# **Theory**

In machine learning, there’s something called the “[No Free Lunch](http://www.no-free-lunch.org/)” theorem which means no one algorithm works well for every problem

If it is labelled data, it’s a supervised learning problem. If it’s unlabeled data with the purpose of finding structure, it’s an unsupervised learning problem. If the solution implies optimising an objective function by interacting with an environment, it’s a reinforcement learning problem

```

y = m\*X + c

```

1. y is the dependent variable

2. X is the independent variable

- Outliers?

> Suppose there is an observation in the dataset which is having a very high or very low value as compared to the other observations in the data, i.e. it does not belong to the population, such an observation is called an outlier. In simple words, it is of extreme value. An outlier is a problem because many times it hampers the results we get

- Multicollinearity?

> When the independent variables(X) are highly correlated to each other then the variables are said to be multicollinear. Many types of regression techniques assume multicollinearity should not be present in the dataset. It is because it causes problems in ranking variables based on their importance. Or it makes the job difficult in selecting the most important independent variable

- Heteroscedasticity?

> When a dependent variable's variability is not equal across values of an independent variable, it is called heteroscedasticity. Example -As one's income increases, the variability of food consumption will increase. A poorer person will spend a rather constant amount by always eating inexpensive food; a wealthier person may occasionally buy inexpensive food and at other times eat expensive meals. Those with higher incomes display a greater variability of food consumption.

- Underfitting and Overfitting?

> Overfitting means that our algorithm works well on the training set but is unable to perform better on the test sets. It is also known as the problem of **high variance**.

When our algorithm works so poorly that it is unable to fit even a training set well then it is said to underfit the data. It is also known as the problem of **high bias**.

- sparse and dense features?

Features with sparse data are features that have mostly zero values. This is different from features with missing data. ... On the other hand, features with dense data have predominantly non-zero values

- What is Regularisation?

> It reduces the overfitting nature of the model. This is done by introducing more errors and making the model learn more.

This will help the model to learn more. And as a result, even if more data is added in the later stage, the model will be able to process those without any issues. Now the model performance will increase and will be better than the unregularized model.

It mainly regularises or reduces the coefficient of features toward zero. In simple words, "*In regularisation technique, we reduce the magnitude of the features by keeping the same number of features*"

Regularisation is generally useful in the following situations:

1. High Multicollinearity
2. A large number of variables
3. Low ratio of number observations to number of variables

**Types of regularisation**

1. Ridge regularisation
2. Lasso regularisation
3. ElascticNet Regression

- Why Scaling?

> Normalisation and Standardisation are the two main methods for the scaling of the data. Which are widely used in the algorithms where scaling is required.

**The Big Question – Normalize or Standardise?**

1. Normalisation is good to use when you know that the distribution of your data does not follow a Gaussian distribution.
2. Standardisation, on the other hand, can be helpful in cases where the data follows a Gaussian distribution.

Ref Link: <https://analyticsindiamag.com/why-data-scaling-is-important-in-machine-learning-how-to-effectively-do-it/>

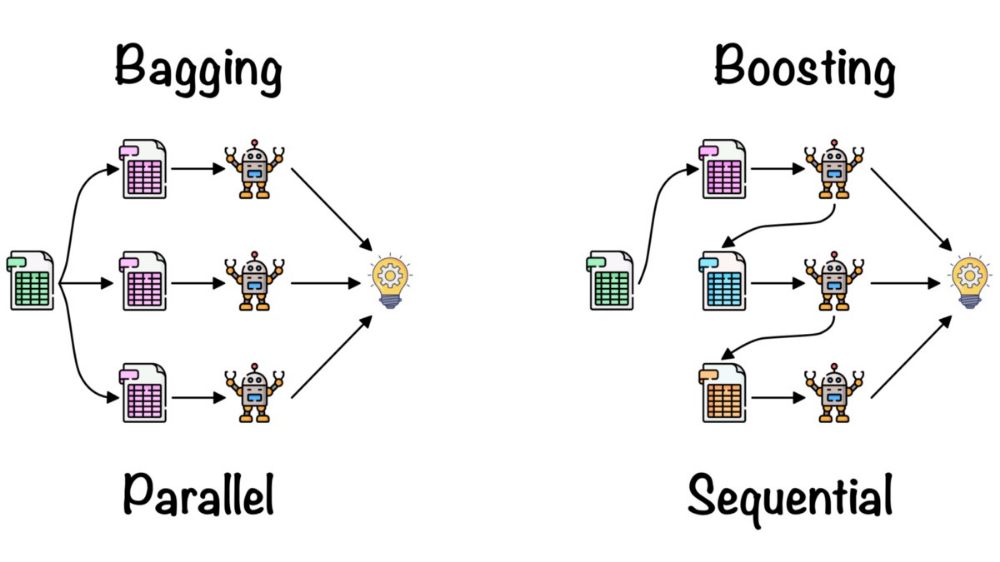
<https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>

- Bagging?

> It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, **Random Forest**

- Boosting?

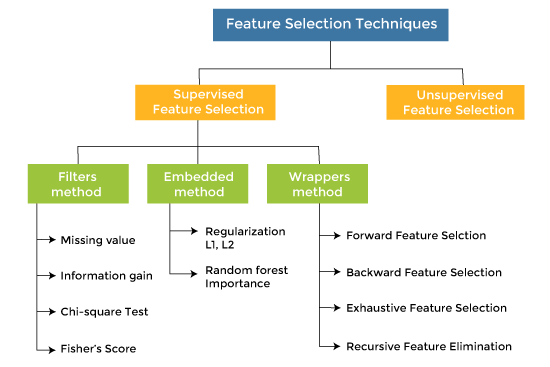
> It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, **ADA BOOST, XG BOOST**



- Feature Selection?

> The classes in the feature\_selection module can be used for feature selection/dimensionality reduction on sample sets, either to improve estimators’ accuracy scores or to boost their performance on very high-dimensional datasets.

The more the number of features the larger the datasets will be. Feature selection is also known as Variable selection or Attribute selection.



Sometimes, **feature selection is mistaken with dimensionality reduction**. But they are different. Feature selection is different from dimensionality reduction. Both methods tend to reduce the number of attributes in the dataset, but a dimensionality reduction method does so by creating new combinations of attributes (sometimes known as feature transformation), whereas feature selection methods include and exclude attributes present in the data without changing them.

Different types of general feature selection methods:

1. Filter methods
2. Wrapper methods
3. Embedded methods

**Chi-square Test**: The Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with the best Chi-square scores. In order to correctly apply the chi-squared in order to test the relation between various features in the dataset and the target variable, the following conditions have to be met: the variables have to be categorical.

