MACHINE LEARNING

What is Machine learning?

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

Machine learning is the concept that a computer program can learn and adapt to new data without human intervention.

- Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.
- The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

In simple words, machine learning is a process of "helping the machine learn how to make decisions logically."

- It makes use of data and algorithms to learn
- And then it is retrained on similar data to improve its accuracy

Machine learning is a part of Artificial Intelligence.

If we wish to recognize an object in a picture, programmers used to have to develop code for each object they intended to recognize, such as a human, a cat, or a vehicle. This isn't a scalable strategy. Today, thanks to machine learning methods, a single system can learn to recognize both by simply exposing it to a large number of examples of both.

{For example, the algorithm can figure out whether a dog is a dog by looking at examples of photographs labelled "this is a dog" or "this is not a dog," and being corrected whenever it makes a mistake about the object in the picture. Then, when presented with a fresh collection of pictures, it begins to recognize cat photos in the new set, just as a kid learns to distinguish between a cat and a dog.}

{For example, **Google spam filter** in this type of machine learning model the algorithm searches some particular keywords (i.e. Free, --%OFF, spam job requirements, and even some particular senders) and it sends them to the spam folder. }

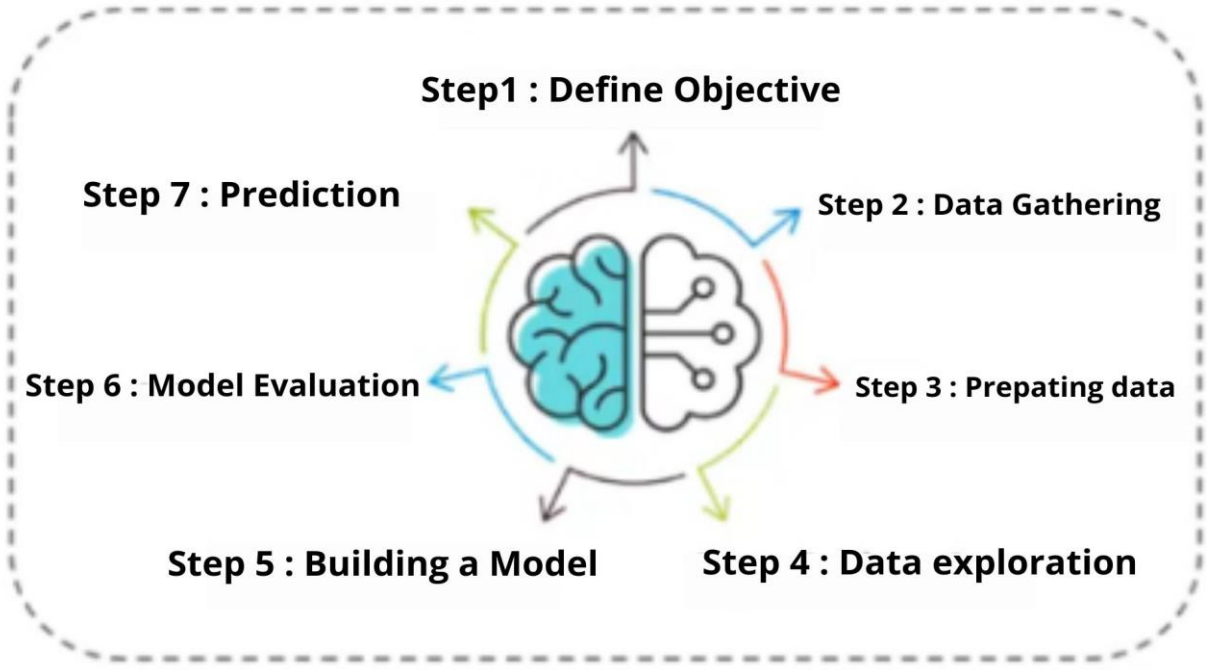
Companies such as Netflix & Amazon build such Machine Learning models by using tons of data in order to identify profitable opportunities and avoid unwanted risks.

Machine Learning Applications

Here are some popular applications:

- **Virtual AI:** Siri, Cortana, Alexa, and Google Assistant.
- **Finance stock marketing**: Prediction of stock prices going up or down.
- **Social Media platforms**: Youtube recommendations, Facebook pages recommendations, etc
- **Retail sector**: Analyzing the sales of a particular product.
- **Customer services sector**: Such as chatbots, etc
- **Health platforms :** Best example is prediction of covid vaccine
- **Search engines**: When you search on Google, the backend keeps an eye on whether you clicked on the first result or went on to the second page -– the data is used to learn from mistakes so that relevant information can be found quicker next time.

Steps in Machine Learning



Step 1: Define the objective of the problem statement:

This is a process of identifying the problem we want to solve and the business benefits we want to obtain.

How to do it? We must be able to ask ourselves a lot of questions, more importantly: the right questions.

The Golden Rule to define a project goal is to ask and refine "sharp" questions that are relevant, specific, and unambiguous; “How can I increase my profit?” is not a good question for any machine learning solution, “which kind of car in my fleet is going to fail first?” or “How much energy my production plant will consume in the next quarter?” are stronger examples of sharp questions.

Step 2: Data gathering:

Data collection is the process of gathering and measuring information from countless different sources. In order to use the data we collect to develop practical artificial intelligence (AI) and machine learning solutions, it must be collected and stored in a way that makes sense for the business problem at hand

Step 3: Preparing data:

The data collected is almost never in the right format. You'll encounter a lot of inconsistencies in the data sets such as missing values, redundant variables, duplicate values, outliers, etc. Removing such inconsistencies is very essential because the might lead to wrongful prediction.

Therefore, at this stage, you scan the data for any inconsistencies and fix them before making any prediction.

Step 4: Data Exploration:

This step is all about diving deep into the data and finding all the hidden relationships between each value, It is also known as EDA or Exploratory Data Analysis. EDA involves understanding the patterns and trends in the data.

For Example: In the case of predicting rainfall, we know that there is a strong possibility of rain if the temperature has fallen low, such correlation must be understood and mapped at this stage.

Step 5: Building a model:

All the insights and patterns derived during the EDA are used to build the Machine Learning model. This stage always begins by splitting the data into two parts (Training data and Testing data)

Training data will be used to build and analyze the model. The logic of the model is based on the Machine Learning Algorithm that is being implemented.

For example: In the case of predicting rainfall, the output will be in the form of True (if it will rain tomorrow) or False (If no rain).

Step 6: Model Evaluation

After building a model by using the training data set, it is finally time to put the model to a test, the testing data set is used to check the efficiency of the model and how accurately it can predict the outcome.

Step 7: Prediction:

Once the model is evaluated and improved, it is finally used to make predictions. The final output can be a Categorical variable (eg: True or False) or it can be a continuous quality (eg: the predicted value of a stock).

Types of Machine Learning

The commonly used types of Machine Learning are:

1. Supervised Learning
2. Unsupervised Learning
3. Semi-Supervised Learning
4. Reinforcement Learning

Supervised Learning:

Supervised learning is a technique in which we teach or train the machine using data that is well labeled.

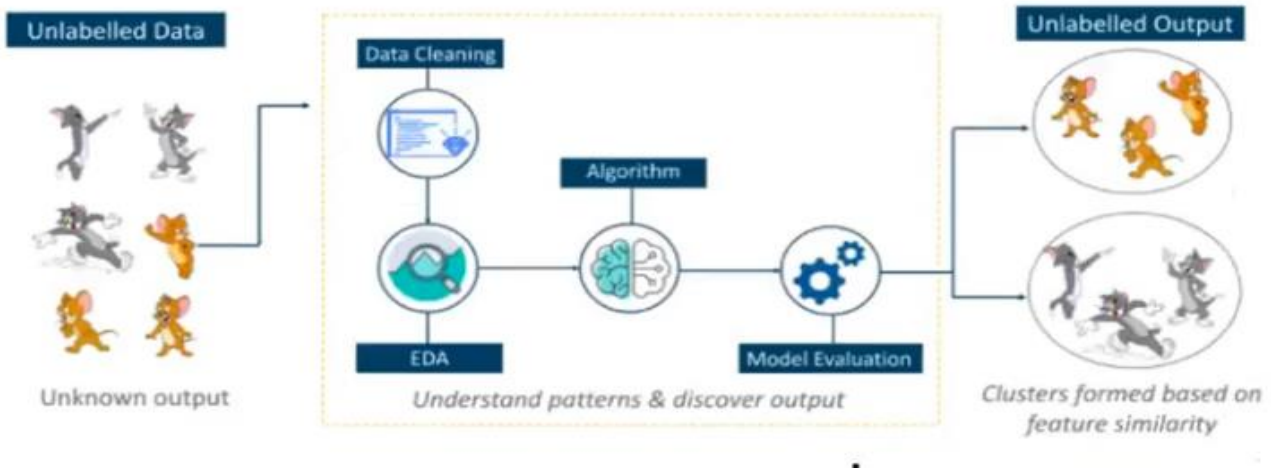
The labeled data represents the class/category each observation belongs to.

For example: As you can see in the above diagram here the data which is being used to teach the model is clearly labeled and with various examples of the pictures of the classes and the model is given a testing photo of Tom and Jerry and checked the accuracy of the prediction.

Unsupervised Learning:

Unsupervised learning is a technique in which we teach or train the machine using data that is not labeled, Think of it as a smart kid who learns without any guidance, Here we use unlabelled data and allow the Machine Learning model to correlate between every data class and understand the underlying patterns and predict the output according to its own understanding.

Most of the data available in the world are unlabelled and hence it makes an important type of Machine Learning model.



For example: As shown in the above figure the data is unlabelled and the model figures out the classes available in the data by its own and the model distinguishes between Tom and Jerry in the form of clusters.

Reinforcement Learning

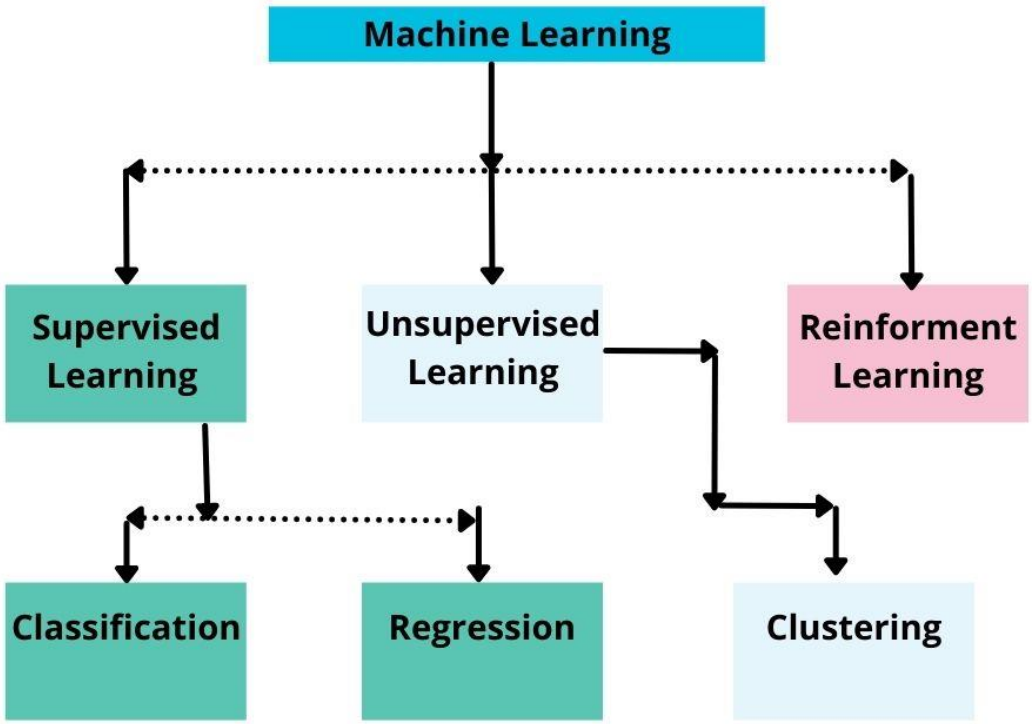
Reinforcement learning is all about making decisions sequentially. In simple words, we can say that the output depends on the state of the current input and the next input depends on the output of the previous input.

