**Supervised learning:**

**1. Regression**  
Regression is used to understand the relationship between dependable and independent variables. Moreover, it is a type of supervised learning that learns from labelled data sets to predict continuous output for different data in an algorithm. It is believed to be widely used in scenarios where the output needs to be a finite value, for instance, height or weight, etc.  
There two types of regression; they are as follows:

1. **Linear regression**

It is used to identify the relationship between two variables, typically used for making future predictions. Moreover, linear regression is sub-divided based on the number of independent and dependent variables.

For instance, if there is one independent and one dependent variable, it is known as simple linear regression. Meanwhile, if there are two or more independent and dependent variables, it is called multiple linear regression.

1. **Logistic regression**

Logistic regression is used when the dependent variable is categorical or has binary outputs like ‘yes’ or ‘no’. Moreover, logistic regression is used to solve binary classification problems; that’s why it predicts discreet values for variables.

1. **Naive Bayes**  
   A Naive Bayes algorithm is used for large datasets. The approach works on the fundamental that every programme in the algorithm works independently. This means that the presence of one feature will not impact the other. Generally, it is used in text classification, recommendation systems, and others.

There are different types of Naive Bayes models, and the decision tree remains the most popular among business organizations. A decision tree is a unique supervised learning algorithm structurally resembling a flowchart. However, they fundamentally perform different roles and responsibilities.

A decision tree consists of control statements containing decisions and their consequences. The output in a decision tree relates to the labelling of unforeseen data. ID3 and CART are some of the popular decision tree algorithms widely used across various industries.

**2. Classification**  
It is a type of supervised learning algorithm that accurately assigns data into different categories or classes. It recognizes specific entities and analyses them to conclude where those entities must be categorized. Some of the classification algorithms are as follows:

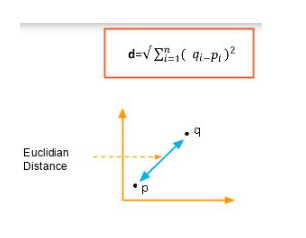
* K-nearest neighbor
* Random forest
* Support vector machines
* Decision tree
* Linear classifiers

**Types of Clustering:**

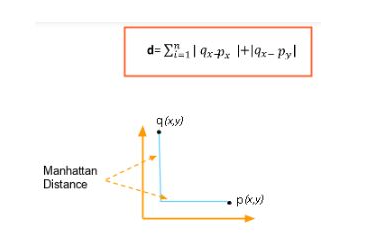
**Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not.

**Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned.

### Euclidean Distance

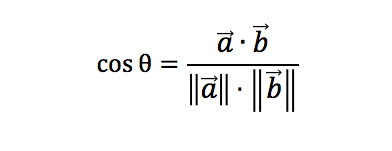


1. Manhattan Distance Measure



1. **Cosine distance:** This distance metric is used mainly to calculate similarity between two vectors. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in the same direction. It is often used to measure document similarity in text analysis.

Formula for similarity measure:



Using this formula we will get a value which tells us about the similarity between the two vectors and **1-cosΘ will give us their cosine distance.**



1. **Hamming Distance**- Hamming distance is a metric for comparing two binary data strings. While comparing two binary strings of equal length, Hamming distance is the number of bit positions in which the two bits are different.

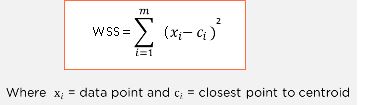
**K-Means Clustering**

K-Means clustering is an unsupervised learning algorithm. There is no labeled data for this clustering, unlike in supervised learning. K-Means clustering supports various kinds of distance measures, such as:

* Euclidean distance measure
* Manhattan distance measure
* A squared euclidean distance measure
* Cosine distance measure

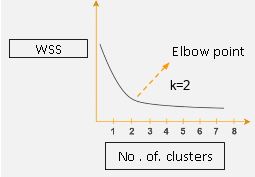
Initially, the Elbow method is used to find the number of clusters.

We use within-sum-of-squares as a measure to find the optimum number of clusters that can be formed for a given data set. Within the sum of squares (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid.



The WSS is measured for each value of K. The value of K, which has the least amount of WSS, is taken as the optimum value.

Now, we draw a curve between WSS and the number of clusters.



Here, WSS is on the y-axis and number of clusters on the x-axis.

K means algorithm works in these steps:

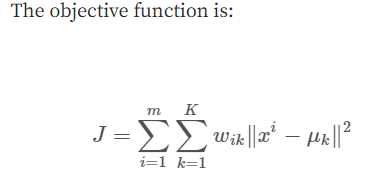
We can randomly initialize points called the cluster centroids.

