million customers worldwide.

It uses this data in a variety of ways to increase revenue, increase customer acquisition and retention, and improve customer service.



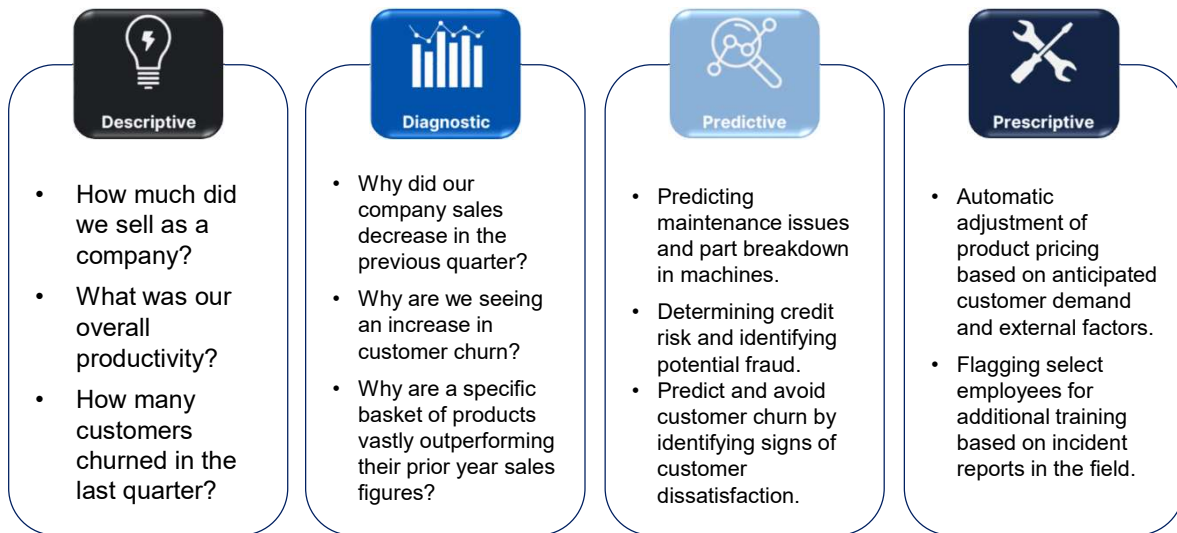
source: intellisoft.io

Four Types of Analytics

Understanding the **What, Why, When, Where** and **How** of your data helps drive analytics maturity.



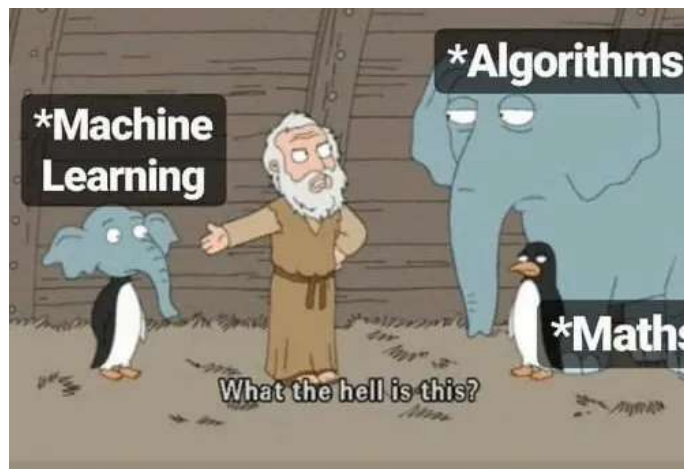
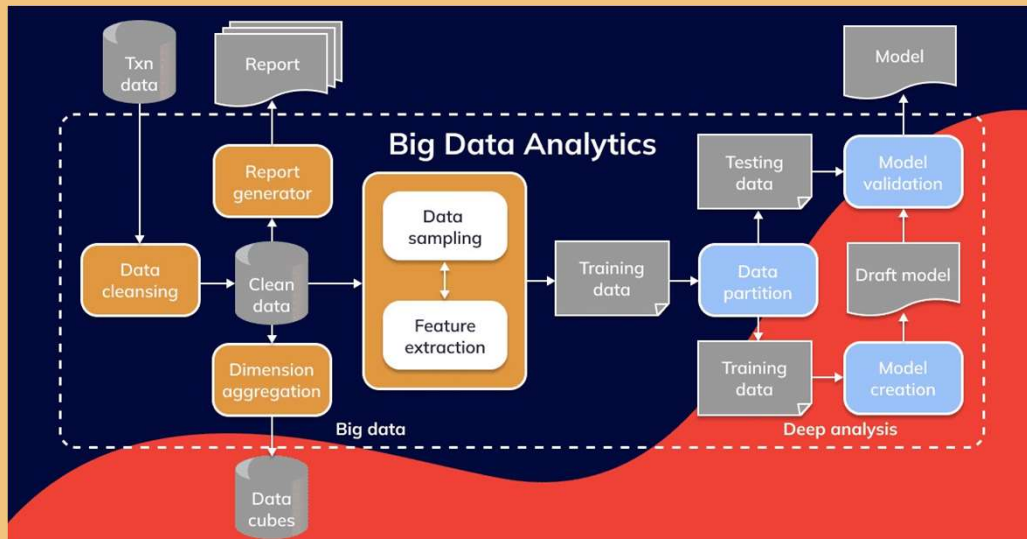
Four Types of Analytics (2)



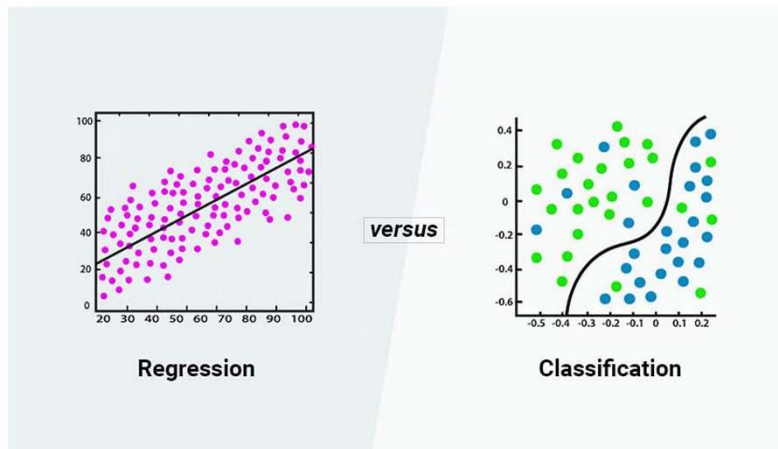
Data Analytics Skills

- **Data management**, or the practices around collecting, organizing and storing data
- **Data engineering**, building data systems to provide a platform that facilitates the process of data visualization and analysis
- **Structured Query Language (SQL)**, a programming language commonly used for databases
- **Probability and statistics**, in order to better analyze and interpret data trends
- **Statistical programming languages**, such as R and Python, commonly used to create advanced data analysis programs
- **Machine learning**, a branch of artificial intelligence that involves using algorithms to spot data patterns
- **Data visualization**, or the ability to use charts and graphs to tell a story with data
- **Econometrics**, or the ability to use data trends to create mathematical models that forecast future trends based
- Etc.

