**Python Interview Questions & Answers:**

[**https://towardsdatascience.com/53-python-interview-questions-and-answers-91fa311eec3f**](https://towardsdatascience.com/53-python-interview-questions-and-answers-91fa311eec3f)

[**https://medium.com/@lokeshsharma596/python-lambda-map-filter-reduce-and-zip-function-c59d8946a3ce**](https://medium.com/@lokeshsharma596/python-lambda-map-filter-reduce-and-zip-function-c59d8946a3ce)

[**https://github.com/Aafreen29/SQL-Interview-Prep-Question/blob/master/queries.sql**](https://github.com/Aafreen29/SQL-Interview-Prep-Question/blob/master/queries.sql)

[**https://towardsdatascience.com/over-100-data-scientist-interview-questions-and-answers-c5a66186769a#e4e2**](https://towardsdatascience.com/over-100-data-scientist-interview-questions-and-answers-c5a66186769a#e4e2)

## Generators:

If the body of any function contains Yield-Keyword then it would be a generator function which returns the generator objects nothing but an iterator and we are getting the objects in iterator by calling the next method.

The main difference between a regular function and generator functions is that the state of generator function is maintained through the use of the Yield keyword and when next time the function is called, execution continues from where it left off, whereas in regular functions the return statement terminates the function completely.

Main advantage of using generators is **saving the memory space** where iterators don’t compute the value of each item when instantiated. They only compute it when you ask for it. This is known as lazy evaluation.

## Decorators:

Decorators are nothing but adding the new functionality to the existing function without modifying its structure. Decorators are usually called before the definition of a function you want to decorate using @ symbol.

It is simple: readability. Python is praised for its clear and concise syntax, and decorators are no exceptions. If there is any behaviour that is common to more than one function, you probably need to make a decorator.

**Multiprocessing and Multi-Threading:**

Executing several tasks simultaneously is the concept of multitasking.

There are two types of Multitasking called as process based multitasking and thread-based multitasking.

Executing several tasks simultaneously where each task is a separate independent process is called process based multi-tasking**(multi-processing**)

**Ex:** while typing python program in the editor we can listen mp3 audio songs from the same system. At the same-time we can download a file from the internet. All these tasks are executing simultaneously and independent of each other. Hence it is process based multi-tasking.

Executing several tasks simultaneously where each task is a separate independent part of the same program, is called thread based multi-tasking**(multi-threading**) and each independent part is called a **Thread**.

**Ex:** Where ever a group of independent jobs are available, then it is highly recommended to execute simultaneously instead of executing one by one then we should go for multi-threading.

**Does Python Supports Multi-Threading?**

Threading is allowed in python but the only problem is that the GIL will make sure that just one thread is executed at a time (no parallelism).

So basically, if you want to multi-thread the code to speed up the calculation, it won't speed it up as just one thread is executed at a time.

The GIL is an interpreter-level lock. This lock prevents the execution of multiple threads at once in the Python interpreter. Each thread that wants to run must wait for the GIL to be released by the other thread, which means your multi-threaded Python application is actually single-threaded. The GIL prevents simultaneous access to Python objects by multiple threads.

Linear Regression:

In the regression model, we are trying to find the best fit line by minimizing the errors, in simple words, we are trying to minimize the length between observed value and predicted value

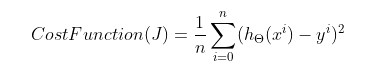
Equation of the regression line is



Where Bo is intercept and B1 is the slope of the line or for example y=4x+2 , here slope is 4 which means unit change in x will change 4 times in y and e is Error term which is actually called as cost function , sum of the squared differenced between actual and predicted values.

**How are we trying to minimize the errors?**

Using Cost Function, it is nothing but calculating the error



Where  is predicted value which is like y=mx+c and is actual value.

Based on the error it will try to update the weight parameters using gradient descent optmizers

Text

Description automatically generated with low confidence

Where  is learning rate which talks about how speed it is taking to converge or to reach the global minima quickly. Theta is the weight parameter

 is derivative of loss with respective to weight parameter nothing but slope of that particular weight. If the slope pointing upwards, then it is positive slope. Then gradient descent is



It means weights are decreasing and if the slope pointing downwards then its negative slope. For negative slope, weights will get increased and will repeat the same untill reach the convergence which is at very low error means low cost function value .

But **Cost function** value should never becomes 0. Though Error is completely 0 but for unseen data we definietly get some error which seems like model performing well for training data and model not performing well for testing data which is called as **Overfitting Condition** which means **Low Bias** and **High Variance** or if model accuracy is bad for both training data and testing data then it is **Underfitting Condition** which means **Low Bias** and **Low Variance** which are not generalized models.

To overcome these problems we use **reguarization** techniques which are L1 (**Lasso**) and L2 (**Ridge**) **Regularization**

So In Regularization, what we do is normally we keep the same number of features, but reducing the magnitude of the coefficients. So main objective of regularization is to scale down the coefficient values

For example, y= 0.9+1.2\*1+20\*2+39\*3 which looks like high coefficients this may cause overfit in real world problems. So we are bringing down the coefficients by using some regularization techniques like Lasso and Ridge. After scaling down , y= 0.9+0.7\*1+2\*2+5\*3.

**Ridge Regularization(L2):**

Ridge regularization is mainly used for preventing Overfitting Condition by adding penalty term to the cost function which is given by **CF = CF + lambda(slope)2 .**

So, the penalty term regularizes the coefficients or weights of the model.

**Lasso Regularization(L1):**

It is like the Ridge Regression except that the penalty term contains only the absolute weights instead of a square of weights which is given by by **CF = CF + lambda(|Slope|).**

Since it takes absolute values, hence, it can shrink the slope to 0, whereas Ridge Regression can only shrink it near to 0. Hence, the Lasso regression can help us to reduce the **overfitting** in the model as well as the **feature selection.**

**Assumptions of Linear Regression:**

1. Relation between dependent and independent variables should be **linear**.
2. If the features, follow **Normal/Gaussian Distribution** then model will get trained well. If the features don’t follow Normal/Gaussian Distribution, then we are applying some transformations to get the Normal/Gaussian Distribution.
3. Normality of Residuals, Errors should follow the Gaussian Distribution
4. It is assumed that there is little or no multicollinearity in the data. **Multicollinearity** occurs when two or more independent variables are highly correlated with one another in a regression model. Dropping one of the correlated features will help in bringing down the multicollinearity between correlated features by using VIF (Variance Inflation Factor).
5. **Homoscedasticity**: Homoscedasticity means says that the variance should not increase or decrease as the error values change. Also, the variance should not follow any pattern as the error terms change.
6. All Independent variables should be uncorrelated to the error term.

**Logistic Regression:**

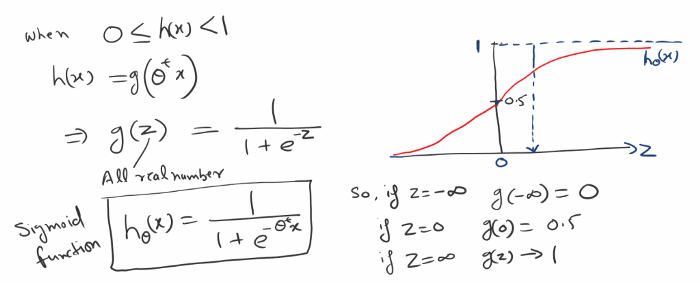
Logistic Regression is mainly used for classification kind of problems.

**Logistic Regression** measures the relationship between dependent and one or more independent variables by estimating probabilities using its logistic function.

In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

**Why can’t we use Linear Regression for Classification Problems?**

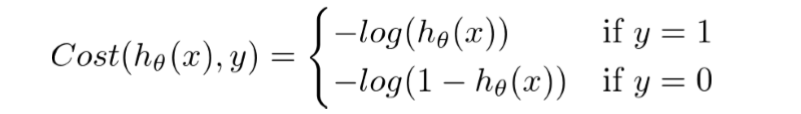
* The first one is that Linear Regression deals with continuous values whereas classification problems deal with probabilities.
* Due to outliers present in the data, best fit line may get deviated with more **steep** which is usually not the good for classification.
* Based on our best fit line we may get very high values like **greater than 1** or very low values like negative values which is **less than 0**. So, for this we must **squash** the best fit line means applying activation function on best fit line using **sigmoid function** which varies from **0 to 1.**



If we try to use the cost function of the linear regression in ‘Logistic Regression’ then it would be of no use as it would end up being a non-convex function with many local minimums, in which it would be very difficult to minimize the cost value and find the global minimum.

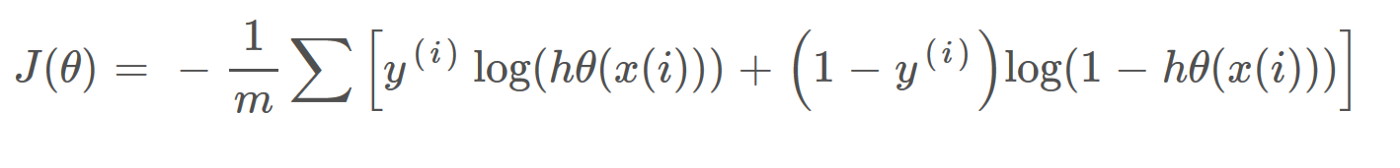
Chart, line chart

Description automatically generated

For logistic regression, the Cost function is defined as:Chart, histogram

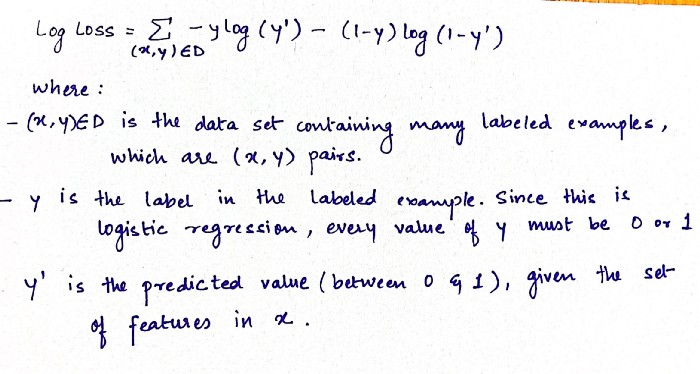
Description automatically generated Chart, line chart

Description automatically generatedThe above two functions can be compressed into a single function i.e.



And using Gradient descent we are trying to repeat the process until reach the **convergence**

Log Loss is the cost function used in Logistic regression



**Why Logistic is called Regression?**

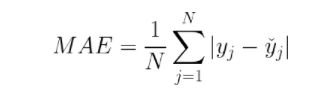
We are trying to **squash** the best fit line in **linear** **regression** using logit function, so it’s called as **Logistic Regression**

**Performance Metrics:**

## Mean Absolute Error:

Mean Absolute Error is the average of the absolute difference between the ground truth and the predicted values. It measures the average of the residuals in the dataset.

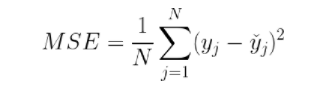
Mathematically, its represented as:



## Mean Squared Error:

## Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals.

Mathematically, its represented as:



MSE is differentiable whereas MAE is non differentiable.

MAE more robust towards outliers than MSE, since it doesn’t exaggerate errors.

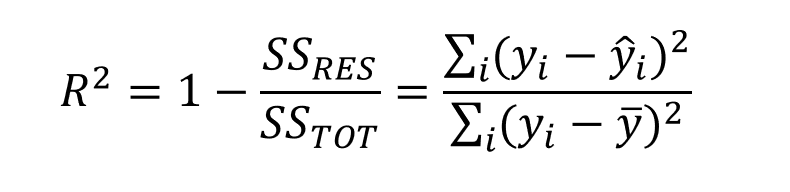
**Root Mean Squared Error:**

Root Mean Squared Error is the square root of Mean Squared error. It measures the standard deviation of residuals.

## R-Squared (R²):

R² represents the proportion of variance explained by your model.

It is noting but variation in y can be explained by using variation of x. More the R2 value, better the model is performing.



where SSRES is the Residual Sum of squares and SSTOT is the Total Sum of squares.

It can be defined as a Ratio of variation to the Total Variation. The value of R squared lies between 0 to 1, the value closer to 1 the better the model.

SSE is the sum of squared errors, the sum of the squared differences between the actual values and predicted values.

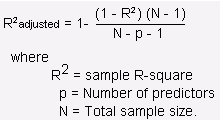
SST is the total sum of squares (shown sometimes as TSS); the sum of the squared differences between the actual values and the mean of the actual values.

More the R² value more generalised model, however if we add any variable to the existing data which is irrelevant to target variable it’s still increasing the R² value so that we are changing the formula of R² which is called as Adjusted R².

**Adjusted R squared:**

The problem/drawback with R2 is that as the features increases, the value of R2 also increases which gives the illusion of a good model.so, Adjusted R squared only considers the features which are important for the model and shows the real improvement of the model.

Adjusted R2 is always lower than R2.



**AUC-ROC:**

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the ‘signal’ from the ‘noise’. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

**Confusion Matrix:**

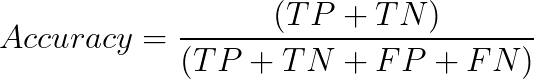
True Positive, True Negative, False Positive and False Negative are usually presented in a tabular format in the so-called **Confusion Matrix.**

Table

Description automatically generated

**Accuracy:**

It is nothing but the Sum the number of predictions correctly predicted as Positive (TP) or correctly predicted as Negative (TN) and divide it by all types of predictions, both correct (TP, TN) and incorrect (FP, FN).



But if the dataset is imbalanced then we should not use Accuracy as a metric. We should gp for precision, recall and F1-Score.

**Precision:**

Precision is nothing but out of all positive predictions hoe many positive predictions are correct. Here focus is to reduce **False Positives.**

Diagram

Description automatically generated

For Example, in **Spam Classification**, actually its not a spam but our model predicts it as spam then it will be a disaster for us so here False positive rate is very important

**Recall:** It is nothing but out of all actual positives how many positives are predicted correctly. Here focus is to reduce False Negatives. Recall is also known as **True Positive Rate** or **Sensitivity**

Diagram

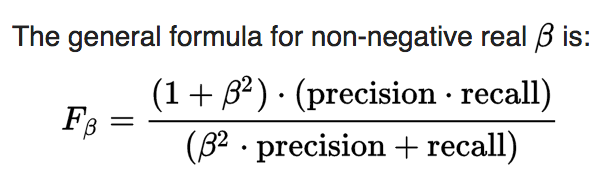
Description automatically generated

For Example, In **Cancer Classification**, though patient is suffering a cancer but model predicts as not a cancer then it will be huge disaster.

**False Positive Rate :** It is exactly opposite to Recall which is nothing but out of all actual negatives , how many negatives are predicted correctly.

FPR = TN/(FP+TN)

**F-beta Score:** Suppose if **FP** and **FN** both are important then we use F1-Score and is given by



* For both FP and FN, we use beta = 1
* For only False positive is important, we use beta = 0.5
* For only False Negative is important, we use beta = 2

Decision Tree**:**

It is used for both **Regression** and **classification** kind of problems.

**Decision Tree Classifier:**

**Steps:**

* Decision Tree works by splitting the complete data into subsets based on a feature, and the first feature that is used to do for splitting is called the **root node**.
* **How are we selecting the particular feature as a root node?**
* For every feature, it will construct the decision tree, and at which feature has the highest **information gain** that feature will be selected as a **root node**
* Once the root node is selected, the algorithm repeats the same steps with other features until it reaches the leaf nodes (the end of the tree). This leaf node now acts as the **output** of the model.

Main aim of decision tree is **to reach the leaf node quickly**. In order to reach the leaf node quickly we should select efficient features/attributes which has **low measured entropy or low Gini impurity**

**How do we know whether needs to be split further or not?**

We are calculating the Purity of split at each node using **Entropy** or **Gini Impurity.**

* Gini impurity is also the same as entropy.

**Entropy:**

It Measures the purity of sub split; at each node we are calculating this entropy to know whether the split is pure or impure.

A picture containing shape

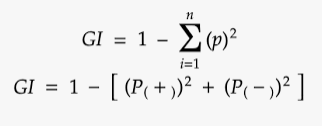
Description automatically generated

Where p is the Probability of **positive** class and q is the Probability of **negative** class

* If Entropy = 0 then it is the purest split, then no needs to be split further
* If entropy not 0 then we are splitting again into sub-splits and repeating the process till entropy is getting zero.

**Gini Impurity:**

It also measures the purity of sub split but only difference is in formula which reduce the complex computation like logarithms used in entropy.



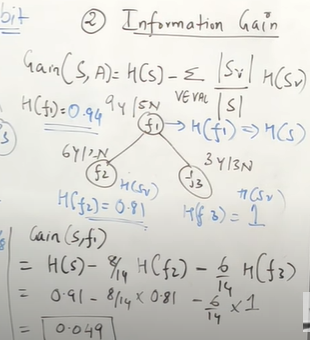
By Default, Decision Tree Classifier, uses Gini impurity which is less computational.

**Information Gain:**

Information gain is mainly used for which feature needs to be selected as a root node and formula is given by the difference between the entropy before splitting and entropy after splitting.



Where T is number of samples before splitting and Tv is Number of samples after splitting



Where H(s) is the Entropy before splitting(nothing but entropy of root node) and H(sv) is the Entropy after splitting the root node, S is total Sample data and Sv is Subset of data.

All Entropy, Gini Impurity and Information gain is ony used for **Decision tree Classifier**.

**Decision Tree Regressor:**

It is also like Decision Tree Classifier only difference is at each node, we are calculating the **MSE or MAE** by taking the **mean** of that node and again splitting the node if we have highest **MSE** or **MAE** and repeating the process to decrease **MSE** or **MAE** instead using **entropy, gini impurity** and whichever the **mean** value is there **at leaf node** acts as the **output** of the model.

