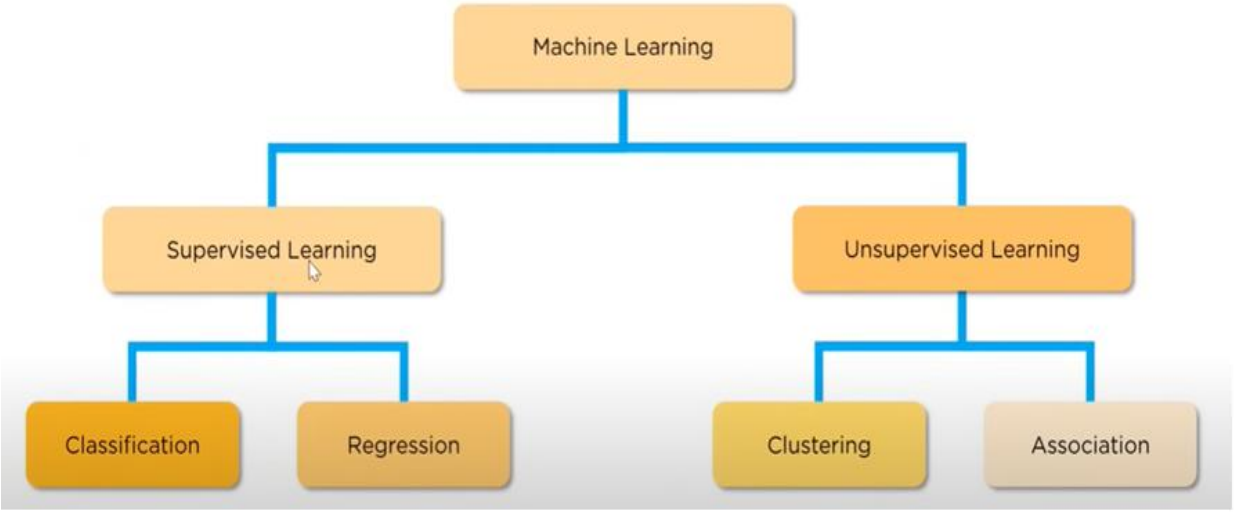
Linear Regression in Machine learning

Machine Learning is a branch of Artificial intelligence that focuses on the development of algorithms and statistical models that can learn from and make predictions on data. **Linear regression** is also a type of machine-learning algorithm more specifically a **supervised machine-learning algorithm** that learns from the labelled datasets and maps the data points to the most optimized linear functions. which can be used for prediction on new datasets.

First of we should know what supervised machine learning algorithms is. It is a type of machine learning where the algorithm learns from labelled data. Labeled data means the dataset whose respective target value is already known. Supervised learning has two types:

- **Classification**: It predicts the class of the dataset based on the independent input variable. Class is the categorical or discrete values. like the image of an animal is a cat or dog?
- **Regression**: It predicts the continuous output variables based on the independent input variable. like the prediction of house prices based on different parameters like house age, distance from the main road, location, area, etc.



-

What is Linear Regression?

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.

When there is only one independent feature, it is known as Simple Linear Regression, and when there are more than one feature, it is known as Multiple Linear Regression.

Similarly, when there is only one dependent variable, it is considered Univariate Linear Regression, while when there are more than one dependent variables, it is known as Multivariate Regression.

Why Linear Regression is Important?

The interpretability of linear regression is a notable strength. The model's equation provides clear coefficients that elucidate the impact of each independent variable on the dependent variable, facilitating a deeper understanding of the underlying dynamics. Its simplicity is a virtue, as linear regression is transparent, easy to implement, and serves as a foundational concept for more complex algorithms.

Linear regression is not merely a predictive tool; it forms the basis for various advanced models. Techniques like regularization and support vector machines draw inspiration from linear regression, expanding its utility. Additionally, linear regression is a cornerstone in assumption testing, enabling researchers to validate key assumptions about the data.

Types of Linear Regression

There are two main types of linear regression:

Simple Linear Regression

This is the simplest form of linear regression, and it involves only one independent variable and one dependent variable. The equation for simple linear regression is: $y=\beta 0+\beta 1X$

where:

- Y is the dependent variable
- X is the independent variable
- β0 is the intercept
- β1 is the slope

Multiple Linear Regression

This involves more than one independent variable and one dependent variable. The equation for multiple linear regression is: $y=\beta 0+\beta 1X+\beta 2X+\ldots\ldots\beta nX$

where:

- Y is the dependent variable
- X1, X2, …, Xp are the independent variables
- β0 is the intercept
- β1, β2, …, βn are the slopes

The goal of the algorithm is to find the best Fit Line equation that can predict the values based on the independent variables.

In regression set of records are present with X and Y values and these values are used to learn a function so if you want to predict Y from an unknown X this learned function can be used. In regression we have to find the value of Y, So, a function is required that predicts continuous Y in the case of regression given X as independent features.

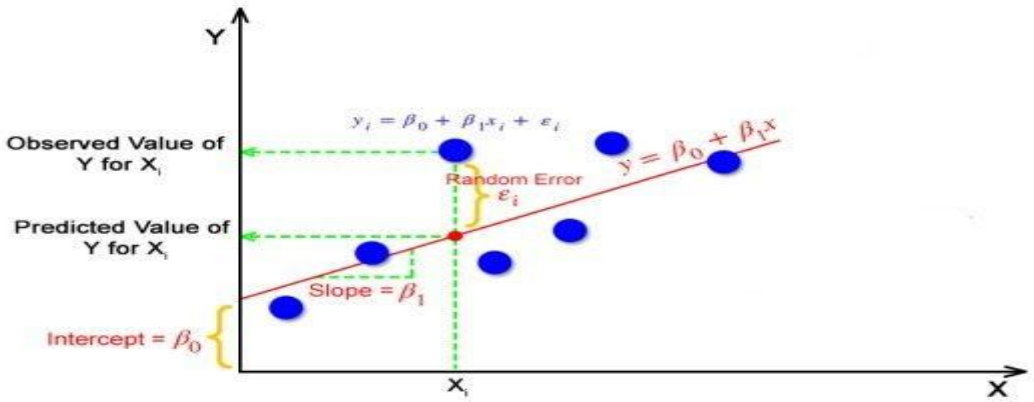
Simple Regression Calculation

To calculate best-fit line linear regression uses a traditional slope-intercept form which is given below,

$\mathbf{Y_i = \beta_0 + \beta_1 X_i}$

where Y_i = Dependent variable, $\boldsymbol{\beta_0}$ = constant/Intercept, $\boldsymbol{\beta_1}$ = Slope/Intercept, $\mathbf{X_i}$ = Independent variable.

This algorithm explains the linear relationship between the dependent(output) variable y and the independent(predictor) variable X using a straight line $Y= B_0 + B_1 X$.



But how the linear regression finds out which is the best fit line?

The goal of the linear regression algorithm is to get the **best values for B_0 and B_1** to find the best fit line. The best fit line is a line that has the least error which means the error between predicted values and actual values should be minimum.

Random Error(Residuals)

In regression, the difference between the observed value of the dependent variable($\mathbf{y_i}$) and the predicted value(**predicted**) is called the residuals.

$\varepsilon_i = y_{predicted} - y_i$

where $y_{predicted} = B_0 + B_1 X_i$

What is the best fit line?

In simple terms, the best fit line is a line that fits the given scatter plot in the best way. Mathematically, the best fit line is obtained by minimizing the Residual Sum of Squares(RSS).

Cost Function for Linear Regression

The cost function helps to work out the optimal values for B_0 and B_1, which provides the best fit line for the data points.

In Linear Regression, generally **Mean Squared Error (MSE)** cost function is used, which is the average of squared error that occurred between the $y_{predicted}$ and y_i.

We calculate MSE using simple linear equation y=mx+b:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (B1x_i + B0))^2$$

Using the MSE function, we'll update the values of B_0 and B_1 such that the MSE value settles at the minima. These parameters can be determined using the gradient descent method such that the value for the cost function is minimum.

Mathematics Behind Simple Linear Regression

The crux of Simple Linear Regression is to derive a **best-fit line** that captures the relationship between any single independent variable and the dependent variable. The best-fit line emphasizes minimizing the overall distance of data points from the **regression line**, ensuring a precise alignment with the trend.

Statistically, the **mathematical equation** that approximately models a **linear relationship** between a dependent variable x, & an independent variable y is

$$y=mx+c$$

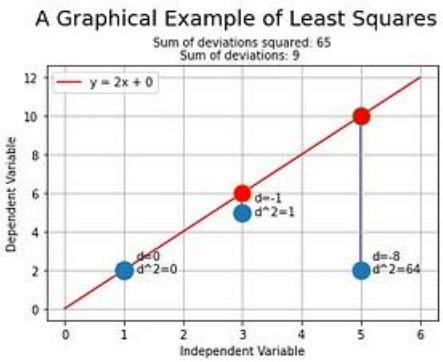
wherein, 'm' is the **slope** of the line, and 'c' is the **intercept**, and together they are referred to as the '**Model's Coefficients**'. This equation is the basis for any Linear Regression model and is often referred to as the **Hypothesis Function**

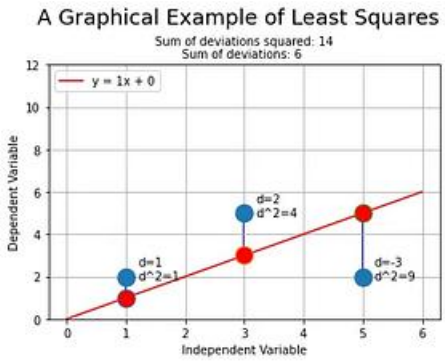
The goal of most machine learning algorithms is to construct a **model** i.e. the **hypothesis** (H) to estimate the dependent variable based on our independent variables such that it minimizes the **Loss Function (**Residual Sum of Squares)

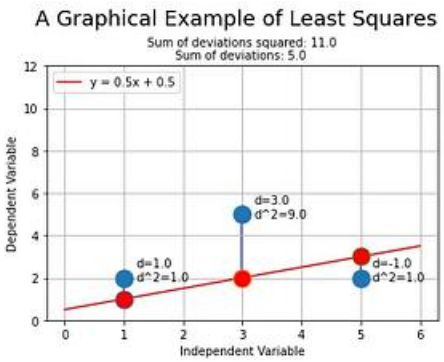
Aptly termed the **Least Squares Method**, this approach seeks to find the most optimal values for the model parameters that minimize the loss function/RSS.

1.2 Exploring the Cost Functions

While **coefficients** provides insights into the **relationship between variables**, we need **metrics** to quantify how accurately these linear models captures the underlying data patterns. The most common metrics are RMSE and R2 Score







One of the primary metric used to assess the model's performance is **MSE,** or **Mean Squared Error**. This metric quantifies the squared difference, between the actual observed values and the predicted values from the regression line. A lower MSE indicates that the model's predictions are closer to actual values.