In Reinforcement learning decision is dependent, so we give labels to sequences of dependent decisions

For example: Imagine you are learning to drive a vehicle and you have fallen, what would you do? Stop learning? Break the bike?

Yes at first everyone would but as time passes you'll slowly learn how not to fall and slowly start practicing and ask one of the experienced friends or family's take their input and you try to improve your driving skills, This is exactly how reinforcement learning works, Depending upon the previous output the model tries to make itself better after getting the input from the previous outputs.

Types of Problems in Machine Learning



There are three main types of problems that can be used to solve in Machine Learning

Classification:

In this type the output is a categorical variable, classifying emails in two classes such as spam or not spam, classification problem can be solved by using supervised learning algorithms such as Support Vector Machines, Naïve Bayes, K Neighbour, etc.

Regression:

In this type the output is a continuous variable, Stock prices prediction and a very simple example would be the kilometres a bike would run on the given amount of fuel. This is a type of supervised Machine Learning model and such problems can be solved using various Models such as Linear Regression Model, Logistic Regression Model.

Clustering:

This type of problem involves assigning the input into two or more clusters based on the feature similarity, for example, clustering the viewers into similar groups based on their interests, age, geography, etc. It can be done by using Unsupervised Learning algorithms such as K-means clustering.

Data Pre-processing

Data pre-processing is a process of preparing the unusable raw data and converting it into a usable format, it is a very important step

It's an ideal condition that we receive and clean and completely formatted data, and it is very important that we clean the data in a formatted way and remove any impurities from the data if there are any.

The aim is to process the data and make it ready for model creation stage.

There are various steps involved in pre-processing:

- ✓ Handling the missing values
- ✓ Treating the outliers
- ✓ Scaling the dataset
- ✓ Encoding the categorical variables
- ✓ Splitting the data into Training and Testing
- ✓ Standardizing the data

Handling the Missing Values

- All real-world datasets are incomplete. Data can be missing due to non-response, lost data or some skip patterns. Most of the features have some or the other missing values.
- These missing values can be divided into as follows:
    - Missing completely at random (MCAR)
    - Missing at random (MAR)
    - Missing not at random (MNAR)

Missing Completely at Random (MCAR)

- ✓ When there is no systematic relation between the observed and the missing values, such values are called Missing Completely at Random.
- ✓ Such values are missing purely by chance. They are not linked or associated to any other value. Such a missing value does not harm the model with biasedness.
- ✓ But yes, these values do need to be handled as they may lead to loss of power due to absent values.

Example:

As you can see that there are some random values which are missing in the table

These values need to be filled if we are working on the model.

Gender	Salary(LPA)
M	200000
F	650000
M	
F	760000
F	
M	890000

Missing at Random (MAR)

- When there is a systematic relationship between the observed and the missing values, such values are called missing at random
  Example: General observations:
- Men are more likely to share age as compared to women.
- Hence we can observe that more missing values of age are coming from the class: women. There is a systematic relation between this missing value and the observed gender values.

As you can see in the table female's age detail is not complete, but there are some females who are comfortable by letting their age known.

Gender	Salary(LPA)	Age
M	200000	20
F	650000	
M	450000	22
F	760000	
F	1000000	
M	890000	35
F	600000	28
M	500000	29
F	560000	

Missing not at random (MNAR)

- When there is a systematic relationship between missing values and 'unobserved' values, such values are called missing not at random.
- This implies that the missingness is related to a factor which are not measured by the researcher
- Example:
  Some survey of employees is done. Most of the data that is missing is for the employees who are on sick leaves.
- Hence the missing values are caused due to this 'unobserved' fact.
- MNAR analysis is complicated as it involves dependences on unrecorded information.

As you can that most of the data is missing as people were sick and the survey was not completed

Gender	Salary(LPA)	Cause
M	200000	
F	650000	
M	450000	
F	760000	
F		Covid
M		Covid
F		Covid
M		Covid
F		Covid

Methods of filling the missing values

- Mean / median / mode replacement: (mode works good for categorical dataset, median is good for outliers dataset , and mean is good for normal data)
- Random sample imputation: Use for quantitative columns. We use sample and then match the index to get the imputation done
- Capturing NaN value with a new feature (we add a new column where Nan is replaced by 1 and rest all rows are replaced by 0)
- End of Distribution: Here the NaN values are replaced by the values that have Z = -3 or +3.
- Arbitrary value imputation: Manually select any arbitrary value and impute it , it may be least value or highest value of the dataset
- Frequent category imputation: Used for categorical dataset, most frequent label is used to fill all missing values
- Treat nan value of categorical as a new category (like ‘missing’)
- Using KNN imputation
- Dropping all NaN values: this is used when the missing values are more than 80% of the data.

Mean, Median, and Mode Imputation

- Mean can be used to fill the missing values when the variable is numerical and it is a normal distribution
- Median can be used to fill the missing values when the variable is numerical and it is skewed (not normally distributed)
- Mode can be used to fill missing values when the variable is categorical

Random sample imputation

- In this method, a random value from the existing set of values is taken and used to fill the missing values.
  Advantages:
- Easy and variance is same as the original dataset

Capturing NAN values

- This method is used where the data is missing due to some cause. That is the data is not missing because of being completely random.
- Here a new feature is created in the dataframe, in that feature 1 is stored for missing value and 0 is stored otherwise.

```
Df[‘C_NAN’] = np.where(df[‘C’].isnull(),1,0)
```

End of Distribution

- This is a special method, where the aim is to fill the missing values with some extreme value of the feature.
- This method is specifically helpful when there are more extreme values in the dataset and the aim is to include those values in the model.
- To find the extreme values use : Mean + /- 3* standard deviation
- Even Z score method can be used. Z = -3 or +3.

Arbitrary value imputation

- This technique was invented by people in Kaggle competition.
- Here each NaN value is replaced by an arbitrary value.
- The decision about the arbitrary value is purely judgment-based.
- It must be a value that is not very frequent in the dataset
- Generally, it can be something like the minimum or the maximum value of the dataset

Frequent category imputation

- When the variable is categorical, the best way to fill the missing values is with the most popular class.
- This easy and fast way to handle categorical missing values
- Disadvantage: this value may end up over-representing the dataset

Treating NaN as a new category

- This is done in order to fill the values by a new category that can indicate that the values are missing.
- So if there are already say three categories in that feature a fourth category is added for all the missing cells. It can be labeled as simple as ‘missing’

Using KNN imputation

- KNN is a machine learning model that relates to the prediction of classes based on the K nearest neighbors.
- The same model or logic can be employed to even fill the missing categorical values

Drop NaN Values

- While dealing with the data set if there are missing values of almost 80% of the data then it is a better option to drop such tables.

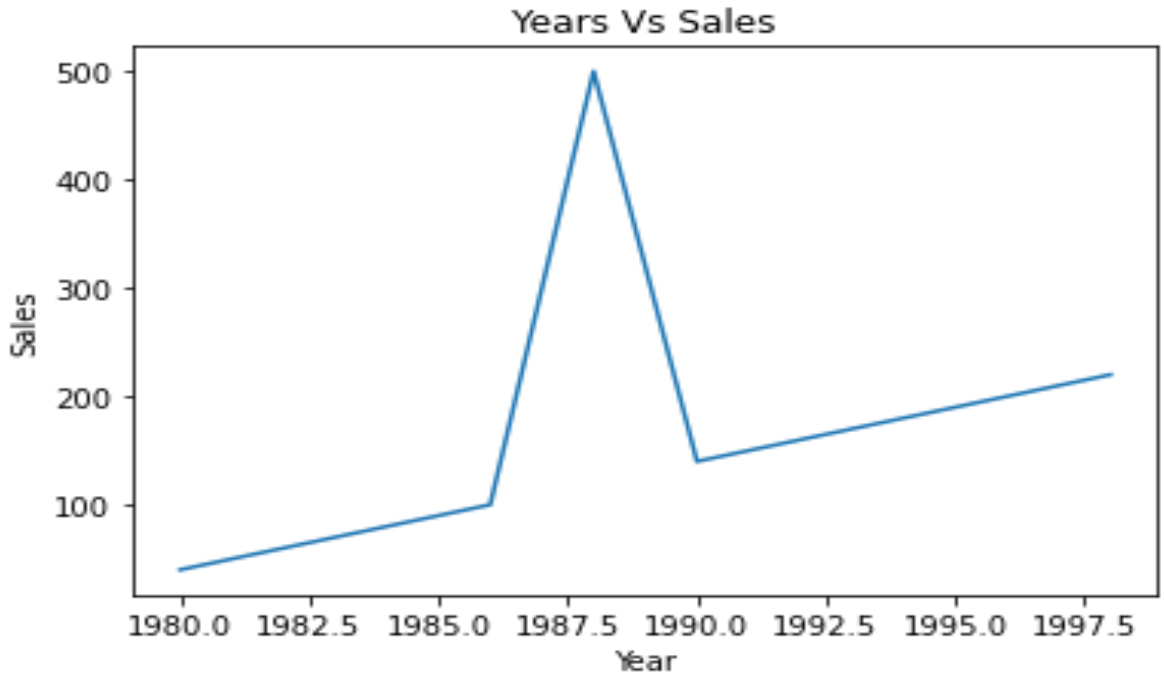
OUTLIERS

A value that differs from the rest of the set of values is called as an outlier. These are the extreme values that are present in the data and should be removed as they can cause lots of miss-prediction in the Machine Learning model.

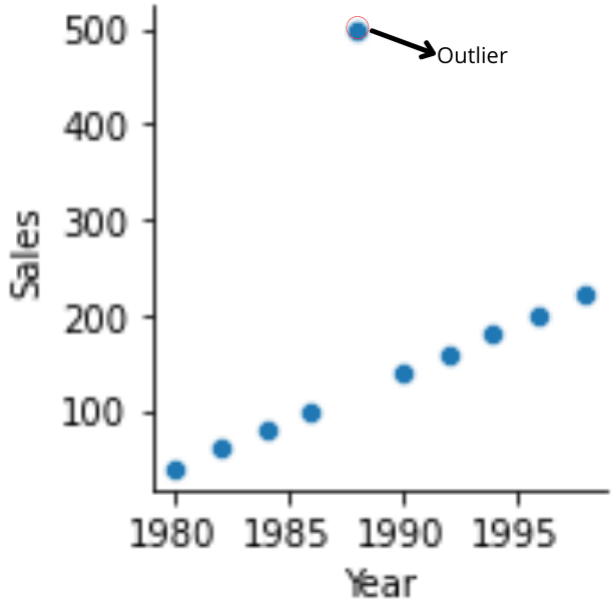
- They are either caused by human error or some technical error which is recorded in the data set.
- The best way to visualize an outlier is through plotting an box plot

An outlier must be interpreted properly and the reason of the outlier must be understood accordingly.

Year	Sales
1980	40
1982	60
1984	80
1986	100
1988	500
1990	140
1992	160
1994	180
1996	200
1998	220



Using Line graph



Using pairplot

As you can see in the above picture in the year 1988 there is a sudden change in the value, this is an example of an outlier, and this clearly shows how a single outlier can make the data so unreliable.

