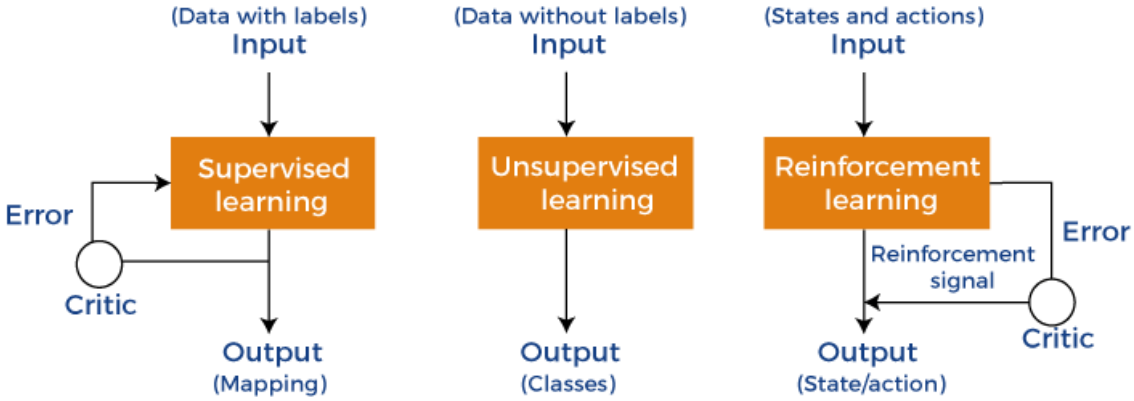
Modeling:

A machine learning model is defined as a mathematical representation of the output of the training process. Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model. A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

Classification of Machine Learning Models:

Based on different business goals and data sets, there are three learning models for algorithms. Each machine learning algorithm settles into one of the three models:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



Supervised Learning is further divided into two categories:

- Classification
- Regression

Unsupervised Learning is also divided into below categories:

- Clustering
- Association Rule
- Dimensionality Reduction

1. Supervised Machine Learning Models

Supervised Learning is the simplest machine learning model to understand in which input data is called training data and has a known label or result as an output. So, it works on the principle of input-output pairs. It requires creating a function that can be trained using a training data set, and then it is applied to unknown data and makes some predictive performance. Supervised learning is task-based and tested on labeled data sets.

We can implement a supervised learning model on simple real-life problems. For example, we have a dataset consisting of age and height; then, we can build a supervised learning model to predict the person's height based on their age.

Supervised Learning models are further classified into two categories:

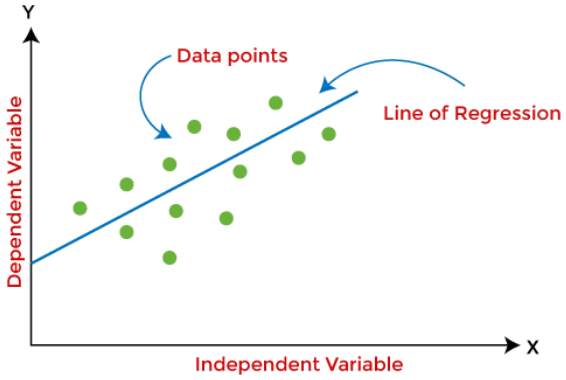
Regression

In regression problems, the output is a continuous variable. Some commonly used Regression models are as follows:

a) Linear Regression

Linear regression is the simplest machine learning model in which we try to predict one output variable using one or more input variables. The representation of linear regression is a linear equation, which combines a set of input values(x) and predicted output(y) for the set of those input values. It is represented in the form of a line:

$$Y = bx + c.$$



The main aim of the linear regression model is to find the best fit line that best fits the data points. Linear regression is extended to multiple linear regression (find a plane of best fit) and polynomial regression (find the best fit curve).

b) Decision Tree

Decision trees are the popular machine learning models that can be used for both regression and classification problems.

A decision tree uses a tree-like structure of decisions along with their possible consequences and outcomes. In this, each internal node is used to represent a test on an attribute; each branch is used to represent the outcome of the test. The more nodes a decision tree has, the more accurate the result will be.

The advantage of decision trees is that they are intuitive and easy to implement, but they lack accuracy.

Decision trees are widely used in **operations research, specifically in decision analysis, strategic planning**, and mainly in machine learning.

c) Random Forest

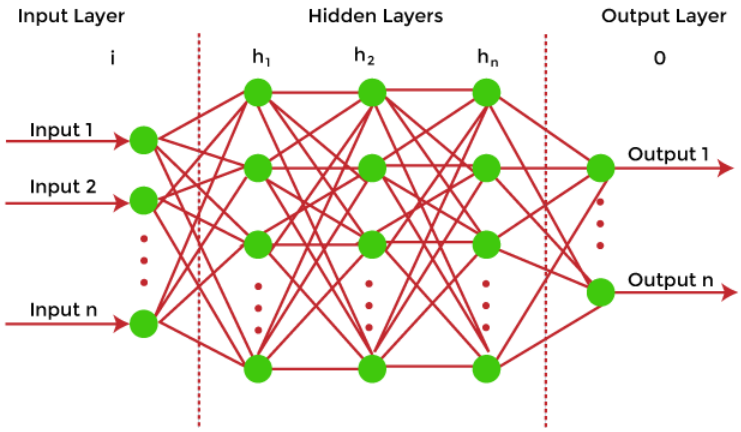
Random Forest is the ensemble learning method, which consists of a large number of decision trees. Each decision tree in a random forest predicts an outcome, and the prediction with the majority of votes is considered as the outcome.