The **R-squared (R2) Score**, also known as the **coefficient of determination** is another vital metric that provides a measure of how well the model explains the variability, in the dependent variable. It ranges from 0 to 1, with a higher value indicating a better fit without explicitly accounting for overfitting data

$$\mathrm{MSE}=\frac{1}{2m}\Sigma(h_\theta(x)^{(i)}-y^i)^2$$

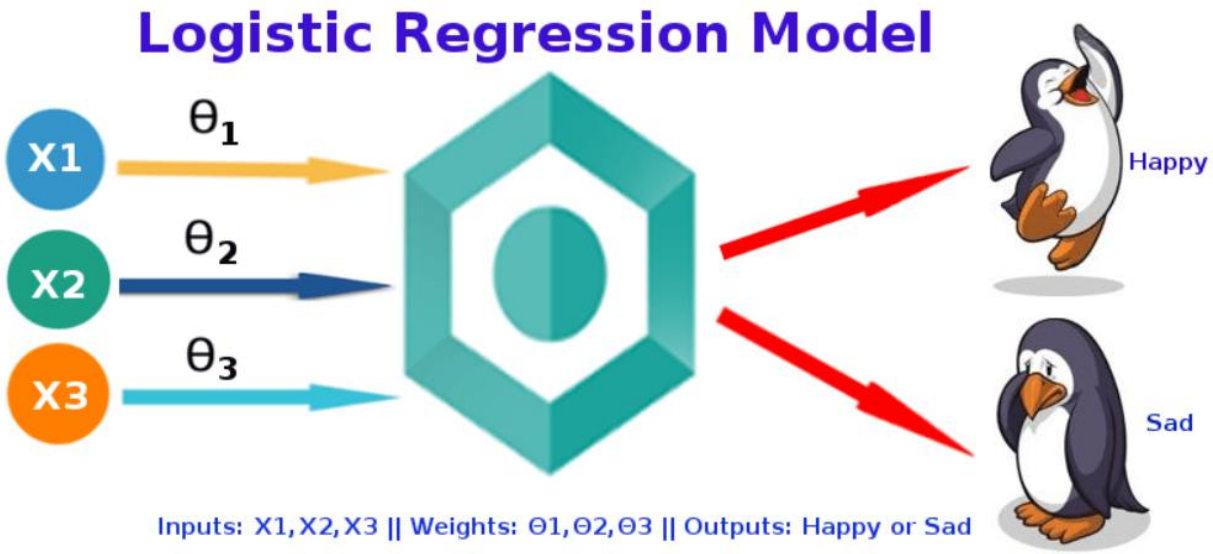
Beyond these, you may also explore other metrices such as: **Residual Sum of Squares,** and **Residual Analysis** that offer a more comprehensive evaluation of model performance.

Logistic Regression

What is Logistic Regression?

Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.

For example, we have two classes Class 0 and Class 1 if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class 1 it belongs to Class 0. It's referred to as regression because it is the extension of linear regression but is mainly used for classification problems.



Why do we use Logistic Regression rather than Linear Regression?

If you have this doubt, then you're in the right place, my friend. After reading the definition of logistic regression we now know that it is only used when our dependent variable is binary and in linear regression this dependent variable is continuous.

The second problem is that if we add an outlier in our dataset, the best fit line in linear regression shifts to fit that point.

Now, if we use linear regression to find the best fit line which aims at minimizing the distance between the predicted value and actual value, the line will be like this:

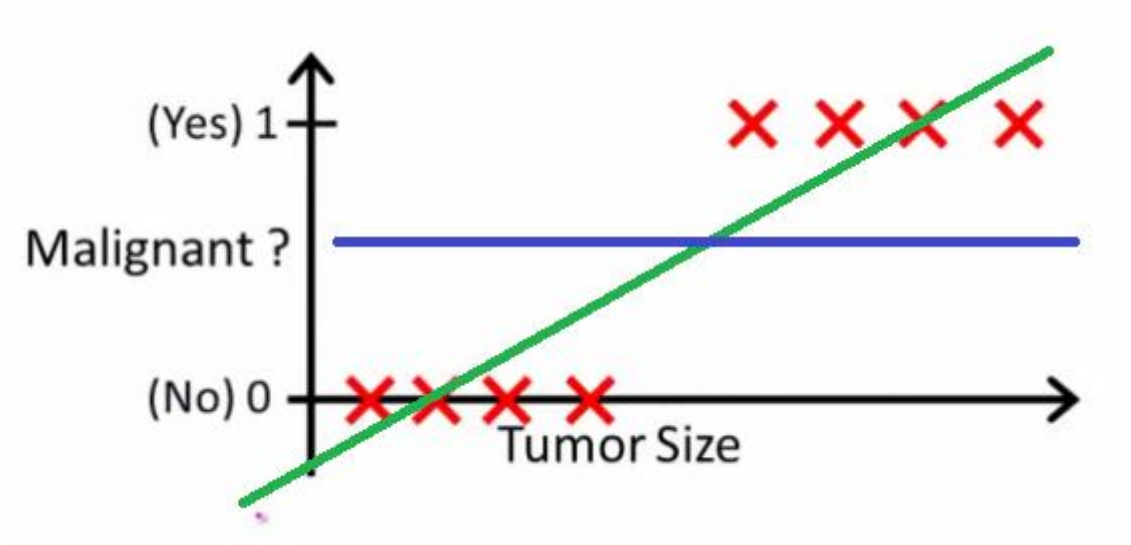


Image Source:towardsdatascience.com

Here the threshold value is 0.5, which means if the value of h(x) is greater than 0.5 then we predict malignant tumor (1) and if it is less than 0.5 then we predict benign tumor (0). Everything seems okay here but now let's change it a bit, we add some outliers in our dataset, now this best fit line will shift to that point. Hence the line will be somewhat like this:

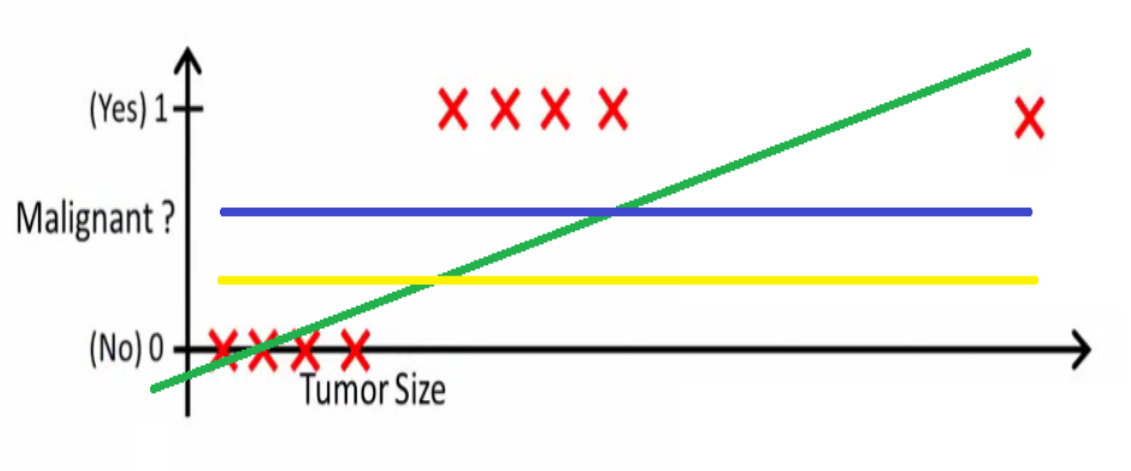


Image Source:towardsdatascience.com

Do you see any problem here? The blue line represents the old threshold and the yellow line represents the new threshold which is maybe 0.2 here. To keep our predictions right we had to lower our threshold value. Hence we can say that linear

regression is prone to outliers. Now here if h(x) is greater than 0.2 then only this regression will give correct outputs.

Another problem with linear regression is that the predicted values may be out of range. We know that probability can be between 0 and 1, but if we use linear regression this probability may exceed 1 or go below 0.

To overcome these problems we use Logistic Regression, which converts this straight best fit line in linear regression to an S-curve using the sigmoid function, which will always give values between 0 and 1. How does this work and what's the math behind this will be covered in a later section?

Key Points:

- Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value.
- It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

Logistic Function – Sigmoid Function

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form.
- The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Types of Logistic Regression

On the basis of the categories, Logistic Regression can be classified into three types:

1. **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
2. **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as “cat”, “dogs”, or “sheep”
3. **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as “low”, “Medium”, or “High”.

Assumptions of Logistic Regression

We will explore the assumptions of logistic regression as understanding these assumptions is important to ensure that we are using appropriate application of the model. The assumption include:

1. Independent observations: Each observation is independent of the other. meaning there is no correlation between any input variables.
2. Binary dependent variables: It takes the assumption that the dependent variable must be binary or dichotomous, meaning it can take only two values. For more than two categories SoftMax functions are used.
3. Linearity relationship between independent variables and log odds: The relationship between the independent variables and the log odds of the dependent variable should be linear.
4. No outliers: There should be no outliers in the dataset.
5. Large sample size: The sample size is sufficiently large

Terminologies involved in Logistic Regression

Here are some common terms involved in logistic regression:

- **Independent variables:** The input characteristics or predictor factors applied to the dependent variable’s predictions.
- **Dependent variable:** The target variable in a logistic regression model, which we are trying to predict.
- **Logistic function:** The formula used to represent how the independent and dependent variables relate to one another. The logistic function transforms the input variables into a probability value between 0 and 1, which represents the likelihood of the dependent variable being 1 or 0.

- **Odds:** It is the ratio of something occurring to something not occurring. it is different from probability as the probability is the ratio of something occurring to everything that could possibly occur.
- **Log-odds:** The log-odds, also known as the logit function, is the natural logarithm of the odds. In logistic regression, the log odds of the dependent variable are modeled as a linear combination of the independent variables and the intercept.
- **Coefficient:** The logistic regression model's estimated parameters, show how the independent and dependent variables relate to one another.
- **Intercept:** A constant term in the logistic regression model, which represents the log odds when all independent variables are equal to zero.
- **Maximum likelihood estimation:** The method used to estimate the coefficients of the logistic regression model, which maximizes the likelihood of observing the data given the model.

Logistic Function

You must be wondering how logistic regression squeezes the output of linear regression between 0 and 1. If you haven't read my article on Linear Regression then please have a look at it for a better understanding.

Well, there's a little bit of math included behind this and it is pretty interesting trust me.

Let's start by mentioning the formula of logistic function:

$$P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

How similar it is too linear regression? If you haven't read my article on Linear Regression, then please have a look at it for a better understanding.

Best Fit Equation in Linear Regression

We all know the equation of the best fit line in linear regression is:

$$y = \beta_0 + \beta_1 x$$

Let's say instead of y we are taking probabilities (P). But there is an issue here, the value of (P) will exceed 1 or go below 0 and we know that range of Probability is (0-1). To overcome this issue we take ***"odds"*** of P:

$$P = \beta_0 + \beta_1 x$$

$$\frac{P}{1-P} = \beta_0 + \beta_1 x$$

Do you think we are done here? No, we are not. We know that odds can always be positive which means the range will always be $(0,+\infty)$. Odds are nothing but the ratio of the probability of success and probability of failure. Now the question comes out of so many other options to transform this why did we only take ***'odds'***? Because odds are probably the easiest way to do this, that's it.