Now the distance of each datapoint from the centroid is measured, and each data point is assigned to the centroid, which is closest to it.

Now compute the actual centroid of data points for the all groups. Reposition the random centroid to the actual centroid.

Once the cluster becomes static, the k-means algorithm is said to be converged.

The approach kmeans follows to solve the problem is called **Expectation-Maximization**. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster.

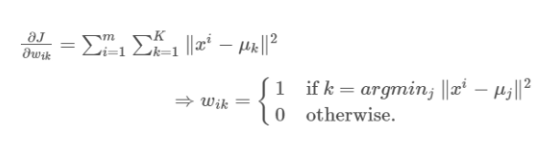


wik=1 for data point xi if it belongs to cluster *k*;

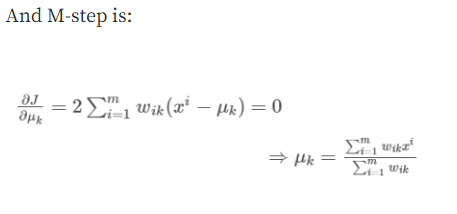
otherwise, wik=0.

μk is the centroid of xi’s cluster.

we differentiate J w.r.t. wik first and update cluster assignments (*E-step*). Then we differentiate J w.r.t. μk and recompute the centroids after the cluster assignments from previous step (*M-step*).



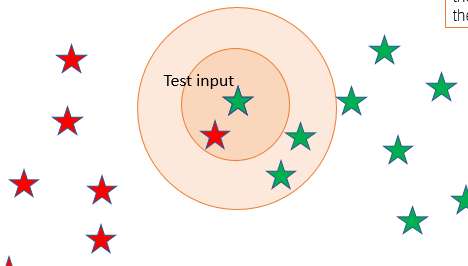
In other words, assign the data point xi to the closest cluster judged by its sum of squared distance from cluster’s centroid.



Which translates to recomputing the centroid of each cluster to reflect the new assignments.

**𝜖-Ball Nearest Neighbors (𝜖-NN)**

* Rather than looking at a fixed number of neighbors, can look inside a ball of a given radius , around the test input



**Hierarchical Clustering**

**Agglomerative:** Bottom up approach

**Divisive:** Topdown approach

Hierarchical clustering is another **unsupervised machine learning algorithm**, which is used to group the unlabeled datasets into a cluster. We develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**. There is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

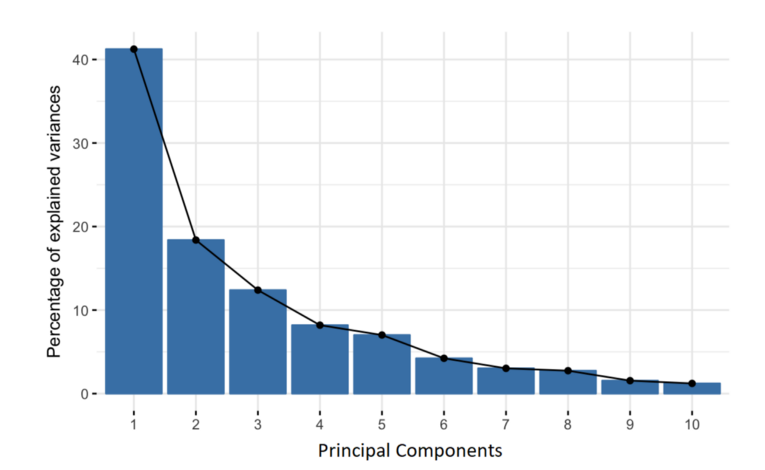
Working:

Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.

Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.

**Principal Component Analysis**

Principal component analysis, or PCA, is a [dimensionality-reduction](https://builtin.com/data-science/dimensionality-reduction-python) method that is often used to reduce the dimensionality of large [data sets](https://builtin.com/data-science), by transforming a large set of variables into a smaller one that still contains most of the information in the large set. Dimensionality reduction is done with a tradeoff with accuracy.

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. 

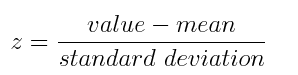
Geometrically speaking, principal components represent the directions of the data that explain a **maximal amount of variance**, that is to say, the lines that capture most information of the data.

The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more information it has.

