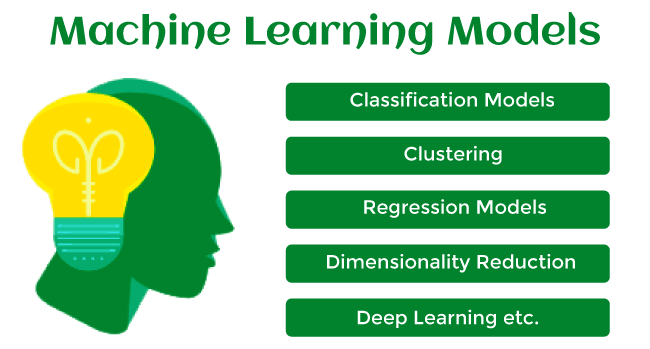
**UNIT V: Machine learning models**

**SYLLABUS:** Machine Learning Models-Types, Calculating accuracy of Regression and Classification Models, Traning Validation and Testing Data, Overfitting and Underfitting, Regularization, Hyperparameter.

Supervised Models: Linear, KNN, Naïve-Bayes.

Machine Learning Models and Types

***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.



Let's understand an example of the ML model where we are creating an app to recognize the user's emotions based on facial expressions. So, creating such an app is possible by Machine learning models where we will train a model by feeding images of faces with various emotions labeled on them. Whenever this app is used to determine the user's mood, it reads all fed data then determines any user's mood.

Hence, in simple words, we can say that a *machine learning model is a simplified representation of something or a process.* In this topic, we will discuss different machine learning models and their techniques and algorithms*.*

What is Machine Learning Model?

Machine Learning models can be understood as a program that has been trained to find patterns within new data and make predictions. These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response. First, these models are trained over a set of data, and then they are provided an algorithm to reason over data, extract the pattern from feed data and learn from those data. Once these models get trained, they can be used to predict the unseen dataset.

30.3K

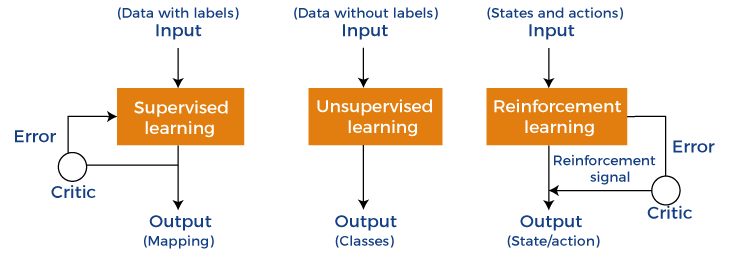
Machine Learning - Preprocessing Structured Data - One Hot Encoding

There are various types of machine learning models available based on different business goals and data sets.

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:

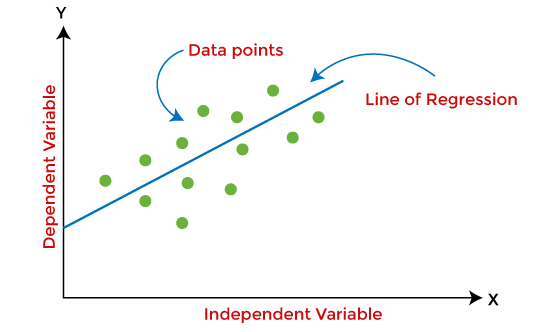
Regression

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**

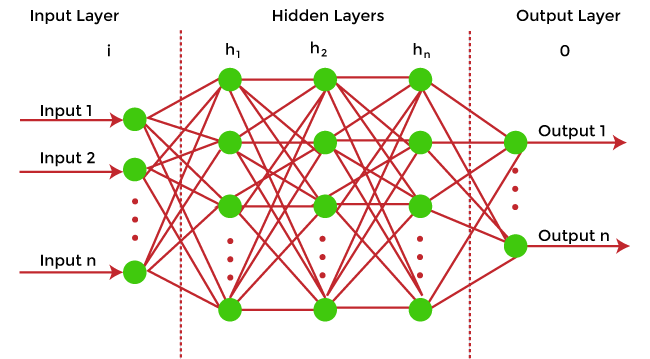
Random Forest is the ensemble learning method, which consists of a large number of decision trees. Each decision tree in a random forest predicts an outcome, and the prediction with the majority of votes is considered as the outcome.

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.