**Variance Threshold**: Variance Threshold is a simple approach to eliminate features based on our expected variance within each feature. Although, there are some down-side with the Variance Threshold method. The Variance Threshold feature selection only sees the input features (X) without considering any information from the dependent variable (y). It is only useful for eliminating features for Unsupervised Modelling rather than Supervised Modelling. We need to transform all of these numerical features before we use the Variance Threshold Feature Selection. All should be on the same scale.

**Recursive Feature Elimination**: Recursive Feature Elimination or RFE is a Feature Selection method utilizing a machine learning model to select the features by eliminating the least important feature after recursively training.

**SelectFromModel**: Like the RFE, SelectFromModel from Scikit-Learn is based on a Machine Learning Model estimation for selecting the features. The differences are that SelectFromModel feature selection is based on the importance attribute (often is coef\_ or feature\_importances\_ but it could be any callable) threshold. By default, the threshold is the mean.

**Sequential Feature Selection**: SFS is a greedy algorithm to find the best features by either going forward or backward based on the cross-validation score of an estimator.

Has Two types:

1. **SFS-Forward**: A feature selection by starting with zero feature and finding the one feature that maximizes a cross-validated score when a machine learning model is trained on this single feature. Once that first feature is selected, the procedure is repeated by adding a new feature to selected features. The procedure is stopped when we find the desired number of features is reached
2. **SFS-Backward**: SFS-Backward follows the same idea but works in the opposite direction: It starts with all the features and greedily removes all the features until it reached the desired number of features

SFS differs from RFE and SelectFromModel because the machine learning model did not need the coef\_ or feature\_importances\_ attribute. Although, it is considerably slower as it evaluated the result by fitting the model multiple times. The higher number of features and data, your selection time would be getting higher as well

**Fisher’s Score**: Fisher score is one of the most widely used supervised feature selection methods. The algorithm which we will use returns the ranks of the variables based on the fisher’s score in descending order. We can then select the variables as per the case

Ref Link: <https://towardsdatascience.com/5-feature-selection-method-from-scikit-learn-you-should-know-ed4d116e4172>

<https://jundongl.github.io/scikit-feature/tutorial.html>

<https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/>

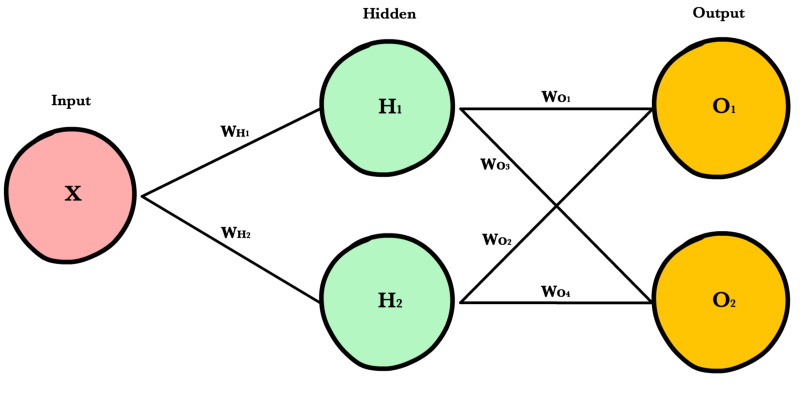
- Dimensionality Reduction?

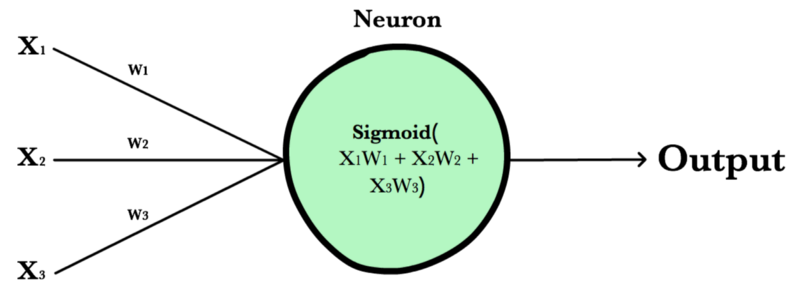
> Type dimensionality reduction methods are:

1. LDA
2. PCA
3. MDS
4. ISOMAP
5. T-SNE
6. LLE
7. UMAP

- Neural networks?

> Neural networks are trained iteratively using optimization techniques like gradient descent. After each cycle of training, an error metric is calculated based on the difference between prediction and target. The derivatives of this error metric are calculated and propagated back through the network using a technique called backpropagation. Each neuron’s coefficients (weights) are then adjusted relative to how much they contributed to the total error. This process is repeated iteratively until the network error drops below an acceptable threshold





**Synapse**: Synapses are like roads in a neural network. They connect inputs to neurons, neurons to neurons, and neurons to outputs. In order to get from one neuron to another, you have to travel along the synapse paying the “toll” (weight) along the way. Each connection between two neurons has a unique synapse with a unique weight attached to it. When we talk about updating weights in a network, we’re really talking about adjusting the weights on these synapses

we just multiply the inputs by the weights, add a bias and apply an activation function to the result and pass the output to the next layer

Ref Link: <https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html>

- What is Epoch?

> The number of times the algorithm runs on the whole training dataset

- What is Batch?

> It denotes the number of samples to be taken to for updating the model parameters

- What is Learning Rate?

> It is a parameter that provides the model a scale of how much model weights should be updated

- Cost Function/Loss Function?

> A cost function is used to calculate the cost that is the difference between the predicted value and the actual value

- Activation Functions?

It is a **function that is added into an artificial neural network in order to help the network learn complex patterns in the data**.

The activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer. They basically decide whether the neuron should be activated or not.

If we do not have the activation function the weights and bias would simply do a linear transformation. A linear equation is simple to solve but is limited in its capacity to solve complex problems and have less power to learn complex functional mappings from data. A neural network without an activation function is just a linear regression model.