**Draw backs of Decision Tree**:

The main **Problem of Decision Tree** is it splits the features into its complete depth so it may cause the **overfitting** problem which means **low bias and high variance.**

**Pruning in Decision Tree:**

In general pruning is a process of removal of selected part of plant such as bud, branches, and roots. In Decision Tree **pruning** does the same task it removes the branches of decision tree to overcome the **overfitting** condition of decision tree.

We can do this in 2 ways using **Post Pruning** and **Pre-Pruning.**

**Post pruning** can be done by **using cost complexity pruning** path function which will give CCP alphas. For each CCP alpha we are creating and calculating the accuracy to check the over fitting problem.

In Simple at any node suppose if we have 8yes/2No then it is 80% of yes then no need to split it again we can cut the branches after this node this is called as **Post Pruning**

**Pre-pruning**, it can be done by using hyper-tuning parameters like **max\_depth and max\_leafs**

**Parameters used in Decision Tree:**

Ensemble Learning**:**

The entire idea behind Ensemble learning is to create multiple learning models instead of one model to predict the values which usually robust the performance.

Bagging Techniques**:**

Bagging is a parallel ensemble learning technique to reduce the variance in the final prediction because of taking the majority voting classifier.

Diagram

Description automatically generated

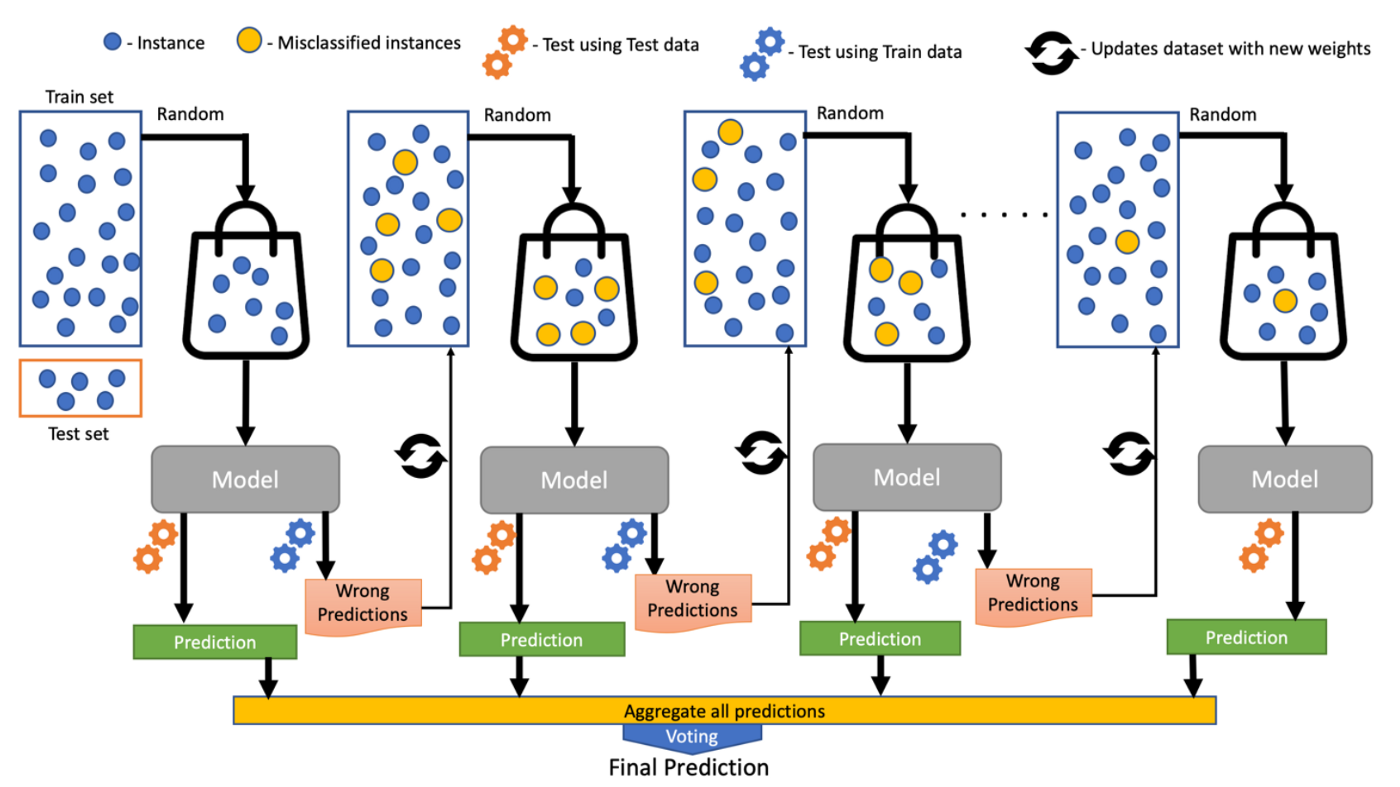
In **Bagging**, we are taking the random sub samples of data from original dataset which is called as **Bootstrapping** and pushing these sub samples of data into multiple Models to train and finally we are aggregating the results which is called as **aggregation** and hence it is also called as **Bootstrap Aggregation**

Examples of Bagging Techniques are Random Forest, Bagging Trees Classifier.

Boosting Techniques**:**

Boosting is a sequential ensemble learning technique to convert weak base learners to strong learner that performs better and is less bias.

Boosting is an iterative method that adjusts the weight of an observation based on the previous classification. If an observation was classified incorrectly, then the weight of that observation is increased in the next iteration. In the same way, if an observation was classified correctly then the weight of that observation is reduced in the next iteration.



Examples of Boosting techniques are AdaBoost, Gradient Boosting, Xtreme Gradient Boosting.

Random Forest Classifier and Regressor:

Random Forest is a **Bagging** Ensemble technique.

1. First it will take random sub samples from the original dataset by **row sampling** and **feature sampling** which is called as **Bootstrap**
2. Then for every sub sample we are training the **decision tree** model.
3. At last, we are using majority **voting** classifier for **classification** and **mean** of all predictions for regressor kind of problems which is called as **aggregation**, hence it is called as **Bootstrap Aggregation.**

**Note:**

* **Standardization** is not required in all trees-based algorithms hence we are just splitting the values
* Some of the samples may present in multiple decision trees

**Out of Bag Error:**

Out of Bag Error is way of **validating** the Random Forest Model.

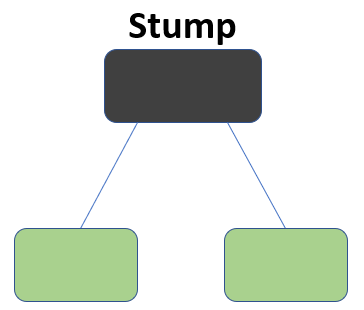
**Intuition:**

1. Let Assume I have 5 rows of original dataset and am using 5 Decision trees inside Random Forest.
2. For the first bootstrap, am taking first 3 rows of data and training the first DT model and whatever the data is left out for DT1 known as **Out of Bag Sample** which is 2 rows.
3. And also assume the same OOB sample (same 2 rows) is also not trained on DT2 & DT3 as well.
4. After training all the models, same OOB sample will be given as unseen data to the DT1, DT2 & DT3 and calculate the error as the number of rows correctly predicted from the OOB Sample which is called **OOB Error.**

Parameters used in Random Forest

AdaBoost**:**

Adaboost is also a Boosting ensemble technique, where we are constructing the decision trees **sequentially** one after the other and each decision tree uses **only one split**. These trees are called as **Decision Stumps.** So, these stumps are called as **Weak Learners.**



Initially it will give equal weights for all datapoints and then after training the first decision tree it will assign higher weights to the points which are wrongly classified and assigns lower weights to the points which are correctly classified and all the points which are higher weights are given to the next decision tree for training and repeating the same process untill and unless a lower error is received or by defining specified number of decision trees.

And for testing data, we are predicting with all decision trees used in training and at last we are using majority voting classifier for final prediction and for regression we are using mean of all predictions as a final prediction

**Refer:**

[AdaBoost Algorithm - A Complete Guide for Beginners - Analytics Vidhya](https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/)

**Parameters Used in Adaboost:**

Gradient Boosting**:**

Gradient Boosting is a Boosting ensemble learning technique which contains a sequence of decision trees where each tree is built based on the previous residuals of decision tree and repeating the same process for a specified number of trees. Here all trees are dependent on each other.

**Intuition:**

**Steps:**

1. First step is creating the base model which always gives the **average** of target variable for **regression** and **0.5 probability for classification problems.**
2. Then we are calculating **residuals** using base model output for each data point.
3. Then we are constructing a decision tree which is **sequential** to the base model using these residuals as target variables and again calculating the **residuals** with first decision tree and repeating the same process for a specified number of iterations here the output of each decision tree is residuals
4. For a new test data, **output** is nothing but **Sum** of all sequential models for **regression** and is given by **Basemodel + α(DT1) + α(DT2) + ……... + α(DTn)**
5. And for Classification, output is nothing but the **Sigmoid** of **Sum** of all sequential models and is given by Sigmoid (**Basemodel + α(DT1) + α(DT2) + ……... + α(DTn))**

Internally **GB regressor** uses **MSE** as loss function at each node whereas in **GB Classifier** uses **Similarity weight** and information gain as loss function

And **Similarity weight** is given by

Diagram

Description automatically generated

And **Information gain** is nothing but the difference between Similarity weight before splitting and Similarity Weight after Splitting.

**Parameters used in Gradient Boosting:**

XGBOOST (Xtreme Gradient Boosting)**:**

Xgboost is ensemble boosting technique which follows the **gradient boosting** framework which contains a sequence of decision trees where each tree is built based on the previous residuals of the decision tree.

**Main features of Xgboost over Gradient boosting:**

* Supports parallel processing - Parallelization of tree construction
* Cache optimization.
* Efficient memory management for large datasets exceeding RAM.
* Has a variety of regularizations which helps in reducing overfitting
* Auto tree pruning – Decision tree will not grow further after certain limits internally.
* Can handle missing values.
* Has inbuilt Cross-Validation.
* Takes care of outliers to some extent.

Main goals of **Xgboost** are model performance and execution speed.

**K Nearest Neighbour Algorithm:**

It is used for both Regression and classification kind of problems and also it is a non-parametric algorithm since it does not make any assumptions on the training dataset like linear and logistic regression.

It is also called a **lazy learner algorithm** because it does not learn anything during training phase and at the time of classification, when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

**Steps:**

1. Primarily, we should select the value of k based on hyperparameter. Let assume K=5.
2. For any unseen data, we are finding the 5 nearest datapoints using Euclidean distance.
3. For **classification**, out of those 5 data points, if 3 belongs to **yes** and 2 belongs to **no** then the given data point belongs to **YES** Category.
4. For **Regression**, we are calculating the **average** of those 5 datapoints

**Note:**

There is no particular method to select the best value of K, hyper tuning is the only option.

**Draw backs:**

It is not good for outliers and for imbalanced data.

## Support vector Machines:

The goal of the SVM algorithm is to create the best line (hyper plane) or decision boundary that can segregate n-dimensional space into n-classes.

Along with the hyperplane, we are creating two parallel marginal lines which are passing through the nearest points from positive and negative sides, those points are called as support vectors. Hence it is called support vector machine algorithm.

Distance between these two parallel marginal lines is called as Marginal Distance.

Supports vectors helps us to calculate the marginal distance. More marginal distance will be the more generalised model.

**SVM Kernels:**

In the SVM algorithm, a kernel function is a special mathematical function. In simple terms, a kernel function takes data as input and converts it into a required form.

Using the kernel function, we can transform the data that is not linearly separable (cannot be separated using a straight line) into one that is linearly separable**.**

For non-linear separable kind of data, we are using SVM Kernels which is usually converting low dimensions to high dimensions. In 3D we can easily be separable with hyper plane.

**Hard and Soft Margins:**

Hard-Margin SVMs have linearly separable training data. No data points are allowed in the margin areas. This type of linear classification is known as Hard margin classification.

Soft-Margin SVMs have training data that are not linearly separable. Margin violation means choosing a hyperplane, which can allow some data points to stay either in between the margin area or on the incorrect side of the hyperplane.

SVM (Support Vector Machines) handles the **outliers** in a better manner than the Logistic Regression.

**Logistic Regression:** Logistic Regression will identify a linear boundary if it exists to accommodate the outliers. To accommodate the outliers, it will shift the linear boundary.

**SVM**: SVM is insensitive to individual samples. So, to accommodate an outlier there will not be a major shift in the linear boundary. SVM comes with inbuilt complexity controls, which take care of overfitting, which is not true in the case of Logistic Regression

## Naive bayes Algorithm:

It is mainly used for Classification kind of problems.

It is a probabilistic classifier, which means it predicts based on the probability of an object.

Naïve bayes algorithm works on bayes theorem and is given by

Text

Description automatically generated

P(A|B) — the probability of event A given B (called posterior)

P(B|A) — the probability of event B given A (called likelihood)

P(A) — the probability of event A (called prior)

P(B) — the probability of event B (called evidence)

Basically, the **bayes theorem** is derived from the two dependent events occurred one after other. Suppose I have bag of 5 marbles which are 3 red and 2 green then probability of red in first event is 3/5 and the probability of green in second event is 2/4 so mathematically is given by

**P(Red and Green) = P(R) + P(G/R)**

P(G/R) is called **Conditional Probability.**

Suppose we have x1, x2, x3 features and output variable y is yes or no, then naive bayes probability with respect to yes and no is given by

Text

Description automatically generated with medium confidence

**Training Process:**

During Training for all unique values present in every feature it will calculate the probabilities with respect to **Yes** and **No** and inference data we are using naive bayes formula with respect to **yes** and **no** and finally taking highest probability class

**Why is it called Naïve Bayes?**

It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features

**Disadvantages:**

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

**Refer:**

[Understanding Naïve Bayes algorithm | by Vaibhav Jayaswal | Towards Data Science](https://towardsdatascience.com/understanding-na%C3%AFve-bayes-algorithm-f9816f6f74c0)

## Clustering:

A Clusters refers to a collection of data points aggregated together because of certain similarities

K Means Clustering**:**

K-means is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid.

Steps:

1. Primarily, choose the numbers of cluster, k assumes k=2
2. Randomly selecting the two data points as centroids.
3. Calculating the distances between every point to centroids and assign all the points to the closest cluster centroid.
4. Calculate the mean of the clusters, which becomes a new centroid and again repeat the same process un till no centroids will change its position

**How to choose the best K Value in K Means Clustering:**

**Using Elbow method:**

Plot a graph between k values and WSS (which tell how far away the points within a cluster are).

Within the sum of squares (WSS), it is defined as the sum of the squared distance between each member of the cluster and its centroid.

At One point of K, where sudden decrease in WSS and then becomes constant would be the right option of choosing the value of K.

**Silhouette Score / Coefficient:**

Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.

If it is close to 1 means, clusters are well apart from each other and clearly distinguished. If it is close to 0 means then clusters are indifferent, or we can say that the distance between clusters is not significant. And it is close to -1 means then clusters are assigned in the wrong way.

**NOTE:**

For distance-based algorithms, it is recommended to apply some standardization techniques to standard the data across the variables since it is sensitive to both the mean and variance of the datasets.

For example, like Standard Scaler or MinMax scaler

**Disadvantages of K Means Clustering:**

* Outliers will cause the centroids to be dragged, or the outliers might get their own cluster instead of being ignored. Outliers should be clipped or removed before clustering.
* If the number of dimensions increase, a distance-based similarity measure converges to a constant value between any given examples. Dimensions should be reduced before clustering them.

**Hierarchical Clustering:**

## DBSCAN Clustering:

**DBSCAN** stands for **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise.

DBSCAN works on the assumption that clusters are dense regions in space separated by lower dense regions.

DBSCAN requires only two parameters: epsilon and min Points. **Epsilon** is the radius of the circle to be created around each data point to check the density and **min Points** is the minimum number of data points required inside that circle for that data point to be classified as a**Core** **point**. If there are no minimum samples inside circle and core point was there inside circle, then that point would be **border point**. If there is no core point and no minimum samples inside a circle, then it would be an **Outlier**.**The most exciting feature of DBSCAN clustering is that it is robust to outliers**. It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

## Recurrent Neural Network:

[Recurrent neural networks](https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce) are used to model the sequential data, it mainly incorporates the technique of context vectorizing.

Context vectorizing is an approach where the input sequence is summarized to a vector such that the vector is then used to predict what the next word could be, so Context vectoring acts as a “memory” which enables RNNs to remember past information.

In the [multi-layer perceptron (MLP)](https://machinelearningmastery.com/neural-networks-crash-course/), we have an input layer, a hidden layer and an output layer. The input layer receives the input, passes it through the hidden layer where activations are applied, and then returns the output.

So here each hidden layer is characterized by its own weights and biases, making them independent to each other. **If the hidden layers are independent to each other, then the structure and the order will be lost**. So, in order to combine these hidden layers together, we should have the same weights and bias. To achieve this continuous and structured relationship between the input and output, we need to feed the input sentence word by word into all the hidden layers in a sequence, simultaneously feeding the output of the previous neuron to the next.

This is the inception of recurrent neural networks, where previous input combines with the current input, thereby preserving some relationship between the current input (x2) with the previous input (x1).

Weights remain constant in each and every input. This is called **Parameter Sharing**. **RNNs share the same weight across several time steps.**

**Back propagation in RNN:**

Since the neural network takes one word at a time, the loss calculated is based on per-word basis, So Loss is computed using the **current output and the actual output** using cross-entropy loss.

**Disadvantages of RNN:**

But two common problems which occur during the backpropagation of sequential data are vanishing and exploding gradients.

**Vanishing/Exploding gradient problems in RNN:**

We also know that the contextual vector, or the hidden state parameter, is shared across the network to **preserve order and continuity**. During initialization, the parameter is assigned with a random number which is close to zero, and when the hidden state moves forward in time it gets multiplied by itself over at different time steps, making the gradient smaller and smaller, essentially zero to a point where it vanishes.