A random forest model can be used for both regression and classification problems.

For the classification task, the outcome of the random forest is taken from the majority of votes. Whereas in the regression task, the outcome is taken from the mean or average of the predictions generated by each tree.

d) Neural Networks

Neural networks are the subset of machine learning and are also known as artificial neural networks. Neural networks are made up of artificial neurons and designed in a way that resembles the human brain structure and working. Each artificial neuron connects with many other neurons in a neural network, and such millions of connected neurons create a sophisticated cognitive structure.



Neural networks consist of a multilayer structure, containing one input layer, one or more hidden layers, and one output layer. As each neuron is connected with another neuron, it transfers data from one layer to the other neuron of the next layers. Finally, data reaches the last layer or output layer of the neural network and generates output.

Neural networks depend on training data to learn and improve their accuracy. However, a perfectly trained & accurate neural network can cluster data quickly and become a powerful machine learning and AI tool. One of the best-known neural networks is **Google's search algorithm.**

Classification

Classification models are the second type of Supervised Learning techniques, which are used to generate conclusions from observed values in the categorical form. For example, the classification model can identify if the email is spam or not; a buyer will purchase the product or not, etc. Classification algorithms are used to predict two classes and categorize the output into different groups.

In classification, a classifier model is designed that classifies the dataset into different categories, and each category is assigned a label.

There are two types of classifications in machine learning:

- **Binary classification**: If the problem has only two possible classes, called a binary classifier. For example, cat or dog, Yes or No,
- **Multi-class classification**: If the problem has more than two possible classes, it is a multi-class classifier.

Some popular classification algorithms are as below:

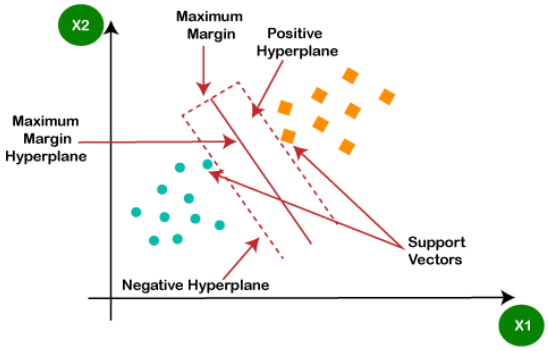
a) Logistic Regression

Logistic Regression is used to solve the classification problems in machine learning. They are similar to linear regression but used to predict the categorical variables. It can predict the output in either Yes or No, 0 or 1, True or False, etc. However, rather than giving the exact values, it provides the probabilistic values between 0 & 1.

b) Support Vector Machine

Support vector machine or SVM is the popular machine learning algorithm, which is widely used for classification and regression tasks. However, specifically, it is used to solve classification problems. The main aim of SVM is to find the best decision boundaries in an N-dimensional space, which can segregate data points into classes, and the best decision

boundary is known as Hyperplane. SVM selects the extreme vector to find the hyperplane, and these vectors are known as support vectors.



c) Naïve Bayes

Naïve Bayes is another popular classification algorithm used in machine learning. It is called so as it is based on Bayes theorem and follows the naïve(independent) assumption between the features which is given as:

$$P(y|X) = \frac{P(X|y) * P(y)}{P(X)}$$

Each naïve Bayes classifier assumes that the value of a specific variable is independent of any other variable/feature. For example, if a fruit needs to be classified based on color, shape, and taste. So yellow, oval, and sweet will be recognized as mango. Here each feature is independent of other features.

2. Unsupervised Machine learning models

Unsupervised Machine learning models implement the learning process opposite to supervised learning, which means it enables the model to learn from the unlabeled training dataset. Based on the unlabeled dataset, the model predicts the output. Using unsupervised learning, the model learns hidden patterns from the dataset by itself without any supervision. Unsupervised learning models are mainly used to perform three tasks, which are as follows:

- **Clustering**

  Clustering is an unsupervised learning technique that involves clustering or groping the data points into different clusters based on similarities and differences. The objects with the most similarities remain in the same group, and they have no or very few similarities from other groups. Clustering algorithms can be widely used in different tasks such as **Image segmentation, Statistical data analysis, Market segmentation**, etc. Some commonly used Clustering algorithms are *K-means Clustering, hierarchal Clustering, DBSCAN*, etc.

- **Association Rule Learning**
  Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc. Some popular algorithms of Association rule learning are ***Apriori Algorithm, Eclat, FP-growth algorithm.***
- **Dimensionality Reduction**
  The number of features/variables present in a dataset is known as the dimensionality of the dataset, and the technique used to reduce the dimensionality is known as the dimensionality reduction technique. Although more data provides more accurate results, it can also affect the performance of the model/algorithm, such as overfitting issues. In such cases, dimensionality reduction techniques are used. "***It is a process of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information***." Different dimensionality reduction methods such ***as PCA(Principal Component Analysis), Singular Value Decomposition, etc.***

Reinforcement Learning

In reinforcement learning, the algorithm learns actions for a given set of states that lead to a goal state. It is a feedback-based learning model that takes feedback signals after each state or action by interacting with the environment. This feedback works as a reward (positive for each good action and negative for each bad action), and the agent's goal is to maximize the positive rewards to improve their performance.