Generally, neural networks use non-linear activation functions, which can help the network learn complex data, compute and learn almost any function representing a question, and provide accurate predictions

1. **Linear**

A straight line function where activation is proportional to input ( which is the weighted sum from neuron

1. **ELU**

Exponential Linear Unit or its widely known name ELU is a function that tends to converge cost to zero faster and produce more accurate results. Different from other activation functions, ELU has an extra alpha constant which should be a positive number

1. **ReLU**

A recent invention stands for Rectified Linear Units. The formula is deceptively simple: max(0,z). Despite its name and appearance, it’s not linear and provides the same benefits as Sigmoid (i.e. the ability to learn nonlinear functions), but with better performance. Although it can suffer from other problems like saturated or “dead” units

1. **LeakyReLU**

LeakyRelu is a variant of ReLU. Instead of being 0 when z<0, a leaky ReLU allows a small, non-zero, constant gradient α (Normally, α=0.01)

1. **Sigmoid**

Sigmoid takes a real value as input and outputs another value between 0 and 1. It’s easy to work with and has all the nice properties of activation functions: it’s non-linear, continuously differentiable, monotonic, and has a fixed output range

1. **Tanh**

Tanh squashes a real-valued number to the range [-1, 1]. It’s non-linear. But unlike Sigmoid, its output is zero-centered. Therefore, in practice, the tanh non-linearity is always preferred to the sigmoid nonlinearity

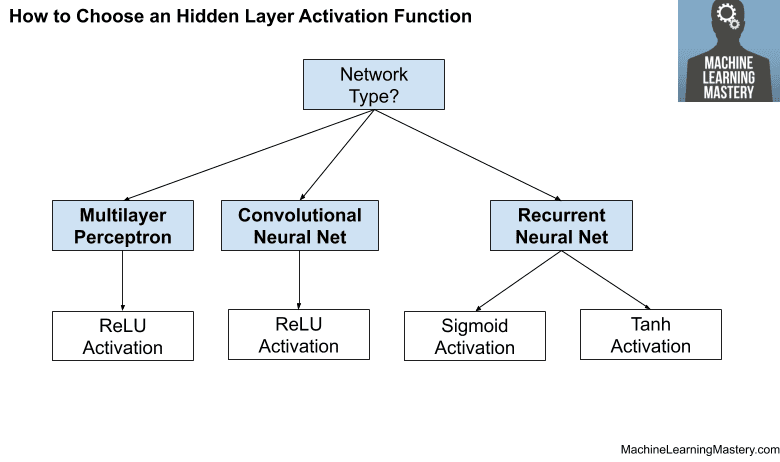
1. **Softmax**

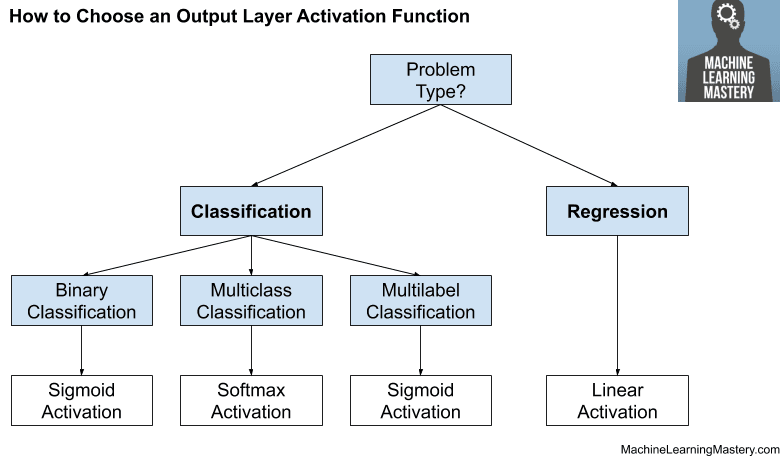
Softmax function calculates the probabilities distribution of the event over ‘n’ different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs

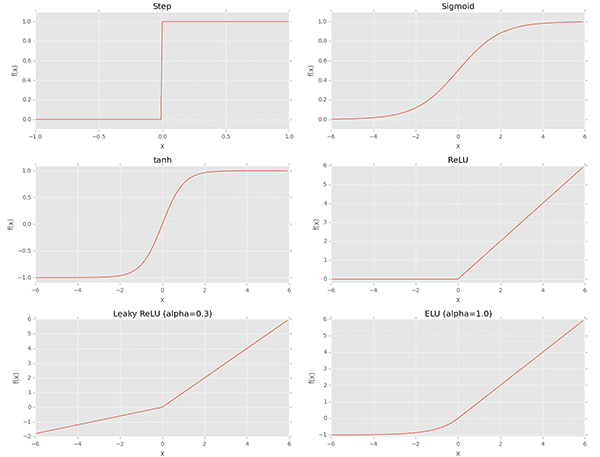
Binary: logistic

Multiclass : softmax

Multiclass : softprob







Ref Link:

<https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html>

<https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/>

<https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253>

<https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441>

- What is an optimizer in a neural network?

> An optimizer is **a function or an algorithm that modifies the attributes of the neural network**, such as weights and learning rate. Thus, it helps in reducing the overall loss and improve the accuracy

List of optimizer:

1. **SGD**
2. **adam**

The name adam is derived from adaptive moment estimation. This optimization algorithm is a further extension of stochastic gradient descent to update network weights during training. Unlike maintaining a single learning rate through training in SGD, Adam optimizer updates the learning rate for each network weight individually

1. **adagrad**

The change in learning rate depends upon the difference in the parameters during training. The more the parameters change, the more minor the learning rate changes. This modification is highly beneficial because real-world datasets contain sparse as well as dense features. So it is unfair to have the same value of learning rate for all the features. It’s downside of AdaGrad optimizer is that it decreases the learning rate aggressively and monotonically. There might be a point when the learning rate becomes extremely small

1. **adadelta**

AdaDelta can be seen as a more robust version of AdaGrad optimizer. It is based upon adaptive learning and is designed to deal with significant drawbacks of AdaGrad and RMS prop optimizer

1. **adamax**
2. **RMSprop**

The problem with RMS Prop is that the learning rate has to be defined manually and the suggested value doesn’t work for every application

1. **nadam**
2. **ftrl**

Ref Link: <https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/>

- What is metrics?

> Metrics are used to monitor and measure the performance of a model (during training and testing)

Keras Regression Metrics

1. Mean Squared Error: mean\_squared\_error, MSE or mse
2. Mean Absolute Error: mean\_absolute\_error, MAE, mae
3. Mean Absolute Percentage Error: mean\_absolute\_percentage\_error, MAPE, mape
4. Cosine Proximity: cosine\_proximity, cosine

Keras Classification Metrics

1. Binary Accuracy: binary\_accuracy, acc
2. Categorical Accuracy: categorical\_accuracy, acc
3. Sparse Categorical Accuracy: sparse\_categorical\_accuracy
4. Top k Categorical Accuracy: top\_k\_categorical\_accuracy (requires you specify a k parameter)
5. Sparse Top k Categorical Accuracy: sparse\_top\_k\_categorical\_accuracy (requires you specify a k parameter)

List of Accuracy:

1. **accuracy**
2. **binary\_accuracy**  
    It computes the mean accuracy rate across all predictions for binary classification problems
3. **categorical\_accuracy**

Calculates the mean accuracy rate across all predictions for multiclass classification problems

1. sparse\_categorical\_accuracy

It is similar to categorical\_accuracy but mostly used when making predictions for sparse targets. A great example of this is working with text in deep learning problems such as word2vec. In this case, one works with thousands of classes with the aim of predicting the next word