Detecting outliers

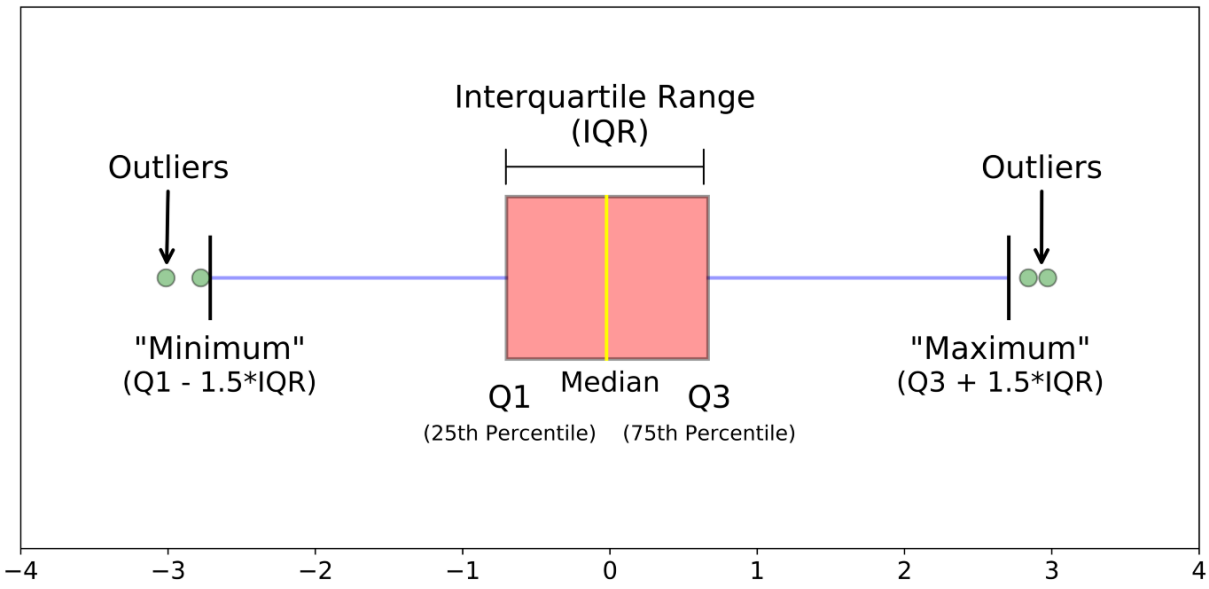
There are many ways we can detect outliers, some of the commonly used techniques to detect outliers are:

- Visualizing the data
- Standard deviation
- Box plots
- Z Score method
- IQR Method
- Hypothesis testing

What should we do with outliers? Should you drop them completely?

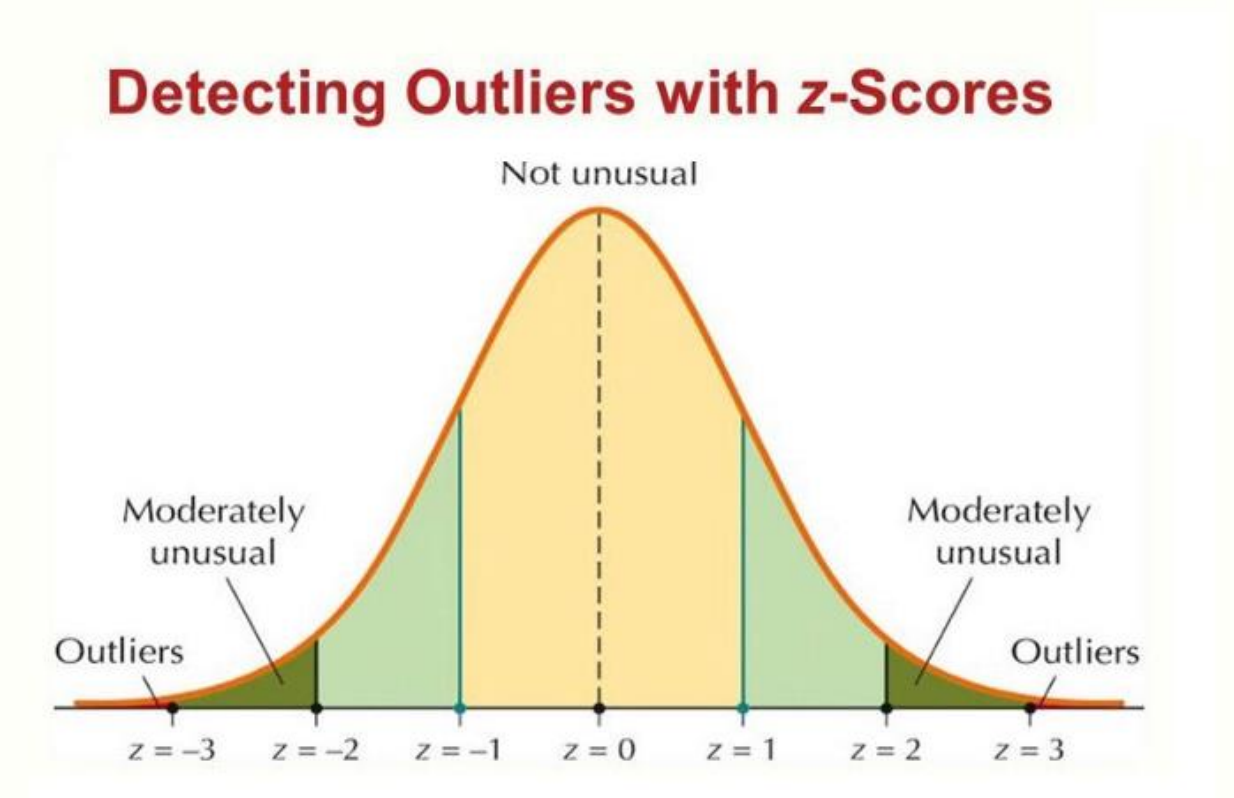
- The answer to this question is not as simple, it purely depends upon the cause the outlier is present and what type of data you are handling.
- We can always use the trial and error method we can drop them and compare the results which we get with and without them
- If you observe that they are incorrectly entered or measured you can drop such outliers.
- If the predictions of the model are getting affected you can certainly drop them.

IQR Method



- When the dataset is skewed or non-normal, IQR method of handling outliers is used.
- Accordingly following defines an outlier:
  Lower outliers = Q1 – 1.5*IQR
  Upper outliers = Q3 + 1.5*IQR
- A code can be created around similar steps to identify the outliers.

Z- Score Method



- This method is used when the dataset follows approximately normal distribution
- Accordingly, any value that has a Z value less than -3 or greater than +3 is called an outlier.
- Convert each value in to Z score and then filter out the values with magnitude greater than 3

Hample Method

- This method is used when the dataset is skewed.
- When IQR method is used, many values are lost. Hampel's method finds a smaller set of values as outliers.
- Accordingly:
  First: find the median of the dataset (Me)
  Second: find the deviations of all values from the median
  Third: find the median of these deviations (MD)
- Outliers = Me +/- 4.5*md

DB Scan Method

- DBSCAN: Density-based spatial clustering of applications with noise (used for multimodal dataset) separates clusters of high density with clusters of low density
- There is one group that is densely filled with data points and then there is another group that has a low density of data points.
- DBSCAN is a method to separate these two groups of different densities

Scaling

Let us take an example of measuring the distance from point A to point B, Some of the data which is available is in different scales such as kilometre's and miles, And if we want to add the complete distance we would have to convert either kilometre to miles or vice versa and then take the total of the distance, Similarly, we'll be having different types of data available we'll have to scale them in a uniform format and then use it.

Feature Scaling Methods

- When different features vary over different ranges, it becomes difficult to put them together in a model. This is more complex specifically when the model is based on distances.
- Hence in such a case it is advised to scale down the features in such a way that all the features fall under the same range and hence can be readily used for model creation
- This process of bringing all the features together in the same range is called feature scaling.

Different Methods of Scaling

There are many ways we can scale data some commonly used methods are:-

- Absolute Maximum Scaling
- Min-Max Scalar
- Normalization
- Standardization
- Robust Scaling

Normalization

- In the methods before, the range of the value was being altered.
- In this method of normalization, the aim is to change the shape of the feature distribution.
- Scaled value = (X – X mean) / (X max – X min)
- Note: here the X min in the numerator is replaced by the X mean value

Min-Max Scalar

- This method follows the below formula for scaling the values.
- Scaled value = (X – Xmin) / (X max – X min)
- The values created with this method lie in the range 0 to 1
- But still, the values are prone to outliers

Code

```
from sklearn.preprocessing import MinMaxScaler

Scalar = MinMaxScaler()

Scalar.fit_transform(df['Age','Salary'])
```

Absolute Maximum Scaling

- According to this method, every value of the feature gets divided by the maximum value of the feature.
- This converts all the values between 0 and +1.
- This method is quite prone to outliers

Code

```
from sklearn.preprocessing import MaxAbsScaler

transformer = MaxAbsScaler().fit(X)

transformer.transform(X)
```

Standardization

- This method converts each value of the feature to a Z score
- Z = (X – Xmean) / X std
- This method of scaling is best suited when the dataset follows normal distribution

Code

```
from sklearn.preprocessing import StandardScalar

Std = StandardScalar()

Std.fit_transform(df['Age','Salary'])
```

Encoding the dataset

In any dataset, there are several categorical features. Since machine learning model may fail to incorporate these variables it is important to convert them into some simple numerical codes.

This process of converting the categories or classes into simple codes is called encoding the dataset.

Methods of Encoding

There are two types of encoding: nominal encoding and ordinal encoding

Nominal encoding

It has three types:

OneHotEncoding: For each feature a number of features are created to encode. For example if a feature has two classes, then two new features are created. One feature captures the presence of first class and second feature captures the presence of second class.

```
Pd.get_dummies()

From sklearn.preprocessing import OneHotEncoder
```

OneHotEncoding with multiple categories: When there are several categories within a feature, this method comes to use. Accordingly only the top 10 most common classes are considered and encoded.

Mean encoding: This method encodes the classes by their respective means. So for each class, the mean value of the target variable is calculated and that replaces each class as its code.

Since the OneHotEncoding method creates additional features it leads to a curse of dimensionality.

Ordinal encoding

It has two types:

LabelEncoding: In this method, the ordinal categories of the feature are automatically encoded by the LabelEncoder.

```
Pd.get_dummies()

From sklearn.preprocessing import LabelEncoder
```

Target guided ordinal encoding: In this method for each category, the mean of the output variable is calculated and ranked. Then these ranks are used to encode the categories.

BUILDING MACHINE LEARNING MODELS

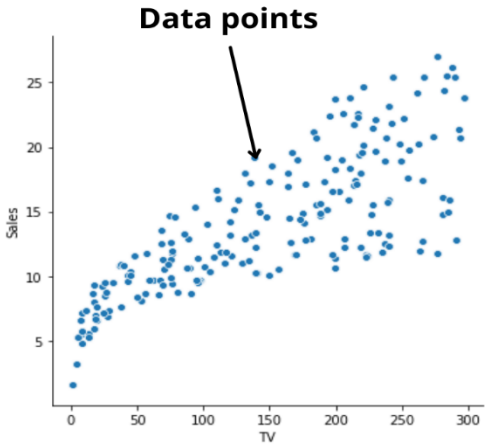
The main goal is to find an equation or distribution that fits the data set. There are various ML models available and for every use case different Models are used, Let us understand some of the famous and commonly used ML models.