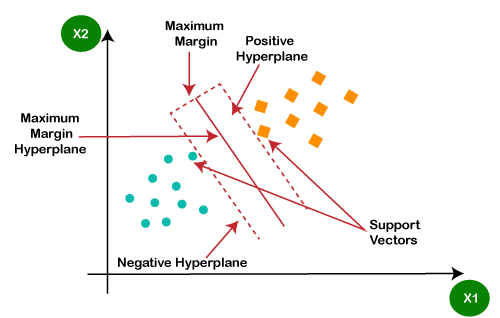
Some popular classification algorithms are as below:

**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.

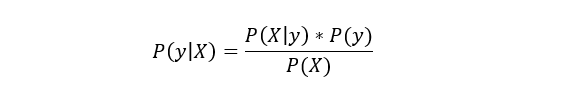
**b) Support Vector Machine**

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:

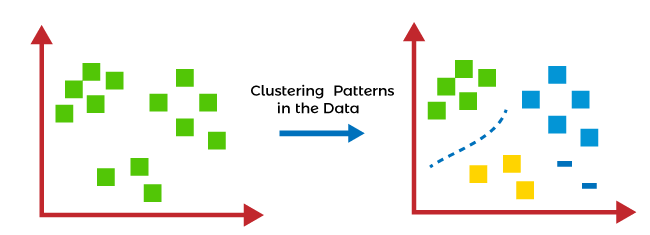


Each naïve Bayes classifier assumes that the value of a specific variable is independent of any other variable/feature. For example, if a fruit needs to be classified based on color, shape, and taste. So yellow, oval, and sweet will be recognized as mango. Here each feature is independent of other features.

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.***

3. 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.

Training Machine Learning Models

Once the Machine learning model is built, it is trained in order to get the appropriate results. To train a machine learning model, one needs a huge amount of pre-processed data. Here pre-processed data means data in structured form with reduced null values, etc. If we do not provide pre-processed data, then there are huge chances that our model may perform terribly.

How to choose the best model?

In the above section, we have discussed different machine learning models and algorithms. But one most confusing question that may arise to any beginner that "which model should I choose?". So, the answer is that it depends mainly on the business requirement or project requirement. Apart from this, it also depends on associated attributes, the volume of the available dataset, the number of features, complexity, etc. However, in practice, it is recommended that we always start with the simplest model that can be applied to the particular problem and then gradually enhance the complexity & test the accuracy with the help of parameter tuning and cross-validation.

Difference between Machine learning model and Algorithms

One of the most confusing questions among beginners is that are machine learning models, and algorithms are the same? Because in various cases in machine learning and data science, these two terms are used interchangeably.

The answer to this question is No, and the machine learning model is not the same as an algorithm. In a simple way, an **ML algorithm is like a procedure or method that runs on data to discover patterns from it** and generate the model. At the same time, a **machine learning model is like a computer program that generates output or makes predictions**. More specifically, when we train an algorithm with data, it becomes a model.

1. Machine Learning ModelModel = Model Data + Prediction Algorithm

**Calculating accuracy of Regression and Classification Models:**

**Calculating accuracy of Regression**

Regression, you might argue, is one of the most basic statistical approach to build predictive models.

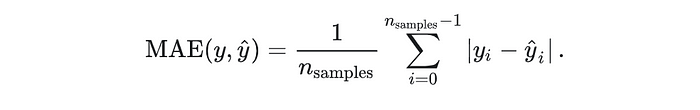
Quantifying the accuracy of a model is an important step to justifying the usage of the model.

The metrics we want to look at are:

1. Mean Absolute Error (MAE)
2. Root Mean Squared Error (RMSE)
3. Mean Absolute Percentage Error (MAPE)
4. R-Squared Score

**Mean Absolute Error (MAE)**

***Definition:***MAE is the average value of error in a set of predicted values, without considering direction. It ranges from 0 to inf., and lower value means better model. It is the simplest to understand regression error metric and is mathematically described as:



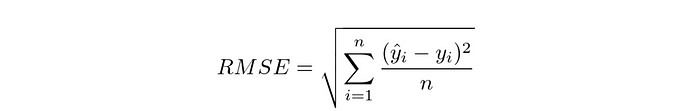
***Intuition:*** It is a natural measure of average error.