The lower the gradient is, the harder it is for the network to update the weights, and if the gradient is zero, the weights will not be updated.

When the differentiating vector goes to zero exponentially fast, which is difficult for the network to learn some long period dependencies (for long sentences), the problem is **vanishing gradient.**

The higher the gradient is, the harder it is for the network to update the weights, and if the gradient is very high, At an extreme, the values of weights can become so large as to overflow and result in NaN values. This problem is **exploding gradient**

**How do You Know if You Have Exploding Gradients?**

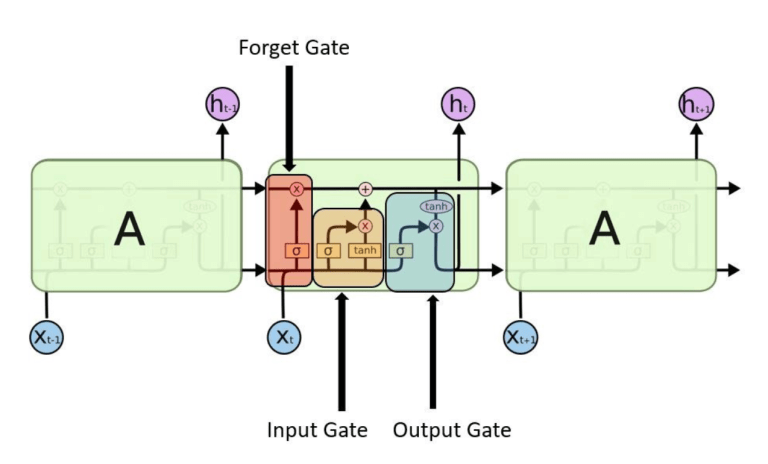
* The model is unstable, resulting in large changes in loss from update to update.
* The model loss goes to NaN during training.
* The model weights quickly become very large during training.
* The model weights go to NaN values during training.
* The error gradient values are consistently above 1.0 for each node and layer during training.

For example, Dega is a good cricketer and am aggressive at scoring runs. Here am refers to dega but there is a huge gap between these two words so that it is very hard for RNN to remember gradients for a longer period of time. This is where **LSTM** comes into the picture

## LSTM (LONG SHORT-TERM MEMORY):

LSTM was mainly built to remember the gradients for a longer period of time which RNN’S lacks.

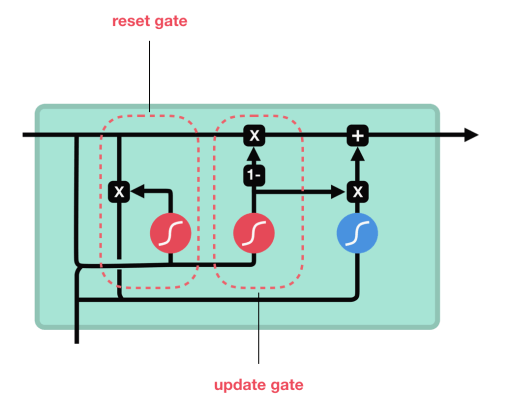
RNNs have a standard architecture where the hidden state formed using some sort of a looping mechanism to preserve and share the information for every time step. even, LSTM also has a looping mechanism. Instead of having a single neural network layer, there are four neural networks, interacting in a way to preserve and share long contextual information.



* **Cell** **state**: Cell state allows previous information to be stored in the LSTM network. It’s modulated by the internal gates namely: forget and input gate.
* **Forget Gate**: Forget gate decides what is relevant to keep from the prior cell state.
* **Input Gate**: Input gate decides what information is relevant to add from the current step
* **Output Gate**: output gate decides what the next hidden state would be

## Gated Recurrent Unit (GRU):

The [GRU](https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be) is the newer generation of Recurrent Neural networks, and it’s the modified version of LSTM but with less complexity. GRUs use the hidden state to transfer information instead of cell state. It also only has two gates, a reset gate and update gate.



**Update Gate:**The update gate selects information that needs to be added and / or ignored. It’s similar to the LSTM input gate.

**Reset Gate:**The reset gate is used to decide how much past information to forget.

Since GRUs are smaller in operation, it’s faster than LSTM networks. And the reason both LSTMs and GRUs are successful, is because their gating mechanism:

* Preserves contextual information and long-term sequences.
* Avoids gradient issues.

## Sequence to Sequence Learning:

It is also called as **Encoder – Decoder** Architecture.

Encoder reads and encodes the input sequence into a fixed context vector. Decoder then outputs a translation from the encoded fixed context vector.

Diagram

Description automatically generated

Here Potential issue is Context Vector. Context Vector may not have all necessary information if inputs have longer sentences rather using the fixed context vector, we can use the encoder of each RNN cell state with the current state to generate Dynamic context vector where Attention comes to the picture. In Attention it chooses a subset of vectors adaptively while decoding the translation.

**Teacher Forcing:**

Instead of passing the output of each state of decoder as next input, passing ground truth as the next input, to more stabilize the model.

**Limitations:**

They were limited. They could only prioritise the importance of words that were most recently processed. As they continued to move along the sentence, the importance or relevance of previous words started to diminish.

Ultimately, they couldn’t create an efficient way to “focus” on the important word in each sentence. This is the problem where Transformer network addressed by using the mechanism we already know as “attention”.

## TRANSFORMERS:

Transformers are mainly designed to improve the performance of machine translation with-out using RNN/LSTMS’s using Attention,it didn’t loop over the input multiple times – instead, the Transformer passes the input through multiple attention layers linearly.

Transformer based models can process the text in parallel unlike sequential manner in RNN’s. This means that at any time the model can be able to look at any word present in the sentence it’s processing.

A transformer model consists of two parts — Encoder Stack (multiple encoder units) and Decoder Stack (multiple decoder units). The decoder stack was mainly used to predict the future words in a sentence, while the encoder stack was used to obtain a meaningful encoded representation of the sentence so that the decoder unit can understand it better.

Diagram

Description automatically generated

In the encoder stack,each encoder layer containing two sublayers, self-attention mechanism followed fully connected feed-forward network. It processes the input text, looks for important parts, and creates an embedding for each word.

In the decoder stack, each decoder layer consists of three sublayers, first two layers are the same as the encoder layers, and the third is encoder-decoder attention over the output of the encoder stack. It takes the output of the encoder, which is an embedding, and then turns that embedding into output which is translation.

Attention can be calculated by using **Query, Key and Values.**

**Refer:** [Different Transformer Models. Introduction | by Vinithavn | Geek Culture | Medium](https://medium.com/geekculture/different-transformer-models-f6f405999e4a#:~:text=Sci-Bert%20is%20another%20domain-specific%20model%20pre-trained%20to%20perform,on%201.14%20Million%20scientific%20papers%20from%20Semantic%20Scholar.)

## Attention vs Self-Attention vs Multi-Head Attention:

**Attention:**

In Encoder-Decoder Architecture**,** Encoder reads and encodes the input sequence into a fixed context vector. Decoder then outputs a translation from the encoded fixed context vector.

Here Potential issue is Context Vector. Context Vector may not have all necessary information if inputs have longer sentences rather using the fixed context vector, we can use the encoder of each RNN cell state with the current state to generate Dynamic context vector where **Attention** comes to the picture. In Attention it chooses a subset of vectors adaptively while decoding the translation.

Attention between the input sequence and the output sequence is called **encoder decoder attention**

**Self-Attention:**

Self-Attention looks all other words present in the input sequence which can help to a better encoding for the current processing word.

Self-attention can be calculated by below steps:

1. Preparing input tokens for each word in the input sequence.
2. Initialising the weights to get Query, key, and Value matrices.
3. Multiplying the inputs with these respective weights to derive the Query, Key and Value matrices.
4. Then for each query, we are the doing the matrix multiplication with all keys present in the input sequence and applying soft-max to these attention scores.
5. Then Multiplying all these attention scores with their respective Values in the input sequence.
6. Finally adding all these weights to generate output for the first input token and repeating the same process parallelly for all input tokens.

**Multi-Head Attention:**

Self-attention is computed multiple times in the Transformer’s architecture, in parallel and independently. It is therefore referred to as Multi-head Attention**.**

**Limitations:**

One of the major drawbacks is that they’re very large models that require a **lot of memory and compute to train** and **content fragmentation** which takes only 512 tokens of input sequence if it is more than 512 needs to split the sentence and pass where we are losing the context.

**Refer**: [Transformers Explained Visually (Part 2): How it works, step-by-step | by Ketan Doshi | Towards Data Science](https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a64f34)

## BERT: Bi-directional Encoder Representation of Transformers

BERT is nothing but the encoder stack of the Transformer.

BERT only uses the encoder part of the original Transformer network since its goal is to create a model that performs a few different NLP tasks

Transformer based models can process the text in parallel unlike sequential manner in RNN’s. This means that at any time the model is able to look at any word in the sentence it’s processing.

Since these models are able to look all the text at once, so its **Bi-directionally.**

BERT doesn’t use any decoder architecture, so the output of BERT is an **embedding**, not a textual output, so we can do anything with this embedding like cosine similarities between any two sentences and so on.

**BERT Training Process:**

BERT uses an innovative training layer which performs the **Masked word Prediction** and **Next Sentence Prediction.**

**Masked word prediction** is nothing but randomly masking some tokens or words in each of the input sentences. The model is trained to predict the masked word to understand the relation between words.

a total of 15% of the words will be chosen for masking. Out of them, 1) 80% of the chosen words will be replaced by [MASK] token; 2) There is a 10% chance to replace the word with a random word, and 3) The remaining 10% words will remain unchanged.

**Next Sentence Prediction** is used to understand if there is any relation between two sentences. The model is trained to predict if a particular sentence follows another sentence or not.

Graphical user interface, application

Description automatically generated

**Why is masking needed?**

For next word generation tasks, Since the model can able to look all words at a time, it can easily predict the next word, BERT can “cheat” by looking at words, it will not learn anything.

For BERT, 15% of the input tokens are masked before the model gets to see them, so there’s no way for it to cheat. To do this, a word is randomly selected and simply replaced with a “[MASK]” token, and then fed into the model.

**Limitations:**

The BERT model is trained on a fixed 512 tokens input limitation. So, it is hard to capture long-term dependencies.

U can use the **max\_position\_embeddings** argument in the configuration while downloading the BERT model into your kernel. with this argument you can choose **512, 1024, 2048** as max sequence length.

There are new architectures that specifically deal with longer sequences, like Longformer, Reformer, Big Bird and Linformer.

**Distil BERT:**

Distil BERT, a distilled version of BERT: smaller, faster, cheaper, and lighter.

The Distil BERT model used the knowledge distillation method to train a model with 97% of the BERT’s ability but 40% smaller in size (66M parameters compared to BERT-based’ s 110M) and 60% faster. The process consists of having a **teacher and student model** and trying to make the student model results close to the teacher one. It is not a model with State-of-the-art (SOTA) results, but it is faster and competitive.

**Roberta:**

Roberta stands for Robustly Optimized BERT Pre-training Approach and the main goal of Roberta is to optimize the training of BERT architecture to take lesser time during pre-training.

**Modifications to BERT:**

* Removing the Next Sentence Prediction (NSP) objective:
* Training with bigger batch sizes & longer sequences.
* Dynamically changing the masking pattern
* with a higher learning rate
* for more epochs
* much more data
* they also changed the tokenization method to byte-level BPE.

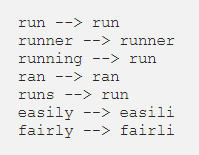
These changes resulted in a model with **125M** parameters (Roberta-base) with a SOTA score on a couple of NLP downstream tasks

**Conclusion:**

If you really need a faster inference speed but can compromise few-% on prediction metrics, Distil BERT is a starting reasonable choice, however, if you are looking for the best prediction metrics, you’ll be better off with Facebook’s Roberta.

## Stemming and Lemmatization:

**Stemming** and **Lemmatization** are the process of producing morphological variants of a root/base word.



But here only difference is that stem may not be an actual word whereas, lemma is an actual language word and lemmatization takes more time than stemming as its need to look at the language models.

**Text Pre-processing:**

* Removing HTML Tags
* Removing Accented Characters - 'Sómě Áccěntěd těxt' using unicodedata
* Expanding Contractions – “Y’all can't expand contractions I'd think” using contractions
* Removing Special Characters - using Regex
* Tweaking with case conversions if required
* Removing extract new lines
* Removing extra white spaces
* Removing stop words

## Bag of words (Count Vectorizer):

Extracting the features from text (means converting the text into numerical representation).

Count Vectorizer is a way to convert a given set of strings into a frequency representation

Text1 = “Natural Language Processing is a subfield of AI”

tag1 = "NLP"

Text2 = “Computer Vision is a subfield of AI”

tag2 = "CV"

**output:**

Table

Description automatically generated with medium confidence

**Limitations:**

1. **Sparsity**: If we use all English words for BOW, the vector will be very long, but very few non zeroes.
2. Frequent words have more power
3. Ignoring word orders (different sentences have same word vector)
4. Different sentences have same word vector
5. BOW cannot handle out of vocabulary words
6. Inability of identifying which are more important and less important words for this using tf-idf

**WHY N-GRAM?**

1. To Overcome BOW drawbacks (ignoring word order)
2. Naïve next word prediction

## TF – IDF (Term Frequency – Inverse Document Frequency):

TF measures how frequently the term occurs in a document and IDF measures the logarithmic of total number of documents divided by number of documents contains this term.

Works on penalising the words which are frequently used and gives more importance which occurred rarely.

So, this is not only based on the frequency of a word in the corpus, but it also provides a numerical representation of how important a word is for statistical analysis.

Table

Description automatically generated

**Advantages:**

1. Keep relevant words score.
2. Lower just frequent words score.

**Limitations:**

1. Both Count vectorizer and TF-IDF basically works on only count of words.
2. Weak on capturing document topic.
3. Similarity is not based on Topic level or situation level it is just based on word level
4. It also doesn’t identify the relationships between words such as linguistic similarity between words.

To Overcome the above issues, we should use word embeddings.

**Why do we use Log in TF-IDF?**

While computing TF, all words are considered equally important. However, it is known that certain words, such as "is", "of", and "that", may appear a lot of times but have little importance**. As we need to weight down the frequent terms to scale up the rare one’s we use Log in IDF.**

## Latent Semantic Analysis:

LSA uses the statistical approach to identify the association among the words in a document.

Singular Value Decomposition is the statistical method that is used to find the latent(hidden) semantic structure of words spread across the document

Here Similarity is based on Topics. **LSA** leverages the context around the words to capture the hidden concepts, also known as topics.

C = collection of documents.

d = number of documents.

n = number of unique words in the whole collection.

M = d X n

The SVD decomposes the M matrix i.e word to document matrix into three matrices as follows

M = U∑VT

U = distribution of words across the different contexts

∑ = diagonal matrix of the association among the contexts

VT = distribution of contexts across the different documents



Steps:

1. Initially it creates the Word – Document Matrix with TF-IDF scores.
2. Then it will reduce the dimensions of the matrix using SVD
3. SVD decomposes a matrix into three other matrices called U, S and Transpose of V.

Example: mobile, phone, cell phone, telephone all are similar but if we pose a query like “The cell phone has been ringing” then the documents which have “cell phone” are only retrieved whereas the documents containing the mobile, phone, telephone are not retrieved

## Word Embeddings:

## Word2Vec:

Word2Vec (W2V) is an algorithm that accepts text corpus as an input and outputs a vector representation of each word in n-dimensional space. Similar words are closer together in n-dimensional space

Diagram

Description automatically generated

Here the Similarity comes from neighbour words and will get the semantic relationship between the words.

There are two flavors of this algorithm namely: CBOW and Skip-Gram.

**Skip-Gram:**

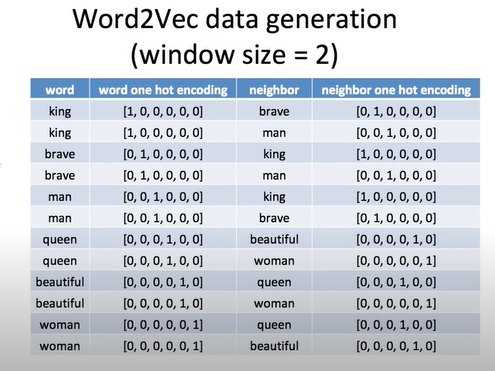
Uses the current word to predict the neighbour words which is its context

Text1: “King brave Men”

Text2:” Queen beautiful woman”

Table

Description automatically generated



**CBOW:**

It tries to predict a target word from a list of context words (neighbour words)

Diagram

Description automatically generated

Similar words are closer together in n-dimensional space.

**Limitations:**

1. It cannot handle out of vocabulary words
2. There is no sub-linear relationship between words.

## GlOVE(Global Vectors):

Glove model uses a co-occurrence counts matrix to make the embeddings. Each row of the matrix represents a word, while each column represents the contexts that word can appear in. The matrix value represents the frequency of a word appears in a given context. Then dimensionality reduction is applied to this matrix to create the resulting embedding matrix (each row will be the embedding vector).

The advantage of Glove is Unlike Word2vec, GLOVE doesn’t just rely on local statistics (local context information of words) but incorporates global statistics (word co-occurrence) to obtain word vectors.

To derive the sub-linear relationships between words, we need to create Word Co-Occurrence matrix (derive the semantics relationship between words)

Example is the cat sat on the mat (here it will calculate the pair of words for given window size so that will get to know the semantic relationship between word pairs)

Both **Word2vec** and **Glove** do not solve the problems like:

* How to learn the representation for out-of-vocabulary words.
* How to separate some opposite word pairs. For example, “good” and “bad” are usually located very close to each other in the vector space, which may limit the performance of word vectors in NLP tasks like sentiment analysis.