The problem here is that the range is restricted and we don't want a restricted range because if we do so then our correlation will decrease. By restricting the range we are actually decreasing the number of data points and of course, if we decrease our data points, our correlation will decrease. It is difficult to model a variable that has a restricted range. To control this we take the ***log of odds*** which has a range from $(-\infty,+\infty)$.

$$\log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x$$

If you understood what I did here then you have done 80% of the maths. Now we just want a function of P because we want to predict probability right? not log of odds. To do so we will multiply by ***exponent*** on both sides and then solve for P.

$$\exp[\log(\frac{p}{1-p})] \;=\; \exp(\beta_0 + \beta_1 x)$$

$$e^{\ln[\frac{p}{1-p}]} = e^{(\beta_0 + \beta_1 x)}$$

$$\frac{p}{1-p} \;=\; e^{\left(\beta_0 + \beta_1 x\right)}$$

$$p \;= e^{\left(\beta_0 + \beta_1 x\right)} - pe^{\left(\beta_0 + \beta_1 x\right)}$$

$$p \;=\; p[\frac{e^{\left(\beta_0 + \beta_1 x\right)}}{p} - e^{\left(\beta_0 + \beta_1 x\right)}]$$

$$1 = \frac{e^{\left(\beta_0 + \beta_1 x\right)}}{p} - e^{\left(\beta_0 + \beta_1 x\right)}$$

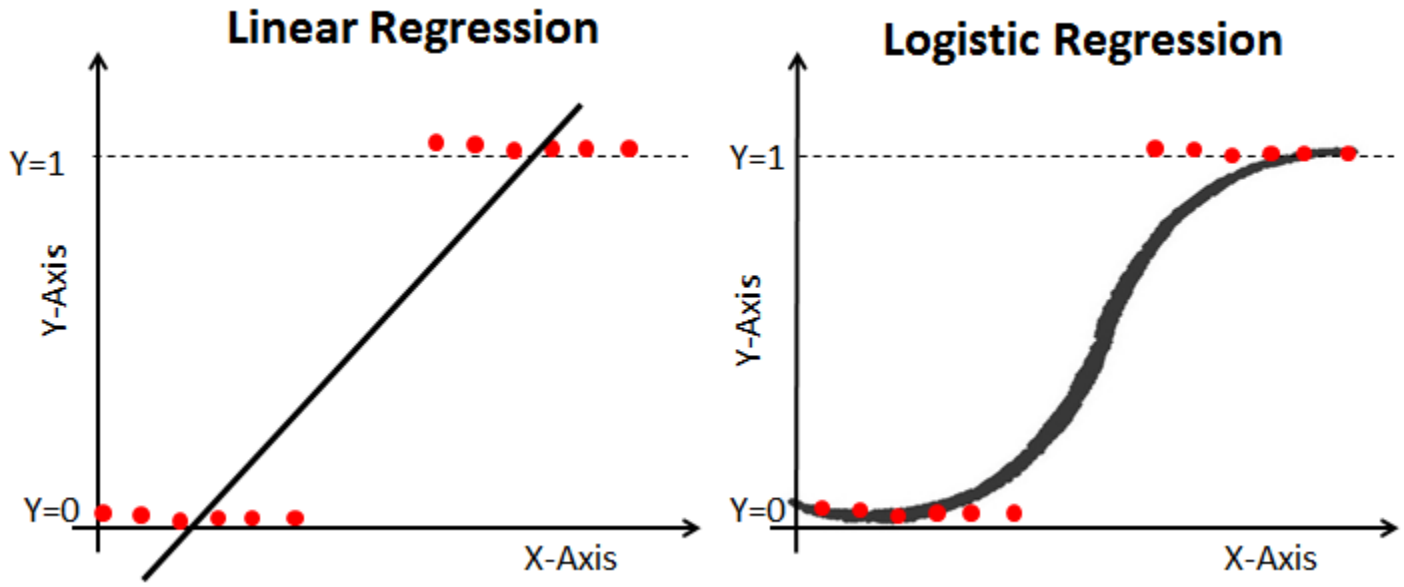
$$p[1 + e^{\left(\beta_0 + \beta_1 x\right)}] = e^{\left(\beta_0 + \beta_1 x\right)}$$

$$p \;= \frac{e^{\left(\beta_0 + \beta_1 x\right)}}{1 + e^{\left(\beta_0 + \beta_1 x\right)}}$$

Now dividing by $e^{\left(\beta_0 + \beta_1 x\right)}$, *we will get*

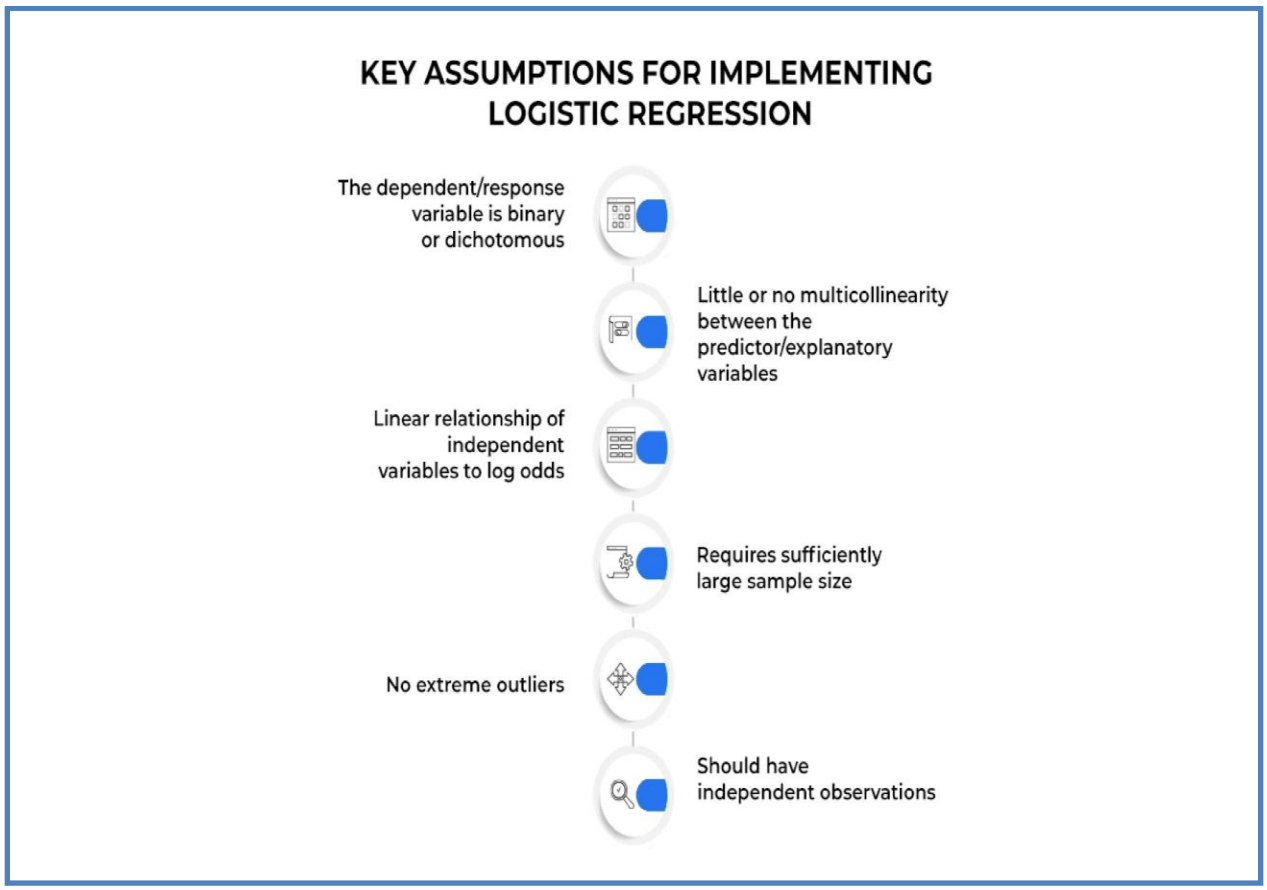
$$p \;=\; \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \quad \textit{This is our sigmoid function.}$$

Now we have our logistic function, also called a sigmoid function. The graph of a sigmoid function is as shown below. It squeezes a straight line into an S-curve.



Key properties of the logistic regression equation

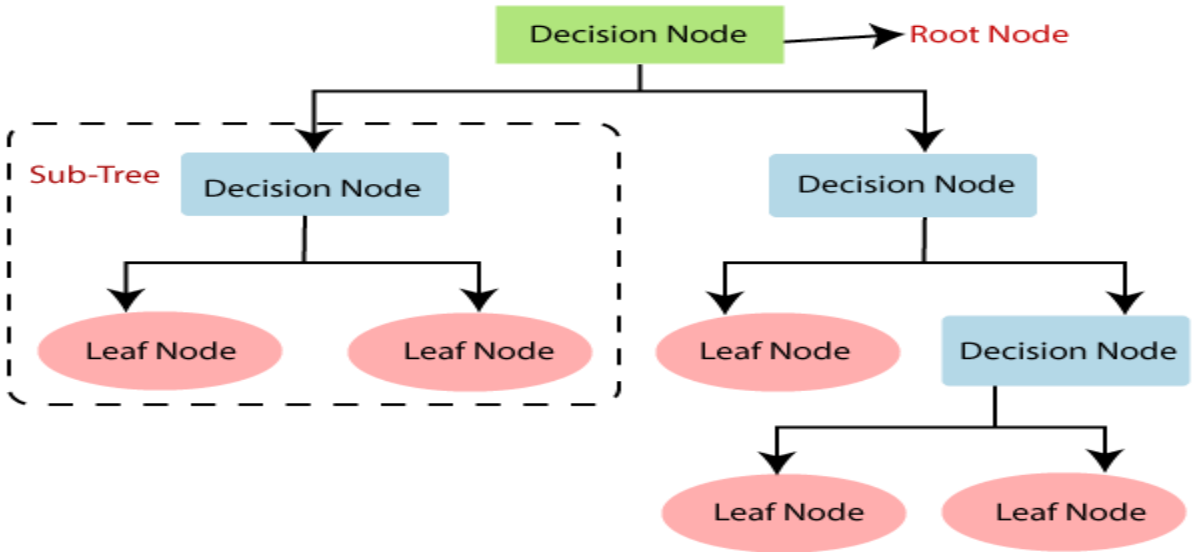
- **Sigmoid Function**: The logistic regression model, when explained, uses a special "S" shaped curve to predict probabilities. It ensures that the predicted probabilities stay between 0 and 1, which makes sense for probabilities.
- **Straightforward Relationship**: Even though the logistic regression model might seem complex, the relationship between our inputs (like age, height, etc.) and the outcome (like yes/no) is pretty simple to understand. It's like drawing a straight line, but with a curve instead.
- **Coefficients**: These are just numbers that tell us how much each input affects the outcome in the logistic regression model. For example, if age is a predictor, the coefficient tells us how much the outcome changes for every one year increase in age.
- **Best Guess**: We figure out the best coefficients for the logistic regression model by looking at the data we have and tweaking them until our predictions match the real outcomes as closely as possible.