***Important properties:***

1. MAE tells us how big of an error we can expect from the forecast on average.
2. One problem with the MAE is that the relative size of the error is not always obvious. When comparing only MAE of two models it is hard to tell which model has a big error vs multiple small error. (unless you have a small dataset that you can quickly eye ball)

**Root Mean Squared Error (RMSE)**

***Definition:***RMSE is the square root of average value of squared error in a set of predicted values, without considering direction. It ranges from 0 to inf., lower means better model and it is always greater in magnitude than MAE. It is most widely used regression error metric and is mathematically described as:



***Intuition:*** It measures of how wide your residuals are spread out.

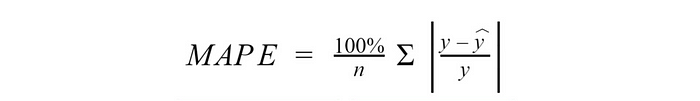
***Important properties:***RMSE can be interpreted as the standard deviation of the unexplained variance. It has several properties worth noting:

* It has the same units as the predicted variable.
* Because of square term it gives a relatively high weight to the outliers or large errors. Hence if you are optimizing your model to reduce RMSE, you are reducing your large error.
* RMSE has following relationship with MAE.



**Mean Absolute Percentage Error (MAPE)**

***Definition:***MAPE is the average percentage error in a set of predicted values, without considering direction. It ranges from 0 to inf., and lower value means better model. It is mathematically described as:



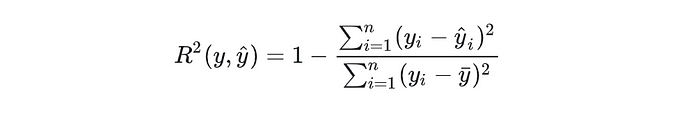
***Intuition:*** It measures the percentage error, on average.

***Important properties:***

* It doesn’t depend on the units of data. It is expressed as percentage.
* Given it is expressed as percentage, it is easier to understand by non-technical staff.
* MAPE can be more than 100%. Thus defining accuracy as (100% — MAPE) is not right. It leads to negative accuracy which is hard to interpret.
* It fails if some of the actual values are equal to zero.
* If any true values are very close to zero, the corresponding absolute percentage errors will be extremely high and therefore bias the MAPE.

**R-Squared/Adjusted R-Squared Score**

*Definition:*It is the coefficient of determination which is proportion of the variance in the dependent variable that is predictable from the independent variable(s). It is mathematically described as:





***Intuition:*** It determines how well the model captures the variance in data.

***Important properties:***

* Its scale is intuitive: it ranges from 0 to 1. Zero indicating that the proposed model does not improve prediction over the mean model, and One indicating perfect prediction. Better the model, higher the R-squared.
* Adjusted R-squared, incorporates the model’s degrees of freedom, and eliminates the problem of artificial increase in R-Squared with addition of more variables.
* R-Squared can be negative. Negative R-Squared means that chosen model does not follow the trend of the data, so fits worse than a horizontal line.

**Calculating accuracy of Classification Models:**

Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

Accuracy=Number of correct predictions/Total number of predictions

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

Accuracy=TP+ TN /TP+FP+FN+TN

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

Let's try calculating accuracy for the following model that classified 100 tumors as [malignant](https://wikipedia.org/wiki/Malignancy) (the positive class) or [benign](https://wikipedia.org/wiki/Benign_tumor) (the negative class):

|  |  |
| --- | --- |
| True Positive (TP):   * Reality: Malignant * ML model predicted: Malignant * Number of TP results: 1 | False Positive (FP):   * Reality: Benign * ML model predicted: Malignant * Number of FP results: 1 |
| False Negative (FN):   * Reality: Malignant * ML model predicted: Benign * Number of FN results: 8 | True Negative (TN):   * Reality: Benign * ML model predicted: Benign * Number of TN results: 90 |

Accuracy=1+90/1+90+1+8=0.91

Accuracy comes out to 0.91, or 91% (91 correct predictions out of 100 total examples). That means our tumor classifier is doing a great job of identifying malignancies, right?

Actually, let's do a closer analysis of positives and negatives to gain more insight into our model's performance.

Of the 100 tumor examples, 91 are benign (90 TNs and 1 FP) and 9 are malignant (1 TP and 8 FNs).

Of the 91 benign tumors, the model correctly identifies 90 as benign. That's good. However, of the 9 malignant tumors, the model only correctly identifies 1 as malignant—a terrible outcome, as 8 out of 9 malignancies go undiagnosed!