## FastText:

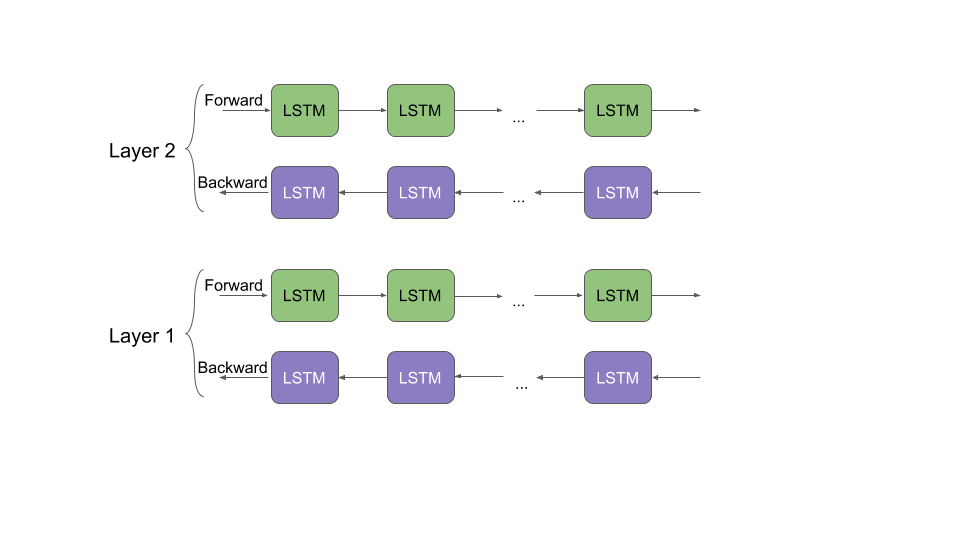
FastText is an extension to Word2Vec proposed by Facebook in 2016. Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words).

For instance, the tri-grams for the word apple *is* app, ppl*, and*ple (ignoring the starting and ending of boundaries of words)

Although it takes longer time to train a FastText model (number of n-grams > number of words), it performs better than Word2Vec and allows **rare words** to be represented appropriately.

## ELMO (Embeddings from Language Models):

Unlike traditional word embeddings such as word2vec and GLOVE, the ELMO assigned a vector to a token or a word depends on current context, So the same word can have different word vectors under different contexts, also ELMO representations are purely character based so they are not limited to any predefined vocabulary.

ELMO word vectors are computed on top of a two-layer bidirectional language model. This bidirectional language model has two layers stacked together. Each layer has 2 passes — forward pass and backward pass:

* The raw word vectors act as inputs to the first layer of bidirectional language model.
* The forward pass contains information about a certain word and the context (other words) before that word.
* The backward pass contains information about the word and the context after it.
* This pair of information, from the forward and backward pass, forms the intermediate word vectors.
* These intermediate word vectors are fed into the next layer of bidirectional language model which results another intermediate word vectors.
* The final representation (ELMO) is the weighted sum of the raw word vectors and the 2 intermediate word vectors.

## BERT (Transformer based word embeddings):

Transformer based models can process the text in parallel unlike sequential manner in RNN’s. This means that at any time the model is able to look at any word in the sentence it’s processing.

Since these models are able to look all the text at once, so its **Bi-directionally.**

BERT only uses the encoder part of the original Transformer network since its goal is to create a model that performs a number of different NLP tasks.

BERT doesn’t use any decoder architecture, so the output of BERT is an **embedding**, not a textual output, so we can do anything with this embedding like cosine similarities between any two sentences and so on.

**The structure of a perceptron**

Diagram, schematic

Description automatically generated

**Perceptron's linear function**

## A picture containing text Description automatically generated

**Perceptron’s non-linear (activation) function**

Timeline

Description automatically generated

## ACTIVATION FUNCTIONS

## Diagram Description automatically generated

**Now the question is – if the activation function increases the complexity so much, can we do without an activation function?**

A neural network without an activation function is essentially just a linear regression model.

every neuron will only be performing a linear transformation on the inputs using the weights and biases. Although linear transformations make the neural network simpler, but this network would be less powerful and will not be able to learn the **complex** patterns from the data.

**To make the nonlinearity** to the neural network we use **activation function.**

**Count only the layers that have weights**: n-1 layers have weights where n is the total number of layers including input, hidden and output layers.

**Sigmoid Function:**

Chart, line chart

Description automatically generated

The output of Sigmoid function always ranges between **0 to 1**. If any value lessthan 0.5 considering it as 0 or any value greater than 0.5 considering it as 1. Derivative of signmoid function is **0 to 0.25**

Currently, we do not usually use the sigmoid function for the hidden layers in MLPs and CNNs. Instead, we **use ReLU or Leaky ReLU** there due to the vanishing gradient problems.

The sigmoid function must be used in the **output layer** when we build **a binary classifier** in which the output is interpreted as a class label depending on the probability value of input returned by the function.

Diagram

Description automatically generated

**Tanh Function:**

The Output of Tanh function ranges between -1 to +1 and derivative of tanh function ranges from 0 to 1.

We never use the tanh function in the output layer. Since we don’t get any -ve values in the output layer.

Chart, line chart

Description automatically generated

We do not usually use the tanh function in the hidden layers because of the following drawbacks.

* The tanh function has the vanishing gradient problem.
* This function is computationally expensive as an e^z term is included.

**ReLU activation function:**

If the input value is 0 or greater than 0, the ReLU function outputs the input as it is. If the input is less than 0, the ReLU function outputs the value 0.

Chart, line chart

Description automatically generated

The convergence is faster than sigmoid and tanh functions. This is because the ReLU function has a fixed derivate (slope) for one linear component(for +ve values) and a zero derivative for the other linear component (-ve values). Therefore, the learning process is much faster than sigmoid and tanh functions.

**Drawbacks:**

* The main drawback of using the ReLU function is that it has a dying ReLU problem.
* The value of the positive side can go very high. That may lead to a computational issue during the training.

**Leaky ReLu Activation Function:**

The main drawback of using the ReLU function is that it has a dying ReLU problem. Especially for -ve values derivative will be 0 so that weights are not going to update during backpropagation. To overcome this issue we are using Leaky ReLu function which outputs a small negative values

For example, -ve values leaky ReLu function is using 0.01 so the derivative of this also becomes small values.

**Softmax Activation Function:**

We must use the softmax function in the **output layer** of a **multiclass** classification problem.

The softmax function calculates the probability value of an event (class) over K different events (classes). It calculates the probability values **for each class**. The **sum of all probabilities is 1** meaning that all events (classes) are mutually exclusive.

Diagram

Description automatically generated

**Summary of Activation Functions:**

* No activation function is required in the input layer nodes of a neural network. So, you don’t need to worry about activation functions when you define the input layer.
* The output layer activation function depends on the type of problem that we want to solve. In a regression problem, we use the linear (identity) activation function with one node. In a binary classifier, we use the sigmoid activation function with one node. In a multiclass classification problem, we use the softmax activation function with one node per class. In a multilabel classification problem, we use the sigmoid activation function with one node per class.
* In MLP and CNN neural network models, **ReLU** or **Leaky ReLu** is the default activation function for hidden layers.
* In RNN neural network models, we use the **sigmoid or tanh** function for hidden layers. The tanh function has better performance.
* We never use softmax and identity functions in the hidden layers.

**Training Process in any Neural Network:**

The entire learning process can be divided into three main parts:

* Forward propagation (Forward pass)
* Calculation of the loss function
* Backward propagation (Backward pass/Backpropagation)

Diagram

Description automatically generated

**Forward propagation:**

First, we assign non-zero random values to weights and biases. This is called **parameter initialization** of the network. Based on these assigned values and the input values, we perform the calculations in each neuron of the network like calculation of neurons activation function.

The **weights** control the level of importance of each input while **biases** determine how easily a neuron **fires or activates.**

**Cost Function:**

The Cost function computes a score between the predicted values and ground truth values which is called the error of the model or loss score. The loss function captures how **well the model performs** in each iteration. We use the **loss score** as a feedback signal to update parameters in the backpropagation part.

Common loss functions **are:**

* **MSE, MAE, RMSE** mainly used for **Regression** problems
* **Binary cross-entropy(Log Loss)** is used to measure the performance the **binary** classification problems
* **Multi-class Cross-entropy/Categorical Cross-entropy** are used to measure the performance of multi-class (more than two classes) classification problems**.**
* **Sparse-categorical-cross-entropy is** used to measure the performance of multi-label classification problems**.**

**Backward propagation:**

In the first iteration, the predicted values are far from the ground truth values because we initially assigned arbitrary values to the network’s parameters (weights and biases). Those values are not optimal values. So, we need to update the values of these parameters in order **to minimize** the loss function. The process of updating network parameters is called **parameter learning or optimization** which is done using an **optimization algorithm (optimizer)** that implements **backpropagation**.

The objective of the optimization algorithm is to find the **global minima** where the loss function has its **minimum value**. If the algorithm is stopped at **a local minimum**, we’ll not get the minimum value for the loss function. Therefore, our model will not perform well.

Chart, line chart

Description automatically generated

Commonly used Optimizers are gradient descent, stochastic gradient descent, Adam, Adagrad, Adadelta, Adamax, RMSProp( Root Mean Squared Propagation).

**The batch size and epochs:**

We do not usually use all training samples (instances/rows) in one iteration during the neural network training. Instead, we specify **the batch size** which determines the number of training samples to be propagated (forward and backward) in one iteration

An **epoch** is an iteration over the entire training dataset.

For example, let’s say we have a dataset of **1000** training samples and we **choose a batch size of 10 and epochs of 20.**

In this case, our dataset will be divided into 100 (1000/10) batches each with 10 training samples.

According to this setting, the algorithm takes the first 10 training samples from the dataset and trains the model. Next, it takes the second 10 training samples and trains the model and so on. Since there is a total of 100 batches, the model parameters will be updated 100 times in each epoch of optimization. This means that one epoch involves 100 batches or 100 times parameter updates. Since the number of epochs is 20, the optimizer passes through the entire training dataset 20 times giving a total of 2000 (100x20) iterations!

**Practical Implemetation:**

|  |
| --- |
|  |
|  | from tensorflow.keras.layers import InputLayer |
|  | from tensorflow.keras.layers import Dense  from tensorflow.keras.models import Sequential |
|  |  |
|  | model = Sequential() |
|  |  |
|  | model.add(InputLayer(input\_shape=(784, ), name='Input\_Layer')) |
|  | model.add(Dense(512, activation='relu', name='Hidden\_Layer\_1')) |
|  | model.add(Dense(256, activation='relu', name='Hidden\_Layer\_2')) |
|  | model.add(Dense(10, activation='softmax', name='Output\_Layer')) |
|  |  |
|  | model.summary() |
|  | model.compile(loss=’ categorical\_crossentropy’, optimizer=’adam’, metrics=[‘accuracy’]) |
|  | model.fit(train\_data,train\_labels,epochs=20,batch\_size=10) |

In input layer 784 means 784 neurons are there in where in first hidden later we have 512 neurons followed by 256 neurons in second hidden layer followed by 10 neurons each for one class in multi calssification problem.

In sequential models, the input, hidden and output layers are stacked in the model sequentially

Diagram

Description automatically generated

**How to calculate Trainable Parameter:**

The total number of trainable parameters is equal to the number of total elements in weight matrices and bias vectors.

* From input layer to the first hidden layer: 784 x 256 + 256 = 200,960
* From the first hidden layer to the second hidden layer: 256 x 256 + 256 = 65,792
* From the second hidden layer to the output layer: 10 x 256 + 10 = 2570
* Total tranable parameters: 200,960 + 65,792 + 2570 = 269,322

Table

Description automatically generated

## Vanishing/ Exploding gradient problem:

During back propagation, in a network of n hidden layers, n derivatives will be multiplied together. If the derivatives are large then the gradient will increase exponentially as we propagate down the model until it eventually explodes, and this is what we call the problem of **exploding gradient**. Alternatively, if the derivatives are small then the gradient will decrease exponentially as we propagate through the model until it eventually vanishes, and this is the **vanishing gradient** problem.

**How do you know that your model has Exploding Gradient Problem?**

* The model is not learning much on the training data therefore resulting in a poor loss.
* The model will have large changes in loss on each update due to the model instability.
* Model weights grow exponentially and become very large when training the model.

**How do you know that your model has Exploding Gradient Problem?**

* The model will improve very slowly during the training phase and it is also possible that training stops very early, meaning that any further training does not improve the model.
* Model weights shrink exponentially and become very small when training model the model.

**Solutions:**

1. Reducing the amount of layers
2. Gradient Clipping – Checking and limiting the size of gradients
3. Weight Initialisation Techniques – Weights should be medium small (he-uniform, he-normal, Xavier-uniform, Xavier-normal)

**Weight Initialization Techniques:**

Its main objective is to prevent the outputs **from exploding or vanishing gradients** during the forward propagation. If either of the problems occurs, loss gradients will either be too large or too small, and the network will take more time to converge.

If we initialized the weights correctly, then optimization will be achieved in the least time otherwise converging to a minimum using optimizers will be impossible.

Some of the important weight initialization techniques are uniform distribution, Xavier distribution for normal & uniform and He - init for normal & uniform

Xavier initialization works well for sigmoid activation functions and He – init works well for ReLu Activation functions

**Loss Functions and Cost Functions:**

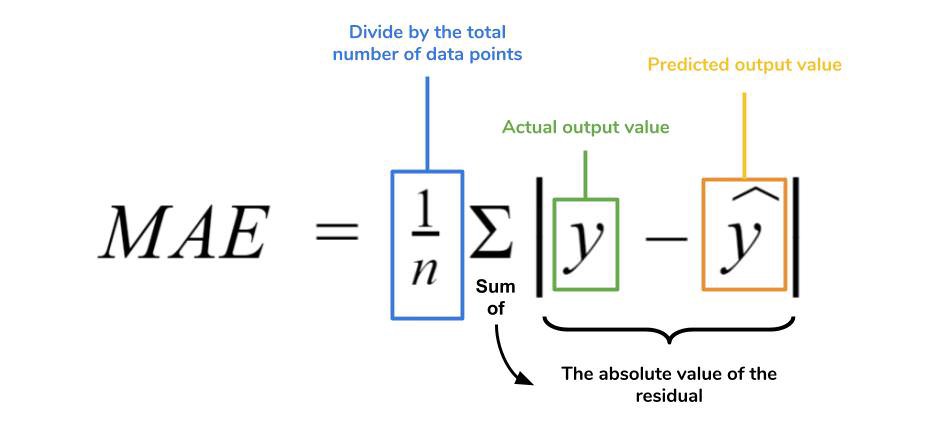
The term “loss” refers to the error in the prediction of a neural network. A loss function is a function that calculates the loss for a certain prediction. The loss function is required by the optimizer in order to decide what steps it should take to minimize the loss.

While the **loss function** calculates the error for a single data point (sample), the **cost function** calculates the loss for the entire dataset. The cost of a neural network is nothing but the **sum of losses** on individual training samples.

MAE, MSE, RMSE are used in regression problems while binary, categorical, sparse categorical cross entropy are used in classification problems.

**Mean Absolute Error (MAE):**

MAE is nothing but the absolute differences between the predicted values and the actual values.

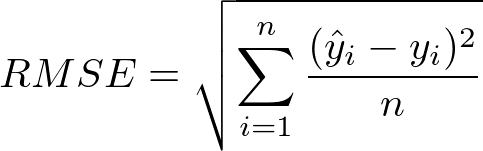


**Mean Squared Error (MSE):**

It is very similar to MAE except it squares the individual errors instead of calculating their absolute values.

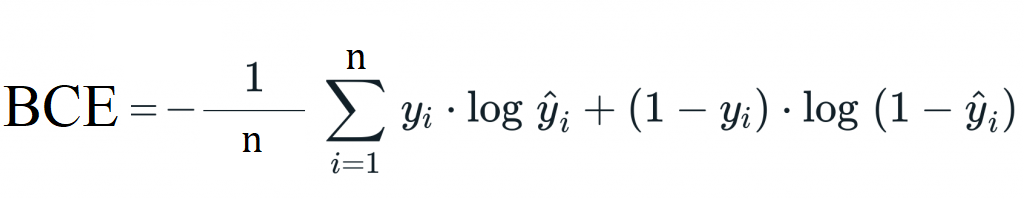
**Root Mean Squared Error (RMSE):**

RMSE can be calculated simply by taking the square root of the MSE for a particular set of true and predicted labels.



**Binary Cross-entropy:**

The binary cross-entropy loss function, also called as log loss, is used to calculate the loss for a neural network performing binary classification, i.e. predicting one out of two classes.



y\_true = [0,  1,  0,  0]

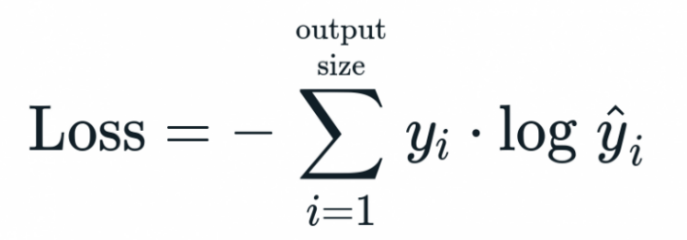
y\_pred = [-18.6,  0.51,  2.94,  -12.8]

BCE = tf.keras.losses.binary\_crossentropy(y\_true, y\_pred).numpy()

BCE = 4.0016456

**Categorical Cross-entropy:**

Categorical cross-entropy is the loss function used **for multi-class classification** tasks. It has pretty much the same formula as binary cross-entropy except a few changes:



y\_true = [[1, 0, 0],

          [0, 1, 0],

          [0, 0, 1],

          [1, 0, 0]]

y\_pred = [[0.98, 0.0, 0.02],

          [0.0, 0.99, 0.01],

          [0.03, 0.01, 0.96],

          [0.42, 0.02, 0.57]]

CCE = tf.keras.losses.categorical\_crossentropy(y\_true, y\_pred).numpy()

CCE = array([0.02020269, 0.01005033, 0.04082202, 0.8774509 ], dtype=float32

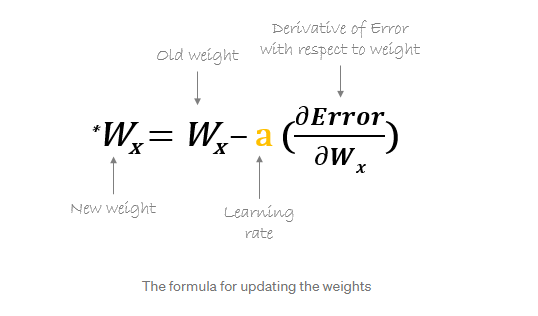
**Sparse Categorical Cross-entropy:**

The categorical cross-entropy becomes very memory in-efficient when we have a large number of classes, say 1000. This means we have a large array of all zeros and a single 1. In such cases, we use the sparse categorical crossentropy loss function. This loss function works on label-encoded data instead of one-hot encoded data, which makes computation very fast when working with a large number of classes.