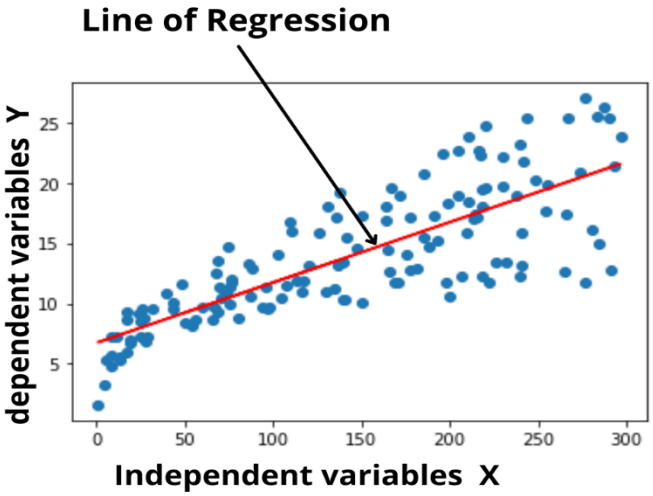
Here are some examples of models:

- Linear Regression
- Logistic Regression
- Decision Tree
- Probabilistic ML model
- Random forest
- KNN
- K –Means
- SVM (Support Vector Machines)

Linear Regression

It is a type of supervised machine learning model, it is used to perform regression tasks. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting





Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (Y) and one or more independent (X) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

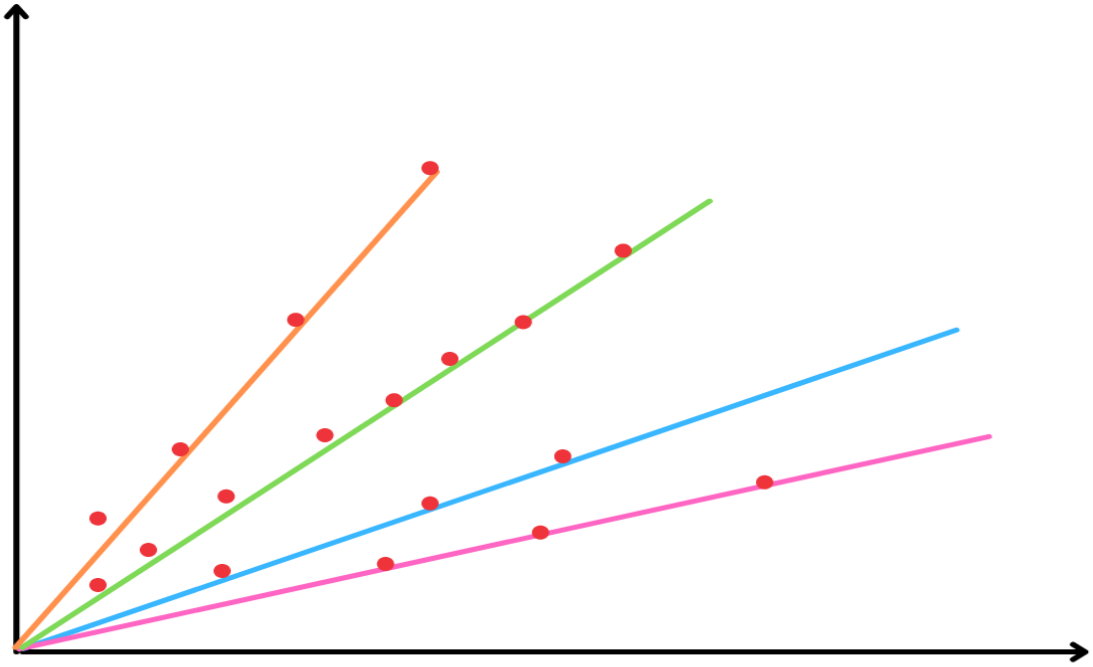
Algorithms for Linear Regression

- Ordinary Least Square (OLS)
- Gradient Descent
- Simple Linear Regression

Ordinary Least Square (OLS)

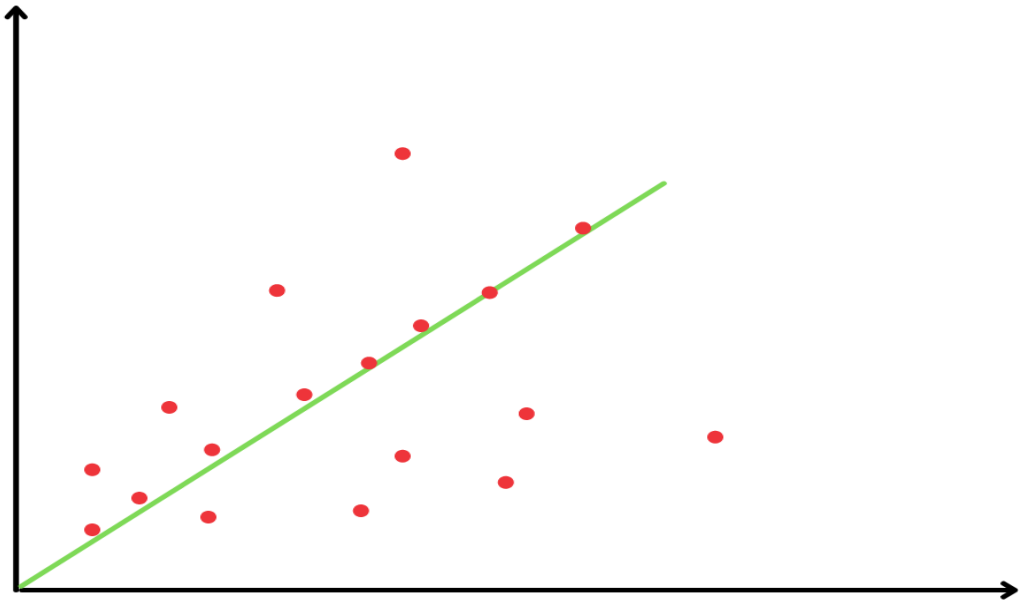
When there are multiple inputs, we can utilize Ordinary Least Squares to estimate the coefficient values.

There can be multiple lines that can be used to represent the set of data values



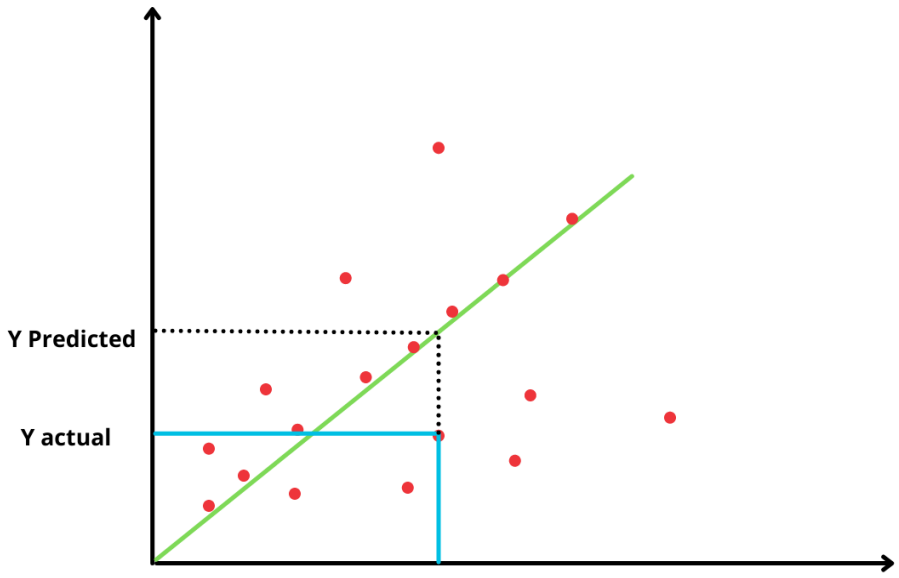
How to decide which is the best line?

- The line with maximum points around it seems to be the best fit.
- The closest the points are to a line, the better fit is that line.
- This implies the distance of points from that line is important



The line with maximum points around it seems to be the best fit.

- So suppose at point X the value Y is denoted by Y actual.
- Now the corresponding value on the line is denoted as Y predicted
- The difference in these two Y values is called error or residual.
- Residual = Y actual – Y predicted



Since some points are above the line so the residual would be positive and some points are below the line so the residual would be negative. Hence when the total residual is calculated it does not give the true picture due to negative signs.

There are two ways to remove this negative sign:

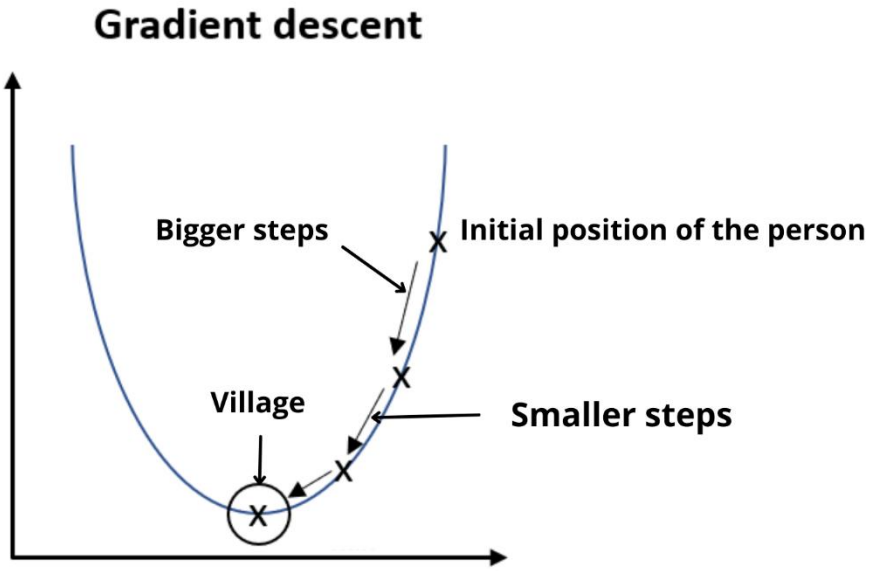
- Find the absolute value of residual
- Find square of each residual

So the concept underlying the linear regression model is to obtain a line that has the least sum of the square of deviations. Hence it is called 'Ordinary Least Square (OLS)

$$\text{Minimize} \rightarrow \text{Residual} = \sum(\mathbf{Y}actual - \mathbf{Y}predicted)^2$$

GRADIENT DESCENT

Let's imagine a person who is coming down the mountain gradually and reach to the valley which is placed at the bottom of the mountain, The person automatically takes large steps when the slope is steep and takes smaller steps when the slope is less steep, Hence till he reaches to the village the person decides his next step based on his current position.



- It is an optimization algorithm and it works step-wise.
- The aim is to decrease the error step-wise.
- In this algorithm, the error is considered as a cost function.
- Hence minimizing the cost function is the final goal.
- The loss is the error in our predicted value of w and b.
- Error is also denoted as SSR (Sum of squares of residuals)

SSR = $\sum$ (Y-Yi) ^2

Here

Y = Y actual

Yi = Y predicted

Let: Yi = wX+b

Therefore

$$H(Ø) = \sum(Y- wX+b)^2$$

To find minima differentiate the equation with respect to slope and then with respect to the constant value

Differentiate with respect to w

$$\frac{d\mathrm{H}(\emptyset)}{dw} = 2(Y - wX - b) * (-X)$$

$$\boldsymbol{\frac{d\mathrm{H}(\emptyset)}{dw} = -2X(Y - wX - b)}$$

Differentiate with respect to b

$$\frac{d\mathrm{H}(\emptyset)}{db} = 2(Y - wX - b) * (-1)$$

$$\boldsymbol{\frac{d\mathrm{H}(\emptyset)}{dw} = -2(Y - wX - b)}$$

The next w value in the slope is given by

$$\mathbf{W}\mathit{next} = \mathbf{W} - \boldsymbol{\alpha}(\mathbf{dw})$$

Here α = Learning ratio (0.1)

Learning rate is a value that helps decided the next slope. The curve of the cost function is a U curve. So learning rate helps us decide the right amount of jump from the previous slope to the next slope such that the minima is reached.