1. top\_k\_categorical\_accuracy
2. sparse\_top\_k\_categorical\_accuracy
3. **binary\_crossentropy**

Computes the crossentropy metric between the labels and predictions. This is the crossentropy metric class to be used when there are only two label classes (0 and 1)

1. **categorical\_crossentropy**

This is the crossentropy metric class to be used when there are multiple label classes (2 or more)

1. sparse\_categorical\_crossentropy
2. kullback\_leibler\_divergence

The Kullback-Leibler divergence (KLD), which is widely used to measure the similarity between two distributions

1. poisson
2. mean\_squared\_error
3. root\_mean\_squared\_error
4. mean\_absolute\_error
5. mean\_absolute\_percentage\_error
6. mean\_squared\_logarithmic\_error
7. cosine\_similarity
8. logcosh

Ref Link: <https://machinelearningmastery.com/custom-metrics-deep-learning-keras-python/>

- What is the loss in Keras?

> A scalar value that we attempt to minimize during our training of the model. The lower the loss, the closer our predictions are to the true labels.

As part of the optimization algorithm, the error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called a loss function, that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation

available loss List:

1. binary\_crossentropy
2. categorical\_crossentropy
3. sparse\_categorical\_crossentropy
4. kl\_divergence
5. poisson
6. mean\_squared\_error
7. mean\_absolute\_error
8. mean\_absolute\_percentage\_error
9. mean\_squared\_logarithmic\_error
10. cosine\_similarity
11. huber\_loss
12. log\_cosh
13. hinge
14. squared\_hinge
15. categorical\_hinge

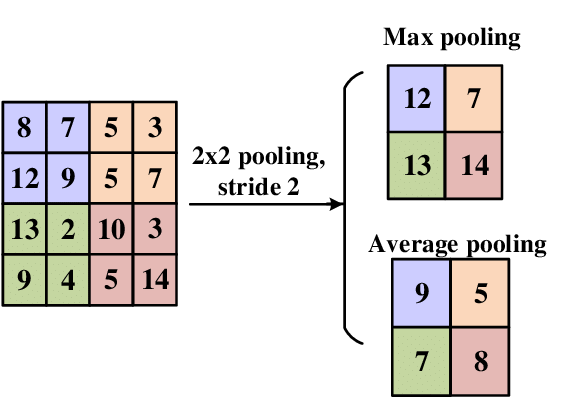
**# Most Uses Loss Function**

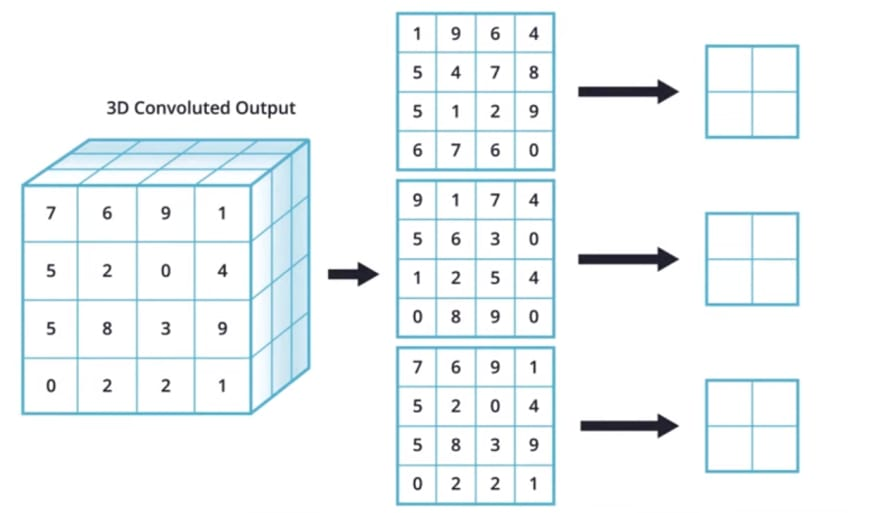
1. **Regression Loss Functions**
   1. Mean Squared Error Loss
   2. Mean Squared Logarithmic Error Loss
   3. Mean Absolute Error Loss
2. **Binary Classification Loss Functions**
   1. Binary Cross-Entropy
   2. Hinge Loss
   3. Squared Hinge Loss
3. **Multi-Class Classification Loss Functions**
   1. Multi-Class Cross-Entropy Loss
   2. Sparse Multiclass Cross-Entropy Loss
   3. Kullback Leibler Divergence Loss

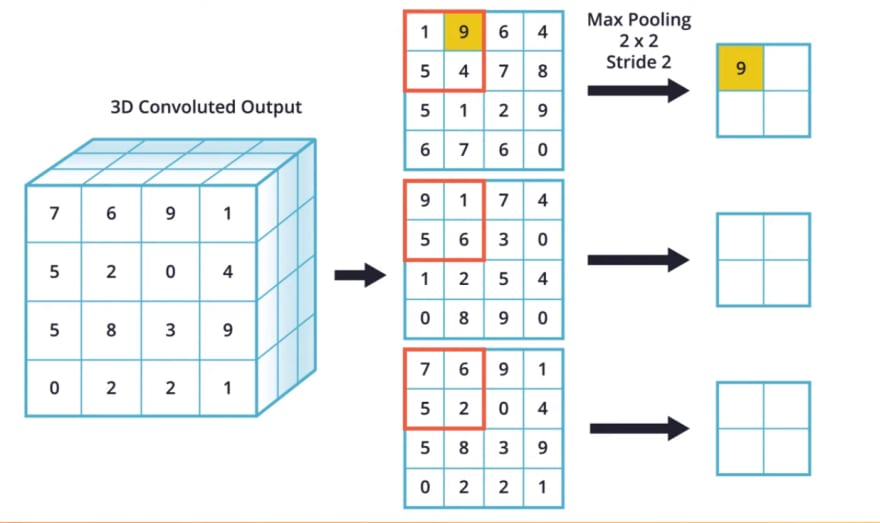
Ref Link: <https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/>

- Max pooling with Stride?