## Optimizers:

During training process of network, optimizer tries to lower the loss function by updating the model parameters in response to the output of the loss function. Thereby helping to reach the Global Minima with the lowest loss and most accurate output.

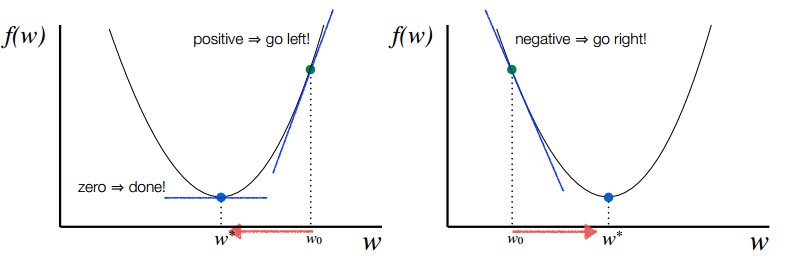
**Updating weights:**



## Gradient Descent:

Gradient Descent is an optimizer which tries to reduce the loss function by updating the weights. Usually Gradient Descent follows the cure ‘U’ Shape (convex function shape)

Suppose if we draw a tangent at any particular weight, based on the tangent line we should know whether we need to increase the weights or decrease the weights. If the tangent line pointing downwards then we can consider it as a negative slope and have to increase the weights for negative slope and If the tangent line pointing upwards then we consider it as a positive slope, for positive slopes we have to reduce the weights.



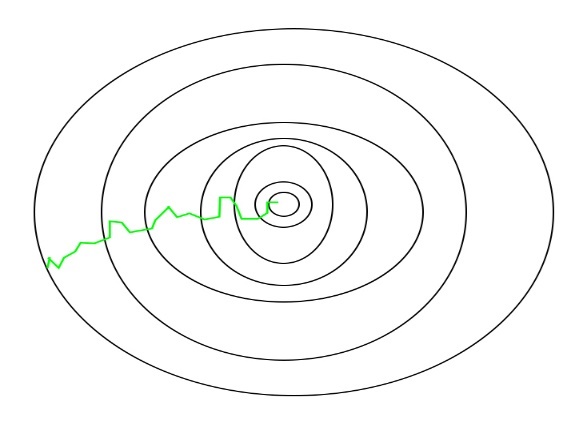
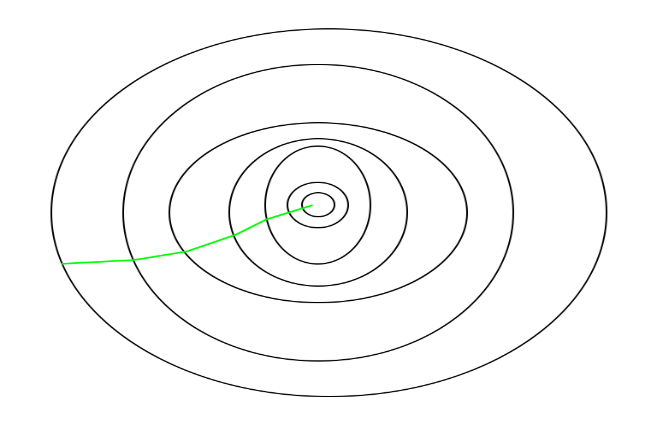
**Global Minima and Local Minima:**

Usually in machine learning techniques, Gradient Descent graph looks like convex function shape but whereas in deep learning techniques, Gradient Descent graph looks like non convex shape

**Stochastic Gradient Descent:**

In Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration like in gradient descent. since only one sample from the dataset is chosen randomly for each iteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn’t matter how the path taken by the algorithm as long as we reach the global minima with a significant shorter time.

Left side is path taken by Gradient descent whereas right side is path taken by SGD.



**Mini-Batch Gradient Descent Optimizers:**

Instead of taking single data point form the dataset, if we consider a batch of data for updating the weights unlike in gradient descent is called mini batch gradient descent optimizer.

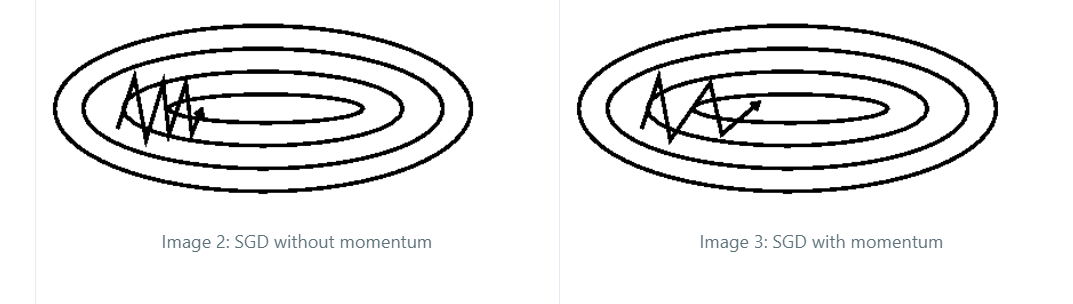
**Drawbacks of all GD optimizers:**

* Gradient Descent uses the whole training data to update weight and bias. Suppose if we have millions of records then training becomes slow and computationally very expensive.
* SGD solved the Gradient Descent problem by using only single records to updates parameters. But, still, SGD is slow to converge because it needs forward and backward propagation for every record. And the path to reach global minima becomes very noisy.
* Mini-batch GD overcomes the SDG drawbacks by using a batch of records to update the parameter. Since it doesn't use entire records to update parameter, the path to reach global minima is not as smooth as Gradient Descent.

**SGD With Momentum:**

Here we are using the Exponentially Weighted Averages to compute Gradient and used this Gradient to update parameter. Exponentially Weighted Averages means over the time for the data points we are giving more importance to the current data and giving less importance to the old data points and so on.

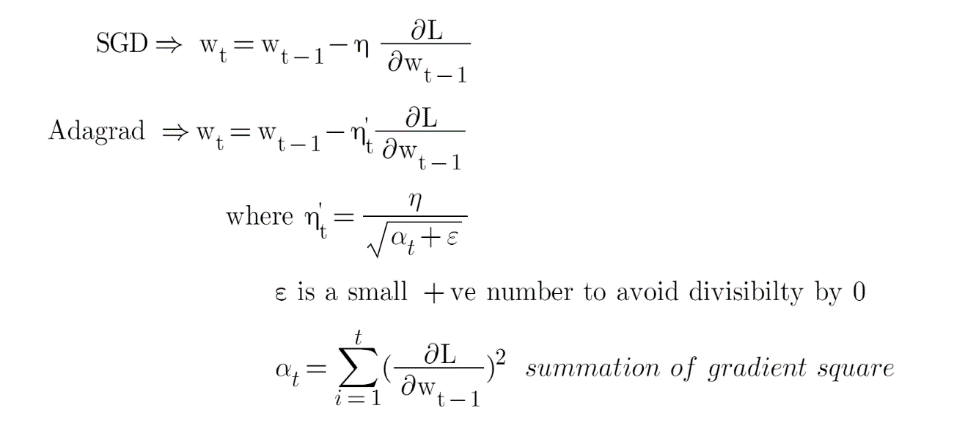
It means the present Gradient is dependent on its previous Gradient and so on. This accelerates SGD to converge faster and reduce the oscillation.



**ADAGRAD Optimizer (Adaptive Gradient Descent):**

The idea behind Adagrad is to use different learning rates for each parameter base on iteration. In real-world dataset, some features are sparse (for example, in Bag of Words most of the features are zero so it’s sparse) and some are dense (most of the features will be non-zero), so keeping the same value of learning rate for all the weights is not good for optimization.

Here using the sum of square of gradients 1 to t time steps called as alpha and learning rate is calculated by



**Advantages of Adagrad:**

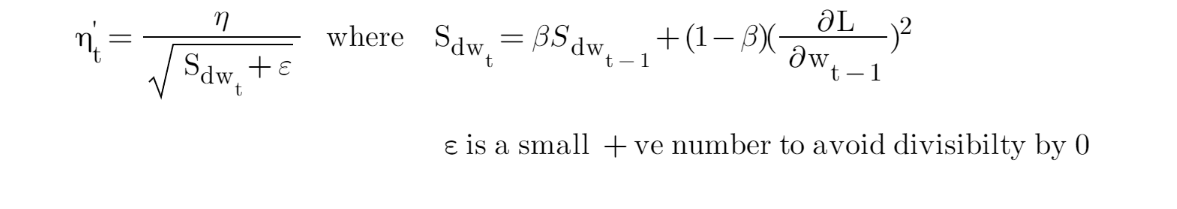
* No manual tuning of the learning rate required.
* Faster convergence
* More reliable

One main disadvantage of Adagrad optimizer is that alpha(t) can become large as the number of iterations will increase and due to this nt’ will decrease at the learing rate. This will make the old weight almost equal to the new weight which may lead to slow convergence.



**Ada-Delta Optimizer :**

This is extension of Adaptive gradient optimizer. Here instead of using the sum of the previous squared gradients, using weighted average of all past squared gradients and averages out.

So, here unlike the alpha “α” in Adagrad, where it increases exponentially after every time step. In Adadelta, using the exponentially weighted averages over the past Gradient, an increase in “Sdw” is under control. This is called **adaptive Learning rate.**

**Adam Optimizer:**

Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter. The idea behind Adam optimizer is to utilize the momentum concept from “SGD with momentum” and adaptive learning rate concept from “Ada delta”.

It means **Adam** uses the weighted average of all past squared gradients like in Adadelta & RMSProp and also uses the weighted average of all the gradients like in SGD with momentum.

**Advantages:**

* Computationally efficient
* Little memory requirements

**Regularization Methods for Neural Networks:**

When training neural networks, the main concern is the problem of **overfitting**. The main reason for overfitting in neural networks is that they are **highly flexible models** that tend to **memorize** the training data and capture the noise in the data.

* It prevents neural networks from overfitting the training data.
* It increases the generalization capability of the neural network models on new unseen data.
* Generally, it improves the overall performance of neural networks.

Timeline

Description automatically generated

**L1 and L2 Regularization:**

Both L1 and L2 regularization is nothing but **weight/parameter** regularization. This type of regularization keeps the weights of the neural network **small** (near zero) by adding **a penalizing term** to the **loss function**

So Neural networks with small weight values are not much sensitive to the noise present in the input data.

L1 norm: λ \* (Sum of the absolute values of the weights)

L2 norm: λ \* (Sum of the squared values of the weights)

Graphical user interface, text, application

Description automatically generated

The differences between L1 and L2 regularization

* **Difference by function**: In L1 regularization, more weights are equal to zero. Therefore, the nodes associated with these weights are **completely removed** from the network. In L2 regularization, the weights have very small values which are not zero, but close to zero. L2 regularization **decreases the effect of the nodes on the output**. However, those nodes are not completely removed from the network.
* **Difference by usage**: L2 regularization is more commonly used in deep neural networks than L1 regularization.
* **Difference by formation**: In L1 regularization, we use **the sum of the absolute values of the weights** multiplied by λ. In L2 regularization, we use the **sum of the squared values of the weight**s multiplied by λ.

**Drop out ratio:**

At every iteration, it randomly selects some nodes and removes them along with all their incoming and outgoing connections. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass, if neurons are randomly dropped out of the network during training, then other neurons will have to step in and handle the things required to make predictions for the missing neurons. This reliant on context for a neuron during training is referred to complex co-adaptations which result in better generalization and is less likely to overfit the training data.

Diagram

Description automatically generated

By randomly removing nodes from the network, a smaller network is used for updating parameters. When the network becomes smaller, it provides less flexibility which results in reducing overfitting.

from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense,

Dropoutmodel = Sequential()

model.add(Dense(256, input\_shape=(784,), activation='relu'))  
**model.add(Dropout(0.2)) # Adds Dropout to the first hidden layer**  
model.add(Dense(256, activation='relu'))  
**model.add(Dropout(0.1)) # Adds Dropout to the second hidden layer**  
model.add(Dense(10, activation='softmax'))

where 0.2 ,0.1 means 2% , 3% of neurons gets deactivated in their respective hidden layers.

**Early Stopping:**

The idea behind early stopping is straightforward. We intentionally stop the training process of the model early right before the model begins to overfit**.**

from tensorflow.keras.callbacks import EarlyStopping  
es\_callback = **EarlyStopping(monitor='val\_loss', mode='auto',  
patience=0)**

history = model.fit(X\_train, y\_train,   
validation\_data=(X\_test, y\_test),  
**epochs=20**,   
**callbacks=[es\_callback]**)

## Statistics:

**Statistics** is the science of collecting, organizing, and analysing the data which makes better decision making.

Facts or pieces of information that can be measured is called **Data**

**Example: -** Age of students in a class is {24,54,55,32,12,43,21}

**Types of statistics:**

**Descriptive statistics:**

It consists of **organizing** and **summarizing** the data.

**Example: -** Calculate the **average, max, mean** marks of a student in a class?

**Inferential Statistics:**

It is the technique where we used the data to make the **conclusions**.

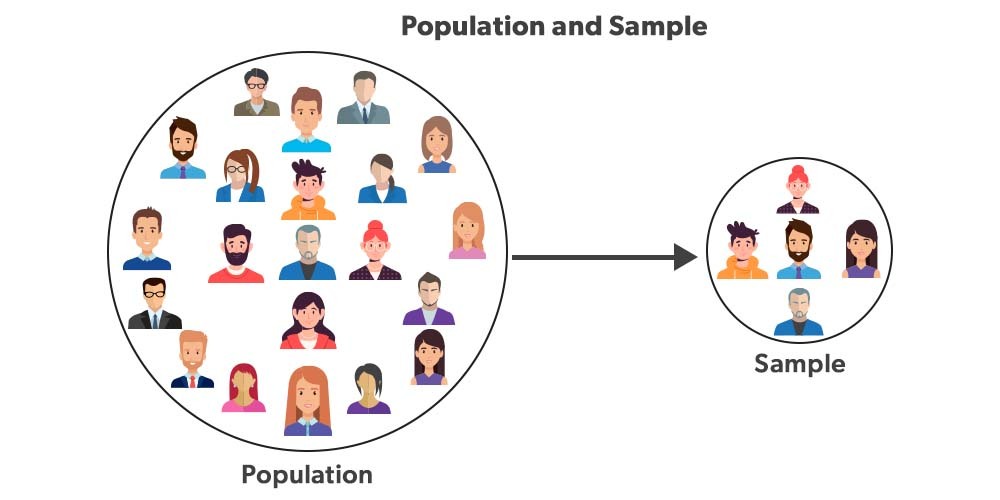
**Example: -** Are the average marks of this class is similar to the maths classroom marks in the college?

Here population dataset is college and sample dataset is maths classroom level data.

**Population and Sample Data:**

In statistics **population** is the entire set of items from which data is drawn in the statistical study and the population is usually denoted with N.

A sample represents a group of the interest of the population which we will use to represent the data. A sample is a group of the elements participating in the survey or study and sample is denoted by the n.



**Sampling and its types:**

The process of collecting info from the sample is called sampling.

**Random Sampling:**

When performing random sampling, every member of population(N) has an equal chance of being selected for your sample(n).

**Example: -** Just select randomly in any population dataset as there is no particular rule to select here.

**Stratified Sampling:**

It is nothing but where the population(N) IS split into non-overlapping groups(strata).

**Examples: -**

* split the dataset based on gender like male and female.
* Split the dataset in age groups wise like (0-10), (11-30), (31-60).
* Split the dataset by profession like doctor, software, and others.

**Systematic Sampling:**

It is nothing but selecting the data for every nth individual.

**Examples: -** For every 8th person am conducting the covid survey, here there is no particular rule of why selecting 8th person it’s just our choice.

**Convenience/Cluster Sampling:**

Consider that am doing a survey only those people who are a domain expert in that particular survey.

For example, conducting the survey who are experts in data science

**Exit Poll** – Random Sampling

**Household Survey** – Convenience Sampling if we apply conditions on household survey like women’s

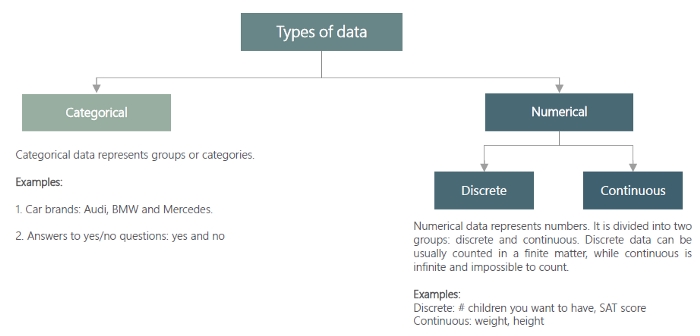
Survey then it would be also a stratified sampling.