While 91% accuracy may seem good at first glance, another tumor-classifier model that always predicts benign would achieve the exact same accuracy (91/100 correct predictions) on our examples. In other words, our model is no better than one that has zero predictive ability to distinguish malignant tumors from benign tumors.

Accuracy alone doesn't tell the full story when you're working with a **class-imbalanced data set**, like this one, where there is a significant disparity between the number of positive and negative labels.

**Precision**

What proportion of positive identifications was actually correct?

Precision is defined as follows:

Precision=TP/TP+FP

**Note:** A model that produces no false positives has a precision of 1.0.

Let's calculate precision for our ML model from the [previous section](https://developers.google.com/machine-learning/crash-course/classification/accuracy) that analyzes tumors:

|  |  |
| --- | --- |
| True Positives (TPs): 1 | False Positives (FPs): 1 |
| False Negatives (FNs): 8 | True Negatives (TNs): 90 |

Precision=1/1+1=0.5

Our model has a precision of 0.5—in other words, when it predicts a tumor is malignant, it is correct 50% of the time.

**Recall**

What proportion of actual positives was identified correctly?

Mathematically, recall is defined as follows:

Recall=TP/TP+FN

**Note:** A model that produces no false negatives has a recall of 1.0.

Let's calculate recall for our tumor classifier:

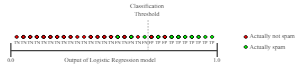
|  |  |
| --- | --- |
| True Positives (TPs): 1 | False Positives (FPs): 1 |
| False Negatives (FNs): 8 | True Negatives (TNs): 90 |

Recall=1/1+8=0.11

Our model has a recall of 0.11—in other words, it correctly identifies 11% of all malignant tumors.

**Precision and Recall: A Tug of War**

To fully evaluate the effectiveness of a model, you must examine **both** precision and recall. Unfortunately, precision and recall are often in tension. That is, improving precision typically reduces recall and vice versa. Explore this notion by looking at the following figure, which shows 30 predictions made by an email classification model. Those to the right of the classification threshold are classified as "spam", while those to the left are classified as "not spam."



**Figure 1. Classifying email messages as spam or not spam.**

Let's calculate precision and recall based on the results shown in Figure 1:

|  |  |
| --- | --- |
| True Positives (TP): 8 | False Positives (FP): 2 |
| False Negatives (FN): 3 | True Negatives (TN): 17 |

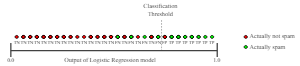
Precision measures the percentage of **emails flagged as spam** that were correctly classified—that is, the percentage of dots to the right of the threshold line that are green in Figure 1:

Precision=8/8+2=0.8

Recall measures the percentage of **actual spam emails** that were correctly classified—that is, the percentage of green dots that are to the right of the threshold line in Figure 1:

Recall=8/8+3=0.73

Figure 2 illustrates the effect of increasing the classification threshold.



**Figure 2. Increasing classification threshold.**

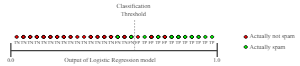
The number of false positives decreases, but false negatives increase. As a result, precision increases, while recall decreases:

|  |  |
| --- | --- |
| True Positives (TP): 7 | False Positives (FP): 1 |
| False Negatives (FN): 4 | True Negatives (TN): 18 |

Precision=7/7+1=0.88

Recall=7/7+4=0.64

Conversely, Figure 3 illustrates the effect of decreasing the classification threshold (from its original position in Figure 1).



**Figure 3. Decreasing classification threshold.**

False positives increase, and false negatives decrease. As a result, this time, precision decreases and recall increases:

|  |  |
| --- | --- |
| True Positives (TP): 9 | False Positives (FP): 3 |
| False Negatives (FN): 2 | True Negatives (TN): 16 |

Precision=9/9+3=0.75

Recall=9/9+2=0.82

Various metrics have been developed that rely on both precision and recall. For example, see [F1 score](https://wikipedia.org/wiki/F1_score).

**F1 Score**

The [F1 Score](https://en.wikipedia.org/wiki/F1_score) is the 2\*((precision\*recall)/(precision+recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.

* The F1 for the All No Recurrence model is 2\*((0\*0)/0+0) or 0.
* The F1 for the All Recurrence model is 2\*((0.3\*1)/0.3+1) or 0.46.
* The F1 for the CART model is 2\*((0.43\*0.12)/0.43+0.12) or 0.19.