The behavior of the model in reinforcement learning is similar to human learning, as humans learn things by experiences as feedback and interact with the environment.

Below are some popular algorithms that come under reinforcement learning:

- **Q-learning:** Q-learning is one of the popular model-free algorithms of reinforcement learning, which is based on the Bellman equation.

It aims to learn the policy that can help the AI agent to take the best action for maximizing the reward under a specific circumstance. It incorporates Q values for each state-action pair that indicate the reward to following a given state path, and it tries to maximize the Q-value.

- **State-Action-Reward-State-Action (SARSA):** SARSA is an On-policy algorithm based on the Markov decision process. It uses the action performed by the current policy to learn the Q-value. The SARSA algorithm stands **for State Action Reward State Action, which symbolizes the tuple (s, a, r, s', a').**
- **Deep Q Network:** DQN **or Deep Q Neural network is Q-learning** within the neural network. It is basically employed in a big state space environment where defining a Q-table would be a complex task. So, in such a case, rather than using Q-table, the neural network uses Q-values for each action based on the state.

Underfitting and Overfitting:

When we talk about the Machine Learning model, we actually talk about how well it performs and its accuracy which is known as prediction errors. Let us consider that we are designing a machine learning model. A model is said to be a good machine learning model if it generalizes any new input data from the problem domain in a proper way. This helps us to make predictions about future data that the data model has never seen. Now, suppose we want to check how well our machine learning model learns and generalizes to the new data. For that, we have overfitting and underfitting, which are majorly responsible for the poor performances of the machine learning algorithms.

Before diving further let's understand two important terms:

- **Bias:** Assumptions made by a model to make a function easier to learn. It is actually the error rate of the training data. When the error rate has a high value, we call it High Bias and when the error rate has a low value, we call it low Bias.
- **Variance:** The difference between the error rate of training data and testing data is called variance. If the difference is high then it's called high variance and when the difference of errors is low then it's called low variance. Usually, we want to make a low variance for generalized our model.

Underfitting:

A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data, i.e., it only performs well on training data but performs poorly on testing data. (It's just like trying to fit undersized pants!) Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have fewer data to build an accurate model and also when we try to build a linear model with fewer non-linear data. In such cases, the rules of the machine learning model are too easy and flexible to be applied to such minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.

In a nutshell, Underfitting refers to a model that can neither performs well on the training data nor generalize to new data.

Reasons for Underfitting:

- High bias and low variance
- The size of the training dataset used is not enough.
- The model is too simple.
- Training data is not cleaned and also contains noise in it.

Techniques to reduce underfitting:

- Increase model complexity
- Increase the number of features, performing feature engineering
- Remove noise from the data.
- Increase the number of epochs or increase the duration of training to get better results.

Overfitting:

A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning

from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

In a nutshell, Overfitting is a problem where the evaluation of machine learning algorithms on training data is different from unseen data.

Reasons for Overfitting are as follows:

- High variance and low bias
- The model is too complex
- The size of the training data

Techniques to reduce overfitting:

1. Increase training data.
2. Reduce model complexity.
3. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
4. Ridge Regularization and Lasso Regularization
5. Use dropout for neural networks to tackle overfitting.

Good Fit in a Statistical Model:

Ideally, the case when the model makes the predictions with 0 error, is said to have a *good fit* on the data. This situation is achievable at a spot between overfitting and underfitting. In order to understand it, we will have to look at the performance of our model with the passage of time, while it is learning from the training dataset.

With the passage of time, our model will keep on learning, and thus the error for the model on the training and testing data will keep on decreasing. If it will learn for too long, the model will become more prone to overfitting due to the presence of noise and less useful details. Hence the performance of our model will decrease. In order to get a good fit, we will stop at a point just before where the error starts increasing. At this point, the model is said to have good skills in training datasets as well as our unseen testing dataset.

Correctness:

Data scientists know that when they build training sets, they need to watch out for data leakage in order to ensure that a model is only trained on the correct data. Data leakage occurs when models are trained on examples that did not really occur in the real world. In time-series models, data leakage typically is caused by adding features to your training set that occurred after a given prediction would have occurred. While all data scientists know data leakage is something they need to watch out for, actually building training sets without data leakage is rarely as straightforward as it seems, especially when machine learning models make predictions in real-time.

The Point-in-Time Correctness Problem

When feature generation, predictions, and label generation occur at different points in time, data leakage can easily be introduced into your training sets. This is often called the

point-in-time correctness problem. How might this problem pop up in a real-time machine learning application?

Imagine you have an e-commerce website that makes product recommendations. The features for this model might include:

- RFM metrics, such as the sum of products purchased by a user over the last week or month or year, calculated every week
- Summary of the items currently in a user's cart that are updated in real time

The label for this model might be: was the product that was recommended actually purchased in the same web session.

A subtle, but important, complication in training a model like this is wrangling the many different timestamps that are present.