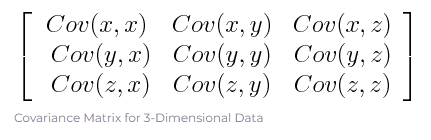
To put all this simply, just think of principal components as new axes that provide the best angle to see and evaluate the data, so that the differences between the observations are better visible.

Steps:

1. Standardisation of values:



1. Covariance matrix computation: Done to check :

* if there is any relationship between variables of input data
* to eliminate variables highly correlated since they contain redundant info.
* 
* See the sign of cov:
* If positive then: the two variables increase or decrease together (correlated)
* If negative then: one increases when the other decreases (Inversely correlated)

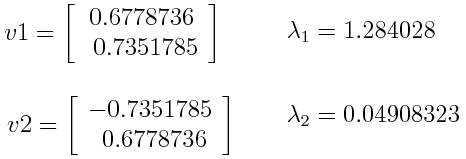
1. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components. Eigenvectors and eigenvalues of the covariance matrix are calculated.

**Eigenvectors** of the Covariance matrix are actually *the**directions of the axes where there is the most variance*(most information).

**Eigenvalues** are simply the coefficients attached to eigenvectors, which give the amount of variance carried in each Principal Component.

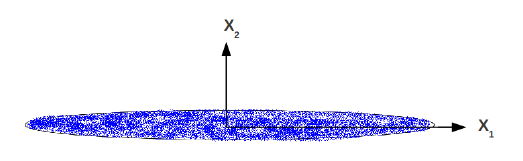
**Example:**

Let’s suppose that our data set is 2-dimensional with 2 variables ***x,y***and that the eigenvectors and eigenvalues of the covariance matrix are as follows:



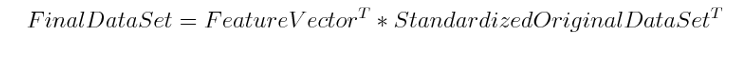
If we rank the eigenvalues in descending order, we get λ1>λ2. So principal component1(PC1) correponds to v1, and PC2 corresponds to v2.

After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96% and 4% of the variance of the data.



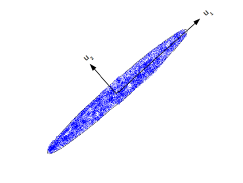
Very little variance across x2.

1. Last step: Recast the data along the principal components axes: use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components.

Applications:

* To compress data by reducing dimensionality. E.g., representing each image in a large collection as a linear combination of a small set of \template" images .
* Visualization (e.g., by projecting high-dim data to 2D or 3D).
* To reduce overfitting problem caused by high-dimensional data.
* To make learning algorithms run faster.

**Change of axes to avoid loss of information:** Eg: figure here has data variance across both x1, x2. Each example x has 2 features: u1, u2.. Change of axes done so that we can drop u2 vector. No variance across u2.



* Roughly speaking, PCA does a **change of axes** that represent the data.
* First PC: Direction of the highest variance.
* Second PC: Direction of next highest variability (orthogonal to the 1st PC)
* Note: All principal components are orthogonal to each other

Deep insights into SVD, LDA and PCA:

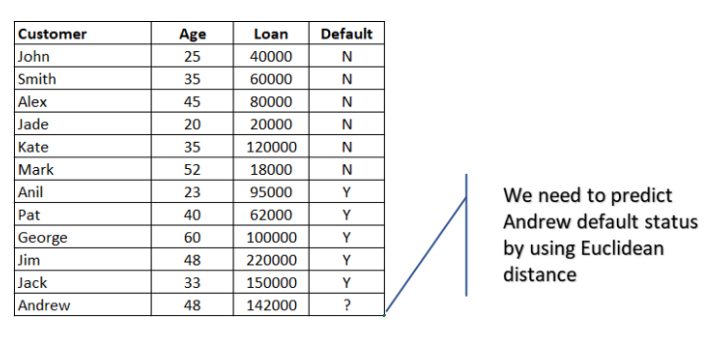
<https://medium.com/nerd-for-tech/dimensionality-reduction-techniques-pca-lca-and-svd-f2a56b097f7c>