A large value of learning rate may result in missing the minima, whereas the very small value of learning rate may lead to an increase in the number of steps and may become way too complex.

The next b value in the slope is given by

$$\mathbf{b}\mathit{next} = \mathbf{W} - \boldsymbol{\alpha}(\mathbf{db})$$

Steps we need to follow to find the gradient descent.

1. Initially let w = 0 and b = 0. Let L be our learning rate. This controls how much the value of m changes with each step. L could be a small value like 0. 1 for good accuracy.

2. Calculate the partial derivative of the loss function with respect to w, and plug in the current values of X, Y, w and b in it to obtain the derivative value of H(Ø)

3. Now we update the current value of w and b using the **W***next* and **b***next*

4. We repeat this process until our loss function is a very small value or ideally 0 (which means 0 error or 100% accuracy). The value of w and b that we are left with now will be the optimum values.

Evaluation Matrix

For each model, there are different parameters used to evaluate the performance of the model.

In the case of the linear regression model some of the parameters of evaluation are:

- MAE(Mean Absolute Error)
- MSE(Mean Squared Error)
- RMSE(Root Mean Square Error)

Mean Absolute Error (MAE):

The mean of absolute errors of the actual response values from the predicted response values**.**

$$\boldsymbol{MAE} = \frac{1}{n}\sum(Y - Yi)$$

Mean Squared Error (MSE):

The mean squared error takes the mean of squared deviations of the values

$$\boldsymbol{MSE} = \frac{1}{n}\sum(Y - Yi)^2$$

Root of Mean Squared Error (RMSE):

Root Mean squared error takes the square root of MSE

$$\boldsymbol{RMSE} = \sqrt[2]{\frac{1}{n}\sum(Y - Yi)^2}$$

.

- MAE value remains the same no matter whether the original error values are the same or highly varying.
- RMSE on the other hand seems to be highly sensitive to the variation in the error values.

Conclusion: two parameters to decide which evaluation parameter to use:

Interpretability: MAE > RMSE > MSE
Sensitivity to extreme variations: MSE > RMSE > MAE

Multicollinearity

- In real life, when many independent variables are involved in the model create to help predict the output variable, there may be a chance that any two independent variables are correlated. This is called multicollinearity.
- For example: in the housing price model, if say carpet area is high, this also implies that there would be more number of rooms. So these two independent variables imply each other. And hence this may influence the resultant model.
- Models are always built on the assumption that all the predictor variables are independent on each other, if this assumption is NOT true it results in a less reliable model. Hence it is important that there should be no multicollinearity.

How to detect multicollinearity

- Visualization: Use heat map
- Statistical measure: Use correlation matrix
- Statistical measure: Variance inflation factor (VIF)
- VIF tells how much the variance in the regression coefficient increases if the predictor variables are related. It is given by: VIF = 1/(1 – R2)
- In a way, it depicts the extent of multicollinearity. A value of 5 or above indicates high multicollinearity

Methods to prevent multicollinearity

- Drop off one of the related independent variables to eliminate the problem of multicollinearity
- Feature engineering: Design the features in a way that the related variables do not harm the model.
- Example: Carpet area and the number of rooms are kind of related independent variables. A ratio can be taken as a new feature in the model that depicts area per room.
- Area per room = carpet area/number of rooms

Logistic Regression

Logistic regression ML model is a classification algorithm. In classification the target variable is categorical. It has different categories also known as classes.

- The important part of logistic regression is that it works ideally when there are only two classes in the output variable. Hence it is called Dichotomous or Binary.
- Example: Yes/No, Pass/Fail, Spam/No Spam, Fraud transaction/Safe transaction, Survived/ Not Survived

In Logistic regression, it has two main classes defined as success and failure

It is denoted by 0 (Failure) and 1 (Success)

It comes under the classification model and not under regression, But the underlying concept is based on linear regression, Here as linear regression, the aim is to create the best fit line but limit its value to 0's and 1's only.

Decision boundary: Let take an example of our school days,

All of us know that the marks required to pass a particular subject are a minimum of 50% if the student comes below that it means that he has failed in that particular subject but if he clears the subject by just 1% he is considered to have successfully completed the subject.

Hence here 50% is the decision boundary and anything above that is considered as success (Denoted by 1) and anything less than that is considered as failed(Denoted by 0).

A decision boundary of 50% ensures that these two classes are divided right between the middle leading to unbiasedness.

Hence we can conclude that the regression model classified the line into two categories that is the reason why logistic regression is called a classification model

Sigmoid Function:

$$\boldsymbol{p = \frac{1}{1 + e^{-y}}}$$

For -∞

$$p = \frac{1}{1 + e^{-(-\infty)}}$$

$$p = \frac{1}{\infty}$$

Anything divided by ∞ is 0, Hence for -∞ it is **p = 0**

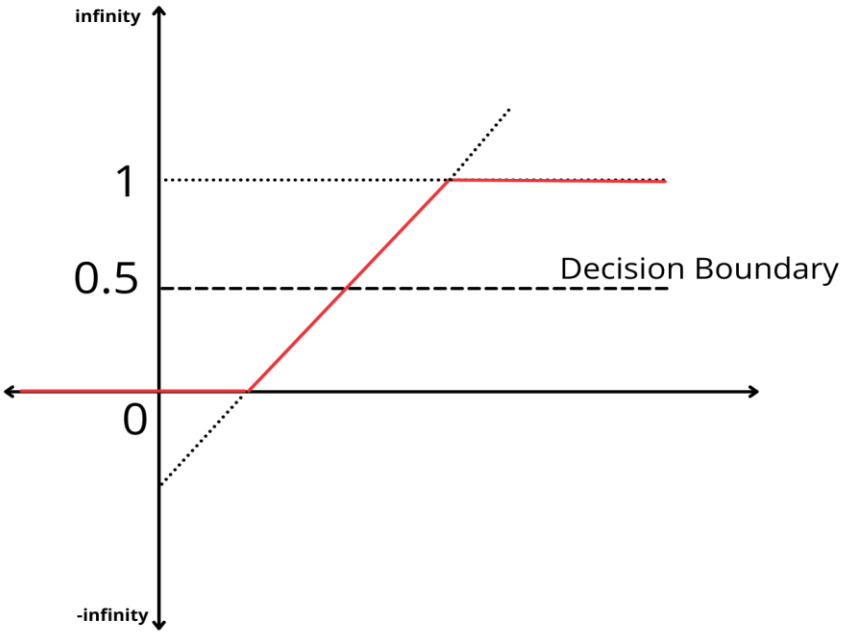
$$p = \frac{1}{1 + e^{-y}}$$

For ∞

$$p = \frac{1}{1 + e^{-(+\infty)}}$$

$$p = \frac{1}{1 + 0}$$

Hence for -∞ it **is p = 1**



Transformation of the sigmoid function

$$p(1 + e^{-y}) = 1$$

$$p + pe^{-y} = 1$$

$$pe^{-y} = 1 - p$$

$$e^{-y} = \frac{1 - p}{p}$$

$$log(e^{-y}) = log\left(\frac{1 - p}{p}\right)$$

$$-y = ln\left(\frac{1 - p}{p}\right)$$

$$y = ln\left(\frac{p}{1 - p}\right)$$

Hence we have converted the sigmoid function such that now it is expressed as providing the value of Y that is the target variable

$$y = ln\left(\frac{p}{1-p}\right)$$

Here (P/ (1-P)) is called “odds ratio”.

Hence Y can be defined as the log of the odds ratio.

Y is also called as Log Odd Function or Logistic Function or Log it Function

Logistic Regression for multi-class classification

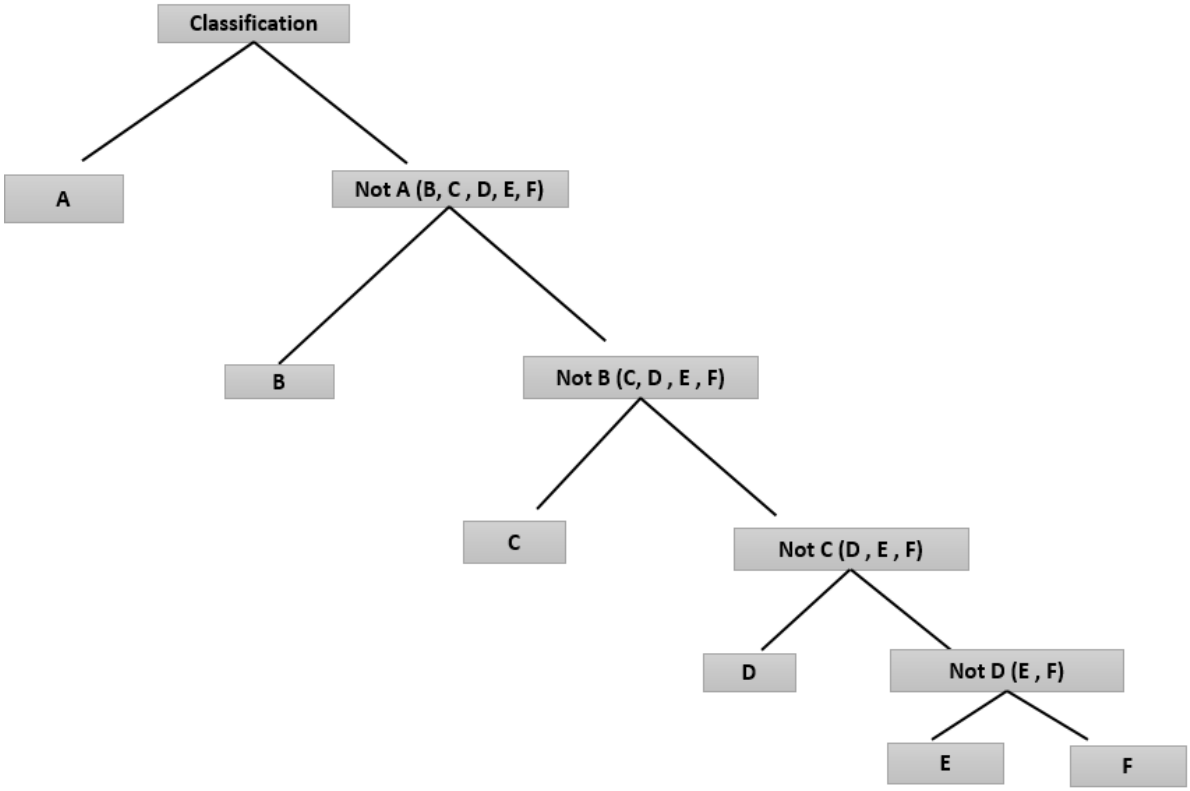
Suppose I work with an output variable that has 6 classes: A, B, C, D, E, and F

First step: classify as A and Not A (this includes B, C D, E, and F)

Second step: classify Not A as B and Not B (this includes C, D, E, and F)

Third step: Classify Not B as C and Not C

And so on…



Evaluation Matrix

The most crucial step in the creation of any machine learning model is to assess its performance. As a result, the topic of how to evaluate the success of a machine learning model arises. How would we determine when to call it a day and stop the training and evaluation?

Machine learning tasks are linked to evaluation measures. Different metrics exist for classification and regression tasks. Some measures, such as precision-recall, are valuable for a variety of purposes. Supervised learning, which accounts for the vast majority of machine learning applications, includes classification and regression. We should be able to increase our model's overall predictive capacity using several metrics for performance evaluation before deploying it on unknown data. When a Machine Learning model is deployed on unseen data without performing a proper evaluation utilizing several evaluation metrics and relying just on the accuracy, it might cause problems and result in bad predictions.

Confusion Matrix

To better understand let us take an example of Covid-19, Imagine we are creating a model to predict the Covid cases there are two outcomes either the person is tested positive or tested negative. But in the real-world these results are not as accurate sometimes the results which come out may be wrong and the situations to handle each case.