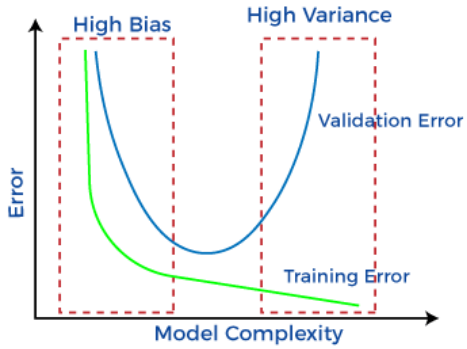
For this machine learning problem, we have 4 timestamps to keep track of:

- When the weekly batch RFM feature aggregation occurs
- When products are added to the cart
- When a product recommendation is generated
- When the purchase is actually made

This is a classic example of a point-in-time correctness problem that pops up whenever you have a real-time machine learning model. When building their training set, the data scientist should join the most up-to-date features that would have been available at prediction time to each training example, and no features that were generated after prediction time. Any feature values added to a row of training data that occurred after the prediction would have actually been generated would constitute data leakage- the real world model wouldn't have access to that data.

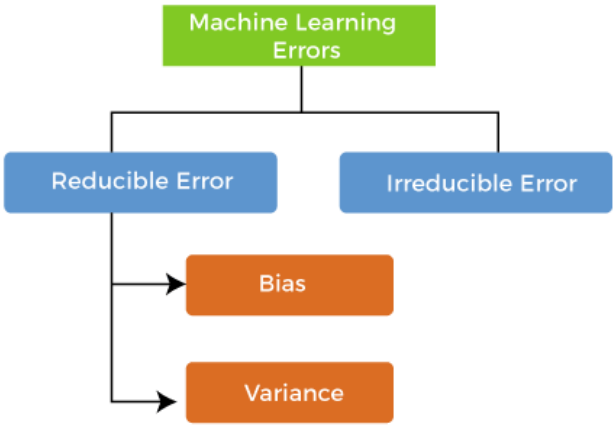
Bias & Variance in Machine Learning:

If the machine learning model is not accurate, it can make predictions errors, and these prediction errors are usually known as Bias and Variance. In machine learning, these errors will always be present as there is always a slight difference between the model predictions and actual predictions. The main aim of ML/data science analysts is to reduce these errors in order to get more accurate results.



There are mainly two types of errors in machine learning, which are:

- **Reducible errors:** These errors can be reduced to improve the model accuracy. Such errors can further be classified into bias and Variance.
- **Irreducible errors:** These errors will always be present in the model regardless of which algorithm has been used. The cause of these errors is unknown variables whose value can't be reduced.



a) Bias

While making predictions, a difference occurs between prediction values made by the model and actual values/expected values, and this difference is known as bias errors or Errors due to bias. It can be defined as an inability of machine learning algorithms such as Linear Regression to capture the true relationship between the data points. Each algorithm begins with some amount of bias because bias occurs from assumptions in the model, which makes the target function simple to learn. A model has either:

- **Low Bias:** A low bias model will make fewer assumptions about the form of the target function.
- **High Bias:** A model with a high bias makes more assumptions, and the model becomes unable to capture the important features of our dataset. A high bias model also cannot perform well on new data.

Generally, a linear algorithm has a high bias, as it makes them learn fast. The simpler the algorithm, the higher the bias it has likely to be introduced. Whereas a nonlinear algorithm often has low bias.

Some examples of machine learning algorithms with ***low bias are Decision Trees, k-Nearest Neighbours and Support Vector Machines.*** At the same time, an algorithm with ***high bias is Linear Regression, Linear Discriminant Analysis and Logistic Regression.***

Ways to reduce High Bias:

High bias mainly occurs due to a much simple model. Below are some ways to reduce the high bias:

- Increase the input features as the model is underfitted.
- Decrease the regularization term.
- Use more complex models, such as including some polynomial features.

b) Variance:

Variance tells that how much a random variable is different from its expected value. Ideally, a model should not vary too much from one training dataset to another, which means the algorithm should be good in understanding the hidden mapping between inputs and output variables. Variance errors are either of *low variance or high variance*.

- **Low variance** means there is a small variation in the prediction of the target function with changes in the training data set.
- **High variance** shows a large variation in the prediction of the target function with changes in the training dataset.

A model that shows high variance learns a lot and perform well with the training dataset, and does not generalize well with the unseen dataset. As a result, such a model gives good results with the training dataset but shows high error rates on the test dataset.

Since, with high variance, the model learns too much from the dataset, it leads to overfitting of the model. A model with high variance has the below problems:

- A high variance model leads to overfitting.
- Increase model complexities.

Usually, nonlinear algorithms have a lot of flexibility to fit the model, have high variance.

Some examples of machine learning algorithms with ***low variance are, Linear Regression, Logistic Regression, and Linear discriminant analysis.*** At the same time, algorithms with ***high variance are decision tree, Support Vector Machine, and K-nearest neighbours.***

Ways to Reduce High Variance:

- Reduce the input features or number of parameters as a model is overfitted.
- Do not use a much complex model.
- Increase the training data.
- Increase the Regularization term.

Bias-Variance Trade-Off

While building the machine learning model, it is really important to take care of bias and variance in order to avoid overfitting and underfitting in the model. If the model is very simple with fewer parameters, it may have low variance and high bias. Whereas, if the model has a large number of parameters, it will have high variance and low bias. So, it is required to make a balance between bias and variance errors, and this balance between the bias error and variance error is known as the Bias-Variance trade-off.

For an accurate prediction of the model, algorithms need a low variance and low bias. But this is not possible because bias and variance are related to each other:

- If we decrease the variance, it will increase the bias.
- If we decrease the bias, it will increase the variance.