**Variables:**

A variable is a property that can take on any value.

**Examples: - height** = {21,21,23,21,23,43,21}

There are two types of variables we nave namely quantitative which measures numerically and qualitative which is nothing but categorical variables which measures groups or categories like cars, gender, brand and so on.



**Discrete** examples are how many children’s we have, how many cars in the street which is a whole number whereas **continuous** examples are what is the height, what is the weight anything which is real or fractional number.

**Variable Measurement Scales:**

**Nominal Variables**: They are not numbers and cannot be put in any order.

**Example**: names, colours, gender

**Ordinal Variables**: Consists of groups and categories that follow a strict order. So here value doesn’t matter order is important.

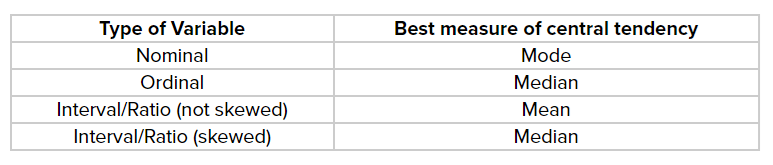
**Example**: Grades (e.g., Bad, Satisfy, Good), rank based on marks

**Interval Variables:** Order matters and value also matters but natural zero is not present.

**Example**: In Temperatures we can have like 30-40, 40-50, 60-70 but 0 is meaningless which is natural zero is not present.

**Ratio:** Represented by numbers and has a true zero.

Example: However, if you analyse temperature in Kelvins, the absolute zero temperature is 0º Kelvin, thus now you can say the temperature value is a Ratio since it has a true zero.



Relation between mean, median and mode: **Mode = 3 Median — 2 Mean.**

**Frequency Distribution:**

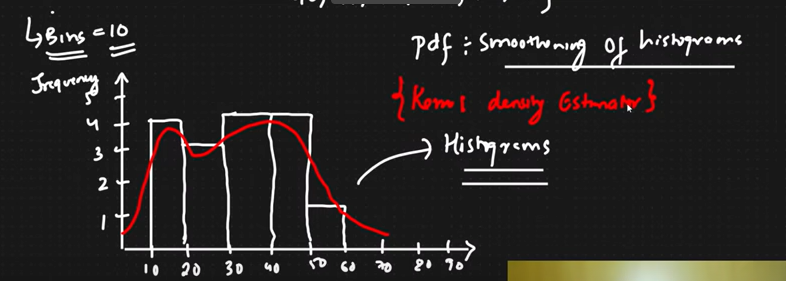
Suppose if we have a dataset like lotus, sunflower, rose, rose, sunflower, lotus, rose, lotus, rose

Then frequency is lotus – 3, sunflower-2, rose-4

So here we can draw a **BAR graph** of flowers frequency distribution since the data is **categorical**.

If the data is **Numerical** data, then we can use **Histogram’s** for frequency distribution like a – {10,22,11,22,13,24,35,43,23,54,43,45,46,56,42,54,32}

Here x-axis is nothing, but bins and y axis is frequency



**Measure of Central Tendency:**

Refers to the measure used to determine the centre of the distribution of data.

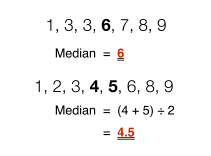
**Mean:** Nothing butdividing the sum by the number of components and it is easily affected by **outliers** so mean might not be the good imputation for missing data when u have outliers in data.

Population mean is denoted by ***μ***

Sample mean is denoted by X̅



**Median:** Nothing but middle value after sorting to ascending order. Median works well for **outliers**.



**Mode:** Nothing but most frequent element. It can be used for numerical as well as categorical variables and mode is also not good for outliers

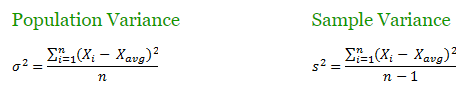
In **Normal distribution** all these, mean, median, model fall at the same middle point.

**Measure of Dispersion:**

Dispersion is nothing but spread it means that how well your data is spread from the mean value.

**Variance** is also one kind of dispersion how well your data is spread from the mean values as variance is **more** then the data in the central region is also **more**.

The variance is nothing but difference between every data point and the mean, squaring that value, and summing for all available data points. In the end, the variance is calculated by dividing the sum by the total number of available points.

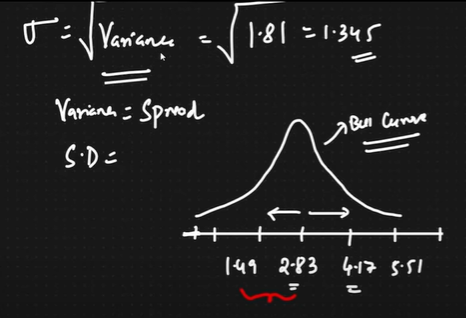


**Why sample variance is n-1 ?**

**Standard Deviation:**

Standard deviation is also one kind of dispersion how well your data is spread from the mean values and is nothing but the **square root of the variance.** Using this SD, we can tell the range of values between one SD to the right and left.

Suppose if we want to represent 5 in with respect to SD then we can say 5 is 1.5 SD from the mean as per below figure.



**Measure of Asymmetry:**

**Modality:**

The modality of a distribution is determined by the number of peaks the data presents. Most distributions are **unimodal** which means it has only one frequently occurring score, clustered at the top while a **bimodal** has two values occurring frequently**.**

Chart

Description automatically generated

**Skewness and Kurtosis:**

These are coefficients that measures how different a distribution is from Normal distribution.

**Skewness:**

Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the centre point.

**Positive Skewed or Right Skewed:**

Data Pushed towards the right side and **mean>median>mode**

**Example:**

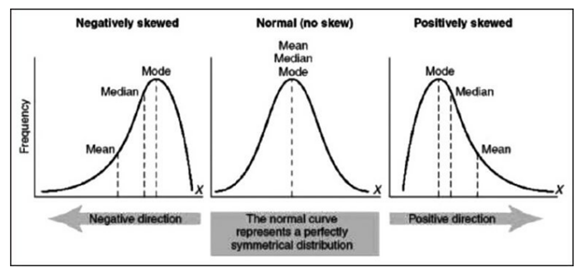
marks obtained by students in a difficult exam, because of the difficulty of the exam there will be fewer and fewer students obtaining higher and higher marks.

**Negative Skewed or Left Skewed:**

Data Pushed towards the left side and **mean<median<mode**

**Example:**

Every student manages to obtain a CGPA in between 7 and 10. There will be very few students getting lower and lower CGPA.

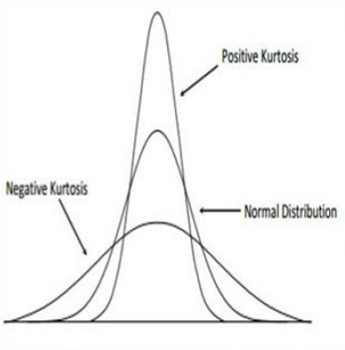


**Kurtosis:**

Kurtosis is a measure of whether the data is **heavy-tailed or light-tailed** relative to a normal distribution. That is, data sets with high kurtosis tend to have heavy tails, or outliers. Data sets with low kurtosis tend to have light tails, or lack of outliers.

Kurtosis refers to **the degree of presence of outliers** in the distribution.

To Overcome this issue, we usually do transformations of data like Log, Box-cox, Exponential, Square root, Reciprocal Transformations.



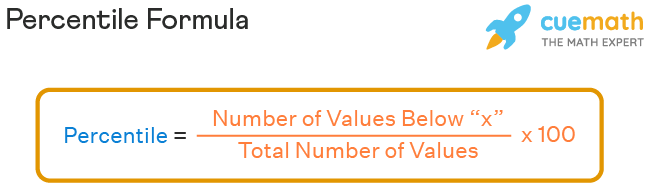
**Percentiles and Quantiles to find the outliers in data:**

**Percentile** is a value below which a certain percentage of distribution lies.

For example, a = 2,2,3,4,5,5,5,6,7,8,8,8,8,8,9,9,10,11,11,12 what is the percentile value of 10?

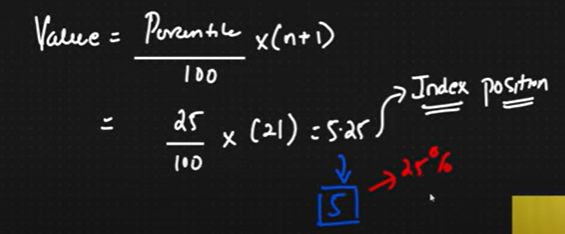
Ans – **80 percentile**

It means that 80% of distribution is present below value 10.



What is the value at **25 percentile?**

Answer – **5**



Usually outliers will be detected by using five number summary points using **IQR(Inter Quantile Range)** and also using Z-Score whichi is far away from 3 SD we can consider it as a Outliers

**Five Number Summary points:**

1. Minimum
2. Quantile(Q1) – 25 percentile
3. Median
4. Quantile(Q3) – 75 percentile
5. Maximum

Outliers can be divided into two ranges namely lower fence and upper fence

**Lower Fence** = **Q1 – 1.5(IQR)**

**Higher Fence** = **Q3 + 1.5(IQR)**

Where **IQR** is **Q3-Q1**

**For example , a=1,2,2,2,3,3,4,5,5,5,6,6,6,6,7,8,8,9,27 what is the outlier here ?**

**Lower Fence =** Q1 – 1.5(IQR)

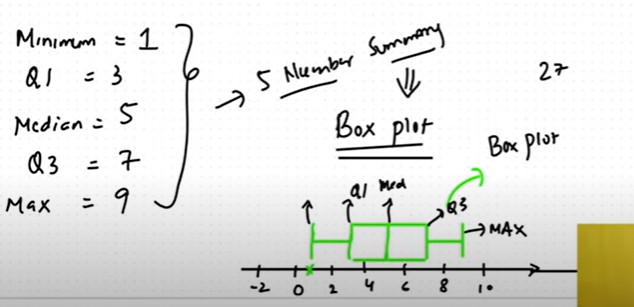
3-1.5(4) = -3

**Lower Fence =** Q3 +1.5(IQR)

7+1.5(4) =13

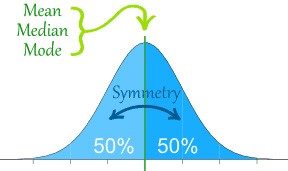
**So outlier is 27.**

**Creating BOX Plot using Five Number Summary:**



## Normal/Gaussian Distribution:

It is also called as Gaussian Distribution, is a probability distribution which is symmetric about the mean. It shows the data which is near to the mean are more frequent in occurrence than the data which is far from the mean.



GD follows bell shape curve, where **mean, median and mode are almost equal**.

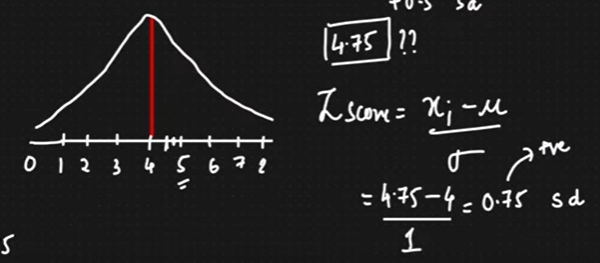
Suppose if we have 100 datapoints then we can draw a below conclusions:

1. 68 % of the data falls in 1st SD from the mean
2. 95 % of the data falls in 2nd SD from the mean
3. 99.7 % of the data falls in 3rd SD from the mean

**Examples**: **Height, Weight, Iris** data follows the gaussian distribution

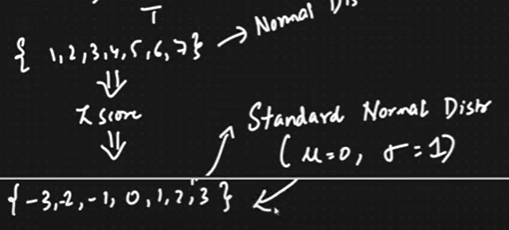
**Question:** If A = {1,2,3,4,5,6,7}, mean = 4 and SD=1 then we can draw a normal distribution and what is the 4.75 value in terms of SD?

**Answer**: We use Z-**Score** for how far the value is away from the mean in terms of **SD, so it is 0.75 SD.**



After applying Z-Score for all elements in A then output will be {-3, -2, -1, 0, 1, 2, 3} so here the main conclusion is **mean =0 and SD =1** hence this is called **as Standard Normal Distribution**

So, if random variable belongs to Standard Normal Distribution, then its **mean** should be 0 and **SD** should be 1.



**So what is the need of Stanndard Normal Distrubution ?**

Suppose if we have features in out dataset like Age (Years), Height(Feets), Salary(Rupees) then for making standardization we are Converting all features into a Standard Normal Distribution. This process is called **Standardization,** whenever we are applying Standardization internally Z-Score is applied.

**Then what is Normalization?**

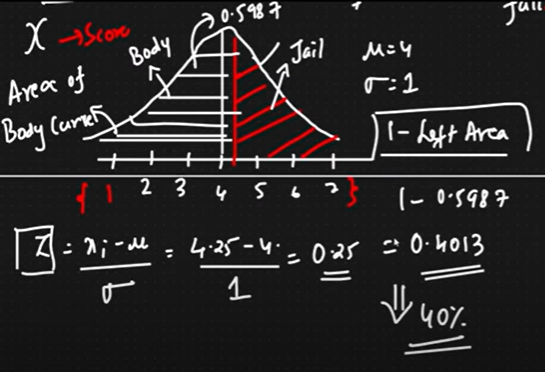
Normalization is also like standardization but only difference is here we are defining the upper and lower boundary. Suppose I want to convert Age column values into 0-1 or -1 to 1. We have different Normalization techniques like Min-Max Scaler and so on .

Foe Example, Normalization is mainly used in CNN Image classification where we are converting 0-255 pixel values into 0-1.

**Practical Question:** If A belongs to Normal distribution, {1,2,3,4,5,6,7} then what is the percentage of values falls above 4.25?

**Answer:** We can use percentiles as well as Z-score to get the answer. Using percentiles, 3/7\*100=40.13

Using Z-score:



Here Z-Score is 0.25 and getting the left area values using left Z-table for 0.25 Z-Score

**Probability:**

Probability is a measure of likelihood of an event.

For example,

* P (rolling a die) = 1/6
* P (Tossing a coin) =1/2

**Addition Rule:** It is nothing but **“Probability OR”**

**Mutually Exclusive Events:** Two events are mutually exclusive if they cannot occur at the same time.

Addition rule for Mutually Exclusive events is

* P (A Or B) = P(A) + P(B)
* P (A ∪ B) = P(A) + P(B)

For example,

* When you toss a coin, you either get heads or tails, but there is no other way of getting both.
* When you are rolling a die, you either get 1 or 2 or 3 or 4 or 5 or 6, but there is no way of getting all 6 at a time.
* In a deck of 52 cards, drawing a **red** card and drawing a **club** are mutually exclusive events because all the clubs are black.

**Non-Mutually Exclusive Events:** Multiple events can occur at the same time

* For example, in a deck of 52 cards, drawing a **red** card and drawing a **heart** card are non-mutually exclusive events because all the heart are red.

Addition rule for Non-Mutually Exclusive events is

* P (A ∪ B) = P(A) + P(B) – P (A and B)

**Questions:** If I toss a coin, what is the probability of the coin landing on head or tails?

**Answer:** This is **Mutually Exclusive event**, soP (H or T) = P(H) +P(T) = ½ + ½ =1

**Question:** When am rolling a die, what is the probability of getting 1 or 3 or 5?

**Answer:** This is Mutually Exclusive event, so P (1 or 3 or 5) = P(1)+P(3)+P(5) = 1/6+1/6+1/6=3/6=1/2

**Question:** A card is drawn at random from a well-shuffled deck of 52 cards. Find the probability that the card drawn is a king or heart?

**Answer:** This is non mutually exclusive event because picking of heart and king may happen both at a time since heart contains king card.

So, P(K or H) = P(K) + P(H) – P(K and H) = 4/52 + 13/52 -1/52=16/52

**Multiplication Rule:** It is nothing **“Probability AND”**

**Independent Events:** Two events are said to be **independent** events if the probability of one event does not affect the probability of another event.

For example, in rolling a die probability of 1 is independent to probability of any other event like 2 or 3 or 4 or 5 or 6.

**Dependent Events**: Two events are said to be **dependent** if the occurrence of one event changes the probability of another event.

For example, in a bag we have 3 red coins and 2 green coins what is the probability of red and green?

First, we need to draw a red coin from the bag which is 1/5 then we have only 4 coins in the bag then probability of green is 2/4 so green is dependent on red so these two events are dependent to each other

**Question:** What is the probability of rolling a 5 and then 4 in dice?

**Answer:** This is independent event 4 is not dependent on 5. so, P(5 and 4) = P(5)\*P(4) = 1/6\*1/6 = 1/36.

**Question:** What is the probability of drawing a Queen and then ace from a deck of cards?

**Answer:** This is dependent event after drawing a Queen we have only 51 cards, so ace is dependent on queen. So, P(Q and A) = P(Q)\*P(A/Q) = 4/52\*4/51

Here **P(A/Q)** is called as **Conditional Probability.**

**Permutations and Combinations:**

A **permutation** is the choice of r things from a set of n things without replacement and where the order matters.

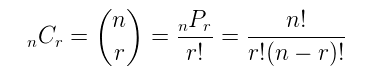
**nPr = (n!) / (n-r)!**

**For example,** we have 6 different chocolates like 5star, eclairs, milky, gems, vennela, Skechers, so in how many ways we select any 3 chocolates?

In first place = 6, second place = 5 third place =4 so ways = 6\*5\*4 =120 **ways**

From 120 ways order may be different but the values might be same so to get the unique combination of values we use **Combinations.**

A **combination** is the choice of r things from a set of n things without replacement and where order does not matter.