**Traning Validation and Testing Data:**

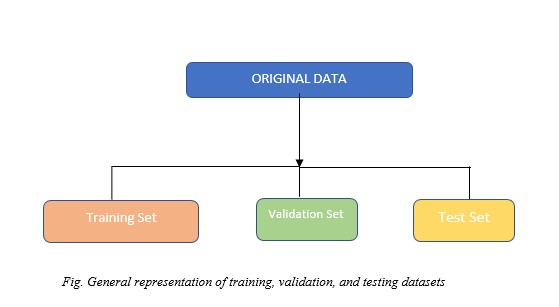
Data splitting is one of the simplest preprocessing techniques we can use in a Machine Learning/Deep Learning task. The original dataset is split into subsets like training, test, and validation sets. One of the prime reasons this is done is to tackle the problem of overfitting. However, there are other benefits as well. Let's have a brief understanding of these terms and see how they are useful.

## Training Set

The training set is used to fit or train the model. These data points are used to learn the parameters of the model. This is the biggest of all sets in terms of size. The training set includes the features and well as labels in the case of supervised learning. In the case of unsupervised learning, it can simply be the feature sets. These labels are used in the training phase to get the training accuracy score. The training set is usually taken as 70% of the original dataset but can be changed per the use case or available data.

*For example*

* While using Linear Regression, the points in the training set are used to draw the line of best fit.
* In K-Nearest Neighbors, the points in the training set are the points that could be the neighbors.



### Applications of Train Set

Training sets are used in supervised learning procedures in data mining (i.e., classification of records or prediction of continuous target values.)

#### **Example**

Let’s consider a dataset containing 20 points

Dataset1 = [1,5,6,7,8,6,4,5,6,7,23,45,12,34,45,1,7,7,8,0]

Train set can be taken as 60 % of the original Dataset1

The train set will contain 12 data points [8,6,4,5,6,7,23,45,12,34,1,5]

## Validation Set

The validation set is used to provide an unbiased evaluation of the model fit during hyperparameter tuning of the model. It is the set of examples that are used to change learning process parameters. Optimal values of hyperparameters are tested against the model trained using the training set. In Machine Learning or Deep Learning, we generally need to test multiple models with different hyperparameters and check which model gives the best result. This process is carried out with the help of a validation set.

For example, in deep LSTM networks, a validation set is used to find the number of hidden layers, number of nodes, Dense units, etc.

### Applications of Validation Set

Validations sets are used for Hyperparameter tuning of AI models. Domains include Healthcare, Analytics, Cyber Security, etc.

#### **Example**

Let’s consider a dataset containing 20 points

Dataset2 = [1,5,6,7,8,6,4,5,6,7,23,45,12,34,45,1,7,7,8,0]

The validation set can be taken as 20 % of the original Dataset2.

The validation set will contain 4 data points [45,1,7,7]

## Testing Set

Once we have the model trained with the training set and the hyperparameter tuned using the validation set, we need to test whether the model can generalize well on unseen data. To accomplish this, a test set is used. Here we can check and compare the training and test accuracies. To ensure that the model is not overfitting or underfitting, test accuracies are highly useful. If there is a large difference in train and test accuracies, overfitting might have occurred.

While choosing the test set the below points should be kept in mind:

* The test should contain the same characteristics as of the train set.
* It should be large enough to yield statistically significant results

### Applications of Test Set

Test sets are used for evaluating metrics like:

Precision, Recall, AUC - ROC Curve, F1-Score

#### **Example**

Let's consider a data set containing 20 points

Dataset3 = [1,5,6,7,8,6,4,5,6,7,23,45,12,34,45,1,7,7,8,0]

The test set can be taken as 20 % of the original Dataset2

The test set will contain 4 data points [6,7,8,0]

### Why do we need a train, validation, and test sets?

The training set is necessary to train the model and learn the parameters. Almost all Machine learning/Deep Learning tasks should contain at least a training set.

The validation set and test sets are optional but highly recommended to use because only then can a trained model's legibility and accuracy can be verified. The validation set can be omitted if we do not choose to perform hyperparameter tuning or model selection. In such cases, a train set and test set will do the job.

A smart way to evaluate a model is to use K-Fold cross-validation.