Bias-Variance trade-off is a central issue in supervised learning. Ideally, we need a model that accurately captures the regularities in training data and simultaneously generalizes well with the unseen dataset. Unfortunately, doing this is not possible simultaneously. Because a high variance algorithm may perform well with training data, but it may lead to overfitting to noisy data. Whereas, high bias algorithm generates a much simple model that may not even capture important regularities in the data. So, we need to find a sweet spot between bias and variance to make an optimal model.

Hence, the ***Bias-Variance trade-off is about finding the sweet spot to make a balance between bias and variance errors.***

Feature Extraction and Selection:

While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning.

- Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features.
- Each machine learning process depends on feature engineering, which mainly contains two processes; which are ***Feature Selection*** and ***Feature Extraction***.
- Although feature selection and extraction processes may have the same objective, both are completely different from each other.
- The main difference between them is that *feature selection* is about selecting the subset of the original feature set, whereas *feature extraction* creates new features. Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model.

Need of feature Selection:

Before implementing any technique, it is really important to understand, need for the technique and so for the Feature Selection. As we know, in machine learning, it is necessary to provide a pre-processed and good input dataset in order to get better outcomes. We collect a huge amount of data to train our model and help it to learn better. Generally, the dataset consists of noisy data, irrelevant data, and some part of useful data. Moreover, the huge amount of data also slows down the training process of the model, and with noise and irrelevant data, the model may not predict and perform well. So, it is very necessary to remove such noises and less-important data from the dataset and to do this, and Feature selection techniques are used.

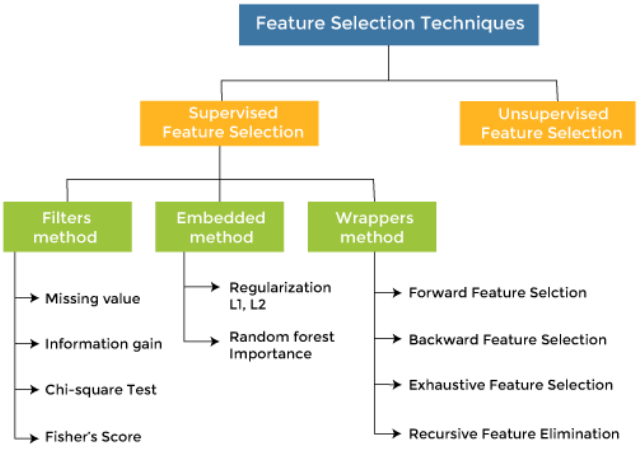
Below are some benefits of using feature selection in machine learning:

- It helps in avoiding the curse of dimensionality.
- It helps in the simplification of the model so that it can be easily interpreted by the researchers.
- It reduces the training time.
- It reduces overfitting hence enhance the generalization.

Feature Selection Techniques

There are mainly two types of Feature Selection techniques, which are:

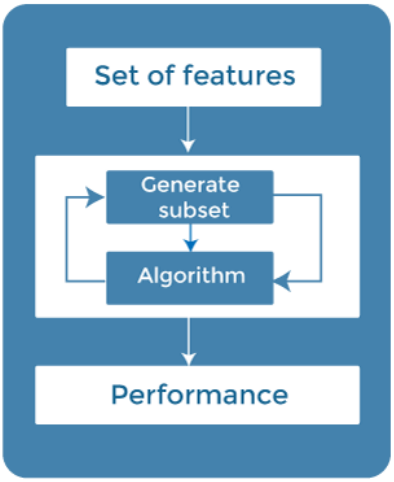
- **Supervised Feature Selection technique**
  Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
- **Unsupervised Feature Selection technique**
  Unsupervised Feature selection techniques ignore the target variable and can be used for the unlabelled dataset.



There are mainly three techniques under supervised feature Selection:

1. Wrapper Methods

In wrapper methodology, selection of features is done by considering it as a search problem, in which different combinations are made, evaluated, and compared with other combinations. It trains the algorithm by using the subset of features iteratively.



On the basis of the output of the model, features are added or subtracted, and with this feature set, the model has trained again.

Some techniques of wrapper methods are:

- **Forward selection** - Forward selection is an iterative process, which begins with an empty set of features. After each iteration, it keeps adding on a feature and evaluates the performance to check whether it is improving the performance or not. The process continues until the addition of a new variable/feature does not improve the performance of the model.
- **Backward elimination** - Backward elimination is also an iterative approach, but it is the opposite of forward selection. This technique begins the process by considering all the features and removes the least significant feature. This elimination process continues until removing the features does not improve the performance of the model.
- **Exhaustive Feature Selection-** Exhaustive feature selection is one of the best feature selection methods, which evaluates each feature set as brute-force. It means this

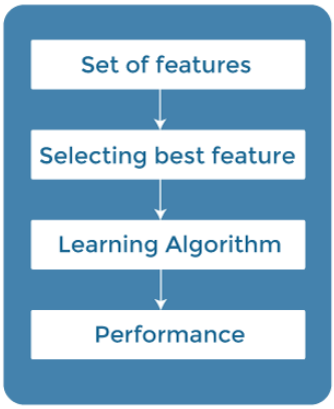
method tries & make each possible combination of features and return the best performing feature set.

- **Recursive Feature Elimination -** Recursive feature elimination is a recursive greedy optimization approach, where features are selected by recursively taking a smaller and smaller subset of features. Now, an estimator is trained with each set of features, and the importance of each feature is determined using *coef_attribute* or through a *feature_importances_attribute.*

2. Filter Methods

In Filter Method, features are selected on the basis of statistics measures. This method does not depend on the learning algorithm and chooses the features as a pre-processing step. The filter method filters out the irrelevant feature and redundant columns from the model by using different metrics through ranking.