True Positive (TP): These are the value that is predicted as positive when the person has Covid

True Negative (TN): These are the value that is predicted as positive when the person does not have Covid

False Positive (FP): These are the value that is predicted as positive when the person does not have Covid

False Negative (FN): These are the value that is predicted as negative when the person has Covid

		Predicted	
		-	+
Actual	-	TN	FP
	+	FN	TP

Accuracy.

It is defined as the ratio of correct predictions to total predictions.

It is calculated as

$$Accuracy = \frac{Total\ correct\ prediction}{Total\ prediction}$$

$$Accuracy = \frac{TN + TP}{TP + TN + FP + FN}$$

Misclassification

It is defined as the ratio of incorrect predictions to total predictions.

$$Misclassification = 1\text{-}Accuracy$$

$$Misclassification = \frac{Incorrect\ prediction}{Total\ prediction}$$

$$Misclassification = \frac{FP + FN}{TP + TN + FP + FN}$$

True Positive Rate (TPR)

It is defined as the ratio of TP to total actual positives.

$$TPR = \frac{TP}{Total\ number\ of\ actual\ positive}$$

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR)

It is the ratio of FP to Total Actual Negatives

$$FPR = \frac{FP}{Total\ number\ of\ actual\ negative}$$

$$FPR = \frac{FP}{FP + TN}$$

True Negative Rate (TNR)

It is defined as the ratio of TN to Total Actual Negatives

$$FPR = \frac{TN}{Total\ number\ of\ actual\ negative}$$

$$FPR = \frac{TN}{FP + TN}$$

Precision

It is the value that has been predicted as positive and how often it is correct

- We use precision when wrong results could lead to loss of business
- It is more useful when FP is a higher concern than FN
- Example:- Email spam(If an important mail is sent to the spam folder and the user misses it due to this reason, It might cause a huge loss to the company, Hence Precision is more useful in this case)

$$precision = \frac{TP}{Total\ predicted\ as\ positive}$$

$$precision = \frac{TP}{FP + TP}$$

Recall

It is the value that tells how many actual positive cases we were able to predict correctly.

- It doesn't matter much if we detect false, But the actual positive cases should not go undetected
- It is more useful when FN is of higher concern than the FP.
- Example: Fraud transaction (If we are evaluating or filtering fraud transactions it is better if the predict a good transaction as fraud rather than predicting fraud as genuine.)

$$recall = \frac{TP}{Total\ actual\ \ positive}$$

$$recall = \frac{TP}{FN + TP}$$

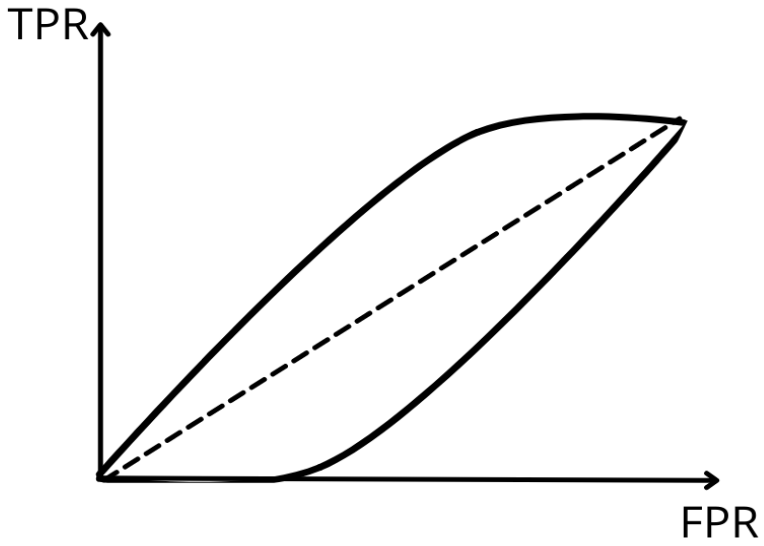
F1 Score

- It is the harmonic mean of precision and the recall
- It gives a combined approximation of both precision and recall.
- When False Positive and False Negative both are important parameters for the business F1 score helps.
- It is precisely used to compare two classifiers. If suppose model A has higher Precision and model B has higher Recall. In that scenario, the F1 score of models A and B is compared.

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{recall}}$$

ROC curve

- ROC means Receiver Operating Characteristics.
- This was initially used by operators of the military radar in 1941, that is why it is named ROC
- ROC curve is a graph plotted between TPR and FPR



Area under the curve

The Area under the Curve (AUC) is a summary of the ROC curve and is a measure of a classifier's ability to discriminate between classes.

The higher the area the better the performance of the model to distinguish between the positive and the negative cases.

Decision Tree Classifier

Decision Tree is a classification and regression supervised ML model. It can be used for a classification problem and also as a regression problem, but it is best suited for a classification problem

- In a decision tree, the dataset is split into different groups based on the features.
- The aim is to use that feature for splitting which results in a purer group or in other words which results in better classification.
- Decision Tree Model is based on 'recursive partitioning'.
- There are different algorithms based on different splitting criteria.

Terminologies

1. **Root Node**: Main node splitting starts here
2. **Splitting**: Process of dividing nodes into sub-nodes.
3. **Decision Node**: Condition node
4. **Parent/ Child Node**: Relative hierarchy in the tree.
5. **Sub Node**: Child Node
6. **Terminal Node/ Leaf Node**: Last node / No further splitting.
7. **Pruning**: Removing Nodes/ Reverse of splitting.

Decision tree

- Supervised machine learning
- Algorithm:
    CART: Classification and Regression Tree (Gini and Gini Impurity)
    C4.5: Entropy
    ID3: Entropy
    CHAID: Chi-Square Automatic Interaction detection
    MARS: Multivariate adaptive regression splines.
- Classification/ Regression
- Hierarchy
- Based on different splitting criteria.

Constraints of tree splitting

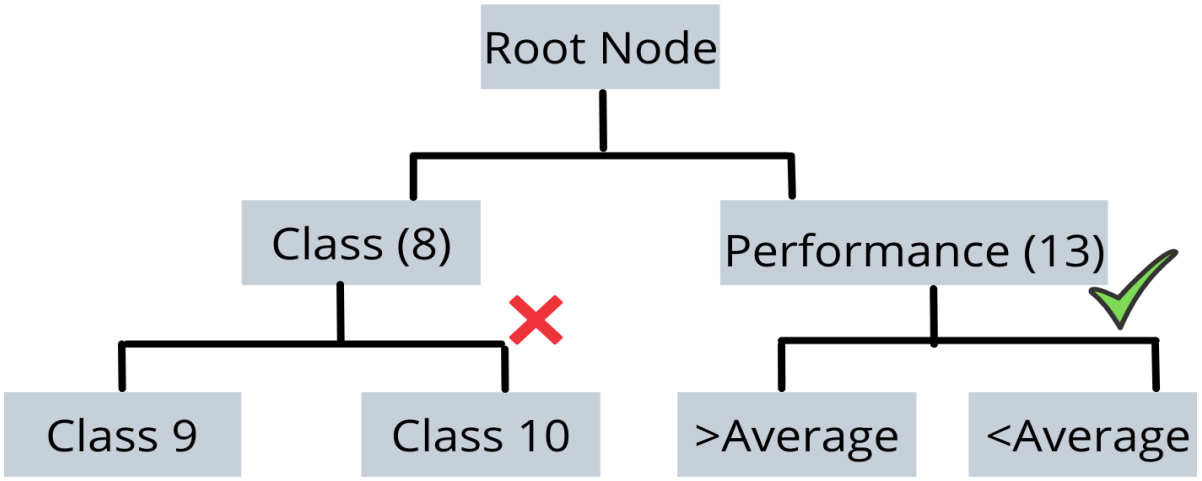
- A decision tree aims to obtain pure nodes. For this it may go on splitting till the time it gets 100% pure nodes. That is the reason it is also known as a 'Greedy Algorithm'
- Since we do not want the tree to continue splitting till the last pure node, we predefine some constraints on when to stop splitting on when should it proceed to split.
- Minimum samples on a node to split
- Minimum samples for a terminal node
- Maximum depth
- Maximum number of terminal nodes
- Maximum features to be considered for a split

Steps of calculation:

- Calculate GINI for each subnode
- Calculate GINI Impurity of each subnode
- Calculate the overall weight GINI impurity of the split.
-

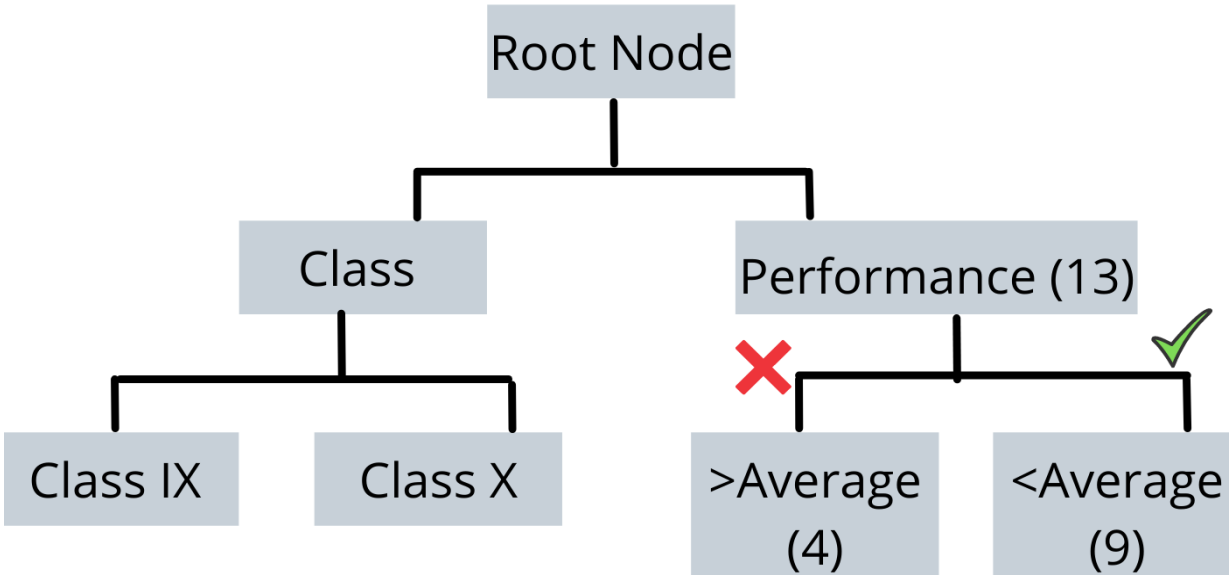
Minimum samples on a decision node to split

Let's say if the rule is that a node must have at least 10 samples to go for the further split



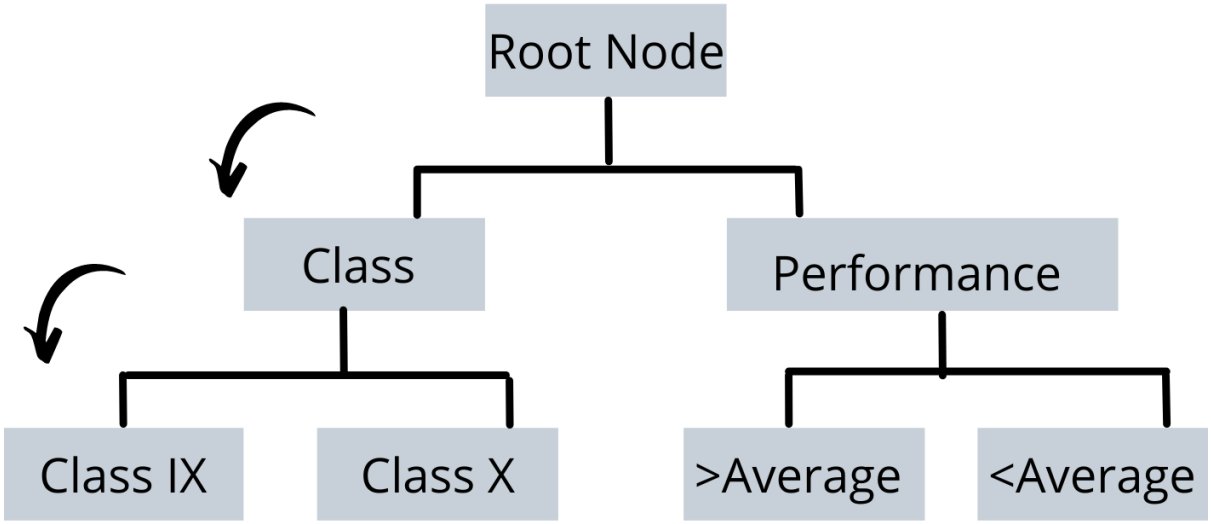
Minimum samples for a terminal node

Suppose if the rule is that minimum samples in the terminal or leaf node must be 5



Maximum depth

Depth is defined as the number of vertical splits the tree needs to do in order to give the final prediction. We can limit this depth by specifying the maximum depth acceptable for pruning purposes.