- **Basic Assumptions**: In logistic regression explained, we assume that our observations are independent, meaning one doesn't affect the other. We also assume that there's not too much overlap between our predictors (like age and height), and the relationship between our predictors and the outcome is kind of like a straight line.
- **Probabilities, Not Certainties**: Instead of saying "yes" or "no" directly, logistic regression gives us probabilities, like saying there's a 70% chance it's a "yes" in the logistic regression model. We can then decide on a cutoff point to make our final decision.
- **Checking Our Work**: In logistic regression explained, we have some tools to make sure our predictions are good, like accuracy, precision, recall, and a curve called the ROC curve. These help us see how well our logistic regression model is doing its job.
- 

Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- 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.

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.

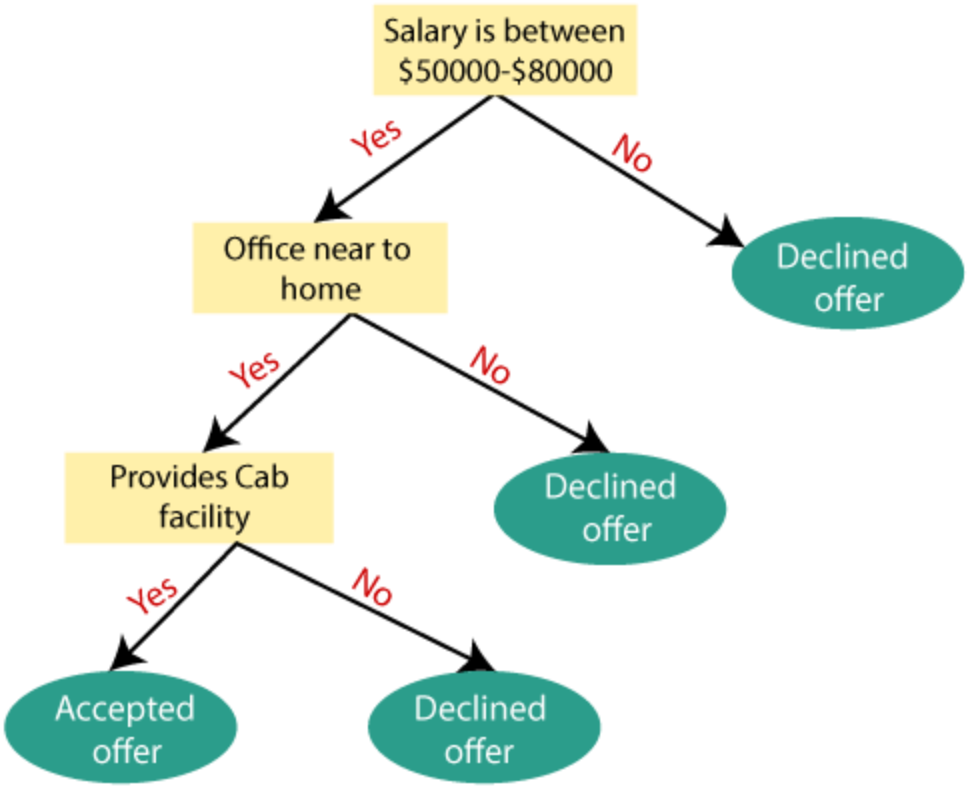
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **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.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



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**

1. 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:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

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

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. 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.
- Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning

tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- 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 of the Decision Tree

- 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.

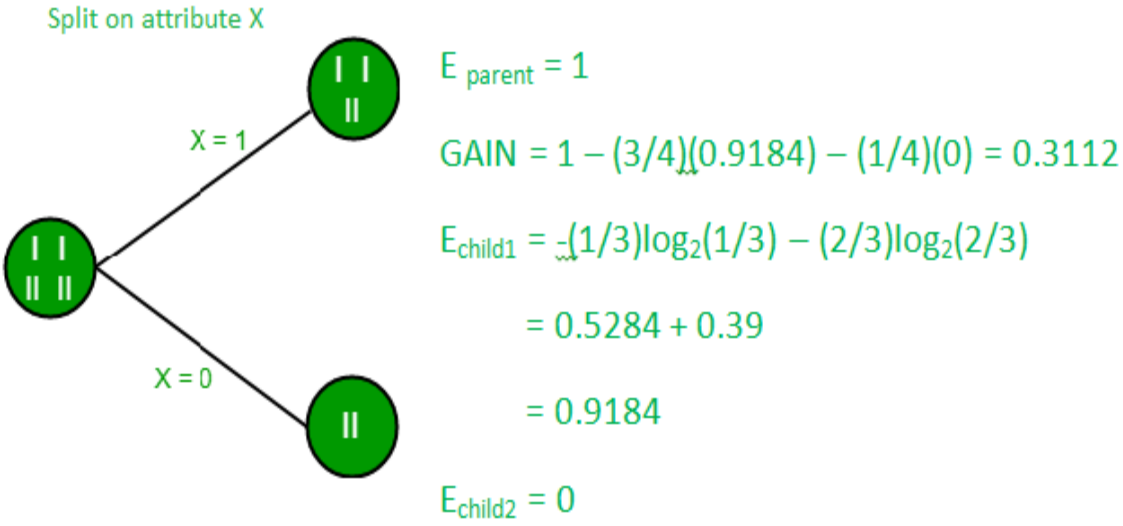
Cart Algorithm

- Step 1: Start
- Step 2: Compute Entropy for entire Dataset
- Step 3: Select first node as root node and calculate the information Gain
- Step 4: repeat step 2 until all the available nodes are not assumed as root node.
- Step 5: Compare the Information Gain of each node and select the root node which is having highest information gain.
- Step 6: Divide the tree into sub nodes.
- Step 7: If sub nodes have all the similar labels
  - Then consider it as leaf node
  - Else recalculate the information gain for all possible attributes and select the attribute with maximum information gain.
- Step 8: Pruning if required.
- Step 9: Display the result.

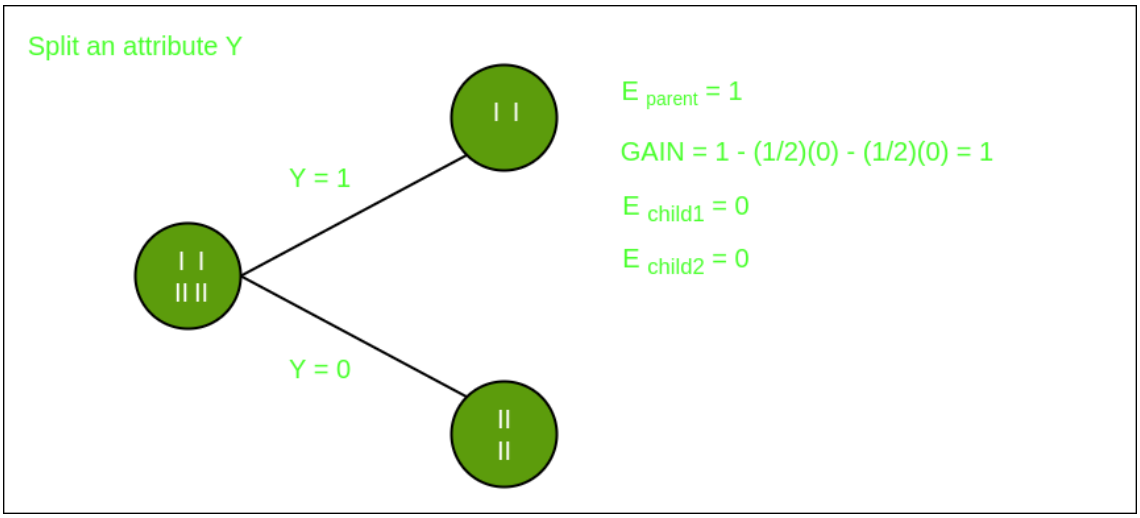
Example: Now, let us draw a Decision Tree for the following data using Information gain. **Training set: 3 features and 2 classes**

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

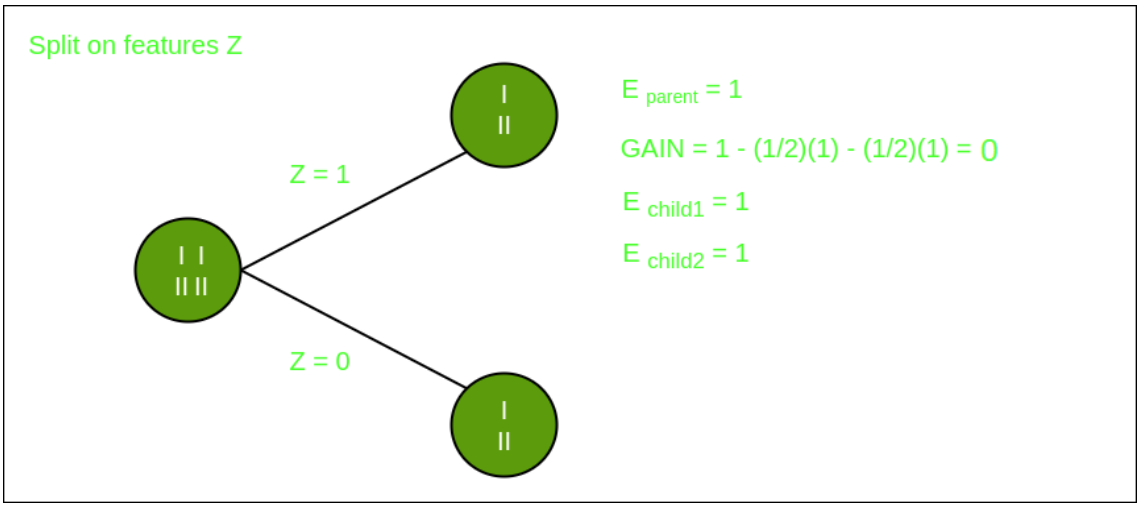
- Here, we have 3 features and 2 output classes. To build a decision tree using Information gain. We will take each of the features and calculate the information for each feature.