The advantage of using filter methods is that it needs low computational time and does not overfit the data.



Some common techniques of Filter methods are as follows:

- Information Gain
- Chi-square Test
- Fisher's Score
- Missing Value Ratio

Information Gain: Information gain determines the reduction in entropy while transforming the dataset. It can be used as a feature selection technique by calculating the information gain of each variable with respect to the target variable.

Chi-square Test: Chi-square test is a technique to determine the relationship between the categorical variables. The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.

Fisher's Score:

Fisher's score is one of the popular supervised technique of features selection. It returns the rank of the variable on the fisher's criteria in descending order. Then we can select the variables with a large fisher's score.

Missing Value Ratio:

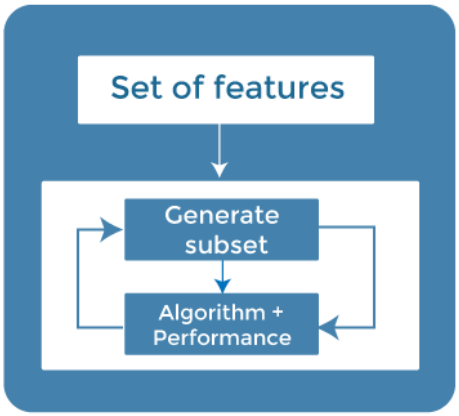
The value of the missing value ratio can be used for evaluating the feature set against the threshold value. The formula for obtaining the missing value ratio is the number of missing

values in each column divided by the total number of observations. The variable is having more than the threshold value can be dropped.

$$\text{Missing Value Ratio}=\frac{Number\ of\ Missing\ values*100}{Total\ number\ of\ observations}$$

3. Embedded Methods

Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost. These are fast processing methods similar to the filter method but more accurate than the filter method.

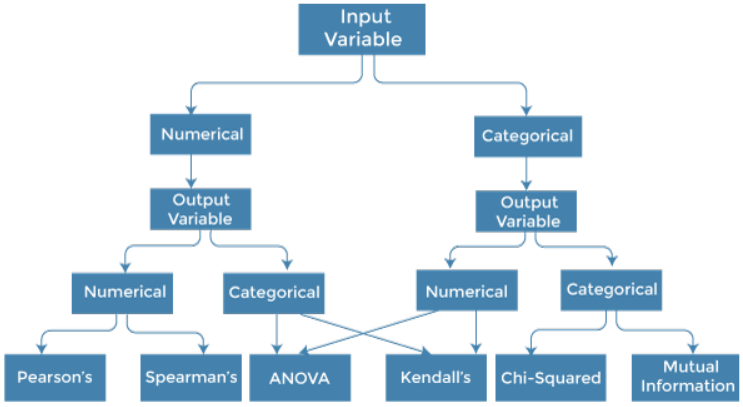


These methods are also iterative, which evaluates each iteration, and optimally finds the most important features that contribute the most to training in a particular iteration. Some techniques of embedded methods are:

- **Regularization**- Regularization adds a penalty term to different parameters of the machine learning model for avoiding overfitting in the model. This penalty term is added to the coefficients; hence it shrinks some coefficients to zero. Those features with zero coefficients can be removed from the dataset. The types of regularization techniques are L1 Regularization (Lasso Regularization) or Elastic Nets (L1 and L2 regularization).
- **Random Forest Importance** - Different tree-based methods of feature selection help us with feature importance to provide a way of selecting features. Here, feature importance specifies which feature has more importance in model building or has a great impact on the target variable. Random Forest is such a tree-based method, which is a type of bagging algorithm that aggregates a different number of decision trees. It automatically ranks the nodes by their performance or decrease in the impurity (Gini impurity) over all the trees. Nodes are arranged as per the impurity values, and thus it allows to pruning of trees below a specific node. The remaining nodes create a subset of the most important features.

How to choose a Feature Selection Method?

For machine learning engineers, it is very important to understand that which feature selection method will work properly for their model. The more we know the datatypes of variables, the easier it is to choose the appropriate statistical measure for feature selection.



To know this, we need to first identify the type of input and output variables. In machine learning, variables are of mainly two types:

- **Numerical Variables:** Variable with continuous values such as integer, float
- **Categorical Variables:** Variables with categorical values such as Boolean, ordinal, nominals.

Below are some univariate statistical measures, which can be used for filter-based feature selection:

1. Numerical Input, Numerical Output:

Numerical Input variables are used for predictive regression modelling. The common method to be used for such a case is the Correlation coefficient.

- Pearson's correlation coefficient (For linear Correlation).
- Spearman's rank coefficient (for non-linear correlation).

2. Numerical Input, Categorical Output:

Numerical Input with categorical output is the case for classification predictive modelling problem**s.** In this case, also, correlation-based techniques should be used, but with categorical output.

- **ANOVA correlation coefficient (linear).**
- **Kendall's rank coefficient (nonlinear).**

3. Categorical Input, Numerical Output:

This is the case of regression predictive modelling with categorical input. It is a different example of a regression problem. We can use the same measures as discussed in the above case but in reverse order.

4. Categorical Input, Categorical Output:

This is a case of classification predictive modelling with categorical Input variables.

The commonly used technique for such a case is Chi-Squared Test. We can also use Information gain in this case.