Maximum number of terminal nodes

Here the maximum number of terminal nodes can be set prior.

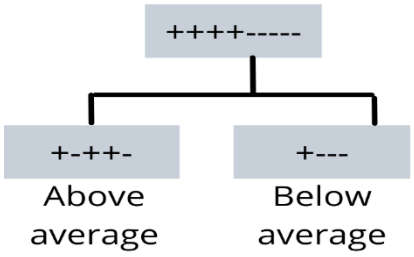
Maximum features to be split

Generally, when there are n features in a dataset, the maximum features to be split is set to sqrt(n) or log(n)

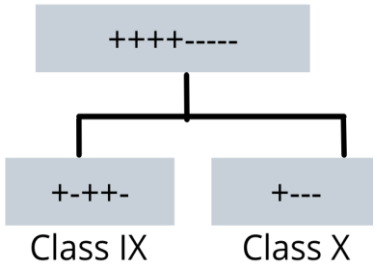
Example

Let the decision node have 20 students. + denotes those who play cricket and – denotes those who do not play cricket

Split by performance



Split by class



GINI / GINI Impurity

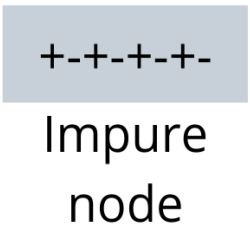
GINI is a measure that talks about purity of a node.

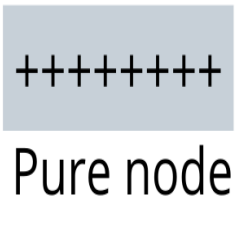
$$GINI = p_1^2 + p_2^2 + p_3^2 + p_4^2 + \cdots$$

$$GINI = p^2 + q^2$$

Information gain/Entropy

- Entropy can be defined as degree of randomness. If there is high randomness in a node, it is less pure.
- On the other hand, if there is more information gain, nodes of higher purity can be obtained.
- When we move from parent node to child node there is information gain.
- Hence information gain leads to higher purity nodes hence reducing their randomness (Entropy)





$$Information\ gain(IG) = 1 - Entropy$$

Calculate Entropy

$$Entropy = -P_1 log P_1 - P_2 log P_2 - P_3 log P_3 \ldots.$$

$$\boldsymbol{Entropy} = -PlogP_1 - QlogQ_2$$

Chi-square Method

This method aims to find the statistical significance between the parent nodes and the sub nodes. It is measured by the sum of squares of differences between the observed and expected frequencies.

It generates a tree called CHAID (Chi-square Automatic Interaction Detector)

The higher the value of Chi-square, the higher is the homogeneity. (Purity)

[So far in GINI impurity and Entropy we were looking for lower values, now we look for higher Chi-square values]

$$\boldsymbol{x^2 Score} = \sqrt[2]{\left(\frac{(Actual - Expected)^2}{Expected}\right)}$$

RANDOM FOREST

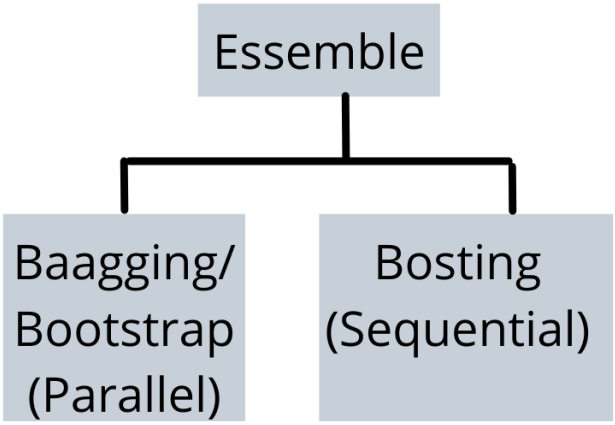
As a forest is made of trees similarly random forest is also made of decision trees, decision tree searches for the best feature while splitting a node random forest searches for the best feature among a random subset of a feature this will result in high diversity and gives better results.

It is a form of supervised learning model.

Ensemble Models

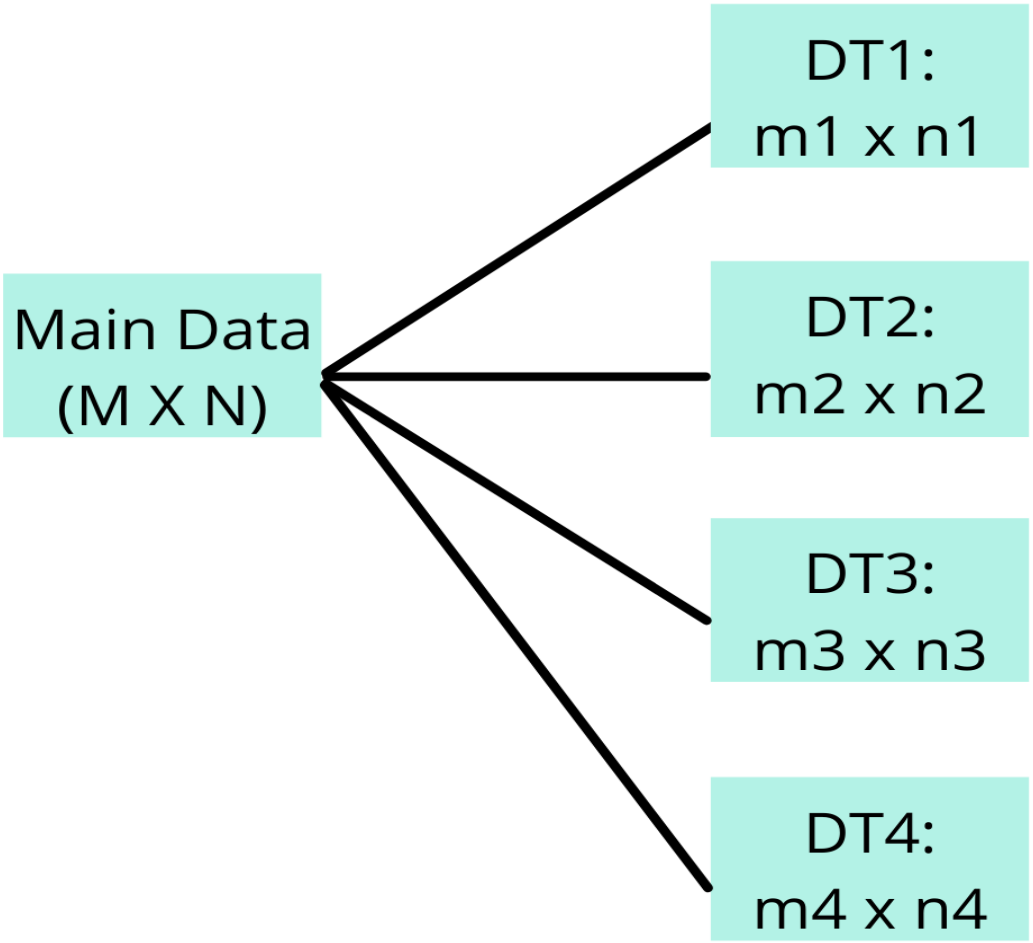
Ensemble implies combination

A technique of combining multiple decision trees either parallelly or sequentially results in an ensemble model.



Bagging/ Bootstrap

- Suppose a dataset has M rows and N columns, so there would be total M x N elements in this dataset.
- For this multiple decision trees can be created from the sub-dataset of the size: m1 x n1, m2 x n2, m3 x n3 and so on…. m < M and n < N
- Note: In the selection of decision trees sampling is done with replacement here each



- This exact process of bagging happens in a random forest ML model.
- The random forest as the name suggests is a combination of multiple trees.
- For random forest, bagging is the ensemble technique used.
- Benefits of bagging: The result of all decision trees are combined. The voting technique is used to decide the final class.
- This increases the predictive power as accuracy does not depend only on 1 decision tree.

Boosting

Boosting is an ensemble technique that works sequentially.

The output of the first decision tree is used to alter the approach of the second decision tree. Then the output of the second decision tree is used to decide the approach for the third decision tree. And in this same format sequential decision trees are creating, improving the process at every point.

Hyperparameters Tuning

In a model, there are some parameters that are calculated by the model itself. And there are some parameters that can be controlled manually. These set of parameters are called 'Hyperparameters'

- Once the model is selected, it can be fine-tuned by altering some of the hyperparameters. This allows checking for which hyperparameters value the model generates better accuracy.
- Examples of hyperparameters: ‘maximum features to include in mode’, ‘minimum samples in the leaf node’, and so on.
- These hyperparameters affect the performance of the model.
- N-jobs are hyperparameters that control the speed of the model. It helps the model know how many jobs can be done simultaneously.
- N- Jobs = 1, which implies one job must be done at a time.
- N- Jobs = 3, which implies 3 jobs can be done simultaneously
- N- Jobs = -1, implies there is no limit on the number of tasks to be done.

When to use to decision tree:

- When you want your model to be simple and explainable
- When you want nonparametric model
- When you don't want to worry about feature selection or regularization or worry about multicollinearity.
- You can overfit the tree and build a model if you are sure the validation or test data set is going to be a subset of the training data set or almost overlapping instead of unexpected.

When to use random forest:

- When you don't bother much about interpreting the model but want better accuracy.
- Random forest will reduce the variance part of error rather than the bias part, so on a given training data set decision tree may be more accurate than a random forest. But on an unexpected validation data set, Random forest always wins in terms of accuracy.

NAÏVE's BAYES

As the name goes this model is based on the Bayes Probability Theorem, This type of model assumes that the features are independent of each other.

- It is labeled as 'Naïve' because of the assumption based on which this model works.
- This model assumes that all the predictor variables are not related to each other at all. In the real world, when we work with several variables, they do have some relation. Hence the assumption is naïve!
- But there are many Use Cases where we have found this assumption to hold true. Text analysis & Sentiment analysis find strong relevance of the Naïve Bayes Model.
- It is originally a classification model**.**

Some of the use cases of the NB model:

- Spam filtering
- Text classification
- Sentiment analysis

Some of the advantages of this type of model are

- Since it is based on probability it is easy to follow.
- This type of model is very fast in very complex datasets.
- It can be scalable.