>



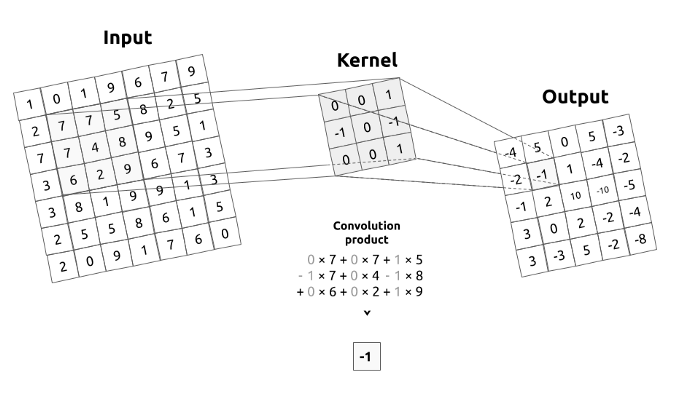


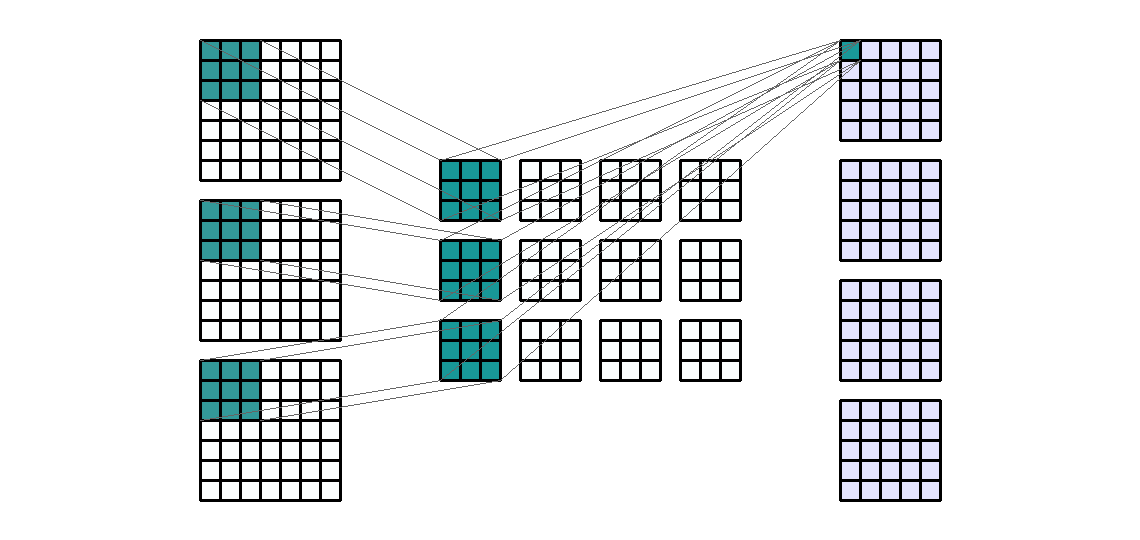


- What is conv2D, Kernel size and Filter, Strides?

>

# conv2D



#RGB Channel

For 4 output channels and 3 input channels, each output channel is the sum of 3 filtered input channels

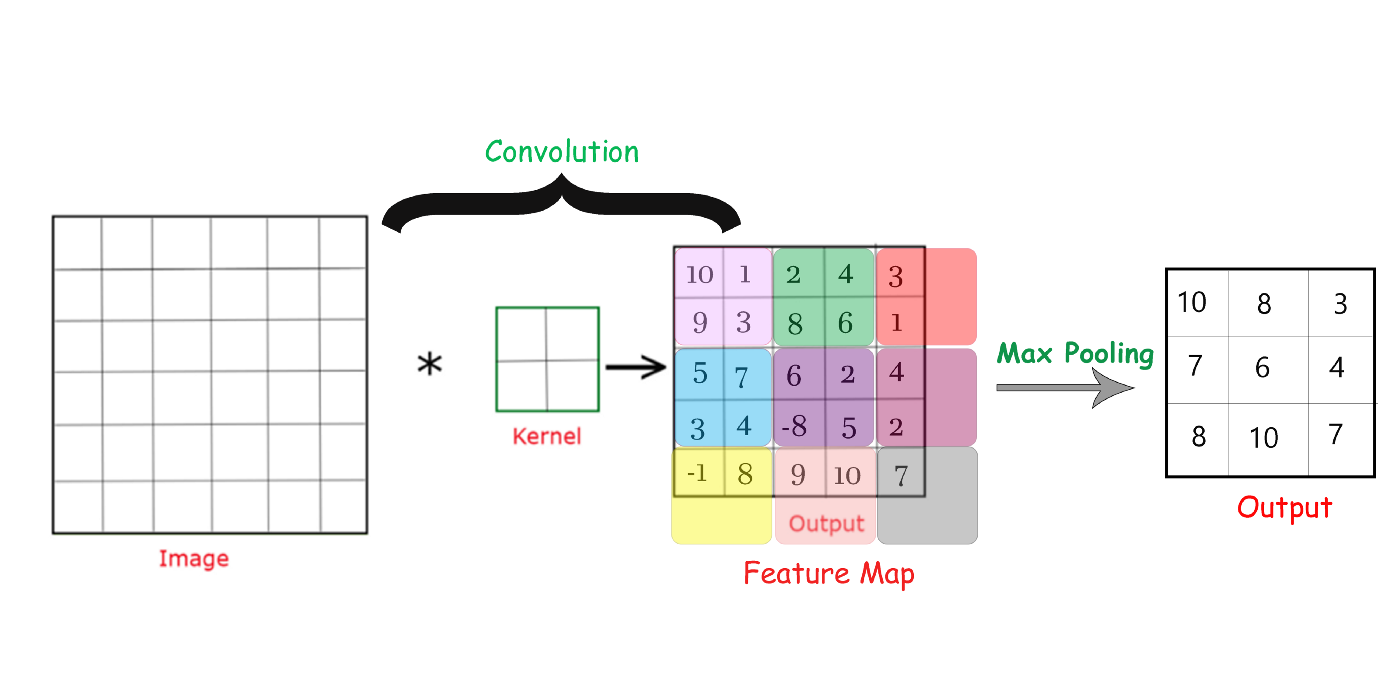
Ref Link:

<https://medium.com/machine-learning-algorithms/what-is-stride-in-convolutional-neural-network-e3b4ae9baedb>

<https://towardsdatascience.com/conv2d-to-finally-understand-what-happens-in-the-forward-pass-1bbaafb0b148>

<https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/>

- How conv2D to pooling works?

> 

- What is image data augmentation?