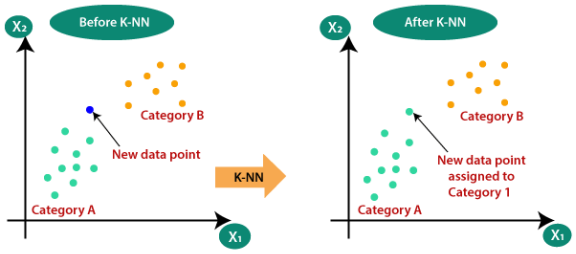
We can summarise the above cases with appropriate measures in the below table:

Input Variable	**Output Variable**	**Feature Selection technique**
Numerical	Numerical	○ Pearson's correlation coefficient (For linear Correlation). ○ Spearman's rank coefficient (for non-linear correlation).
Numerical	Categorical	○ ANOVA correlation coefficient (linear). ○ Kendall's rank coefficient (nonlinear).
Categorical	Numerical	○ Kendall's rank coefficient (linear). ○ ANOVA correlation coefficient (nonlinear).
Categorical	Categorical	○ Chi-Squared test (contingency tables). ○ Mutual Information.

K-Nearest neighbor (KNN) Algorithm:

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

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



How does K-NN work?

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

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

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

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

Step-6: Our model is ready.

Generating KNN Regressor Model:

```
from sklearn.neighbors import KNeighborsRegressor
scores=[]
for i in range(1,10):
  model = KNeighborsRegressor(n_neighbors=i)
  model.fit(x_train, y_train)
  scores.append(model.score(x_test,y_test))


k = scores.index(max(scores))
model = KNeighborsRegressor(n_neighbors=k)
model.fit(x_train, y_train)
print("Train Score",model.score(x_train,y_train))
print("Test Score",model.score(x_test,y_test))
```

```
Train Score 0.9556178882661865
Test Score 0.9276949091790976
```

Bayesian Classification:

- Bayesian classifiers are statistical classifiers.
- They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class.
- Bayesian classification is based on Bayes' theorem.

Bayes' Theorem:

- Let X be a data tuple. In Bayesian terms, X is considered — "evidence" and it is described by measurements made on a set of n attributes.
- Let H be some hypothesis, such as that the data tuple X belongs to a specified class C.

- For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the ―evidence‖ or observed data tuple X.
- P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X.
- Bayes' theorem is useful in that it provides a way of calculating the posterior probability, P(H|X), from P(H), P(X|H), and P(X).

$$\boldsymbol{P(H|X) = \frac{P(X|H)P(H)}{P(X)}}$$

Naïve Bayesian Classification:

The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let=be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x1, x2, …,xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2, …, An.
2. Suppose that there are m classes, C1, C2, …, Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naïve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if

$$\boldsymbol{P(C_i|X) > P(C_j|X) \quad for \quad 1 \leq j \leq m, j \neq i.}$$

Thus we maximize$P(C_j|X)$. The class Ci for which $P(C_j|X)$. is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$\boldsymbol{P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}}$$

3. As P(X) is constant for all classes, only P(X|Ci)P(Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = …= P(Cm), and we would therefore maximize P(X|Ci). Otherwise, we maximize P(X|Ci)P(Ci).
4. Given data sets with many attributes, it would be extremely computationally expensive to compute P(X|Ci). In order to reduce computation in evaluating P(X|Ci), the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple. Thus,

$$\boldsymbol{P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)}$$
$$\boldsymbol{= P(x_1|C_1) \times P(x_2|C_2) \times \ldots\ldots\times P(x_n|C_i)}$$

5. We can easily estimate the probabilities P(x1|Ci), P(x2|Ci), : : : , P(xn|Ci) from the training tuples.
6. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute P(X|Ci), we consider the following:

- If A_k is categorical, then $P(x_k|Ci)$ is the number of tuples of class Ci in=havingthe value x_k for A_k, divided by |Ci ,D| the number of tuples of class Ci in D.
- If A_k is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward.

Example:

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

We wish to predict the class label of a tuple using naïve Bayesian classification, given the same training data above. The training data were shown above in Table. The data tuples are described by the attributes *age*, *income*, *student*, and *credit rating*. The class label attribute, *buys computer*, has two distinct values (namely, {yes, no}). Let *C*1 correspond to the class *buys computer=yes* and *C*2 correspond to b*uys computer=no*. The tuple we wish to classify is

X={age= "youth", income= "medium", student= "yes", credit_rating= "fair"}

We need to maximize P(X|Ci)P(Ci), for i=1,2. P(Ci), the prior probability of each class, can be computed based on the training tuples:

P(*buys computer = yes*) = 9/14 = 0.643
P(*buys computer = no*) = 5/14 = 0.357

To compute P(X|Ci), for *i* = 1, 2, we compute the following conditional probabilities:

P(*age = youth | buys computer = yes*) = 2/9 = 0.222
P(*income=medium | buys computer=yes*) = 4/9 = 0.444
P(*student=yes | buys computer=yes*) = 6/9 = 0.667
P(*credit rating=fair | buys computer=yes*) = 6/9 = 0.667

P(*age=youth | buys computer=no*) = 3/5 = 0.600
P(*income=medium | buys computer=no*) = 2/5 = 0.400
P(*student=yes | buys computer=no*) = 1/5 = 0.200
P(*credit rating=fair | buys computer=no*) = 2/5 = 0.400