6cr = 6! /3! (6-3)! = 20 combinations. So, from 120 ways we have only 20 unique **combinations**

**Covariance and Correlation:**

Suppose if we want to calculate the relationship between any two variables then we should go for covariance. For Example:

* If X increases and Y increases or if X decreases and Y decreases, then it is Positive Correlation
* If X increases and Y decreases or if X decreases and Y increases, then it is Negative Correlation

Covariance of X and Y is given by:

Schematic

Description automatically generated

Where x bar and y bar are the means of X and Y variables and n is the sample size.

But the basic **disadvantage** is magnitude of **covariance** varies from – infinity to +infinity so Its only gives the **direction** of the magnitude but doesnot gives the **strength** of the magnitude like how much it is positively correlated or how much it is negatively correlated.

To get the **Strength** of the relationship we should go for **Pearson Correlation coefficient**.

So **Pearson Correlation coefficient** values varies from -1 to +1, more towards the +1 more positively correlated and more towards the -1 more negatively correlated and person correlation coefficient is given by

Text, letter

Description automatically generated

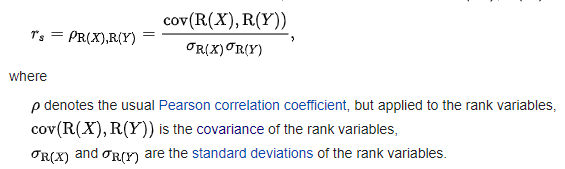
Pearson correlation coefficient is good at satisfying the linear properties of data as shown in the below figure.

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

But this is not good for non linear properties of data. In the above figure it is clearly says that for every X increases Y also increases but pearson correlation is 0.88 actually it should be 1. So to overcome this we go for **Spearman Correlation** which is for above data and Spearman Correlation is given by



For example,

|  |  |  |  |
| --- | --- | --- | --- |
| Weight | Height | Rank(weight) | Rank(Height) |
| 23 | 5 | 1 | 2 |
| 21 | 4 | 3 | 3 |
| 22 | 6 | 2 | 1 |

**Probability Mass Function** – PMF is used for calculating the probability of **discrete** values in each distribution.

Example - If given data is 1,1,2,3.

pmf(1) = 0.5, pmf(2) = 0.25, pmf(3) = 0.25

**Probability Density Function** - defines a probability distribution for a **continuous** random variable as opposed to a discrete random variable (as in PMF). When the PDF is graphically portrayed, the area under the curve will indicate the interval in which the variable will fall. The PDF for a discrete value is always 0!!!! (surprisingly). It can be taken as integration with limits.

**Example** - The probability of life expectancy between 60–65 is given by PDF while probability of it to be exactly 65 will be given by PMF. And will be 0 according to PDF.

**Cumulative Mass/Density Function** - It is the cumulative probability calculated over PDF/PMF i.e. summation of all probabilities associated with a comparatively lower value.

**Example -** Data-1,1,2,3

cmf(1) = 0.5, cmf(2) = cmf(1) + pmf(2) = 0.75, cmf(3) = cmf(2) + pmf(3)

Likewise, CDF is for PDF as well!!

**Types of Distributions:**

[Important Distributions in Probability & Statistics | by Saurabh Raj | Analytics Vidhya | Medium](https://medium.com/analytics-vidhya/important-distributions-in-probability-statistics-a868283fa127)

**Curse of Dimensionality:**

Curse of Dimensionality refers to a set of problems that arise when working with high-dimensional data. A dataset with a large number of features with more than 100, is referred to as high dimensional data.

Some of the difficulties were to analyse or visualise the data to identify patterns, and some manifest while training machine learning models. The difficulties related to training machine learning models due to high dimensional data are referred to as the ‘Curse of Dimensionality’.

**Dimensionality Reduction:**

When we are dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the “essence” of the data. This is called dimensionality reduction.

Dimension reduction can be done by using PCA and SVD

**Covariance Matrix:**

The diagonal entries of the covariance matrix are the variances, and the other entries are the covariances. For this reason, the covariance matrix is sometimes called the \_variance-covariance matrix\_.

## PCA (Principal Component Analysis):

It is a Dimensionality Reduction Technique which reduces the features that are correlated to each other.

**Steps:**

1. Compute the covariance matrix of the features from the dataset.
2. Perform eigen-decomposition on the covariance matrix which will give the Eigen vectors (principal components) and its eigen values.
3. The eigenvectors represent the principal components (the directions of maximum variance) of the covariance matrix. The eigenvalues are their corresponding magnitude. The eigenvector that has the largest corresponding eigenvalue represents the direction of maximum variance.
4. Then projecting all the data points on the selected eigen vectors (principal components).

**Limitations:**

1. The concept of eigenvalues is only applicable to square matrices.
2. For square matrices, the presence of complex eigenvalues restricts the amount of real eigenspaces.

SVD is like Principal Component Analysis (PCA), but more general. PCA assumes that input square matrix, SVD doesn’t have this assumption.

**Singular Value Decomposition:**

The idea behind SVD is to take this matrix of order *m* x *n* and will split this matrix into three matrices.

A picture containing text

Description automatically generated

* **A**: *Input data matrix* — m x n matrix (eg. m documents, n terms)
* **U:***Left**Singular Vectors —*m x r matrix (m documents, r concepts)
* **Σ:***Singular Values —*r x r diagonal matrix (strength of each ‘concept’) where r is rank of matrix **A**
* **V:***Right Singular Vectors —*n x r matrix (n terms, r concepts)

**Note:** Singular Value is a diagonal matrix with non-zero elements and is sorted in descending order

## Central Limit Theorem

The theorem states that as the size of the sample increases, the distribution of the mean across multiple samples will approximate a Gaussian distribution.

If we calculate the mean of all samples, it will be an estimate of the mean of the population distribution. But, like any estimate, it will be wrong and will contain some error. If we draw multiple independent samples, and calculate their means, the distribution of those means will form a Gaussian distribution.

**How to calculate average size of sharks in the sea?**

From total number of sharks,if we draw ‘n’ number of samples , then the distribution of all means of samples will be a guassian distribution. Mean of all such samples is the mean of the size of sharks in the sea.

**Cross Validation:**

Cross-validation is one of the most effective techniques to avoid overfitting and to understand the performance of a predictive model well.

Cross-validation means dividing our training data into different portions and testing our model on a subset of these portions.

The test dataset continues to be used for the final evaluation, while the model performances are evaluated on the portions generated by the cross-validation.

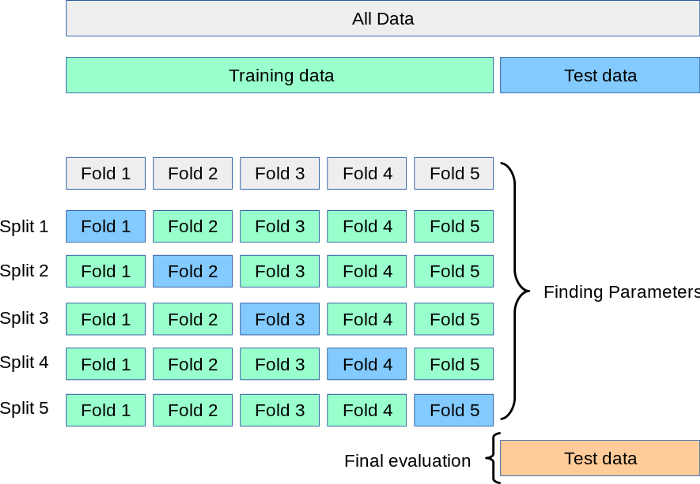
**K-Fold Cross Validation:**

The training set is divided into K-folds (we read “portions”) and the model is trained on k-1 portions. The remaining portion is used to evaluate the model.

For example, if K=5 , then we are dividing our training dataset into 5 portions and the model is going to train on 4 portions and the remainig portion will be considered as a model evaluation and we need to repeat the same process of n splits based on specified number of iterations and at the end we are the calculating the mean of the overall performances.

**Where, number\_of\_splits = 1 + log2(N)**

**where N is the total number of samples.**



**Stratified k-Fold cross-validation:**

It is also similar to K-Fold CV but only difference here is each fold contains approximately the same percentage of samples of each target class, where as in regression mean target value is approximately equal in all folds.

## Chebyshevs Inequality:

If any Random Variable doesnot belongs to Gaussian/Normal Distribution, the maximum number of values within k standard deviations of the mean will be 1–1/k2.

**P-Value or Probabaility Value:**

P-Value or Probabaility Value, nothing but how likely our data would have occurred by random choice. It is the probability of the null hypothesis to be true.

**Example: Touching the space bar key Experiment**

1. If P=0.80 means , if we repeat this experiment ( touching the space bar key) 100 times, number of times touching that particular area would be 80 times ( which is at the center of space bar).
2. If P=0.05 means , if we repeat this experiment ( touching the space bar key) 100 times, number of times touching that particular area would be 5 times ( which is at the corner of space bar).
3. If any random variable follows the gaussian distribution, very less amount of data falls in the two tails region (because of outliers).

**Significance/Alpha Value & Confidence Interval & Hypothesis Testing:**

**Hypothesis testing** is nothing but performing statistical experiments which uses data from a sample to draw conclusions about a population distribution.

So when we are performing Hypothesis Testing we we need to define a Null Hypothesis, Alternate Hypothesis, Defining the Significance Value, State Decision Rule to get the decision boundary, Performing Test Statistics and Making the decisions.

**Experiment:** Coin test, probability of head is 30 times whether coin is fair or not ?

**Null Hypothesis** - coin is fair

**Alternative Hypothesis** - coin is not fair

Then Domain expert will define the **significance value** which is **alpha value** lets say for now alpha is **0.05** then **confidence interval** **region** is **95%** so if we draw a bell shape curve then 30 will falls in confidence interval region so it is true accepting the null hypothesis

**P-Value** is not same as the **Significance value(alpha).** Usually in statsmodel Ztest also gives the P-Value including the Z-Score if that P-Value is lessthan Significance value then we are rejecting the Null Hypothesis and accepting the alternate hypothesis.

**How to calculate P-Value?**

We are calculating P-Value using Z-Score, after calculating the Z-Score, have to look at the Z-Table to get the area of tail region.

For example, assume alpha = 0.05, Z-Score for 2.307, Z-Table value is 0.99111 then this is confidence interval region and tail region is 1-0.9911 =0.00889 which is less than alpha value so that we need to reject the null hypothesis and accept the alternate hypothesis

**Confidence interval** **region**= 1- alpha

If **alpha - value > 0.05 ( 0.05 gave ny domain expert)** then confidence interval is 95 % So any value falls in **confidence interval region** then we can accept the null hypothesis else rejecting the null hypothesis and accepting the alternative hypothesis.

**Type 1 and Type 2 Error:**

When we are performing Hypothesis Testing, first we need to define a Null Hypothesis, Alternate Hypothesis, Defining the Significance Value, State Decision Rule to get the decision boundary, Performing Test Statistics and Making the decisions.

So In any experiment we have always four possible outcomes :

1. We Reject the Null Hypothesis when in reality it is fake. This outcome is **True Negative.**
2. We Reject the Null Hypothesis when in reality it is True. This outcome is **Type-1 Error** also **False Poitive.**
3. We accept the Null Hypothesis when in reality it is fake. This outcome is **Type-2 Error** also called as **False negative.**
4. We accept the Null Hypothesis when in reality it is true. This outcome **is True Positive**

**Example:**

Ho – He is innocent

H1—He is not innocent

As we know that everyone that he is innocent , but if we fail to prove that he is innocent then it is called Type 1 Error and vice-versa.

**One tail and two tail test:**

## If the alternate hypothesis gives the alternate in both directions(less than and greater than) of the value of the parameter specified in the null hypothesis is called a two tailed test.

## If the alternate hypothesis gives the alternate in only one direction(either less than or greater than) of the value of the parameter specified in the null hypothesis is called a one tailed test.

**Examples:**

**If mean of H0=100 and H1= not equal to 100.**

According H1, mean can be greater than 100 or less than 100. This is an example of **Two tailed test.**

## Similarly, if mean of H0>=100 and H1 <100.

## Here the mean is less than 100 which is only one direction. This is called a One tailed test

## Question: Colleges in Karnataka have an 85% placement rate. A new college was recently openedand it was found thata sample of 150 students had a placement of 88% with a standard deviation of 10.

## Does this college has a different placemcent rate?

## Answer: Different placement rate means either greater than 85% or less than 85% this is called 2-Tailed test. Let assume significance value=0.05

## Diagram Description automatically generated Diagram Description automatically generated

## Does this college has a placemcent rate greater than 85%?

## Answer: Greater than 85% means only one direction which is greater than 85% so it should not be 85 or lessthan 85% hence it is called as One tailed Test

## Confidence Interval:

## Confidence interval is nothing but the lower boundary and higher boundary in the confidence interval region.

## Suppose if significance value is 0.05 then confidence interval region is 95% but we don’t know the interval points. To know that we have CI = point estimate+-Margin of errors.

## Point Estimate is the value of any statistic which estimates the value of parameters. Suppose in inferential statistics by taking the sample data we are making estimations on population data which is point estimator. Suppose mean of sample is 2.9 and mean of population is 3 , it may be same or almost same .

## Question: In the quant test of CAT Exam, the standard deviation is known to be 100. A sample of 25 test takers has a mean of 520 score.Construct a 95% CI about the mean which is getting the Lower and upper boundary?

## Answer: Given that SD= 100, sample mean = 520, CI=95% So Alpha = 0.05

## 

## Upper boundary = 520 + 1.96(20) = 559.2

## Lower Boundary = 520 - 1.96(20) = 480.8.

## Note: Here is Z is nothing Z-test nothing Z-Score using Z-table we are getting the 1.96 Value. If Population Standard Deviation is not given then We have to use T-Test Instead of Z-table have to use T-Table. To get the T-Table we have to find the Degree of Freedom, which is n-1 where n is sample.

## So In the above problem, Degree of freedom is 25-1 = 24.

## Question: In the quant test of CAT Exam. A sample of 25 test takers has a mean of 520 score with standard deviation of 80.Construct a 95% CI about the mean which is getting the Lower and upper boundary?

## Answer: So here Population SD is not given so we have to use T-Test Confidence Interval formula,

## 

## Upper boundary = 520 + 2.064(80/5) = 553.024

## Lower Boundary = 520 – 2.064(80/5) = 486.97

## Where 2.064 is from T-table with respect to degree of freedom of 24

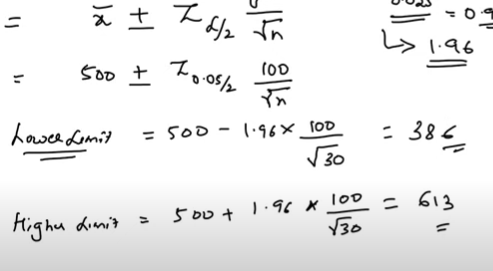
## 

**Question: How to find the Average size of sharks in the sea?**

Confidence intervals = means of sample (+-) marginal errors

**Assumptions :**

1. SD of Population = 100
2. Each Sample Size = 30
3. Mean of each sample = 500
4. Significance value =0.05



**One Sample Z-Test:**

Whenever we are performing an Hypothesis Testing,first we need to define a Null Hypothesis, Alternate Hypothesis, Defining the Significance Value, State Decision Rule to get the decision boundary, Performing Test Statistics and Making the decisions.

In **State Decision Rule**, we have to find the Confidence interval points with the help of significance value and Z-table then we are performing the test. So if the Population Standard Deviation is given and Sample Size is greater than or equal to 30 then we can use **Z-Test** which is nothing but **Z-Score**

**Question:** In a Population, the average IQ is 100 with SD of 15. Reasearchers wants to test a medication for positive or negative intelligence or not effect at all. A sample of 30 participants who have taken the medication has mean IQ of 140. Did the medication affect the intelligence?

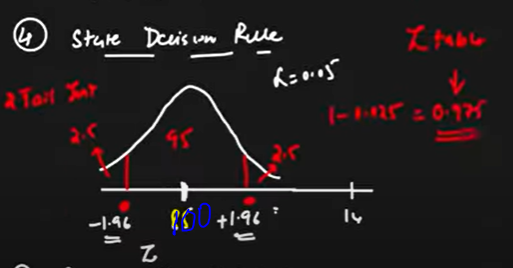
**Answer:**

**Step1 : Defining the Null Hypothesis =** IQ mean after using medication is 100. ( means it is **2 tailed test** either it would be greater than 100 or less than 100)

**Step2 : Defining the Alternate Hypothesis =** mean after using medication is not equal to 100.

**Step3 : Defining the Significance Value =** 0.05 which means CI region is 95%.

**Step4 : Performing the State Decision to get the decision boundary**using the Significance value and Z-table.



Here SDR contains -1.96 SD and +1.96 SD range if anything comes in this region accepting the null hypothesis else rejecting the Null Hypothesis

**Step5 : Performing Z-test =** (mean of sample – mean of population)/(SD/sqrt(sample size))

= (140 – 100)/(15/sqrt(30))

= 14.60 SD which is greater than the 1.96 so **rejecting the Null Hypothesis**

**One Sample T-Test:**

Also the Same like One Sample Z-Test but when the Population SD is not given **we use T-Test**

**Question:** In a Population, the average IQ is 100. Reasearchers wants to test a medication for positive or negative intelligence or not effect at all. A sample of 30 participants who have taken the medication has mean IQ of 140 with the SD of 20. Did the medication affect the intelligence?