**K-Nearest Neighbor(KNN)**

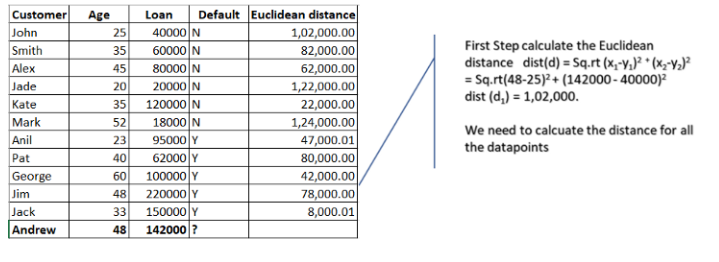
* In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.
* In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors.
* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* ‘k’ in KNN algorithm is based on feature similarity choosing the right value of K is a process called parameter tuning and is important for better accuracy.
* keep the value of k odd in order to avoid confusion between two classes of data.
* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* One way to choose K is though cross-validation. Take the small portion from the training dataset and call it a validation dataset, and then use the same to evaluate different possible values of K. This way we are going to predict the label for every instance in the validation set using with K equals to 1,2,3,.... and then we look at what value of K gives us the best performance on the validation set and we can take that value and use that as the final setting of our algorithm so we are minimizing the validation error .

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

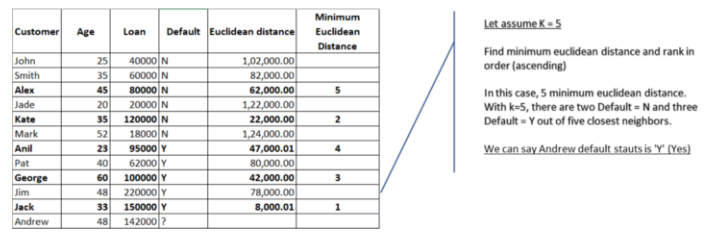
* **Step-1:** Select the number K of the neighbors.



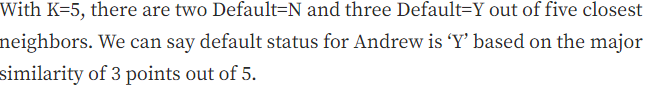
* **Step-2:** Calculate the Euclidean distance of the point with all the **neighbors.**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.



* **Step-4:** Among these k neighbors, count the number of the data points of each different category.



* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.



There are six types of kernels in SVM:

* Linear kernel - used when data is linearly separable.
* Polynomial kernel - When you have discrete data that has no natural notion of smoothness.
* Radial basis kernel - Create a decision boundary able to do a much better job of separating two classes than the linear kernel.
* Sigmoid kernel - used as an activation function for neural networks.

**What Are Common Bias Types?**

A phenomenon that occurs when an algorithm produces results that are systemically prejudiced/biased due to erroneous assumptions in the machine learning process.

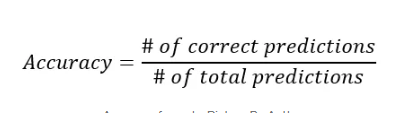
* Anchoring bias,
* Availability bias,
* Confirmation bias,
* Stability bias.

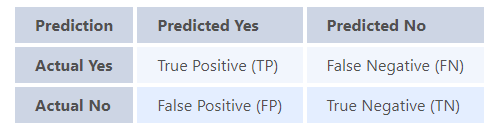
**Difference Between Classification and Regression?**

* Classification is used to classify data into some specific categories. For example, classifying emails into spam and non-spam categories.
* Whereas, regression deals with continuous data and predict the relationship between data   
  For example, predicting stock prices at a certain point in time.

**What is F1 score? How would you use it?**

**Accuracy-** The accuracy metric computes how many times a model made a correct prediction across the entire dataset. This can be a reliable metric only if the dataset is class-balanced; that is, each class of the dataset has the same number of samples.





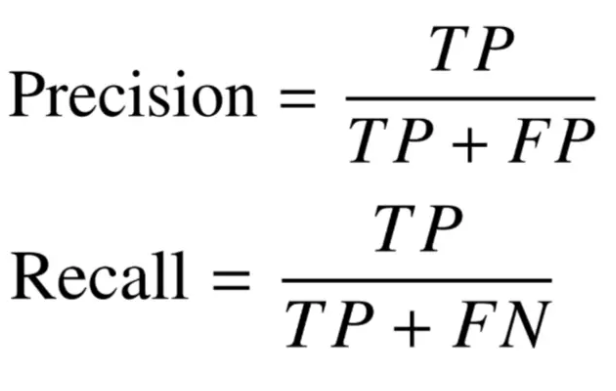
In binary classification we consider the F1 score to be a measure of the model’s accuracy. The F1 score is a weighted average of precision and recall scores.

**Precision**: A precise system is which may not detect all positives but whatever positives are detected are true positives. Precision focuses on the accuracy of positive predictions. A high precision indicates that the model is good at not misclassifying negative instances as positive.