Using these probabilities, we obtain

$$P(\boldsymbol{X} \mid buys\ computer=yes) = P(age=youth \mid buys\ computer=yes) \times P(income=medium \mid buys\ computer=yes) \times P(student=yes \mid buys\ computer=yes) \times P(credit\ rating=fair \mid buys\ computer=yes) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044.$$

Similarly,

$$P(\boldsymbol{X} \mid buys\ computer=no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

To find the class, C_i, that P(X|Ci)P(Ci), we compute

$$P(\boldsymbol{X} \mid buys\ computer=yes)\ P(buys\ computer=yes) = 0.044 \times 0.643 = 0.028$$

$$P(\boldsymbol{X} \mid buys\ computer=no)\ P(buys\ computer=no) = 0.019 \times 0.357 = 0.007$$

Therefore, the naïve Bayesian classifier predicts *buys computer = yes* for tuple ***X***.

Types of Naïve Bayes Model:

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

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

Fitting Naive Bayes to the Training Set:

After the pre-processing step, now we will fit the Naive Bayes model to the Training set. Below is the code for it:

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

In the above code, we have used the GaussianNB classifier to fit it to the training dataset. We can also use other classifiers as per our requirement.

Gradient Descent:

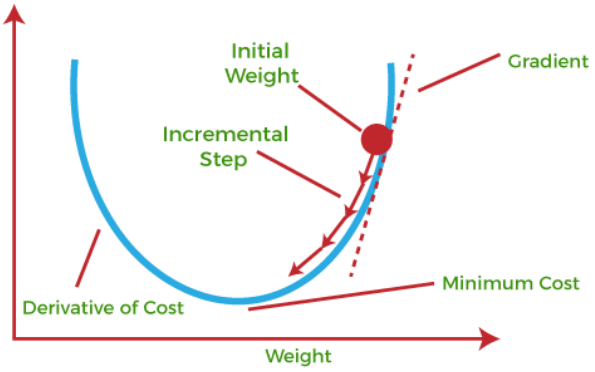
Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results. Further, gradient descent is also used to train Neural Networks.

In mathematical terminology, Optimization algorithm refers to the task of minimizing/maximizing an objective function f(x) parameterized by x. Similarly, in machine learning, optimization is the task of minimizing the cost function parameterized by the model's parameters. The main objective of gradient descent is to minimize the convex function using iteration of parameter updates. Once these machine learning models are optimized, these models can be used as powerful tools for Artificial Intelligence and various computer science applications.

Gradient descent was initially discovered by "Augustin-Louis Cauchy" in mid of 18th century. Gradient Descent is defined as one of the most commonly used iterative optimization algorithms of machine learning to train the machine learning and deep learning models. It helps in finding the local minimum of a function.

The best way to define the local minimum or local maximum of a function using gradient descent is as follows:

- If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the ***local minimum*** of that function.
- Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the ***local maximum*** of that function.



This entire procedure is known as Gradient Ascent, which is also known as steepest descent. The main objective of using a gradient descent algorithm is to minimize the cost function using iteration. To achieve this goal, it performs two steps iteratively:

- Calculates the first-order derivative of the function to compute the gradient or slope of that function.
- Move away from the direction of the gradient, which means slope increased from the current point by alpha times, where Alpha is defined as Learning Rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.

The ***cost function*** is defined as the measurement of difference or error between actual values and expected values at the current position and present in the form of a single real number.

Types of Gradient Descent

Based on the error in various training models, the Gradient Descent learning algorithm can be divided into Batch gradient descent, stochastic gradient descent, and mini-batch gradient descent. Let's understand these different types of gradient descent:

1. Batch Gradient Descent:

Batch gradient descent (BGD) is used to find the error for each point in the training set and update the model after evaluating all training examples. This procedure is known as the training epoch. In simple words, it is a greedy approach where we have to sum over all examples for each update.

Advantages of Batch gradient descent:

- It produces less noise in comparison to other gradient descent.
- It produces stable gradient descent convergence.
- It is Computationally efficient as all resources are used for all training samples.

2. Stochastic gradient descent

Stochastic gradient descent (SGD) is a type of gradient descent that runs one training example per iteration. Or in other words, it processes a training epoch for each example within a dataset and updates each training example's parameters one at a time. As it requires only one training example at a time, hence it is easier to store in allocated memory. However, it shows some computational efficiency losses in comparison to batch gradient systems as it shows frequent updates that require more detail and speed. Further, due to frequent updates, it is also treated as a noisy gradient. However, sometimes it can be helpful in finding the global minimum and also escaping the local minimum.

Advantages of Stochastic gradient descent:

In Stochastic gradient descent (SGD), learning happens on every example, and it consists of a few advantages over other gradient descent.

- It is easier to allocate in desired memory.
- It is relatively fast to compute than batch gradient descent.
- It is more efficient for large datasets.

3. MiniBatch Gradient Descent:

Mini Batch gradient descent is the combination of both batch gradient descent and stochastic gradient descent. It divides the training datasets into small batch sizes then performs the updates on those batches separately. Splitting training datasets into smaller batches make a balance to maintain the computational efficiency of batch gradient descent and speed of stochastic gradient descent. Hence, we can achieve a special type of gradient descent with higher computational efficiency and less noisy gradient descent.

Advantages of Mini Batch gradient descent:

- It is easier to fit in allocated memory.
- It is computationally efficient.
- It produces stable gradient descent convergence.