Key Tech Solutions Enabling Big Data Analytics for Businesses



Data Analytics Problem



Regression vs Classification



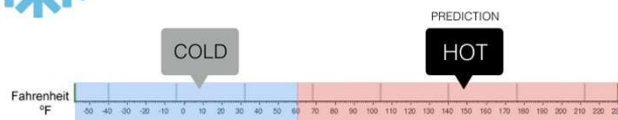
Regression

What is the temperature going to be tomorrow?



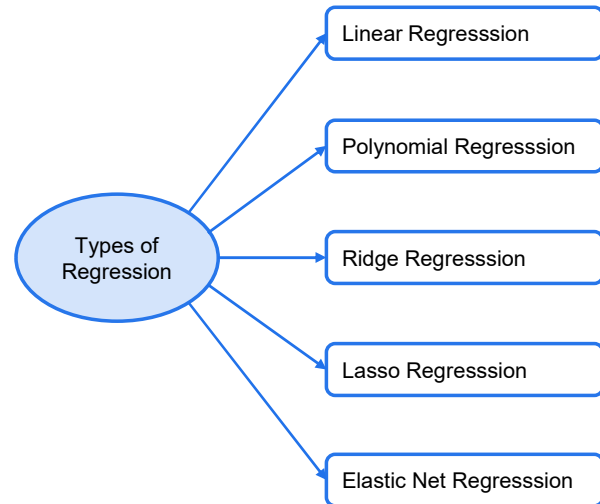
Classification

Will it be Cold or Hot tomorrow?



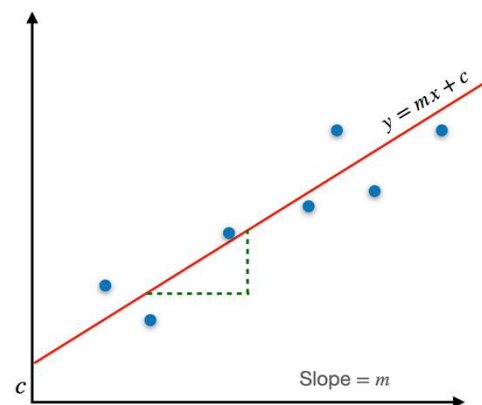
Regression

- Regression algorithms used to predict numerical values based on input data.
- Regression algorithms attempt to find a relationship between the input variables and the output variable by fitting a mathematical model to the data.
- The goal of regression is to find a mathematical relationship between the input features and the target variable that can be used to make accurate predictions on new, unseen data.



1. Linear Regression

- Linear regression is a popular and widely used algorithm for predicting continuous numeric values.
- It models the relationship between independent variables (input features) and a dependent variable (target variable) by fitting a linear equation to the observed data.
- The linear regression algorithm aims to find the best-fit line that represents the relationship between the input features (x) and the target variable (y).



For multi-number of variables, we use:

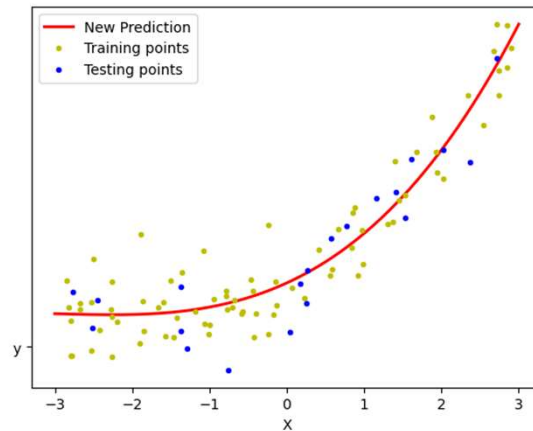
$$y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_nx_n$$

2. Polynomial Regression

- Polynomial regression is a statistical technique for modeling a relationship between a dependent variable (y) and two or more independent variables (x) using a polynomial function.
- The general form of the polynomial regression model is:

$$y = \beta_0 + \beta_1x + \beta_2x^2 + \dots \beta_nx^n$$

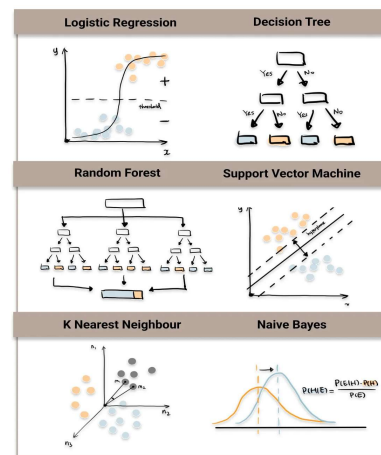
- It's important to note that while polynomial regression allows for a more flexible fit to the data, it also runs the risk of overfitting, especially with higher-degree polynomials.
- Overfitting occurs when the model captures noise or fluctuations in the training data, leading to poor generalization to new, unseen data.



Classification

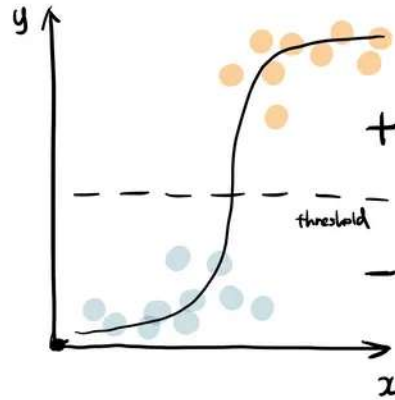
A number of classification techniques are known, which can be broadly classified into the following categories:

- Statistical-Based Methods
 - Logistic Regression
 - Bayesian Classifier
- Distance-Based Classification
 - K-Nearest Neighbors (KNN)
- Classification using Vector Machine
 - Support Vector Machine (SVM)
- Tree-Based Classification
 - Decision Tree, Random Forest
- Classification using Neural Network



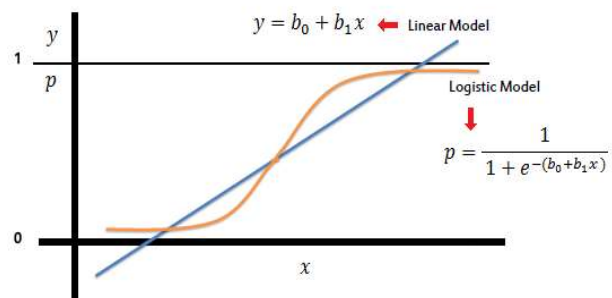
1. Logistic Regression

- Don't let the name logistic regression tricks you, it falls under the category of the classification algorithm instead of regression algorithm.
- Logistic regression uses sigmoid function to return the probability of a label. It is widely used when the classification problem is binary — true or false, win or lose, positive or negative, ...
- The sigmoid function generates a probability output. By comparing the probability with a pre-defined threshold, the object is assigned to a label accordingly.