- **Split on feature X**

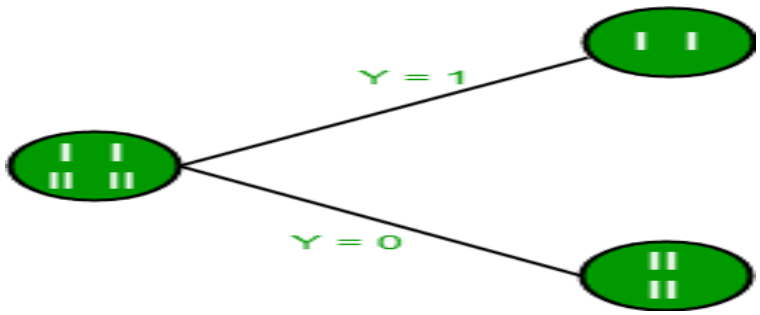


Split on feature Y



Split on feature Z

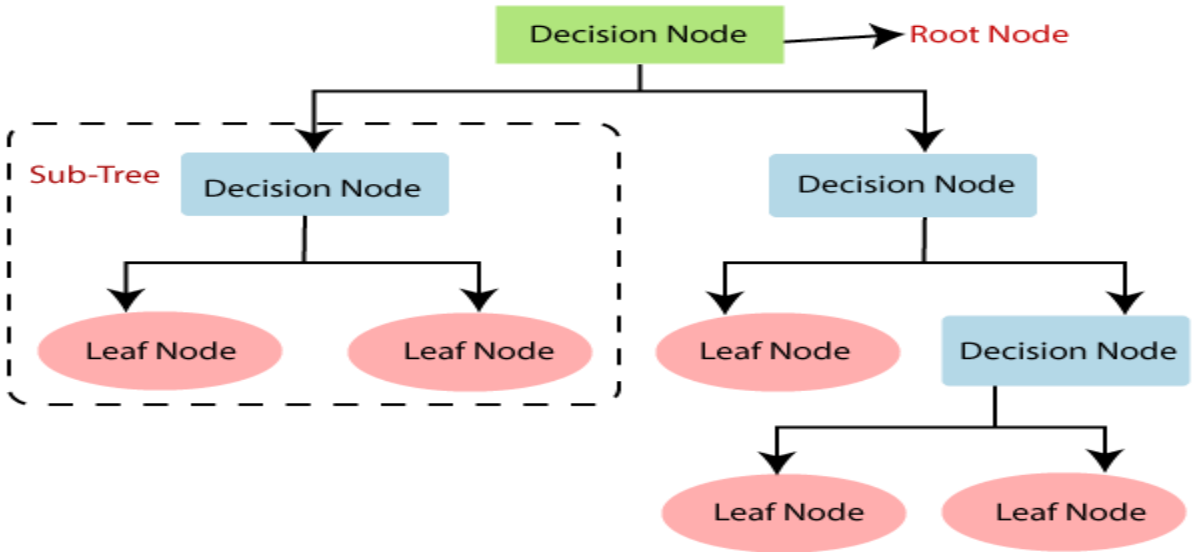
From the above images, we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best-suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains a pure subset of the target variable. So we don't need to further split the dataset. The final tree for the above dataset would look like this:



Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- 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.

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.

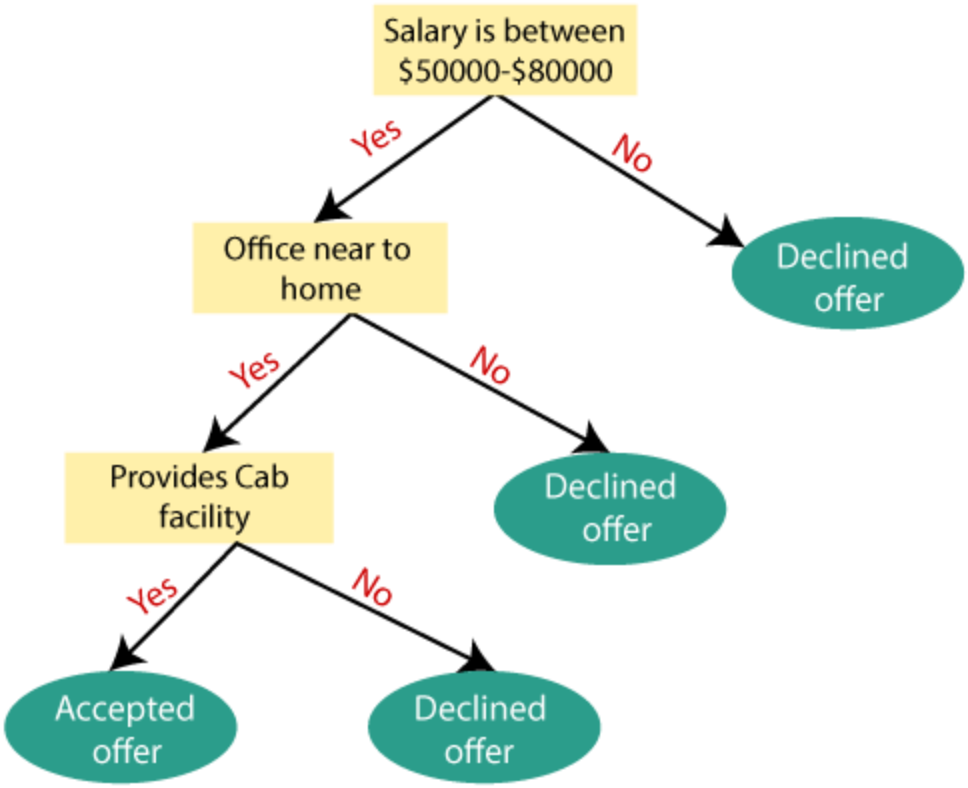
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **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.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



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**

1. 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:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

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

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. 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.
- Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning

tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- 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 of the Decision Tree

- 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.

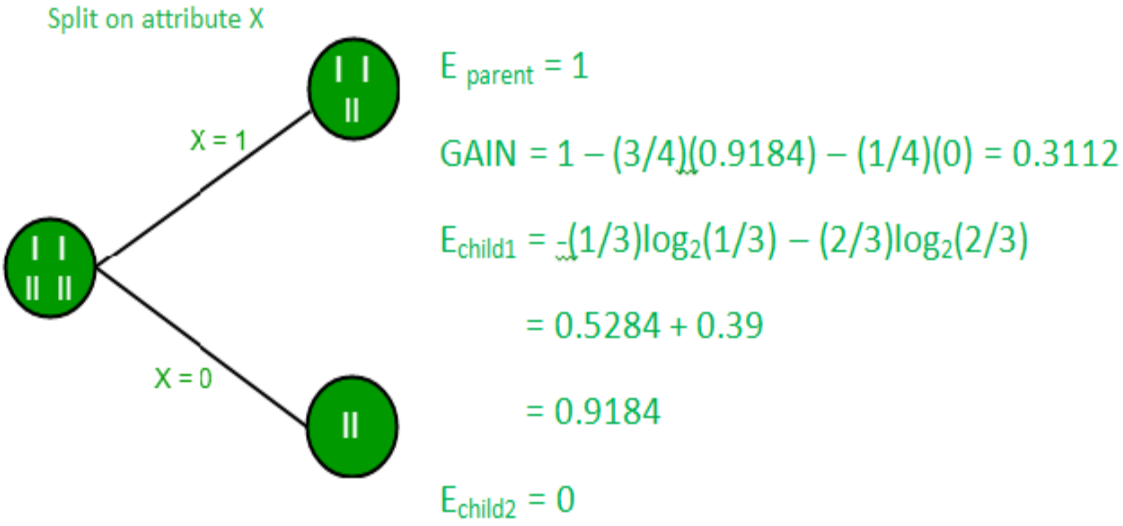
Cart Algorithm

- Step 1: Start
- Step 2: Compute Entropy for entire Dataset
- Step 3: Select first node as root node and calculate the information Gain
- Step 4: repeat step 2 until all the available nodes are not assumed as root node.
- Step 5: Compare the Information Gain of each node and select the root node which is having highest information gain.
- Step 6: Divide the tree into sub nodes.
- Step 7: If sub nodes have all the similar labels
  - Then consider it as leaf node
  - Else recalculate the information gain for all possible attributes and select the attribute with maximum information gain.
- Step 8: Pruning if required.
- Step 9: Display the result.

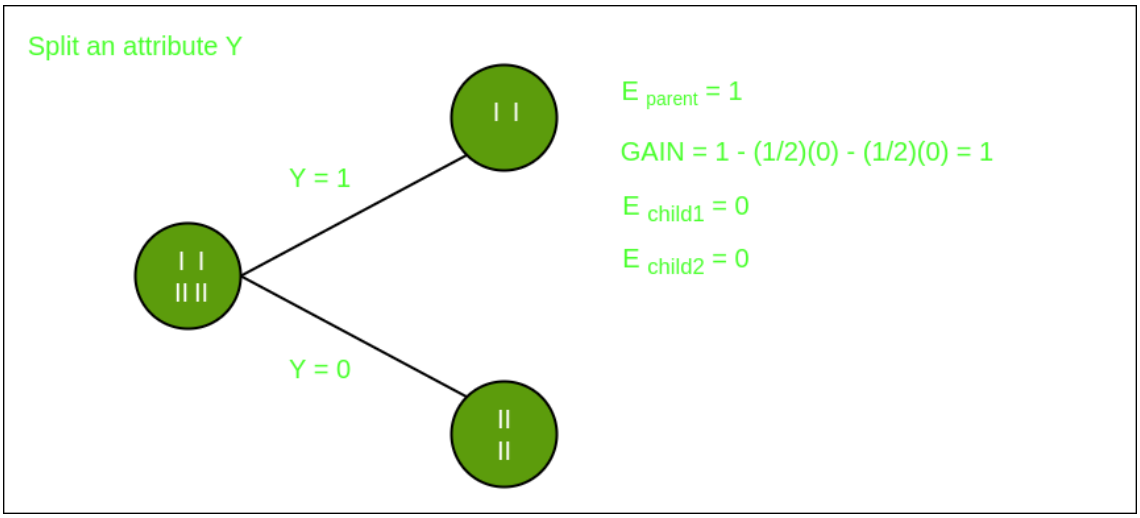
Example: Now, let us draw a Decision Tree for the following data using Information gain. **Training set: 3 features and 2 classes**

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

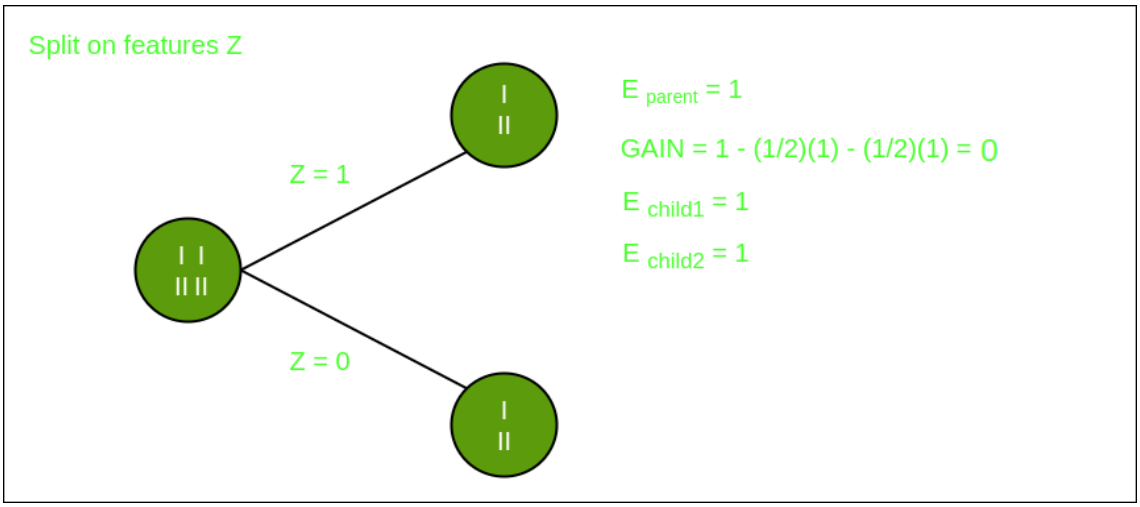
- Here, we have 3 features and 2 output classes. To build a decision tree using Information gain. We will take each of the features and calculate the information for each feature.