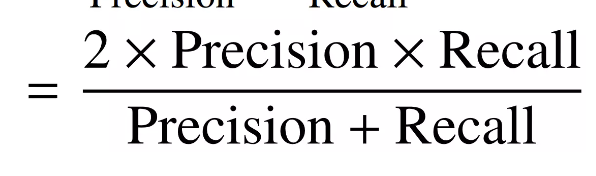
* **Of all the instances predicted as positive, how many were actually positive?"**

**Recall**: Recall, also known as sensitivity or true positive rate (TPR), measures the proportion of correctly predicted positive cases (true positives) out of all actual positive instances, including both true positives and false negatives.

* **Of all the actual positive instances, how many were correctly predicted?"**



The F1 score is calculated as the harmonic mean of the precision and recall scores. A higher F1 score denotes a better quality classifier.



* A model will obtain a **high F1 score** if both Precision and Recall are high and **low F1 score** if any one of them is low. It is the harmonic mean of recall and precision and tells if any algo is having good precision and recall or not.
* A model will obtain a **medium F1 score** if one of Precision and Recall is low and the other is high

Why is the F1 score calculated using the harmonic mean instead of simple arithmetic or geometric means?

* To put it simply: the harmonic mean encourages similar values for precision and recall. That is, the more the precision and recall scores deviate from each other, the worse the harmonic mean.

**Roc and AUC curve**

**Auc:**

* The AUC quantifies the overall performance of a binary classification model by computing the area under the ROC curve.
* AUC ranges from 0 to 1, where a higher value indicates better model performance.
* An AUC of 0.5 suggests random guessing while an AUC of 1 indicates perfect classification.
* AUC is a useful metric because it measures the model's ability to rank true positives higher than false positives across all possible thresholds.

[**Pipeline**](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline) can be used to chain multiple estimators into one. This is useful as there is often a fixed sequence of steps in processing the data, for example feature selection, normalization and classification. Pipeline help as:

* call [fit](https://scikit-learn.org/stable/glossary.html#term-fit) and [predict](https://scikit-learn.org/stable/glossary.html#term-predict) only once on the data to fit a whole sequence of estimators.
* You can [grid search](https://scikit-learn.org/stable/modules/grid_search.html#grid-search) over parameters of all estimators in the pipeline at once. Eg :

Pipeline(steps=[('reduce\_dim', PCA()), ('clf', SVC(C=10))])

param\_grid = dict(reduce\_dim\_\_n\_components=[2, 5, 10], clf\_\_C=[0.1, 10, 100])

grid\_search = GridSearchCV(pipe, param\_grid=param\_grid)

**How to tackle overfitting of data?**

1. [Hold-out](https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d#c287)- Splitting data into train and test data.
2. Cross Validation - diving data into n section and training n-1 sections and checking on that one section. Each time one section which is to be left out changes.



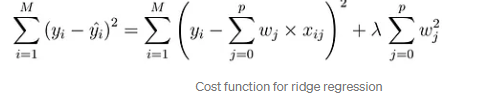
1. [Data augmentation](https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d#f80d)- A larger dataset would reduce overfitting. If we cannot gather more data we can artificially increase the size of our dataset. For example, if we are training for an image classification task, we can perform various image transformations to our image and add to the dataset.
2. Reduce number of [features](https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d#253a)- If we have only a limited amount of training samples, each with a large number of features, we should select the most important features for training.
3. [L1 / L2 regularization](https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d" \l "d178)- Used in regression models. Try increasing lambda so that weights value decrease.

Examples of regularization, includes:

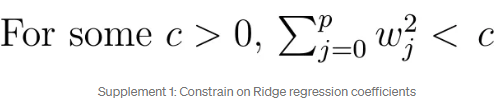
* K-means: Restricting the segments for avoiding redundant groups.
* Neural networks: Confining the complexity (weights) of a model.
* Random Forest: Reducing the depth of tree and branches (new features)

L1 regularization -***Lasso Regression***. Adds “*absolute value of magnitude*” of coefficient as penalty term to the loss function. It shrinks the less important feature’s coefficient to zero thus, removing some feature altogether. this works well for **feature selection** in case we have a huge number of features.

L2 regularization -***Ridge Regression***. Adds “*squared magnitude*” of coefficient as penalty term to the loss function. Loss function for L1 :



i.e. coefficients are restricted. And for large coeff. cost fn. increases. Hence coeff. should be small.