Linear vs Logistic Regression

- Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the target variable, rather than the probability.
- A linear regression is not appropriate for predicting the value of a binary variable for two reasons:
 - ✓ A linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1)
 - ✓ Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.
- On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1.



$$\frac{p}{1-p} = \exp(b_0 + b_1x)$$

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1x$$

$$p = \frac{1}{1 + e^{-(b_0 + b_1x_1 + b_2x_2 + \dots + b_px_p)}}$$

for multi-number
of variables

Dataset

Example: There is a dataset given which contains the information of various users obtained from the social networking sites.

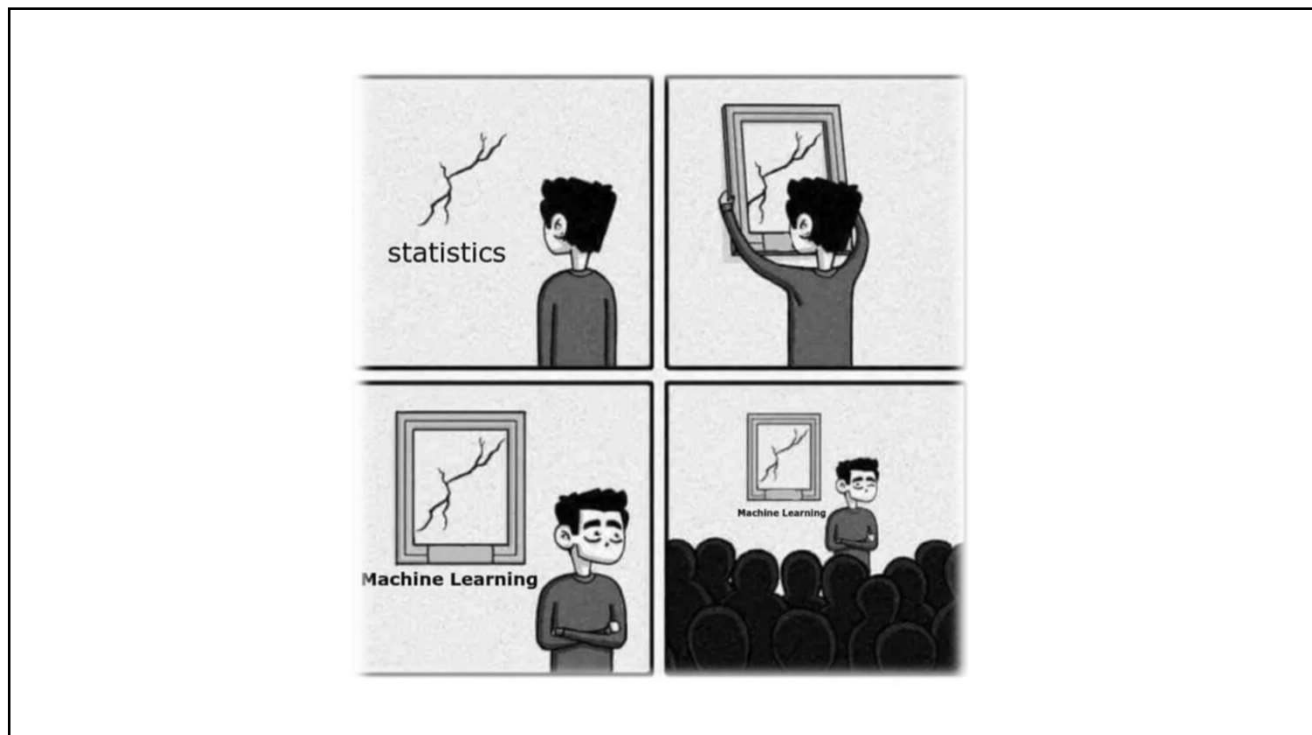
There is a car making company that has recently launched a new SUV car.

So the company wanted to check how many users from the dataset, wants to purchase the car.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	15000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

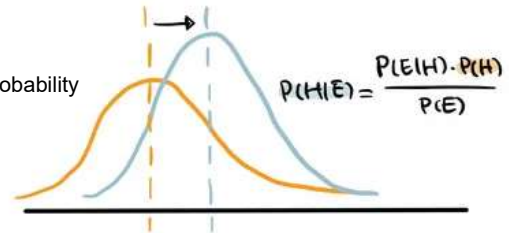
Let's get it on...





2. Naïve Bayes

- A statistical classifier
Performs probabilistic prediction, i.e., predicts class membership probabilities
- A supervised learning algorithm
Classes must be labeled first
- Foundation
Based on Bayes' Theorem - an approach to calculate conditional probability based on prior knowledge
- Assumptions
The classes are mutually exclusive and exhaustive.
The attributes are independent given the class.
Called "Naïve" classifier because of those assumptions
- Advantage
The biggest advantage of Naive Bayes is that, while most machine learning algorithms rely on large amount of training data, it performs relatively well even when the training data size is small.

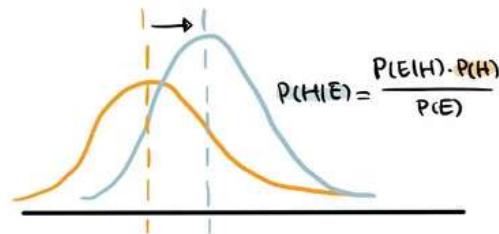


Bayes' Theorem

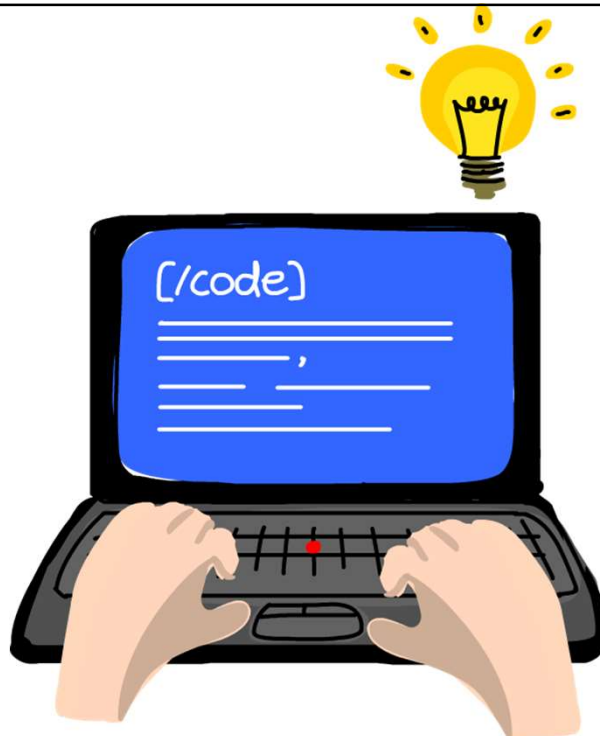
- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