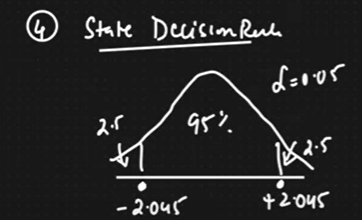
**Answer:**

**Step1 : Defining the Null Hypothesis =** IQ mean after using medication is 100. ( means it is **2 tailed test** either it would be greater than 100 or less than 100)

**Step2 : Defining the Alternate Hypothesis =** mean after using medication is not equal to 100.

**Step3 : Defining the Significance Value =** 0.05 which means CI region is 95%.

**Step4 : Performing the State Decision Rule to get decision boundary-** Hence Population SD is not given have to perform T-Test. To get the boundaries first have to find degree of freedom which 30-1 = 29 . and then using the t-table with respect to degree of freedom 29 , lower boundary is -2.045 and upper boundary is +2.045 which is shown in the below:



**Step5 :** Performing T-Test so value is 10.96 which is greater than 2.045 hence rejecting the null hypothesis. And confirming that medication is positive

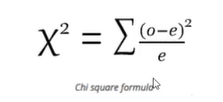


**Real World Problems:** Bank wants to open New ATM Machine in particlar or not ?

**Chi-Square Test:**

Chi-Square text claims about population proportions. It is a non parametric test which is performed on **categorical** varibales like **nominal or ordinal variables.**

And whenever do Chi-Square test, we need to calculate degree of freedom, **n-1** here n is **number of categories.**



**Where O is Observed value, e is Expected value**

**Question**: In the indian 2000 census, the age of the individual in a small town were found to be the following :

|  |  |  |
| --- | --- | --- |
| **Less than 18** | **18-35** | **Greater than 35** |
| 20% | 30% | 50% |

And in 2010 census, age of 500 individuals were sampled and below are the results:

|  |  |  |
| --- | --- | --- |
| **Less than 18** | **18-35** | **Greater than 35** |
| 121 | 288 | 91 |

Using alpha = 0.05, Would you concludethat population distribution of ages has changed in last 10 years?

**Answer:** Hence Age is categorical Variable we are performing Chi-Square Test in Hypothesi Testing. And when we are performing Chi-Square Test first we need to get the **observation(O)** given in the data and find the **expected(e)** data.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Less than 18** | **18-35** | **Greater than 35** |
| **Observation** | 121 | 288 | 91 |
| **Expected** | 500\*0.2 = 100 | 500\*0.3 = 150 | 500\*0.5 = 250 |

Now we can start the hypothesis testing:

**Step1 : Defining the Null Hypothesis –** Given observation meets the original distribution in 2000.

**Step2 : Defining the Alternate Hypothesis –** Given observation does not meets the original distribution in 2000.

**Step3 : Defining the Significance Value –** Alpha = 0.05 then CIR = 95%

**Step4 :** **Defining the Decision boundary -** Using Degree of freedom and significance value we can get the decision boundary in Chi-Square table. So here ( +5.99 or -5.99)

**Step5 : Then Performing the Test –**

**Chi-Squre = Sum of (observed – expected)2/expected**

**Chi-Squre** = (121-100)2/100 + (288-150)2/150 + (91-250)2/250

**Chi-Squre** = 232.94

So Chi-Squre value is greater than 5.99 so we are rejecting the Null Hypothesis Testing. So output is Given observation doesnot meets the orginial distribution of data.

**F-Test or ANOVA Test:**

If you want to perform Hypothesis test for **both categorical and numerical** variables then test is **ANOVA** test.

**Example:**

Suppose If you want to calculate average height of people in the india, it will be very difficult to draw a conclusion from population of data , so we are taking the sample of data and making conclusions using hypothesis testing.

**Steps to perform Hypothesis Testing:**

1. Define a Null Hypothesis and Alternate Hypothesis
2. Define Alpha value ( usually it would be 0.05 or else depends on Business domain)
3. Define decision rule – ( 1-0.05) =0.9750 . we need to check the z-table to get the upper limit and lower limit for the confidence range.
4. Performing Z-test/ T-test/Annova test/ and anaysing the results.

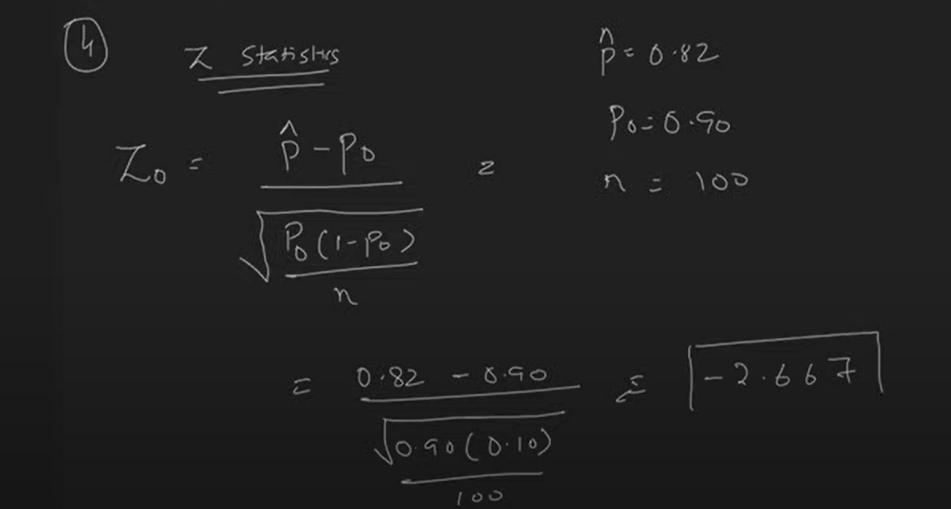
**Example:**

**A survey claims that a 9 out of 10 doctors recommend aspiron to their patients for headache. To test this claim, a random sample of 100 doctors is taken, out of these 100 doctors, 82 indicate that they recommend aspirons. Is that claim accurate? ( given alpha =0.05)**

**Ans:**

**Steps :**

1. Define Null Hypothesis and Alternate Hypothesis , h0 =0.90 and h1!=0.90
2. Signficance level = 0.05
3. Decision rule ( 1- 0.05=0.9750, from z table we will get the intervals = -1.96 to +1.96)
4. Perfomring z-statistics:



**Note :** -2.667 does not falls in confidence interval range so we should reject the null hypothesis and accept the alternate hypothesis**.** It mean **Claim is not Accurate.**

**Difference between fit and transform:**

for train data: fit creates formula for all the features in dataset ,transform will transform data with created formula.for test data: formula already created just transform it accordingly.

## Convolutional Neural Network:

Convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

We have three basic components to define a basic convolutional network.

1. The convolutional layer
2. The Pooling layer[optional]
3. The output layer

**The Convolution Layer:**

We define a weight matrix which is called as a filters or kernels which extracts certain features from the images using convolution operation. **Convolution** is nothing but performing the element wise multiplication of the weight matrix and the part of the given image . This process will repeat for all the pixels are covered atleast once.

Suppose,if the given image has size of 6\*6 matrix and kernal size of 3\*3 matrix, after performing the convolution operation, will get the convolved output of size 4\*4 matrix, means here we are loosing some of the information during convoultion.

To Overcome this problem, we use Padding for the given images. Padding the input image with zeros around the image which is called as **Same Padding.** Valid Padding is nothing but, we are not using Padding.

**Formula to get the size of Convolved output**:

N+2P-F+1

Where N is size of input image,P is Padding, F is Kernal Size.

**The Pooling Layer :**

Sometimes when the images are too large, we would need to reduce the number of trainable parameters so that we are periodically introducing pooling layers between subsequent convolution layers. Pooling is done for the sole purpose of reducing the spatial size of the image**.** The most common form of pooling layer generally applied is the max pooling.

So the combination of Convolution,Padding and Pooling is called a one convolution layer. We may have multiple convolutional layers present in our network.

**The Output layer:**

After multiple layers of convolution and padding, we would need to apply a fully connected layer to generate an output equal to the number of classes we need. The output layer has a loss function like categorical cross-entropy, to compute the error in prediction. Once the forward pass is complete the backpropagation begins to update the weight and biases for error and loss reduction.

**Note:**

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

## Why Not ANN for Image Classification:

While Solving am Image Classification Problem using ANN, the first step is to convert a 2-dimensional image into a 1-dimensional vector to train the model. This has 2 drawbacks

1. The number of trainable parameters increases drastically with an increase in the size of image. For example, if the size of the image is 224\*224, then the number of trainable parameters at the first hidden layers with just 4 neurons is 6 lakhs. That’s Huge!
2. ANN loses the spatial features of an image, Spatial features refers to the arrangement of the pixels in an image.

## Object Detection Algorithms:

[**https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/**](https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/)

## 1. A Simple Way of Solving an Object Detection Task (using Deep Learning – CNN):

1. First, we take an image as input
2. Then we divide the image into various regions
3. We will then consider each region as a separate image.
4. Pass all these regions (images) to the CNN and classify them into various classes.
5. Once we have divided each region into its corresponding class, we can combine all these regions to get the original image with the detected objects

**Limitations:**

Here we would require a very large number of regions resulting in a huge amount of computational time. To solve this problem and reduce the number of regions, we can use region-based CNN.

## Region-Based Convolutional Neural Network (RCNN):

RCNN algorithm proposes a bunch of boxes in the image and checks if any of these boxes contain any object. RCNN uses selective search to extract these boxes from an image (these boxes are called regions). Selective search identifies the various regions based on the colours, textures, scales, enclosures.

**How Selective Search Works:**

1. It first takes image as an input
2. Then, it generates initial sub-segmentations so that we have multiple regions from this image.
3. Then combining the similar regions to form a larger region (based on colour similarity, texture similarity, size similarity, and shape compatibility)
4. Finally, these regions then produce the final object locations (Region of Interest).

**Steps involved in Object Detection:**

1. First will take a pre-trained convolutional neural network.
2. We train the last layer of the network based on the number of classes that need to be detected.
3. Then, we get the Regions of Interest (ROI) for each image using Selective search method.
4. All these regions are then reshaped as per the input of the CNN, and each region is passed to the Convolution Neural Network.
5. CNN then extracts features for each region and SVMs are used to divide these regions into different classes.
6. Finally, a bounding box regression is used to predict the bounding boxes for each identified region.

**Limitations:**

Training an RCNN model is expensive and slow because of extracting regions from each image and each region is passed into the convolution neural network makes very slow.

## Fast RCNN:

Instead of running CNN for multiple regions in an image in RCNN, we are passing one image into CNN.

**Steps involved in Object Detection:**

1. First will take image as an input.
2. This image is passed to a Convolution Neural Network which returns the region of interests using Selective search method.
3. Then we apply the ROI pooling layer on the extracted regions of interest to make sure all the regions are of the same size.
4. Finally, these regions are passed on to a fully connected network which classifies them, as well as returns the bounding boxes using soft-max and linear regression layers simultaneously.

**Limitations:**

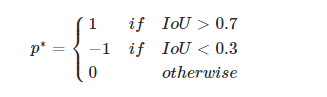
It also uses selective search as a proposal method to find the Regions of Interest, which is a slow and time-consuming process.

## Faster RCNN:

Faster RCNN is the modified version of Fast RCNN. The major difference between them is that Fast RCNN uses selective search for generating Regions of Interest, while Faster RCNN uses “**Region Proposal Network**”, aka RPN. **RPN** takes convolved feature maps as an input and generates a set of object proposals, each with an object score as output.

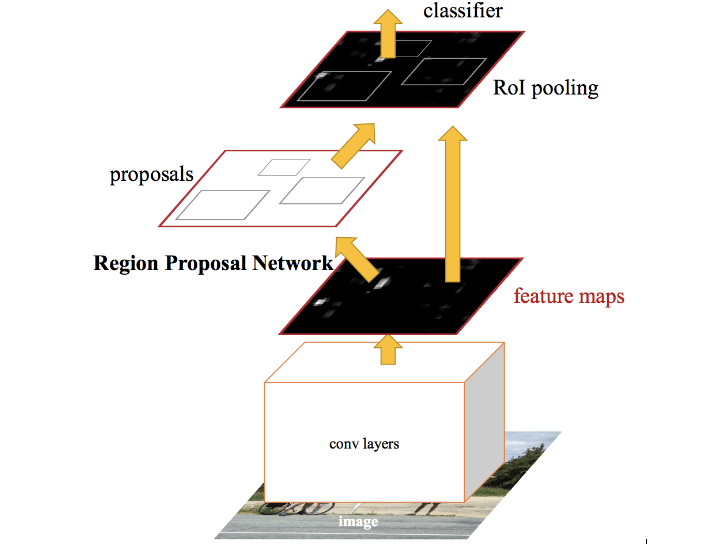
**How RPN Works:**

1. We take an image as input and pass it to the Convolution Neural Network which returns the convolved feature map for that image.
2. Region proposal network is applied on these convolved feature maps, then this RPN uses n\*n sliding window of size ‘n’ here n=3 run spatially on these feature maps, for each sliding window, a set of 9 anchor boxes are generated with different sizes and shapes.
3. Then for each anchor box, RPN Predicts two things
4. P\* (probability that an anchor is an object) is computed which indicates that how much these anchor boxes are overlap with the ground-truth bounding boxes.
5. Second is the bounding box regressor for adjusting the anchor boxes to better fit the objects which are ROI proposals.

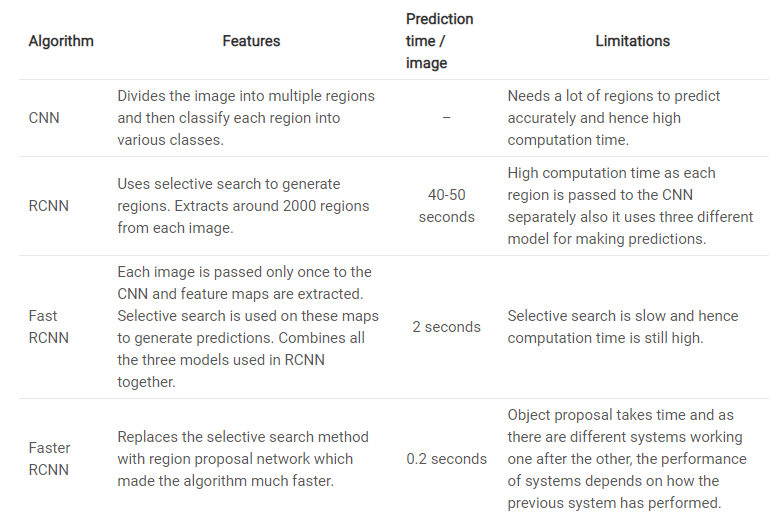


**Steps involved in Object Detection:**

1. We take an image as input and pass it to the Convolution Neural Network which returns the feature map for that image.
2. Region proposal network is applied on these feature maps. This returns the object proposals along with their object score.
3. A ROI pooling layer is applied on these proposals to bring down all the proposals to the same size.
4. Finally, the proposals are passed to a fully connected layer which has a soft-max layer and a linear regression layer at its top, to classify and output the bounding boxes for objects.



## Difference between Algorithms:



## YOLO (YOU ONLY LOOK ONCE):

## The YOLO framework (You Only Look Once) takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. ****The biggest advantage of using YOLO is its superb speed**** – it’s incredibly fast and can process 45 frames per second.

## Steps involved in Object Detection:

1. YOLO first takes an input image

## The framework then divides the input image into grids (say a 3 X 3 grid)

## Suppose we have 3 objects to detect, so for each grid, the label y will be an eight-dimensional vector.

## 

## Where pc defines whether an object is present in the grid or not (it is the probability)

## bx, by, bh, bw specify the bounding box if there is an object

## c1, c2, c3 represent the classes. So, if the object is a car, c2 will be 1 and c1 & c3 will be 0.

## For each image will get 3\*3\*8 size.

## After creating the labels for each image, feeding into the CNN to train the model which will give the output of 3\*3\*8.

## YOLO may predicted multiple bounding boxes for single object. To remove those unwanted bounding boxes, we are using Non-Max Suppression.

## Non-Max Suppression:

1. First it will discard all the bounding boxes having probabilities less than or equal to a pre-defined threshold (say, 0.5)
2. For the remaining boxes:
   1. Pick the box with the highest probability and take that as the output prediction
   2. Picking the box which has high probability score and calculating the IOU score and the boxes which have high IOU with the current box are suppressed.

**Summary:**

YOLO is faster in comparison to Faster-RCNN. Their accuracies are comparatively similar. YOLO does not work pretty well for **small** objects.  
In order to improve its performance on smaller objects, you can try the following things:

* Increase the number of anchor boxes
* Decrease the threshold for IOU

This may give better results.

**Data Augmentation Techniques:**

Image augmentation is a technique of altering the existing data to create some more data for the model training process.

**Image rotation:**

Even if you rotate the image, the information on the image remains the same.

**Image Shifting:**

By shifting the images, we can change the position of the objects in the image and hence give more variety to the model. Which eventually can result in a more generalized model.

**Image Flipping:**

Flipping can be considered as an extension of rotation. it allows us to flip the image in the Left-Right direction as well as the Up-Down direction.

**Image Noising:**

Image Noising where we add noise to the image. This technique allows our model to learn how to separate the signal from the noise in the image. This also makes our model more robust to changes in the image.

**Image Blurring:**

Images come from different sources and hence the quality of images will not be the same from each source. Some images might be of very high quality and others must be very bad. In such scenarios we can blur the original images, this will make our model more robust to the quality of the image being used in the test data.

## Image Segmentation

In Object detection, we are finding the objects present in the image and their location using bounding boxes whereas in object segmentation, we are finding the objects present in the image and their location exactly. In simple words, instead of drawing a rough rectangular box around the object, we draw a polygon around the object.

**Semantic Segmentation:**

All the pixels belonging to a particular class are represented by the same colour.

**Instance Segmentation:**

All the pixels belonging to a particular class are represented by the different colour.

## Mask RCNN

Mask R-CNN is the current state-of-the-art for image segmentation and runs at 5 fps and it is an extension of Faster RCNN. Usually, FRCNN predicts two things for each object in the image which are its class and bounding box coordinates, whereas Mask RCNN outputs the mask for the objects