> When you don't have a large image dataset, it's a good practice to artificially introduce sample diversity by applying random yet realistic transformations to the training images, such as random horizontal flipping or small random rotations. This helps expose the model to different aspects of the training data while slowing down overfitting

# **Regression**

1. Linear Regression

It is a technique in which the dependent variable is continuous in nature. The relationship between the dependent variable and independent variables is assumed to be linear in nature

When you have only 1 independent variable and 1 dependent variable, it is called simple linear regression. When you have more than 1 independent variable and 1 dependent variable, it is called Multiple linear regression

1. Lasso [L1]

The (least absolute shrinkage and selection operator) adds the “Absolute value of magnitude” of coefficient as a penalty term to the loss function. It is called an L1 penalty.

sse = np.sum ((y-b1x1-b2x2-…-bo) \*\*2) + (alpha \* (|b1|+|b2|+|b3|+…+|bo|))

1. Ridge [L2]

It adds the “Squared magnitude” of coefficient as a penalty term to the loss function. It is called an L2 penalty.

sse = np.sum ((y-b1x1-b2x2-…-bo) \*\*2) + (alpha \* (b1\*\*2+b2\*\*2+b3\*\*2+…+bo\*\*2))

Ridge regression is computationally more efficient over lasso regression

1. ElascticNet (combine of L1+L2)

It is the combination of both Ridge and Lasso regularization.

sse = np.sum ((y-b1x1-b2x2-…-bo) \*\*2) + (alpha\_ridge \* (b1\*\*2+b2\*\*2+b3\*\*2+…+bo\*\*2)) +(alpha\_lasso \* (|b1|+|b2|+|b3|+…+|bo|))

Elastic Net regression is preferred over both ridge and lasso regression when one is dealing with highly correlated independent variables

1. SGD Regressor

Gradient descent is an iterative optimization algorithm used in machine learning to minimize a loss function.

Types of Gradient Descent

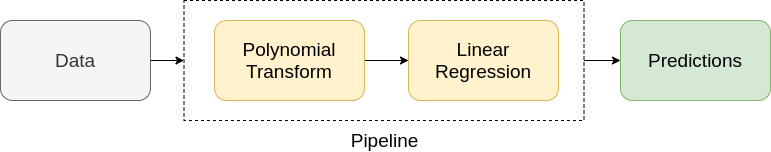
1. **Batch Gradient Descent**: Parameters are updated after computing the gradient of error with respect to the entire training set
2. **Stochastic Gradient Descent**: Parameters are updated after computing the gradient of error with respect to a single training example
3. **Mini-Batch Gradient Descent**: Parameters are updated after computing the gradient of error with respect to a subset of the training set
4. Polynomial Regressor

Deal with the non-linear relationships.

```

y = a0 + a1x1 + a2x12 + … + anx1n

```



Ref Link: <https://www.analyticsvidhya.com/blog/2021/07/all-you-need-to-know-about-polynomial-regression/>

1. SVM

The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three. Effective in high dimensional spaces

**Type of SVM:**

1. Linear SVM
2. Non-linear SVM

Ref Link: <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/#h2_5>

1. Decision Tree

Scikit-Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees.

Always remember that the higher the Entropy, the lower will be the purity and the higher will be the impurity.

The more the value of max\_depth, the more complex your tree will be. The training error will off-course decrease if we increase the max\_depth value but when our test data comes into the picture, we will get a very bad accuracy

Pruning – is nothing but cutting down some nodes to stop overfitting.

There are mainly 2 ways for pruning

- Pre-pruning

> we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance while growing the tree.

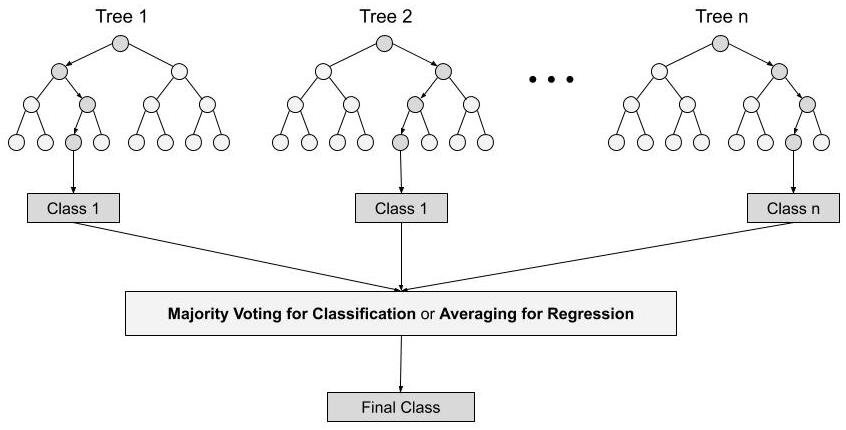
- Post-pruning

> once our tree is built to its depth, we can start pruning the nodes based on their significance

Ref Link: <https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/>

1. Random Forest

It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.



Ref Link: <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

1. Bagging Regressor

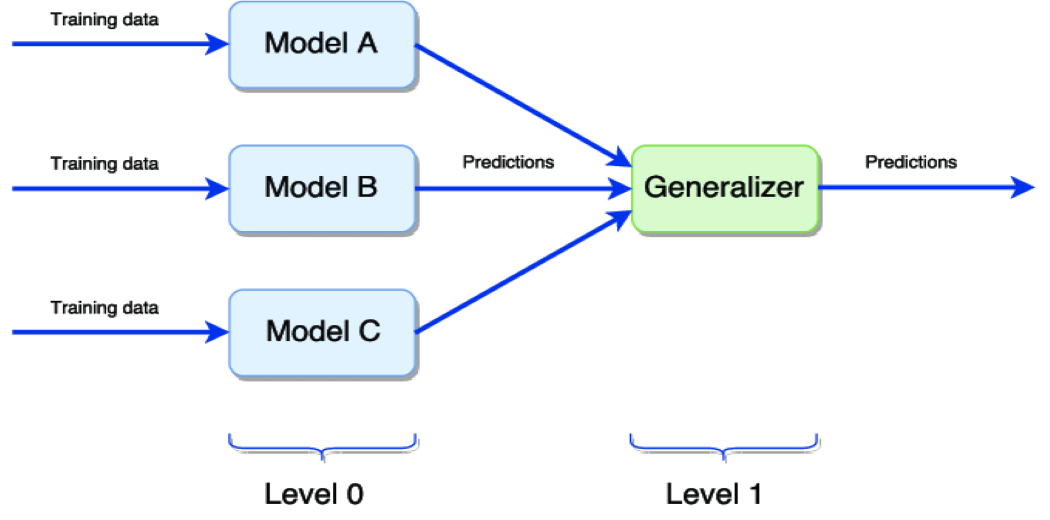
It is used to train them on different random subsets of the training set. When sampling is performed with replacement, this method is called bagging (short for bootstrap aggregating).

When sampling is performed without replacement, it is called pasting

1. Voting Regressor

A voting regressor is an ensemble meta-estimator that fits several base regressors, each on the whole dataset. Then it averages the individual predictions to form a final prediction

1. Stacking Regressor



Level 0 – Training different models on the same dataset then making predictions.

Level 1 – Generalize the predictions made by different models to get the final output.

1. Gradient Boosting

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor’s error.