- Where,
 $P(H|E)$ is **Posterior** probability: Probability of hypothesis A on the observed event B.
 $P(E|H)$ is **Likelihood** probability: Probability of the evidence given that the probability of a hypothesis is true.
 $P(H)$ is **Prior** probability: Probability of hypothesis before observing the evidence.
 $P(E)$ is **Marginal** probability: Probability of Evidence.



Let's get it on...



Training the Naive Bayes model on the Training set

```
In [11]: from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

Out[11]: GaussianNB(priors=None, var_smoothing=1e-09)
```

Predicting a new result

```
In [12]: print(classifier.predict(sc.transform([[50,187000]])))

[1]
```

Predicting the Test set results

```
In [13]: y_pred = classifier.predict(X_test)
print(y_pred)

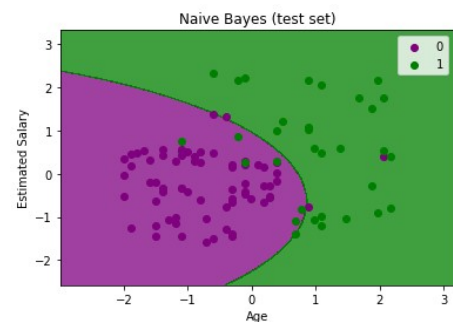
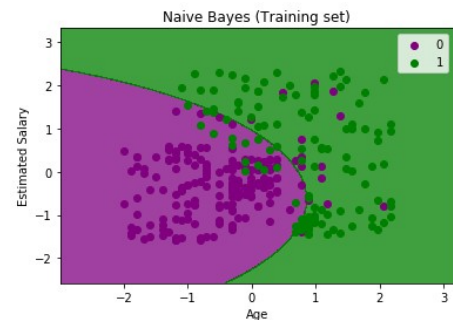
[0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1]
```

Making the Confusion Matrix

```
In [14]: from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)

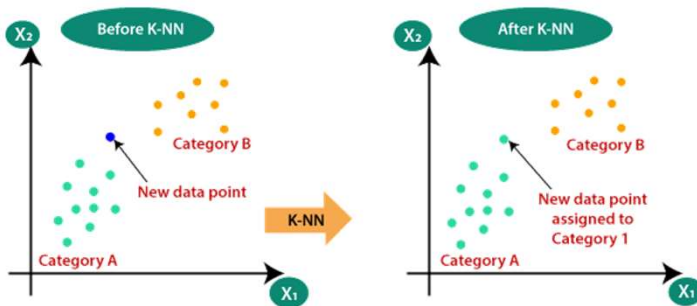
[[65  3]
 [ 7 25]]

Out[14]: 0.9
```



3. K-Nearest Neighbors

- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity.



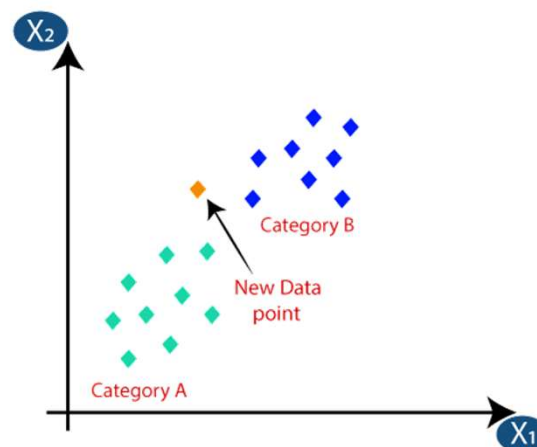
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

1. Select the number K of the neighbors
2. Calculate the Euclidean distance of K number of neighbors
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

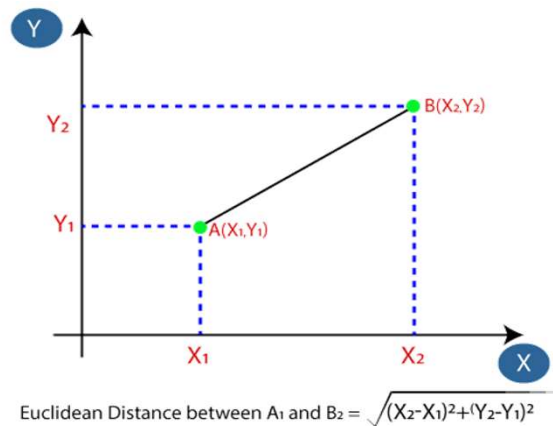


How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

1. Select the number K of the neighbors
2. Calculate the Euclidean distance of K number of neighbors
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.

- Firstly, we choose the number of neighbors, so we will choose the $k=5$.
- Next, we calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

1. Select the number K of the neighbors
2. Calculate the Euclidean distance of K number of neighbors
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.

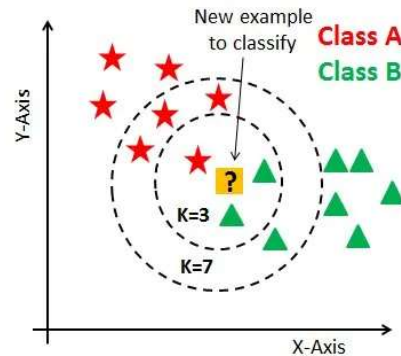
- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the K value?

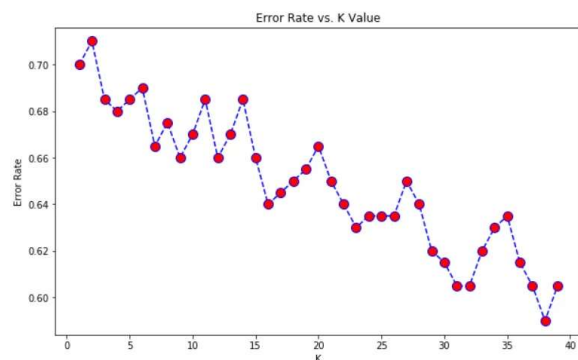
- We have to compute distances between test points and trained labels points.
- Updating distance metrics with every iteration is computationally expensive, and that's why KNN is a [lazy learning algorithm](#).
- As you can see from the beside image, if we proceed with $K=3$, then we predict that test input belongs to class B, and if we continue with $K=7$, then we predict that test input belongs to class A.
- That's how you can imagine that the K value has a powerful effect on KNN performance.