- **Split on feature X**

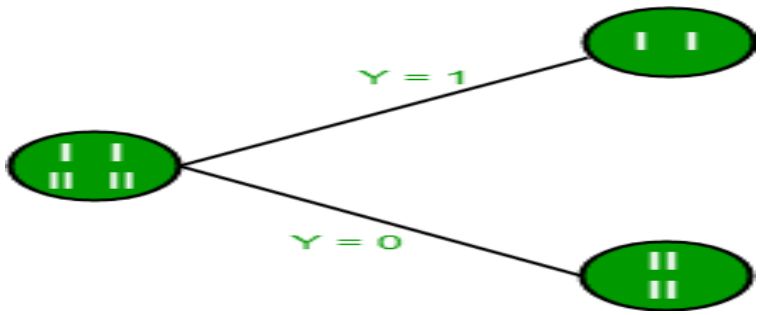


Split on feature Y



Split on feature Z

From the above images, we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best-suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains a pure subset of the target variable. So we don't need to further split the dataset. The final tree for the above dataset would look like this:



Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solvingclassification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building thefast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifyingarticles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

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

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2

Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)=

0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)=

P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that

P(Yes|Sunny)>P(No|Sunny)Hence on a Sunny day, Player can play

the game.

dvantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship betweenfeatures.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

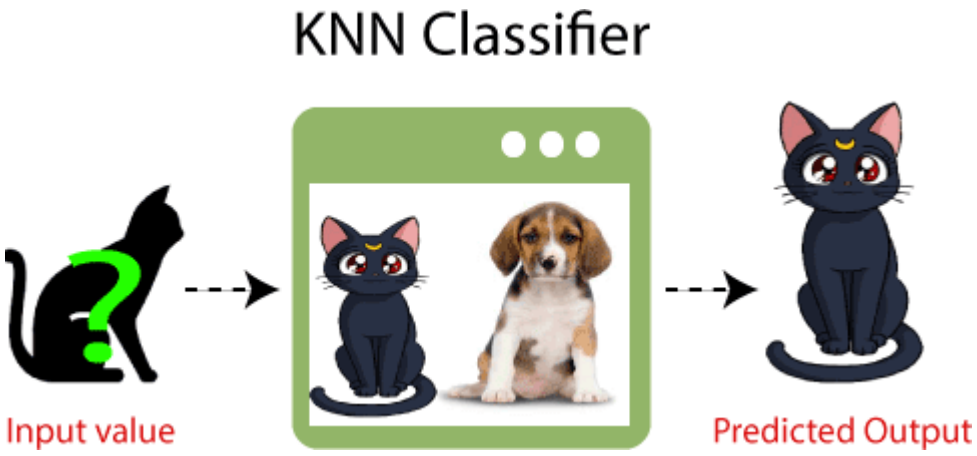
Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to whichcategory such
  - **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. Thismodel is also famous for document classification tasks.

K-Nearest Neighbor(KNN) Algorithm for Machine Learning

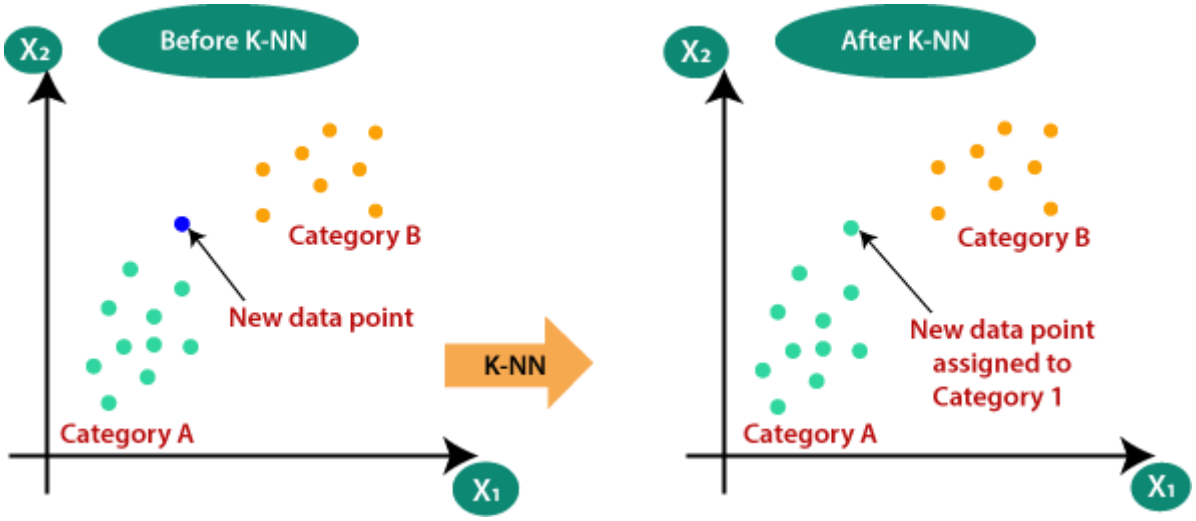
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learningtechnique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case intothe category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for theClassification problems.
- 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.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we

have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

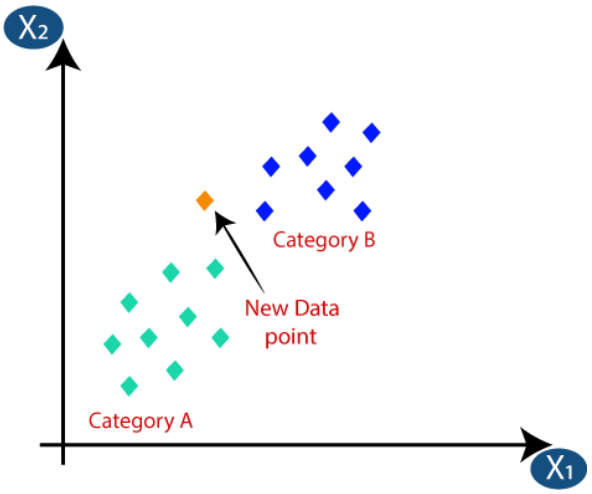


How does K-NN work?

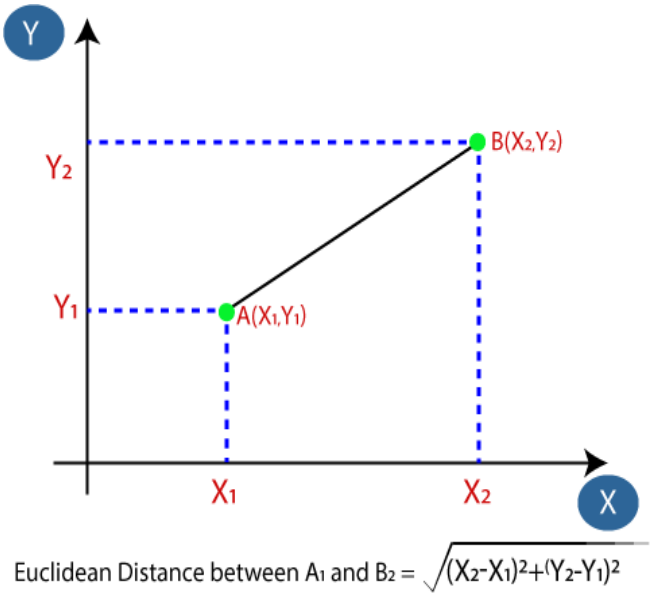
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

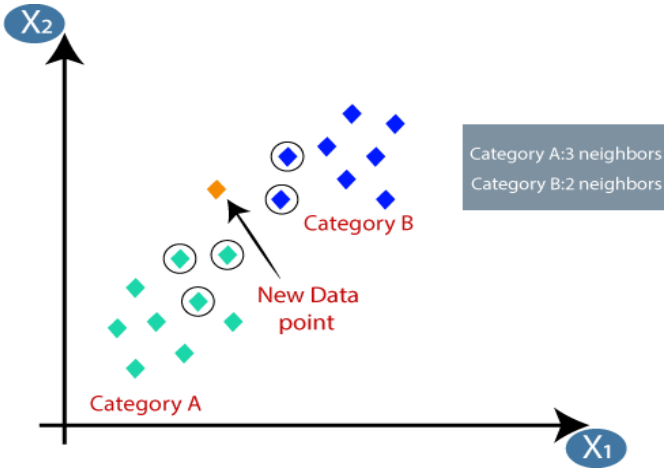
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will 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:



- 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 value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm: There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.

- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of the KNN Algorithm

- **Easy to implement** as the complexity of the algorithm is not that high.
- **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- **Few Hyperparameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

Disadvantages of the KNN Algorithm

- **Does not scale** – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.
- **Curse of Dimensionality** – There is a term known as the peaking phenomenon according to this the KNN algorithm is affected by the curse of dimensionality which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.
- **Prone to Overfitting** – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well. Hence generally feature selection as well as dimensionality reduction techniques are applied to deal with this problem.

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solvingclassification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building thefast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifyingarticles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

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

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2

Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)=

0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)=

P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that

P(Yes|Sunny)>P(No|Sunny)Hence on a Sunny day, Player can play

the game.

dvantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship betweenfeatures.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to whichcategory such
  - **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. Thismodel is also famous for document classification tasks.

What is Linear Discriminant Analysis?

Linear Discriminant Analysis (LDA) is a statistical technique for categorizing data into groups. It identifies patterns in features to distinguish between different classes. For instance, it may analyze characteristics like size and color to classify fruits as apples or oranges. LDA aims to find a straight line or plane that best separates these groups while minimizing overlap within each class. By maximizing the separation between classes, it enables accurate classification of new data points. In simpler terms, LDA helps make sense of data by finding the most effective way to separate different categories, aiding tasks like pattern recognition and classification.

Why Linear Discriminant Analysis (LDA)?

- Logistic Regression is one of the most popular linear classification models that perform well for binary classification but falls short in the case of multiple classification problems with well-separated classes. While LDA handles these quite efficiently.
- Linear Discriminant Analysis (LDA) also reduces the number of features in data preprocessing, akin to Principal Component Analysis (PCA), thereby significantly reducing computing costs.
- LDA is also used in face detection algorithms. In Fisherfaces LDA is used to extract useful data from different faces. Coupled with eigenfaces it produces effective results.

Shortcomings:

- Linear decision boundaries may not effectively separate non-linearly separable classes. More flexible boundaries are desired.
- In cases where the number of observations exceeds the number of features, LDA might not perform as desired. This is called *Small Sample Size* (SSS) problem. Regularization is required.