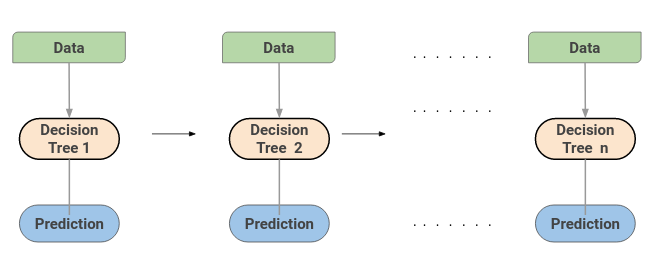
The errors in machine learning algorithms are broadly classified into two categories i.e. Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model

A Gradient Boosting Machine or GBM combines the predictions from multiple decision trees to generate the final predictions. Keep in mind that all the weak learners in a gradient boosting machine are decision trees.

**Here is the trick – the nodes in every decision tree take a different subset of features for selecting the best split. This means that the individual trees aren’t all the same and hence they are able to capture different signals from the data**.

Additionally, **each new tree takes into account the errors or mistakes made by the previous trees. So, every successive decision tree is built on the errors of the previous trees**. This is how the trees in a gradient boosting machine algorithm are built sequentially.

Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is **Mean Square Erro**r (MSE) and when it is used as a classifier then the cost function is **Log loss**



1. XGBoost

The working procedure of XGBoost is the same as GBM. **XGBoost used a more regularized model formalization to control over-fitting, which gives it better performance**.

Additionally, if you are using the XGBM algorithm, you don’t have to worry about imputing missing values in your dataset. The XGBM model can handle the missing values on its own. During the training process, the model learns whether missing values should be in the right or left node

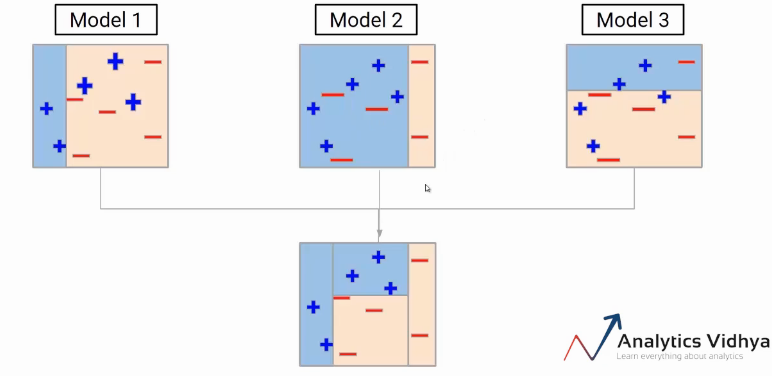
1. AdaBoost

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level which means with Decision trees with only 1 split. These trees are also called Decision Stumps

This algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received

Ref Link: [https://www.analyticsvidhya.com/blog/2021/03/introduction-to-adaboost-algorithm-with-python-implementation/](https://www.analyticsvidhya.com/blog/2021/03/introduction-to-adaboost-algorithm-with-python-implementation/#h2_3)

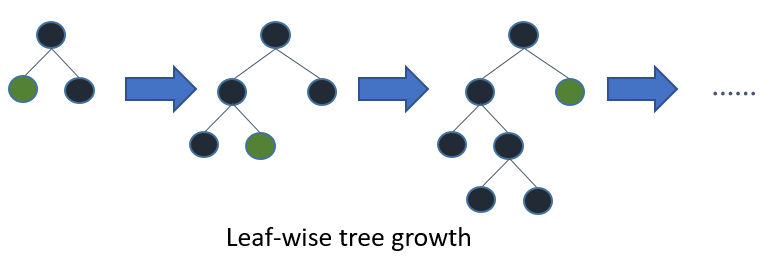
<https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/>



1. Light GBM

The LightGBM boosting algorithm is becoming more popular by the day due to its speed and efficiency. LightGBM is able to handle huge amounts of data with ease. But keep in mind that this algorithm does not perform well with a small number of data points.

The trees in LightGBM have a leaf-wise growth, rather than a level-wise growth. After the first split, the next split is done only on the leaf node that has a higher delta loss.



LightGBM uses a histogram-based method for selecting the best split. For any continuous variable, instead of using the individual values, these are divided into bins or buckets. This makes the training process faster and lowers memory usage

1. CatBoost

As the name suggests, CatBoost is a boosting algorithm that can handle categorical variables in the data. Most machine learning algorithms cannot work with strings or categories in the data. Thus, converting categorical variables into numerical values is an essential preprocessing step.

**CatBoost can internally handle categorical variables in the data**. These variables are transformed into numerical ones using various statistics on combinations of features

1. MARS
2. Principal Components Regression (PCR)

PCR is a regression technique that is widely used when you have many independent variables OR multicollinearity exist in your data. It is divided into 2 steps:

1. Getting the Principal components

2. Run regression analysis on principal components

The most common features of PCR are:

1. Dimensionality Reduction

2. Removal of multicollinearity

Principal components analysis is a statistical method to extract new features when the original features are highly correlated. We create new features with the help of original features such that the new features are uncorrelated. PCR is not a feature selection technique instead it is a feature extraction technique

1. Partial Least Squares (PLS) Regression

It is an alternative technique of principal component regression when you have independent variables highly correlated. It is also useful when there are a large number of independent variables

1. Poisson Regression

Poisson regression is used to predict a dependent variable that consists of "count data" given one or more independent variables

Application of Poisson Regression -

1. Predicting the number of calls in customer care related to a particular product

2. Estimating the number of emergency service calls during an event

Poisson regression assumes the variance equal to its mean

# **Classification**

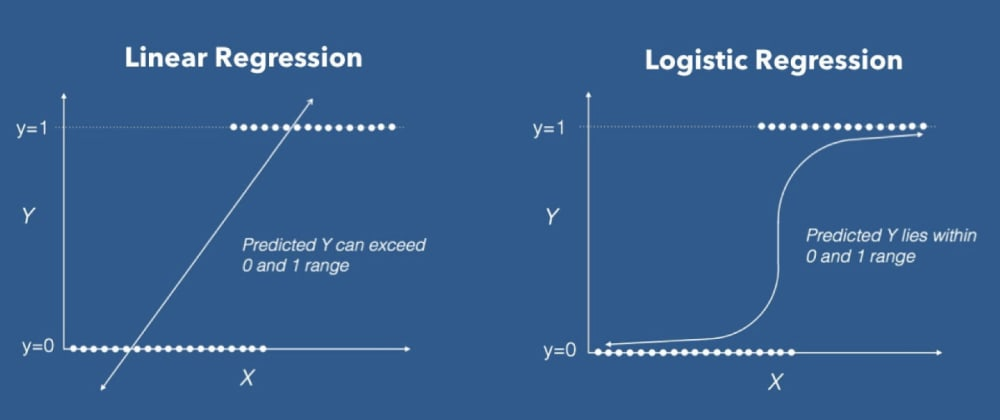
1. Logistic Regression

Logistic regression is usually used for Binary classification problems. Although it is said Logistic regression is used for Binary Classification, it can be extended to solve multiclass classification problems.