Then how to select the optimal K value?

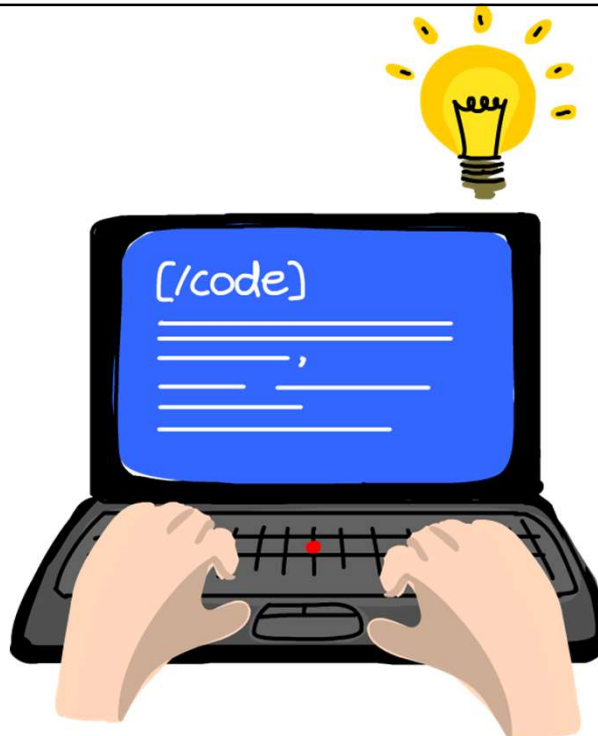
- There are no pre-defined statistical methods to find the most favorable value of K.
- Initialize a random K value and start computing.
- Choosing a small value of K leads to unstable decision boundaries.
- The substantial K value is better for classification as it leads to smoothening the decision boundaries.
- **Derive a plot between error rate and K denoting values in a defined range. Then choose the K value as having a minimum error rate.**

Minimum error:- 0.59 at K = 37

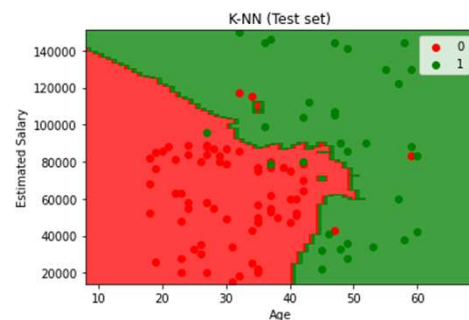
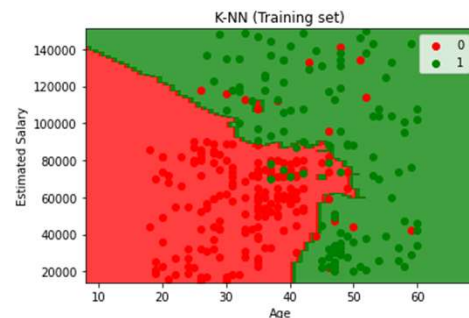


It is known as "Elbow" method.

Let's get it on...

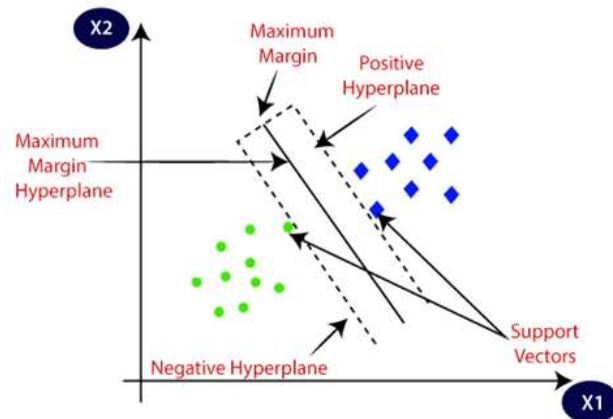


```
Jupyter k_nearest_neighbors Last Checkpoint: 05/30/2023 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
+ - - - - - Run - - - - - Code
Training the K-NN model on the Training set
In [11]: from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
Out[11]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=5, p=2,
weights='uniform')
Predicting a new result
In [12]: print(classifier.predict(sc.transform([[50,187000]])))
[1]
Predicting the Test set results
In [ ]: y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
Making the Confusion Matrix
In [14]: from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
[[64  4]
 [ 3 29]]
Out[14]: 0.93
```



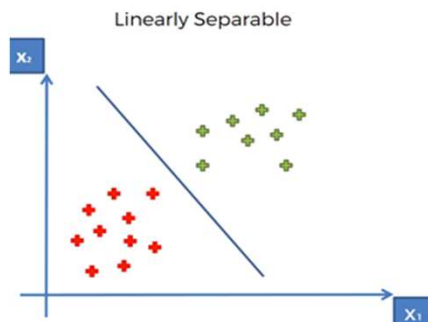
4. Support Vector Machine

- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.
- This best decision boundary is called a **hyperplane**.
- SVM chooses the extreme points/vectors that help in creating the hyperplane.
- These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

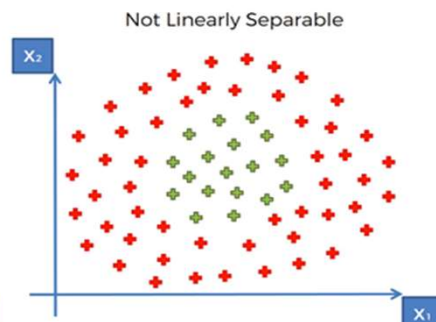


Types of SVM

SVM can be of two types:



Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.



Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

How does SVM work?