Sum of squared residual (loss function) + lamda\*(square mag of coefficients)

Eg: in linear regression loss fn is : sum of (Yi-XjBj)2 . Now for ridge regresion line loss fn will have added term of lambda. This will minimize the error also. Refer to this link: https://builtin.com/data-science/l2-regularization

[6. Remove layers / number of units per layer](https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d#87f3)  
[7. Dropout](https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d#6f6a)  
[8. Early stopping](https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d#98ac)

**Bias and Variance**

**Bias**

* Bias comes when the model is underfit.
* It is the error rate of the algorithm on the training accuracy.
* Suppose training accuracy is 85% and error rate is 15%. If this 15% can be reduced further, it is called avoidable bias.

**Variance**

* Variance comes when the model overfits the training data.
* It is how much worse the algorithm performs on the test data as compared to training accuracy.
* Suppose the test accuracy is 84% and error rate is 16%, the extra 1% which is coming in test accuracy, can be due to variance, i.e overfitting of the mode.

**Ensemble learning** is a method that combines multiple machine learning models to create more powerful models. The model should always have a balance between bias and variance. By leveraging the diverse strengths of different models, ensemble learning aims to mitigate errors, enhance performance, and increase the overall robustness of predictions. Two of them are:

* Bagging,
* Boosting,

An Outlier is an observation in the dataset that is far away from other observations in the dataset. Tools used to discover outliers are:

* Box plot
* Z-score
* Scatter plot, etc.

**Covariance and Correlation:**

* **Covariance** indicates the direction of the linear relationship between variables.
* **Correlation** measures both the strength and direction of the linear relationship between two variables.
* A correlation between variables, however, does not automatically mean that the change in one variable is the cause of the change in the values of the other variable.
* The value of covariance lies between -∞ and +∞ whereas correlation is betw -1 to 1.

**Boosting vs Bagging**

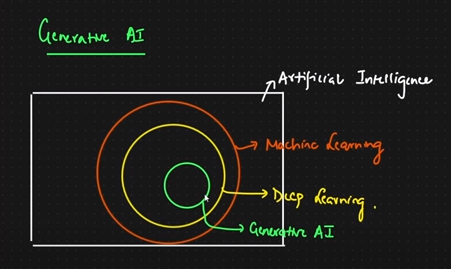
| S.NO | Bagging | Boosting |
| --- | --- | --- |
| 3. | Each model receives equal weight. | Models are weighted according to their performance. |
| 4. | Each model receives an equal weight. | Models are weighed based on their performance. |
| 5. | Training data subsets are drawn randomly with a replacement for the training dataset. | Every new subset contains the elements that were misclassified by previous models. |
| 6. | Bagging applied where the classifier is unstable and has a high variance. | Applied where the classifier is stable and simple and has high bias. |
| 8. | In this base classifiers are trained parallelly. | In this base classifiers are trained sequentially. |
| 9 | Example: The Random forest model uses Bagging. | Example: The AdaBoost uses Boosting techniques |

1. **Bagging** - homogeneous weak learners’ model that learns from each other independently in parallel and combines them for determining the model average.
2. **Boosting** - homogeneous weak learners’ model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.  Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

**Decision tree**

* A decision tree is a supervised learning algorithm used for classification and predictive modelling tasks.
* Entropy is calculated at every node to check the quality of split. 0<=entropy<=1. Worst split has entropy =1 when both classes has same number of data. Eg: will it rain- yes(50%) no(50%). Best split has 0 entropy.
* Entropy= -(Po\*log(Po)+ P1\*log(P1))
* Decision tree algorithms are popular in machine learning because they can handle complex datasets with ease and simplicity.
* By asking a sequence of questions and following the corresponding branches, decision trees enable us to classify or predict outcomes based on the data's characteristics.
* Gini impurity: 1- sum ((Pi)^2) ,where Pi=P0, P1.
* 0<=Gini Imp<=0.5.
* Gini impurity is better since it is computationally faster, more efficient since value only goes till 0.5 max.

**Generative AI**



* It is a subset of deep learning
* Chatgpt, bard- these are LLM(large language models) models, trained on huge huge datasets, based on generative AI.
* Chatgpt is giving output for new data, given trained on some dataset.
* When output is probability, text, number, categories- it is not Gen AI.
* When output is text, image , audio, viedo framaes- Gen AI.
* Chatgpt 3.5 trained with 170 billion parameters. Trained on internet data