The below table summarizes Training, Validation, and Testing sets.

|  |  |  |
| --- | --- | --- |
| **Training Set** | **Validation Set** | **Testing Set** |
| It is used to fit the model to learn the parameters of the model | It is used to provide an unbiased evaluation of the model fit during hyperparameter tuning of the model | It is used to test whether the model can generalize well on unseen data. |
| Larger in size as compared to validation and test sets | Smaller in size. | Smaller in size as compared to the train set. |
| In the case of supervised learning, it comprises features and labels. In unsupervised learning, it includes only features | Contains both features and labels in supervised learning and only features in supervised learning | Contains both features and labels in supervised learning and only features in supervised learning |
| Slower on larger datasets but the job can be run in parallel using multiprocessing | Usually slower on a single core, if hyperparameters under observation are large. Can be run in parallel. | Faster than both train and validation sets. Used to get the metrics on test data based on the trained model |

**Overfitting and Underfitting:**

Overfitting and Underfitting are the two main problems that occur in machine learning and degrade the performance of the machine learning models.

The main goal of each machine learning model is **to generalize well**. Here **generalization** defines the ability of an ML model to provide a suitable output by adapting the given set of unknown input. It means after providing training on the dataset, it can produce reliable and accurate output. Hence, the underfitting and overfitting are the two terms that need to be checked for the performance of the model and whether the model is generalizing well or not.

Before understanding the overfitting and underfitting, let's understand some basic term that will help to understand this topic well:

* **Signal:** It refers to the true underlying pattern of the data that helps the machine learning model to learn from the data.
* **Noise:** Noise is unnecessary and irrelevant data that reduces the performance of the model.
* **Bias:** Bias is a prediction error that is introduced in the model due to oversimplifying the machine learning algorithms. Or it is the difference between the predicted values and the actual values.
* **Variance:** If the machine learning model performs well with the training dataset, but does not perform well with the test dataset, then variance occurs.

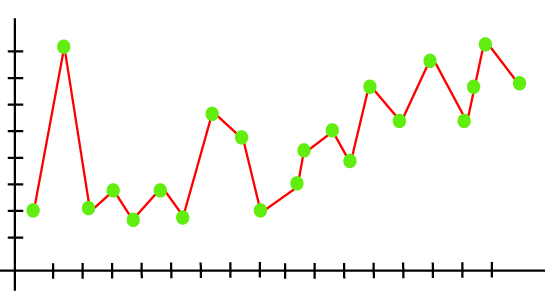
Overfitting

Overfitting occurs when our [machine learning](https://www.javatpoint.com/machine-learning) model tries to cover all the data points or more than the required data points present in the given dataset. Because of this, the model starts caching noise and inaccurate values present in the dataset, and all these factors reduce the efficiency and accuracy of the model. The overfitted model has **low bias** and **high variance.**

The chances of occurrence of overfitting increase as much we provide training to our model. It means the more we train our model, the more chances of occurring the overfitted model.

Overfitting is the main problem that occurs in [supervised learning](https://www.javatpoint.com/supervised-machine-learning).

**Example:** The concept of the overfitting can be understood by the below graph of the linear regression output:



As we can see from the above graph, the model tries to cover all the data points present in the scatter plot. It may look efficient, but in reality, it is not so. Because the goal of the regression model to find the best fit line, but here we have not got any best fit, so, it will generate the prediction errors.

How to avoid the Overfitting in Model

Both overfitting and underfitting cause the degraded performance of the machine learning model. But the main cause is overfitting, so there are some ways by which we can reduce the occurrence of overfitting in our model.

* **Cross-Validation**
* **Training with more data**
* **Removing features**
* **Early stopping the training**
* **Regularization**
* **Ensembling**

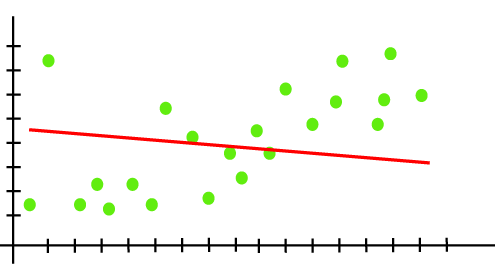
Underfitting

Underfitting occurs when our machine learning model is not able to capture the underlying trend of the data. To avoid the overfitting in the model, the fed of training data can be stopped at an early stage, due to which the model may not learn enough from the training data. As a result, it may fail to find the best fit of the dominant trend in the data.

In the case of underfitting, the model is not able to learn enough from the training data, and hence it reduces the accuracy and produces unreliable predictions.

An underfitted model has high bias and low variance.