Assumptions

Linear Discriminant Analysis (LDA) makes some assumptions about the data:

- It assumes that the data follows a normal or Gaussian distribution, meaning each feature forms a bell-shaped curve when plotted.
- Each of the classes has identical covariance matrices.

However, it is worth mentioning that LDA performs quite well even if the assumptions are violated.

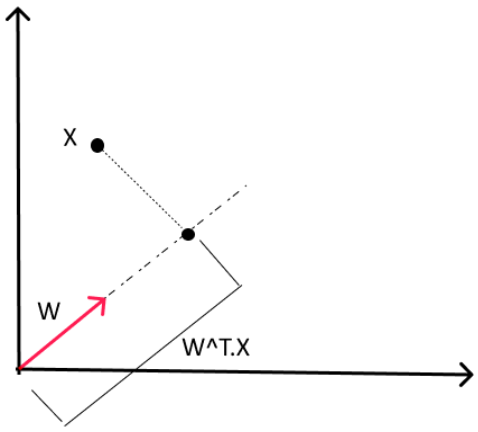
Fisher's Linear Discriminant:

Linear Discriminant Analysis (LDA) is a generalized form of FLD. Fisher in his paper used a discriminant function to classify between two plant species *Iris Setosa* and *Iris Versicolor.*

The basic idea of FLD is to project data points onto a line to maximize the between-class scatter and minimize the within-class scatter.

This might sound a bit cryptic but it is quite straightforward. So, before delving deep into the derivation part we need to get familiarized with certain terms and expressions.

- Let's suppose we have **d**-dimensional data points $x_1 \ldots . x_n$ with 2 classes $\mathbf{C_{i=1,2}}$ each having N_1 & N_2 samples.
- Let W be a unit vector onto which the data points are to be projected (took unit vector as we are only concerned with the direction).
- Number of samples : N = N1 + N2
- If x(n) are the samples on the feature space then WTx(n) denotes the data points after projection.
- Means of classes before projection: $\mathbf{m_i}$
- Means of classes after projection: $\mathbf{M_i = W^T m_i}$



Datapoint X before and after projection

Scatter matrix: Used to make estimates of the covariance matrix. IT is a m X m positive semi-definite matrix.

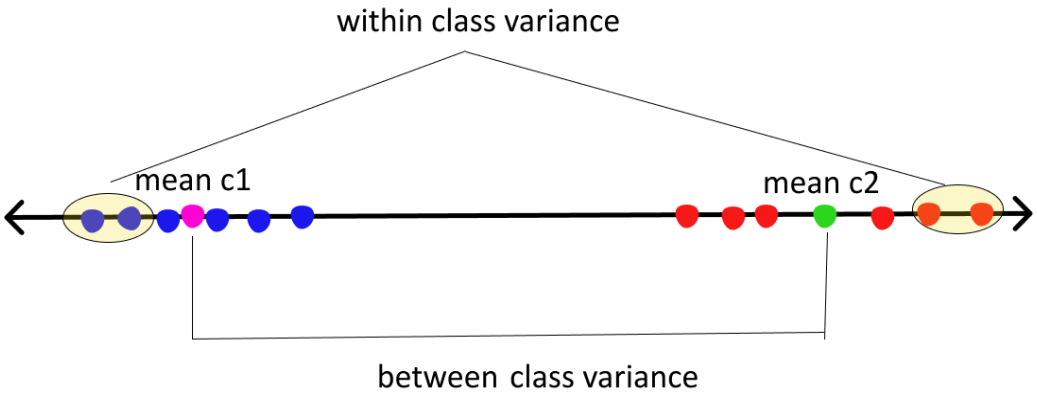
Given by: sample variance * no. of samples.

Note: Scatter and variance measure the same thing but on different scales. So, we might use both words interchangeably. So, do not get confused.

Two Types of Scatter Matrices

Here we will be dealing with two types of scatter matrices

- Between class scatter = $\mathbf{S_b}$ = measures the distance between class means
- Within class scatter = $\mathbf{S_w}$ = measures the spread around means of each class



Now, assuming we are clear with the basics let's move on to the derivation part.

As per Fisher's LDA :

arg max J(W) = $(M_1 - M_2)^2 / S_1^2 + S_2^2$ ..... (1)

The numerator here is **between class scatter** while the denominator is **within-class scatter**. So to maximize the function we need to maximize the numerator and minimize the denominator, simple math. To maximize the above function we need to first express the above equation in terms of W.

Numerator:

$$(M_1 - M_2)^2$$

$$= (W^T m_1 - W^T m_2)(W^T m_1 - W^T m_2)^T \quad = W^T (m_1 - m_2)(m_1 - m_2)^T W$$

$$= W^T S_b W \qquad \ldots\ldots\ldots \quad (2)$$

S_b = between class scatter

Denominator:

For denominator we have $\mathbf{S_1}^2 + \mathbf{S_2}^2$.

Class Scatter before projection

$$S_i = \sum_{x(n) \in C_i} (x(n) - m_i)(x(n) - m_i)^T$$

Scatter for projected samples:

$$S_i^2 = \sum_{x(n) \in C_i} (W^T x(n) - M_i)(W^T x(n) - M_i)^T$$

$$M_i = W^T m_i$$

With little re-arrangement we will get:

$$S_i^2 = W^T \left(\sum_{x(n) \in C_i} (x(n) - m_i)(x(n) - m_i)^T \right) W$$

$$S_i^2 = W^T S_i W$$

So,

$$S_1^2 + S_2^2 = W^T (S_1 + S_2) W = W^T S_W W \quad \text{..... (3)}$$

$S_W =$ within class scatter

Now, we have both the numerator and denominator expressed in terms of W

$J(W) = W^T S_b W / W^T S_w W$

Upon differentiating the above function w.r.t W and equating with 0, we get a generalized eigenvalue-eigenvector problem

$$\mathbf{S_b W = v S_w W}$$

$\mathbf{S_w}$ being a full-rank matrix , inverse is feasible

$$\mathbf{=> S_w^{-1} S_b W = v W}$$

Where **v** = **eigen value**

W = eigen vector

Evaluation Matrix of Regression

Machine learning is an effective tool for predicting numerical values, and regression is one of its key applications. In the arena of regression analysis, accurate estimation is crucial for measuring the overall performance of predictive models. This is where the famous machine learning library Python Scikit-Learn comes in. Scikit-Learn gives a complete set of regression metrics to evaluate the quality of regression models. In this article, we are able to explore the basics of regression metrics in scikit-learn, discuss the steps needed to use them effectively, provide some examples, and show the desired output for each metric. Regression Regression fashions are algorithms used to expect continuous numerical values primarily based on entering features. In scikit-learn, we will use numerous regression algorithms, such as Linear Regression, Decision Trees, Random Forests, and Support Vector Machines (SVM), amongst others. Before learning about precise metrics, let's familiarize ourselves with a few essential concepts related to regression metrics:

1. True Values and Predicted Values:

In regression, we've got two units of values to compare: the actual target values (authentic values) and the values expected by our version (anticipated values). The performance of the model is assessed by means of measuring the similarity among these sets.

2. Evaluation Metrics:

Regression metrics are quantitative measures used to evaluate the nice of a regression model. Scikit-analyze provides several metrics, each with its own strengths and boundaries, to assess how well a model suits the statistics.

Types of Regression Metrics

Some common regression metrics in scikit-learn with examples

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- R-squared (R^2) Score
- Root Mean Squared Error (RMSE)

1. **Mean Absolute Error (MAE)**

In the fields of statistics and machine learning, the Mean Absolute Error (MAE) is a frequently employed metric. It's a measurement of the typical absolute discrepancies between a dataset's actual values and projected values.

Mathematical Formula

The formula to calculate MAE for a data with "n" data points is:

Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=0}^{n-1}|y_i - \hat{y}_i|$$

Where:

- xi represents the actual or observed values for the i-th data point.
- yi represents the predicted value for the i-th data point.

2. **Mean Squared Error (MSE)**

A popular metric in statistics and machine learning is the Mean Squared Error (MSE). It measures the square root of the average discrepancies between a dataset's actual values and projected values. MSE is frequently utilized in regression issues and is used to assess how well predictive models work.

Mathematical Formula

For a dataset containing 'n' data points, the MSE calculation formula is:

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2$$

where:

- xi represents the actual or observed value for the i-th data point.
- yi represents the predicted value for the i-th data point.

3. **Root Mean Squared Error (RMSE)**

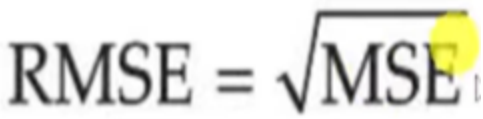
RMSE stands for Root Mean Squared Error. It is a usually used metric in regression analysis and machine learning to measure the accuracy or goodness of fit of a predictive model, especially when the predictions are continuous numerical values.

The RMSE quantifies how well the predicted values from a model align with the actual observed values in the dataset. Here's how it works:

1. **Calculate the Squared Differences:** For each data point, subtract the predicted value from the actual (observed) value, square the result, and sum up these squared differences.
2. **Compute the Mean:** Divide the sum of squared differences by the number of data points to get the mean squared error (MSE).
3. **Take the Square Root:** To obtain the RMSE, simply take the square root of the MSE.

Mathematical Formula

The formula for RMSE for a data with 'n' data points is as follows:



Where:

- RMSE is the Root Mean Squared Error.
- xi represents the actual or observed value for the i-th data point.
- yi represents the predicted value for the i-th data point.

4. **Relative Mean Square (RMS) or Relative Mean Squared Error (RMSE)**

Relative Mean Square Error normalizes the error by the observed values, providing a relative measure of prediction accuracy. It is useful when comparing errors across datasets with different scales.