A few examples of Binary classification are Yes/No, Pass/Fail, Win/Lose, Cancerous/Non-cancerous, etc.

**Types of Logistic Regression**

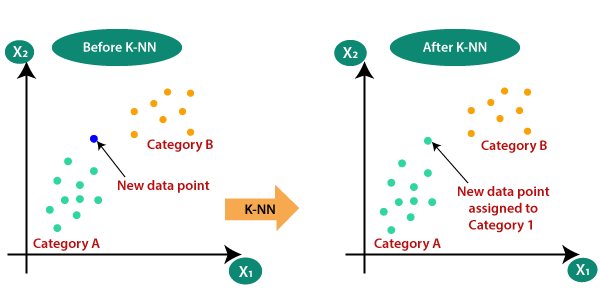
1. Simple Logistic Regression: a single independent is used to predict the output
2. Multiple logistic regression: multiple independent variables are used to predict the output
3. Multinomial Logistic Regression: The output variable is discrete in three or more classes with no natural ordering



1. SVM
2. KNN

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-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.



Ref Link: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

1. Decision Tree
2. Gaussian Naive Bayes

Types of Naive Bayes Classifiers

1. Gaussian Naive Bayes: This classifier is employed when the predictor values are continuous and are expected to follow a Gaussian distribution
2. BernoulliNB: When the predictors are boolean in nature and are supposed to follow the Bernoulli distribution, this classifier is utilized
3. Multinomial: It is suitable for classification with discrete features (e.g., word counts for text classification)

It is based on the Bayes Theorem for calculating probabilities and conditional probabilities. It is extremely fast relative to other classification algorithms. On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.

Ref Link: <https://www.javatpoint.com/machine-learning-naive-bayes-classifier>

1. Random Forest
2. SGD Classifier

SGD Classifier implements regularised linear models with Stochastic Gradient Descent.

Using hyper-parameters optimised by parfit, we can get equivalent performance from SGDClassifier in third of the time taken by LogisticRegression

1. LinearSVC

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. **LinearSVC minimizes the squared hinge loss while SVC minimizes the regular hinge loss**.

Between SVC and LinearSVC, one important decision criterion is that LinearSVC tends to be faster to converge the larger the number of samples is. This is due to the fact that the linear kernel is a special case, which is optimized for in Liblinear, but not in Libsvm

1. Bagging Classifier
2. Voting Classifier

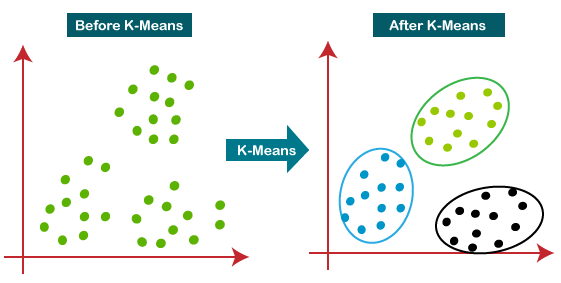
It has Two Type

1. **Hard Voting**: Suppose three classifiers predicted the output class(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction
2. **Soft Voting**: Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40). So the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier
3. Stacking Classifier
4. Gradient Boosting
5. Hist Gradient Boosting
6. XGBoost
7. AdaBoost
8. Light GBM
9. CatBoost

# **Clustering**

1. KMeans

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.



k-means clustering is a distance-based algorithm. This means that it tries to group the closest points to form a cluster. So, we first define the number of groups that we want to divide the population into – that’s the value of k. Based on the number of clusters or groups we want, we then randomly initialize k centroids. The data points are then assigned to the closest centroid and a cluster is formed. The centroids are then updated and the data points are reassigned. This process goes on iteratively until the location of centroids no longer changes. **centroids of the clusters are updated iteratively using the mean value**.

Indeed, the K-Means algorithm does not behave very well when the blobs have very different diameters since all it cares about when assigning an instance to a cluster is the distance to the centroid.

Instead of assigning **each instance to a single cluster, which is called hard clustering**, it can be useful to just give each instance a score per cluster: this is called soft clustering

Ref Link: <https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning>

1. Hierarchical Clustering

A Hierarchical clustering method works via grouping data into a tree of clusters.

Initially consider every data point as an individual Cluster and at every step, merge the nearest pairs of the cluster. (It is a bottom-up method). At first, every data set is considered as an individual entity or cluster. At every iteration, the clusters merge with different clusters until one cluster is formed

1. GaussianMixture

So instead of using a distance-based model, we will now use a **distribution-based model**. Gaussian Mixture Models (GMMs) assume that there are a certain number of Gaussian distributions, and each of these distributions represents a cluster.

**Gaussian Mixture Models use the soft clustering technique for assigning data points to Gaussian distributions**

Ref Link: <https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/>

2. Affinity Propagation

Affinity Propagation does not require you to specify the number of clusters. Affinity Propagation creates clusters by sending messages between data points until convergence.

1. DBSCAN

K-Means and Hierarchical Clustering both fail in creating clusters of arbitrary shapes. They are not able to form clusters based on varying densities. That’s why we need DBSCAN clustering.

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise.

DBSCAN requires only two parameters: epsilon and minPoints. Epsilon is the radius of the circle to be created around each data point to check the density and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point

Ref Link: <https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/>

# Optimize Hyperparameters

1. Grid search
2. Random search
3. Bayesian optimization

# Boosting Algorithms in Machine Learning

1. Gradient Boosting (GB) [Ref Link](https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/#h2_5)
2. Extreme Gradient Boosting Machine (XGBM) [Ref Link](https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#h2_8)
3. LightGBM [Ref Link](https://www.analyticsvidhya.com/blog/2021/08/complete-guide-on-how-to-use-lightgbm-in-python/)
4. CatBoost [Ref Link](https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/#h2_3)

### Flight Ticker Predict

<https://www.analyticsvidhya.com/blog/2022/01/flight-fare-prediction-using-machine-learning/>

### Diabetes Prediction

<https://www.analyticsvidhya.com/blog/2022/01/diabetes-prediction-using-machine-learning/>

# OpenCV

### PAN card fraud

<https://www.analyticsvidhya.com/blog/2021/06/pan-card-fraud-detection-using-computer-vision/>