Linear SVM

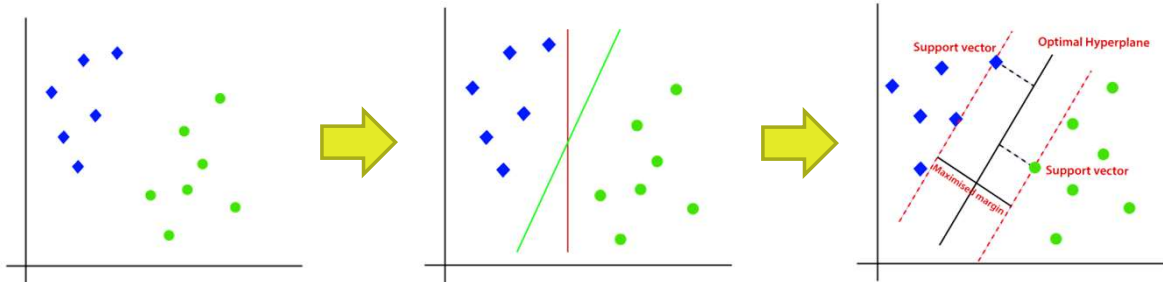
Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x_1 and x_2 . We want a classifier that can classify the pair(x_1 , x_2) of coordinates in either green or blue.

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes.

In SVM we try to find a decision surface which is the line in our picture by **finding the maximum difference (margin) between the data points that lie closest to the decision surface**.

These decision points are called support vectors and in our picture shown as the two parallel dotted lines.

By maximizing the distance, we can reduce the possibility of points get misclassified.



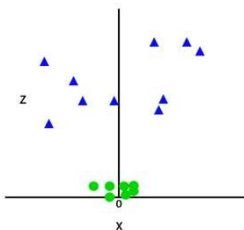
Non-linear SVM

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line.

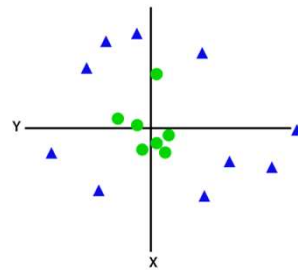
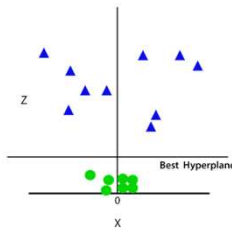
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y , so for non-linear data, we will add a third dimension z . It can be calculated as:

$$z = x^2 + y^2$$

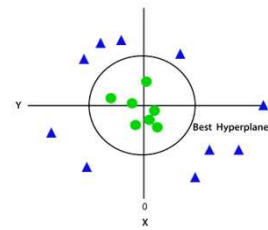
By adding the third dimension, the sample space will become

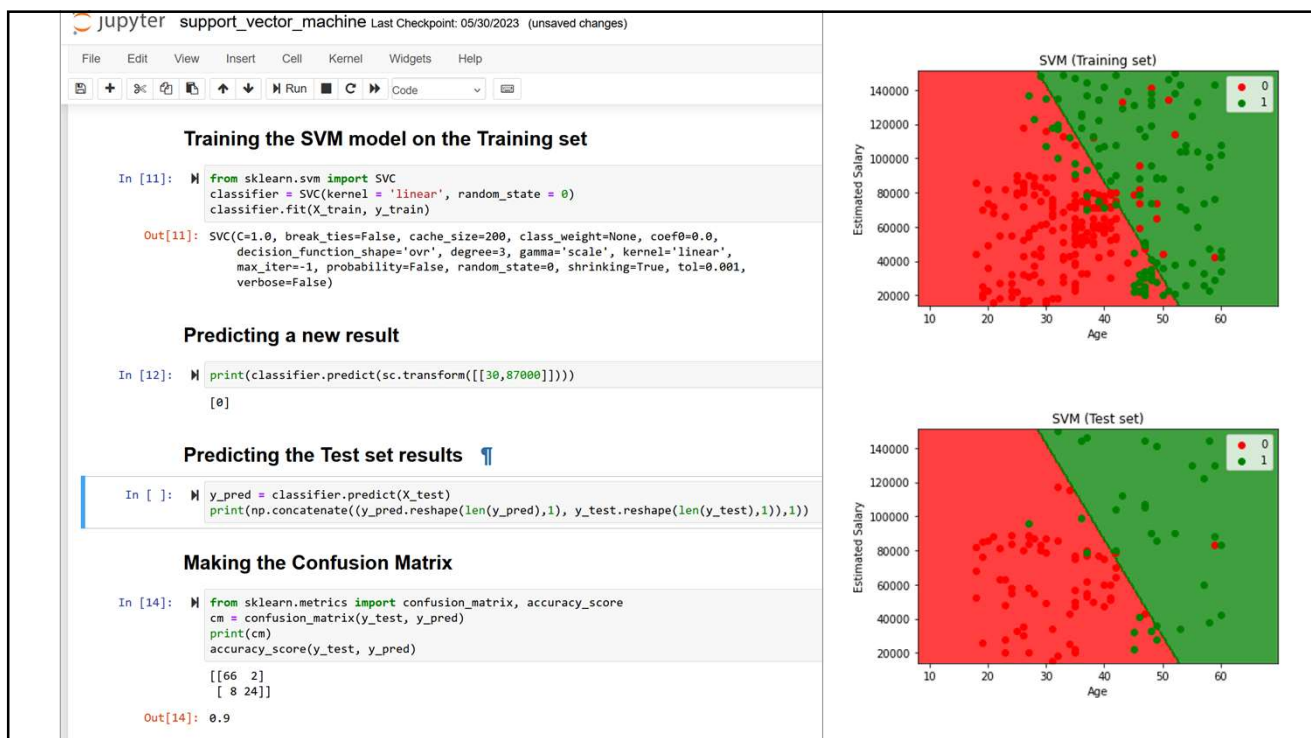
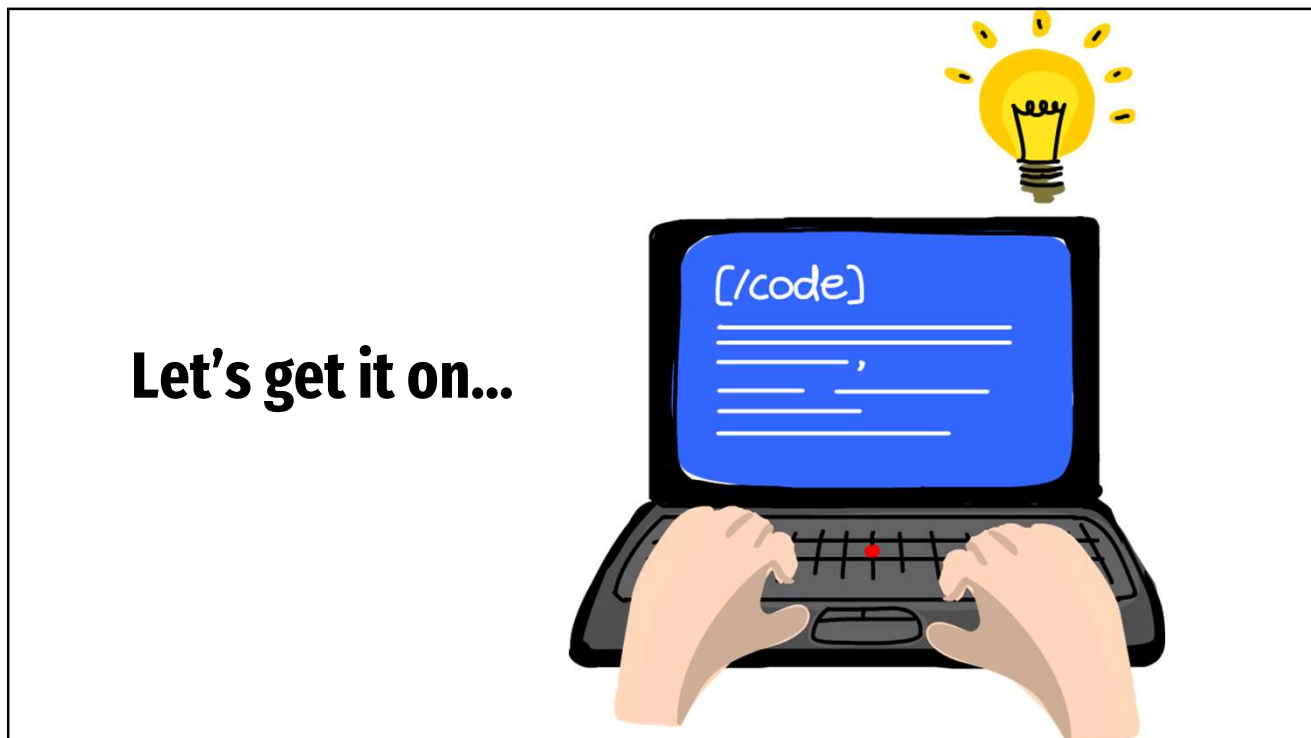