Calculation:

- **Relative MSE**

$$\text{RelMSE} = \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y}_i)^2}$$

5. **The coefficient of variation**

The coefficient of variation (CV), also known as relative standard deviation (RSD), is a standardized measure of the dispersion of a probability distribution or frequency distribution. It helps us in understanding how the spread is the data in two different tests. Standard deviation is the most common measure of variability for a single data set. But why do we need yet another measure, such as the coefficient of variation? Well, comparing the standard deviations of two different data sets is meaningless, but comparing coefficients of variation is not.

$$CV\ (\%) = \left(\frac{Standard\ deviation}{Mean}\right) \times 100$$

6. **Sum of Squared Errors (SSE)**

   SSE is the sum of the squared differences between observed values and predicted values**.**

$$\text{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

7. **R-squared (R^2)**

R-squared (R^2) R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It is calculated as:

$$R^2 = SSE/SST$$

where SST (Total Sum of Squares) is the total variance in the observed data:

$$\text{SST} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Example Dataset

Observed (y_i)	Predicted ($\hat{y}_i$)
10	8
15	14
14	12
10	10
12	13

1. Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Calculations:

$$\begin{aligned} \text{MAE} &= \frac{|10 - 8| + |15 - 14| + |14 - 12| + |10 - 10| + |12 - 13|}{5} \\ &= \frac{2 + 1 + 2 + 0 + 1}{5} \\ &= \frac{6}{5} \\ &= 1.2 \end{aligned}$$

2. Mean Squared Error (MSE)

$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Calculations:

$$\begin{aligned}\text{MSE} &= \frac{(10-8)^2 + (15-14)^2 + (14-12)^2 + (10-10)^2 + (12-13)^2}{5} \\ &= \frac{2^2 + 1^2 + 2^2 + 0^2 + 1^2}{5} \\ &= \frac{4+1+4+0+1}{5} \\ &= \frac{10}{5} \\ &= 2\end{aligned}$$

3. Root Mean Square Error (RMSE)

$\text{RMSE} = \sqrt{\text{MSE}}$

Calculations:

$$\begin{aligned}\text{RMSE} &= \sqrt{2} \\ &\approx 1.414\end{aligned}$$

4. Relative Mean Squared Error (Relative MSE)

$$\text{Relative MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(y_i-\hat{y}_i)^2}{y_i^2}\right)$$

Calculations:

$$\text{Relative MSE} = \frac{1}{5}\left(\frac{(10-8)^2}{10^2} + \frac{(15-14)^2}{15^2} + \frac{(14-12)^2}{14^2} + \frac{(10-10)^2}{10^2} + \frac{(12-13)^2}{12^2}\right)$$

$$= \frac{1}{5}\left(\frac{4}{100} + \frac{1}{225} + \frac{4}{196} + \frac{0}{100} + \frac{1}{144}\right)$$

$$= \frac{1}{5}(0.04 + 0.0044 + 0.0204 + 0 + 0.0069)$$

$$= \frac{1}{5} \times 0.0717$$

$$= 0.01434$$

5. Coefficient of Variation (CV) of the Prediction Errors

$$\text{Coefficient of Variation} = \frac{\sigma_{\text{error}}}{\mu_{\text{observed}}}$$

First, calculate the prediction errors ($e_i = y_i - \hat{y}_i$):

Observed (y_i)	Predicted ($\hat{y}_i$)	Error (e_i)
10	8	2
15	14	1
14	12	2
10	10	0
12	13	-1

Mean of observed values (μ_{observed}):

$$\mu_{\text{observed}} = \frac{10 + 15 + 14 + 10 + 12}{5}$$

$$= \frac{61}{5}$$

$$= 12.2$$

Mean of the errors ($\bar{e}$):

$$\bar{e} = \frac{2 + 1 + 2 + 0 - 1}{5}$$

$$= \frac{4}{5}$$

$$= 0.8$$

Variance of the errors:

$$\text{Variance} = \frac{1}{5}\left((2 - 0.8)^2 + (1 - 0.8)^2 + (2 - 0.8)^2 + (0 - 0.8)^2 + (-1 - 0.8)^2\right)$$

$$= \frac{1}{5}\left(1.44 + 0.04 + 1.44 + 0.64 + 3.24\right)$$

$$= \frac{1}{5} \times 6.8$$

$$= 1.36$$

Standard deviation of errors (σ_{error}):

$$\sigma_{\text{error}} = \sqrt{1.36}$$

$$\approx 1.166$$

Coefficient of Variation:

$$\text{CV} = \frac{1.166}{12.2}$$

$$\approx 0.0956$$

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Calculations:

$$\begin{aligned}
\text{SSE} &= (10 - 8)^2 + (15 - 14)^2 + (14 - 12)^2 + (10 - 10)^2 + (12 - 13)^2 \\
&= 2^2 + 1^2 + 2^2 + 0^2 + 1^2 \\
&= 4 + 1 + 4 + 0 + 1 \\
&= 10
\end{aligned}$$

R-squared measures the proportion of variance in the dependent variable that is predictable from the independent variables.

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$$

Where SST (Total Sum of Squares) is calculated as:

$$\text{SST} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

First, calculate the mean of the observed values ($\bar{y}$):

$$\bar{y} = \frac{10 + 15 + 14 + 10 + 12}{5} = \frac{61}{5} = 12.2$$

Next, calculate SST:

$$\begin{aligned}
\text{SST} &= (10 - 12.2)^2 + (15 - 12.2)^2 + (14 - 12.2)^2 + (10 - 12.2)^2 + (12 - 12.2)^2 \\
&= (-2.2)^2 + 2.8^2 + 1.8^2 + (-2.2)^2 + (-0.2)^2 \\
&= 4.84 + 7.84 + 3.24 + 4.84 + 0.04 \\
&= 20.8
\end{aligned}$$

Now calculate R^2:

$$\begin{aligned} R^2 &= 1 - \frac{\text{SSE}}{\text{SST}} \\ &= 1 - \frac{10}{20.8} \\ &= 1 - 0.4808 \\ &\approx 0.5192 \end{aligned}$$

Cross-Validation in Machine Learning

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. ***We can also say that it is a technique to check how a statistical model generalizes to an independent dataset***.

In machine learning, there is always the need to test the stability of the model. It means based only on the training dataset; we can't fit our model on the training dataset. For this purpose, we reserve a particular sample of the dataset, which was not part of the training dataset. After that, we test our model on that sample before deployment, and this complete process comes under cross-validation. This is something different from the general train-test split.

Hence the basic steps of cross-validations are:

- Reserve a subset of the dataset as a validation set.
- Provide the training to the model using the training dataset.
- Now, evaluate model performance using the validation set. If the model performs well with the validation set, perform the further step, else check for the issues.

Methods used for Cross-Validation

There are some common methods that are used for cross-validation. These methods are given below:

1. **Validation Set Approach**
2. **Leave-P-out cross-validation**
3. **Leave one out cross-validation**

4. **K-fold cross-validation**
5. **Stratified k-fold cross-validation**

Validation Set Approach

We divide our input dataset into a training set and test or validation set in the validation set approach. Both the subsets are given 50% of the dataset.

But it has one of the big disadvantages that we are just using a 50% dataset to train our model, so the model may miss out to capture important information of the dataset. It also tends to give the underfitted model.

Leave-P-out cross-validation

In this approach, the p datasets are left out of the training data. It means, if there are total n datapoints in the original input dataset, then n-p data points will be used as the training dataset and the p data points as the validation set. This complete process is repeated for all the samples, and the average error is calculated to know the effectiveness of the model.

There is a disadvantage of this technique; that is, it can be computationally difficult for the large p.

Leave one out cross-validation

This method is similar to the leave-p-out cross-validation, but instead of p, we need to take 1 dataset out of training. It means, in this approach, for each learning set, only one datapoint is reserved, and the remaining dataset is used to train the model. This process repeats for each datapoint. Hence for n samples, we get n different training set and n test set. It has the following features:

- In this approach, the bias is minimum as all the data points are used.
- The process is executed for n times; hence execution time is high.
- This approach leads to high variation in testing the effectiveness of the model as we iteratively check against one data point.

K-Fold Cross-Validation

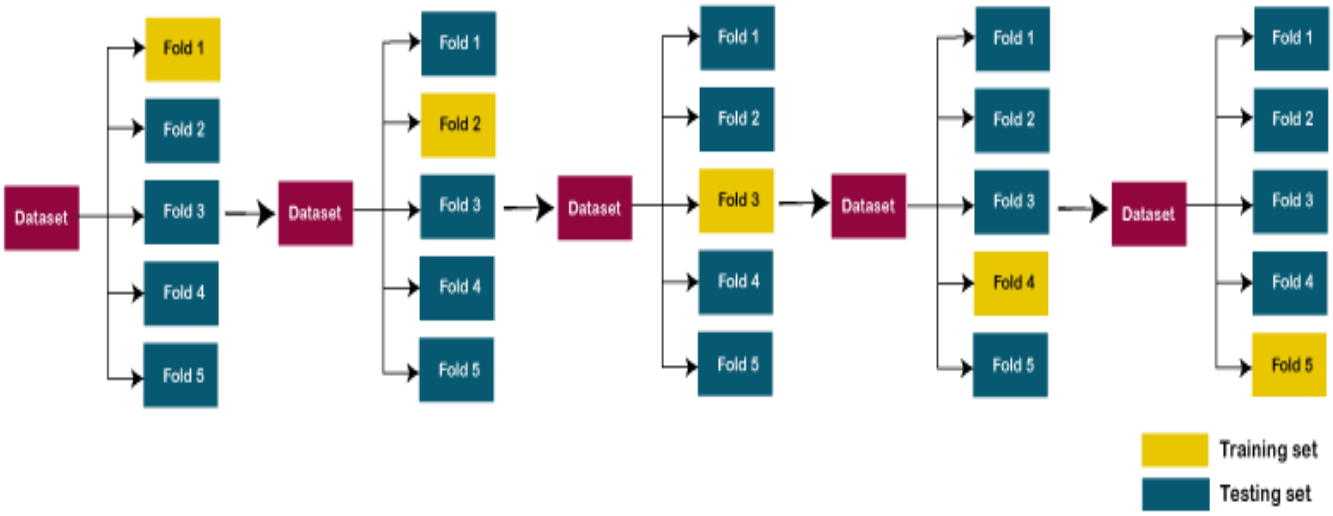
K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.

The steps for k-fold cross-validation are:

- Split the input dataset into K groups
- For each group:
    - Take one group as the reserve or test data set.
    - Use remaining groups as the training dataset
    - Fit the model on the training set and evaluate the performance of the model using the test set.

Let's take an example of 5-folds cross-validation. So, the dataset is grouped into 5 folds. On 1st iteration, the first fold is reserved for test the model, and rest are used to train the model. On 2nd iteration, the second fold is used to test the model, and rest are used to train the model. This process will continue until each fold is not used for the test fold.

Consider the below diagram:



Stratified k-fold cross-validation

This technique is similar to k-fold cross-validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.

It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k-fold cross-validation technique is useful.

Holdout Method

This method is the simplest cross-validation technique among all. In this method, we need to remove a subset of the training data and use it to get prediction results by training it on the rest part of the dataset.

The error that occurs in this process tells how well our model will perform with the unknown dataset. Although this approach is simple to perform, it still faces the issue of high variance, and it also produces misleading results sometimes.

Comparison of Cross-validation to train/test split in Machine Learning

- **Train/test split:** The input data is divided into two parts, that are training set and test set on a ratio of 70:30, 80:20, etc. It provides a high variance, which is one of the biggest disadvantages.
    - **Training Data:** The training data is used to train the model, and the dependent variable is known.
    - **Test Data:** The test data is used to make the predictions from the model that is already trained on the training data. This has the same features as training data but not the part of that.
- **Cross-Validation dataset:** It is used to overcome the disadvantage of train/test split by splitting the dataset into groups of train/test splits, and averaging the result. It can be used if we want to optimize our model that has been trained on the training dataset for the best performance. It is more efficient as compared to train/test split as every observation is used for the training and testing both.

Applications of Cross-Validation

- This technique can be used to compare the performance of different predictive modeling methods.
- It has great scope in the medical research field.
- It can also be used for the meta-analysis, as it is already being used by the data scientists in the field of medical statistics.
-