**Example:** We can understand the underfitting using below output of the linear regression model:



As we can see from the above diagram, the model is unable to capture the data points present in the plot.

How to avoid underfitting:

* By increasing the training time of the model.
* By increasing the number of features.

Goodness of Fit

The "Goodness of fit" term is taken from the statistics, and the goal of the machine learning models to achieve the goodness of fit. In statistics modeling, *it defines how closely the result or predicted values match the true values of the dataset.*

The model with a good fit is between the underfitted and overfitted model, and ideally, it makes predictions with 0 errors, but in practice, it is difficult to achieve it.

As when we train our model for a time, the errors in the training data go down, and the same happens with test data. But if we train the model for a long duration, then the performance of the model may decrease due to the overfitting, as the model also learn the noise present in the dataset. The errors in the test dataset start increasing, *so the point, just before the raising of errors, is the good point, and we can stop here for achieving a good model.*

There are two other methods by which we can get a good point for our model, which are the **resampling method** to estimate model accuracy and **validation dataset**.

Regularization

Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it.

Sometimes the [machine learning](https://www.javatpoint.com/machine-learning) model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.

This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.

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

How does Regularization Work?

Regularization works by adding a penalty or complexity term to the complex model. Let's consider the simple linear regression equation:

y= β0+β1x1+β2x2+β3x3+⋯+βnxn +b

In the above equation, Y represents the value to be predicted

X1, X2, …Xn are the features for Y.

β0,β1,…..βn are the weights or magnitude attached to the features, respectively. Here represents the bias of the model, and b represents the intercept.

Linear regression models try to optimize the β0 and b to minimize the cost function. The equation for the cost function for the linear model is given below:

Regularization in Machine Learning

Now, we will add a loss function and optimize parameter to make the model that can predict the accurate value of Y. The loss function for the linear regression is called as **RSS or Residual sum of squares.**

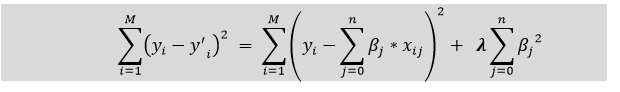
Techniques of Regularization

There are mainly two types of regularization techniques, which are given below:

* **Ridge Regression**
* **Lasso Regression**

Ridge Regression

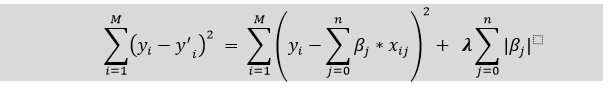
* Ridge regression is one of the types of linear regression in which a small amount of bias is introduced so that we can get better long-term predictions.
* Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as **L2 regularization**.
* In this technique, the cost function is altered by adding the penalty term to it. The amount of bias added to the model is called **Ridge Regression penalty**. We can calculate it by multiplying with the lambda to the squared weight of each individual feature.
* The equation for the cost function in ridge regression will be:



* In the above equation, the penalty term regularizes the coefficients of the model, and hence ridge regression reduces the amplitudes of the coefficients that decreases the complexity of the model.
* As we can see from the above equation, if the values of **λ tend to zero, the equation becomes the cost function of the linear regression model.** Hence, for the minimum value of λ, the model will resemble the linear regression model.
* A general linear or polynomial regression will fail if there is high collinearity between the independent variables, so to solve such problems, Ridge regression can be used.
* It helps to solve the problems if we have more parameters than samples.

Lasso Regression:

* Lasso regression is another regularization technique to reduce the complexity of the model. It stands for **Least Absolute and Selection Operator.**
* It is similar to the Ridge Regression except that the penalty term contains only the absolute weights instead of a square of weights.
* Since it takes absolute values, hence, it can shrink the slope to 0, whereas Ridge Regression can only shrink it near to 0.
* It is also called as **L1 regularization.** The equation for the cost function of Lasso regression will be:



* Some of the features in this technique are completely neglected for model evaluation.
* Hence, the Lasso regression can help us to reduce the overfitting in the model as well as the feature selection.

Key Difference between Ridge Regression and Lasso Regression

* **Ridge regression** is mostly used to reduce the overfitting in the model, and it includes all the features present in the model. It reduces the complexity of the model by shrinking the coefficients.
* **Lasso regression** helps to reduce the overfitting in the model as well as feature selection.

**Hyperparameters**

**Hyperparameters in Machine learning are those parameters that are explicitly defined by the user to control the learning process.** These hyperparameters are used to improve the learning of the model, and their values are set before starting the learning process of the model.