Conditional Probability

$$P(X \mid Y) = \frac{P(X \text{ and } Y)}{P(Y)} \qquad P(X \text{ and } Y) = P(X \mid Y) * P(Y)$$

$$P(Y \mid X) = \frac{P(X \text{ and } Y)}{P(X)} \qquad P(Y \mid X) = \frac{P(X \mid Y) * P(Y)}{P(X)}$$

If there are different independent variables like X1, X2, and so on… we can rewrite the above equation as:

$$P(Y \mid X_1 X_2 .. X_n) = \frac{P(X_1 \mid Y) * P(X_2 \mid Y) ..... * P(X_n \mid Y)\, P(Y)}{P(X_1) * P(X_2) * ... P(X_n)}$$

Drawbacks:

- The problem of 'Zero Frequency': If we look at P (X | Y), it turns out to be zero.
- This will give the class probability zero.
- This is one of the flaws of the Naïve Bayes Theorem.
- In order to address this flaw, Naïve Bayes uses Laplace's Correction.
- Replace 0 by 1 in the entire table so that some idea about the probability of each class can be obtained. This goes internally in the algorithm and accordingly the probability of each class is evaluated.

- The problem of correlation: If there are any correlated predictor variables, it is important to remove such variables or merge such features so that the assumption of Naïve Bayes is valid

Classifiers:

- If target variable is continuous: Gaussian classifier is used
- If the target variable has more than 2 classes: A multinomial classifier is used
- If the target variable has only two classes: Bernoulli classifier is used

KNN Machine Learning Model

It is a supervised ML model used for classification (and regression).

More commonly used for classification.

From sklearn.neighbors you can import KNeighborsClassifier or KNeighborsRegressor

Steps for KNN Model

- Decide the value of k
- Calculate the distance of all the data points from the unknown point.
- Sort the values in ascending order
- Get the top k rows from the sorted array
- Identify their respective classes and find the most frequently occurred class amount those k points.
- That becomes the predicted class on the unknown point.

How to select the optimal value of k?

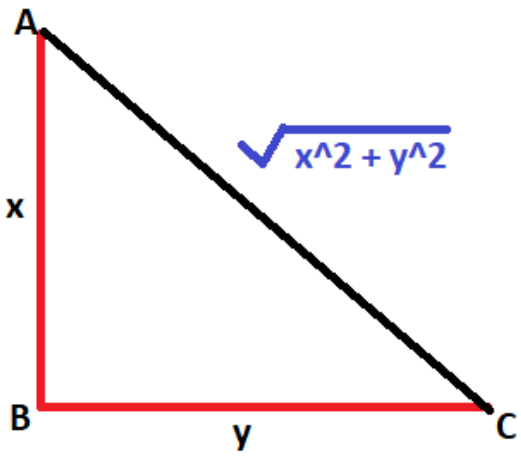
- Some suggestions for how the k value can be selected:
- Prefer odd k values, as there are chances of tying in even values.
- If still when you take k as odd, and tie occurs, increase or reduce the value of k
- K should not be too small.
- Thumb rule: k should be generally sqrt(n), where n denotes the total number of data points
- Further, one can try different values of k, and then observe the evaluation metrics to decide upon the best value of k

How is the distance measured?

Since KNN model classifies the points based on proximity or distance. It is important to understand the different ways to calculate the distance are.

Euclidean distance

By default, KNN uses Euclidean distance (Minkowski with power p = 2)



Manhattan distance / city distance/ taxi distance

It is the sum of horizontal and vertical absolute distances.

$$\text{M distance} = \sum\left[\left|x_2 - x_1\right| + \left|y_2 - y_1\right|\right]$$

Imagine you are taking a taxi from block I to Block A the path which you think a taxi takes and the overall distance is calculated by the sum of all the vertical and the horizontal values.

Block A	Block B	Block C
Block D	Block E	Block F
Block G	Block H	Block I

KNN is a Lazy Learner model

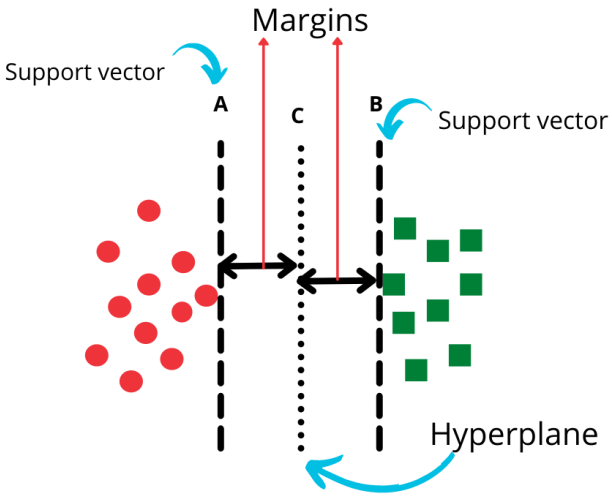
- Almost all the models get trained on the training dataset, but KNN does not get trained on the training dataset.
- When we use knn.fit(X train, Y train), this model 'memorizes' the dataset. It does not understand it or tries to learn the underlying trend.
- Now when we ask the model to predict some value, then it takes a lot of time because now it actually will have to recall all the points and work around them so that it can help predict the correct value.
- Hence where most of the models, take time during training, this model does not take any time during training.
- Most of the models take no time in prediction, but the KNN model takes a lot of time during the prediction stage.

Important points

- Since it is a distance-based model, feature scaling is a must for it.
- Besides logistic regression, the rest all the classification models can work on multi-class classification.

Support vector machines (SVM)

- It is a supervised machine learning algorithm
- It separates data using hyper planes.



Different classes are separated by two lines known as support vectors, and the line that runs between them is known as the hyperplane. The support vectors are positioned on the edge of the class in which they each lie. SVM works very well on multi-dimensional sets.

Support vector: The line passing through 1st class and a parallel line for the 2nd class

Hyperplane: A-line passing in between the 2 support vectors and dividing it into two equal parts and these parts are known as margin

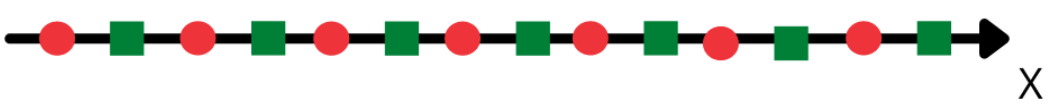
Our aim is to consider the highest margin (Because throughout our classification we want our classification to be properly defined)

Example: Imagine a village where 2 brothers are fighting for a piece of land and come to you for a solution what solution would you give? Yes, you'll try to divide it into 2 equal parts, similarly here the hyperplane divides it into two equal parts.

Kernel trick

It allows us to operate in the original space without capturing the data into higher space.

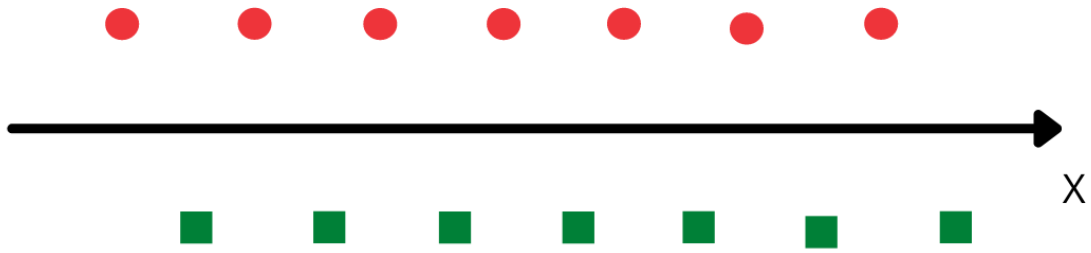
Let's say I have some classes as shown in the below diagram which is non-separable.



How can we separate it with just 1D, Hence well have to go to one higher dimension

$$\emptyset(x) = x \; mod \; 2$$

After the transformation of all the values, it'll separate all the classes which we have.



As we can see the point were not separable in 1D, Which was the original dimension but after applying the transformation it became linearly separable.

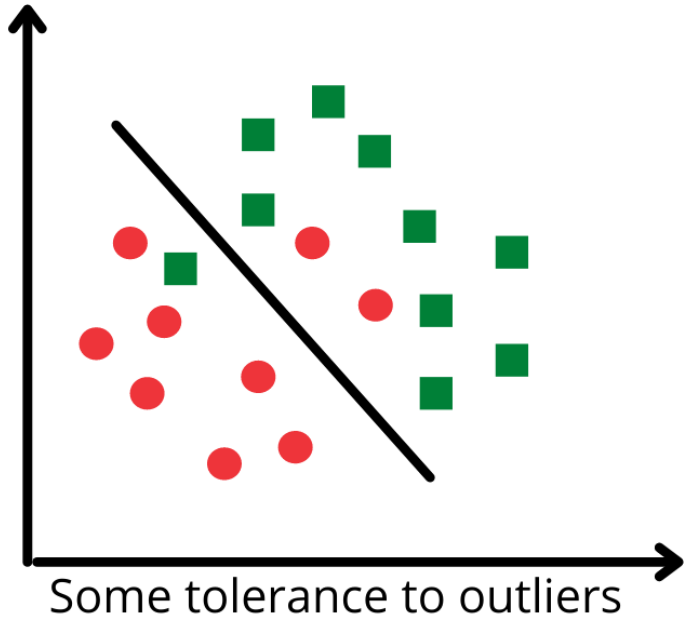
We keep on increasing the dimensions unless the classes are separable.

There are different kernel functions available:

- Gaussian (RBF kernel- Radial Basis Function)
- Polynomial (Whenever we are passing this we'll have to give the degree as well)
- Sigmoid
- ANOVA
- Basil

Hyperparameters of SVM

In real-time, the classes are not distinctly separated there are some outliers that cross the hyperplane and enter the other classes as shown in the figure below.

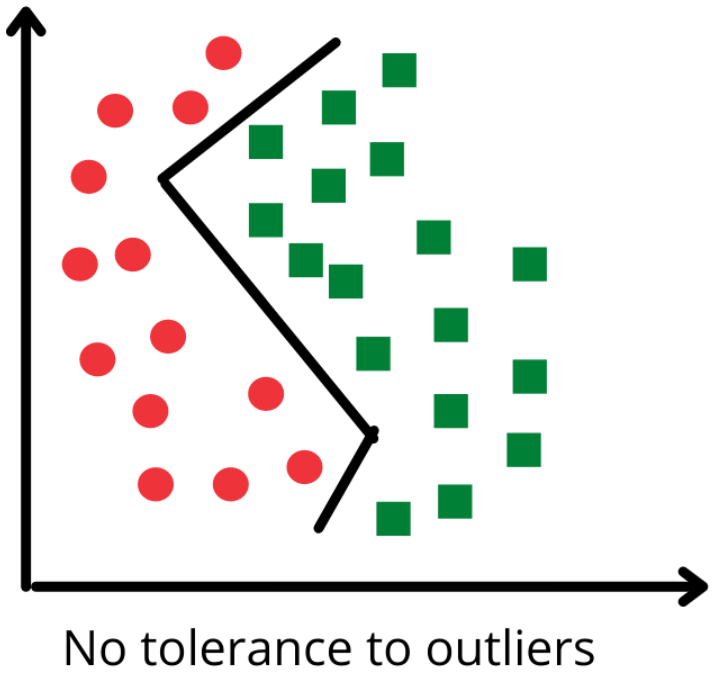


If we give some tolerance (C) to the outliers in the model then a simple plane will be created.

Here the C is low

Tolerance(C) is also called a regularization parameter.

Here high margin can be obtained hence also called a high margin classifier.



If we give no tolerance (C) to the outliers in the model then a complex plane will be created.

Here the C is high

Here small margin is be obtained hence also called a low margin classifier.

Gamma:

Two key points to remember:

- Low Gamma means “FAR”
- High Gamma means “Nearby”

Imagine you have a data set with 2 classes and it is separated by support vectors, if you are making the Gamma high it means that the model has to give more attention to a nearby point, similarly, if we pass low Gamma it means that the model will give attention to far points.