So now, SVM will divide the datasets into classes in the following way.



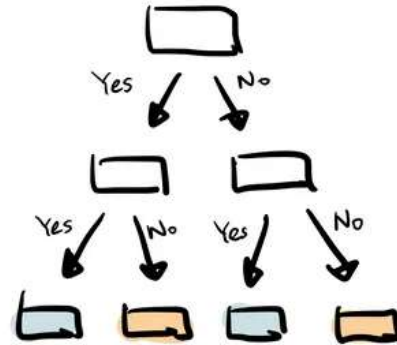
Since we are in 3-d Space, hence it is looking like a plane parallel to the x -axis. If we convert it in 2d space with $z=1$, then it will become as:





5. Decision Tree

- Decision tree builds tree branches in a hierarchy approach and each branch can be considered as an if-else statement.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- Internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- The branches develop by partitioning the dataset into subsets based on most important features.
- Final classification happens at the leaves of the decision tree.



Decision Tree Terminologies

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

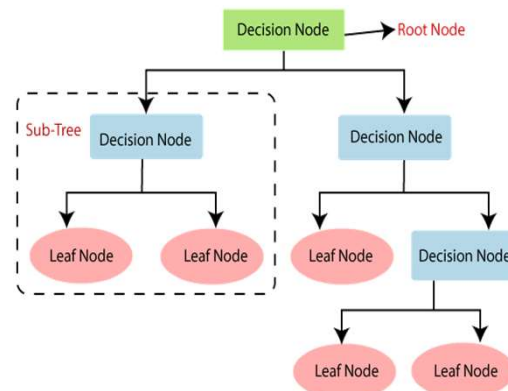
Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

Branch/Sub Tree: A tree formed by splitting the tree.

Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.



How does the Decision Tree algorithm Work?

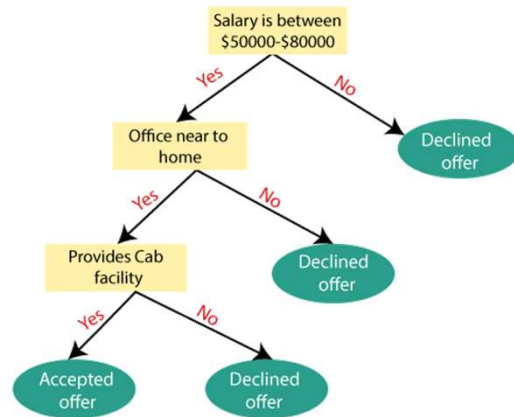
Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM)**.

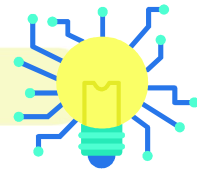
Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in **Step-3**. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.



Decision Tree advantages vs disadvantages



Advantages



- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

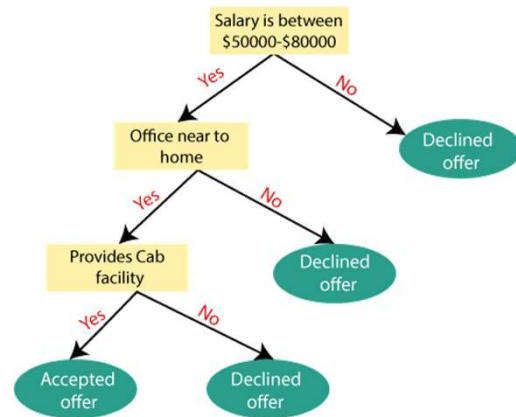
Disadvantages



- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

Attribute Selection Measures

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes.
- So, to solve such problems there is a technique which is called as **Attribute Selection Measure or ASM**.
- By this measurement, we can easily select the best attribute for the nodes of the tree.
- There are two popular techniques for ASM, which are:
 - **Information Gain**
 - **Gini Index**



Information Gain

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

- **Entropy** is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$\text{Entropy}(s) = -P(\text{yes}) \log_2 P(\text{yes}) - P(\text{no}) \log_2 P(\text{no})$$

Where,

S = Total number of samples

P(yes) = probability of yes

P(no) = probability of no

A data set with a high entropy is a data set that is not well classified.

$$\text{Information Gain} = \text{Entropy}(S) - [(\text{Weighted Avg}) * \text{Entropy}(\text{each feature})]$$

Gini Index

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
- An attribute with the **low Gini index should be preferred** as compared to the high Gini index.
- It only creates **binary splits**, and the CART algorithm uses the Gini index to create binary splits.
- The Gini Index or Gini Impurity is calculated by **subtracting the sum of the squared probabilities of each class from one**.
- Gini index can be calculated using the formula:

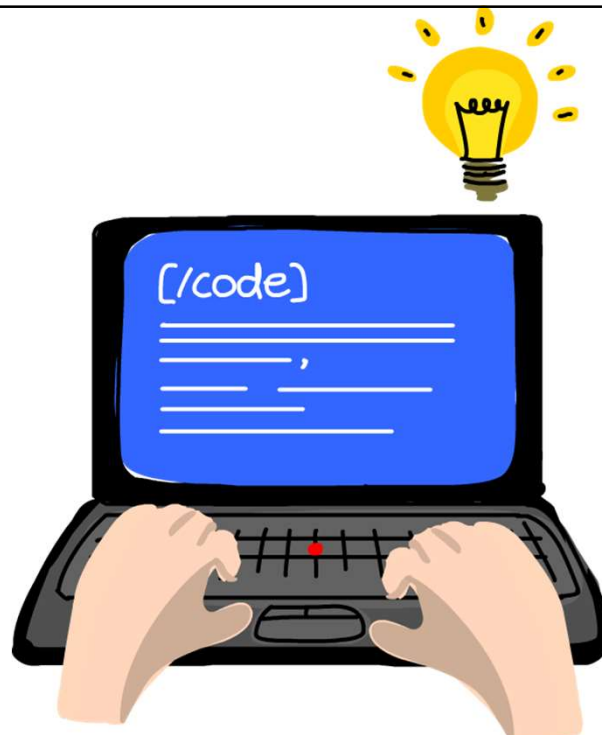
$$G = \sum_{i=1}^C p(i) * (1 - p(i))$$

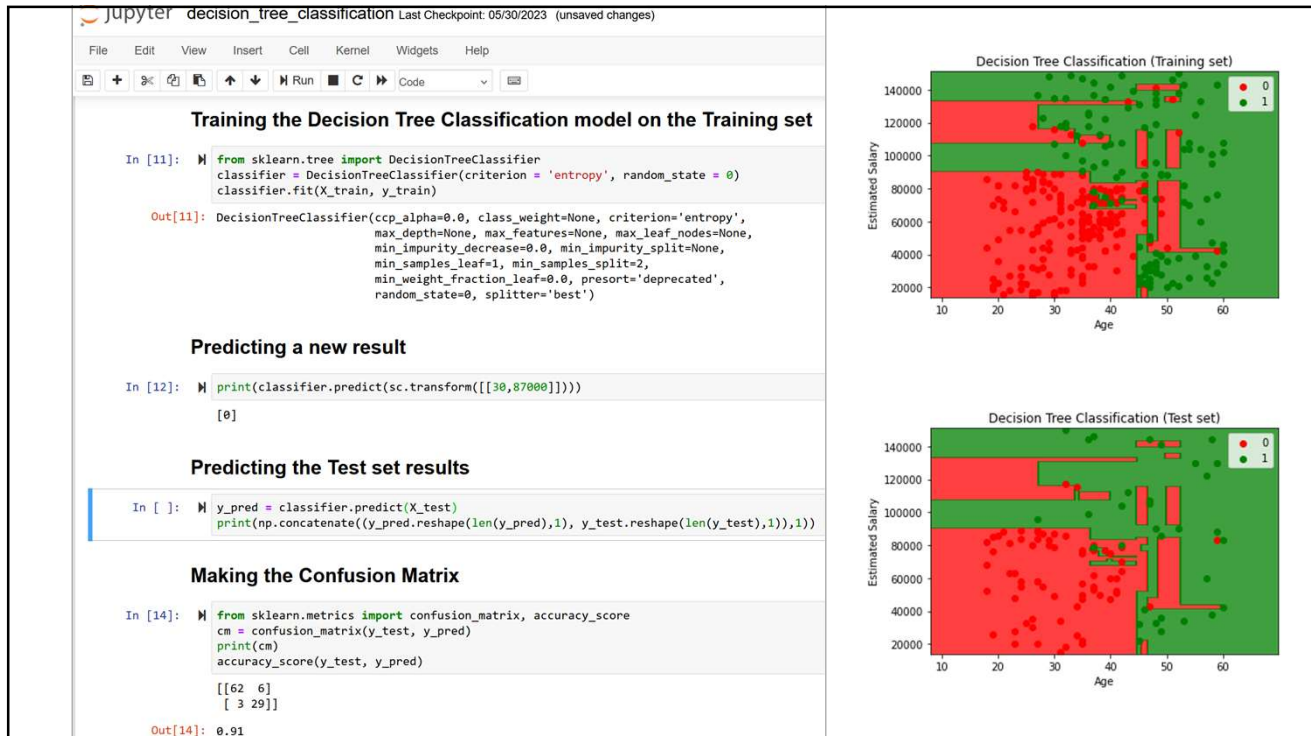
Where,

C is the total number of classes and

p(i) is the probability of picking the data point with the class **i**.

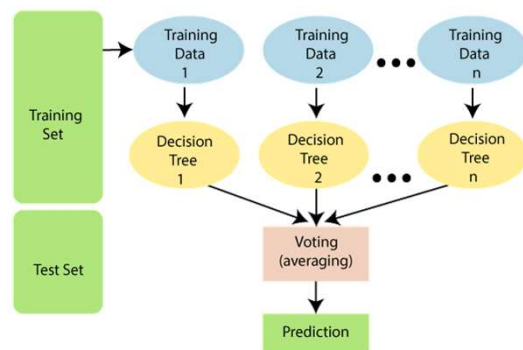
Let's get it on...





6. Random Forest

- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- It is based on the concept of **ensemble learning**, which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model*.



The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting

Assumption for Random Forest

- Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output.
- Therefore, below are two assumptions for a better Random forest classifier:
 - There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
 - The predictions from each tree must have very low correlations.



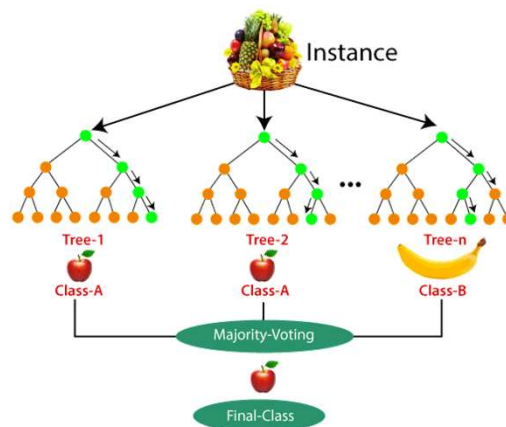
Random Forest can maintain accuracy even if a large proportion of data is missing.

How does the Random Forest algorithm Work?

The Working process can be explained in the below steps and diagram:

1. Select random K data points from the training set
2. Build the decision trees associated with the selected data points (*subsets*)
3. Choose the number N for decision trees that you want to build
4. Repeat Step 1 & 2
5. For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Random Forest works in two-phase. First is to create the random forest by combining N decision tree and second is to make predictions for each tree created in the first phase.



Let's get it on...