## What are hyperparameters?

In Machine Learning/Deep Learning, a model is represented by its parameters. In contrast, a training process involves selecting the best/optimal hyperparameters that are used by learning algorithms to provide the best result. So, what are these hyperparameters? The answer is, "**Hyperparameters are defined as the parameters that are explicitly defined by the user to control the learning process."**

Here the prefix "hyper" suggests that the parameters are top-level parameters that are used in controlling the learning process. The value of the Hyperparameter is selected and set by the machine learning engineer before the learning algorithm begins training the model. **Hence, these are external to the model, and their values cannot be changed during the training process**.

### Some examples of Hyperparameters in Machine Learning

* The k in kNN or K-Nearest Neighbour algorithm
* Learning rate for training a neural network
* Train-test split ratio
* Batch Size
* Number of Epochs
* Branches in Decision Tree
* Number of clusters in Clustering Algorithm

## Difference between Parameter and Hyperparameter?

There is always a big confusion between Parameters and hyperparameters or model hyperparameters. So, in order to clear this confusion, let's understand the difference between both of them and how they are related to each other.

### Model Parameters:

Model parameters are configuration variables that are internal to the model, and a model learns them on its own. For example**, W Weights or Coefficients of independent variables in the Linear regression model**. or **Weights or Coefficients of independent variables in SVM, weight, and biases of a neural network, cluster centroid in clustering.** Some key points for model parameters are as follows:

* They are used by the model for making predictions.
* They are learned by the model from the data itself
* These are usually not set manually.
* These are the part of the model and key to a machine learning Algorithm.

### Model Hyperparameters:

Hyperparameters are those parameters that are explicitly defined by the user to control the learning process. Some key points for model parameters are as follows:

* These are usually defined manually by the machine learning engineer.
* One cannot know the exact best value for hyperparameters for the given problem. The best value can be determined either by the rule of thumb or by trial and error.
* Some examples of Hyperparameters are **the learning rate for training a neural network, K in the KNN algorithm,**

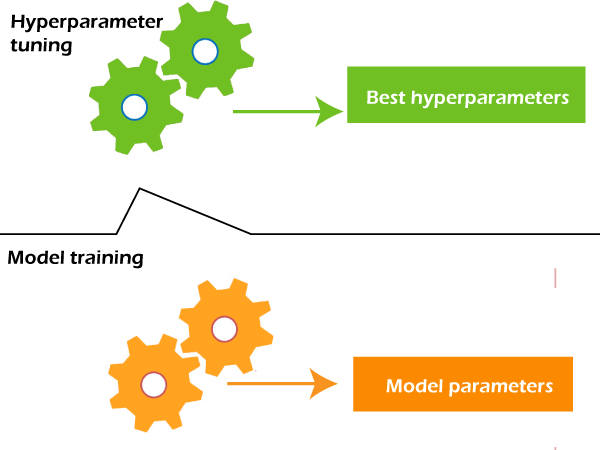
## Categories of Hyperparameters

Broadly hyperparameters can be divided into two categories, which are given below:

1. **Hyperparameter for Optimization**
2. **Hyperparameter for Specific Models**

### Hyperparameter for Optimization

The process of selecting the best hyperparameters to use is known as hyperparameter tuning, and the tuning process is also known as hyperparameter optimization. Optimization parameters are used for optimizing the model.



Some of the popular optimization parameters are given below:

* **Learning Rate:** The learning rate is the hyperparameter in optimization algorithms that controls how much the model needs to change in response to the estimated error for each time when the model's weights are updated. It is one of the crucial parameters while building a neural network, and also it determines the frequency of cross-checking with model parameters. Selecting the optimized learning rate is a challenging task because if the learning rate is very less, then it may slow down the training process. On the other hand, if the learning rate is too large, then it may not optimize the model properly.

#### **Note: Learning rate is a crucial hyperparameter for optimizing the model, so if there is a requirement of tuning only a single hyperparameter, it is suggested to tune the learning rate.**

* **Batch Size:** To enhance the speed of the learning process, the training set is divided into different subsets, which are known as a batch. **Number of Epochs:** An epoch can be defined as the complete cycle for training the machine learning model. Epoch represents an iterative learning process. The number of epochs varies from model to model, and various models are created with more than one epoch. To determine the right number of epochs, a validation error is taken into account. The number of epochs is increased until there is a reduction in a validation error. If there is no improvement in reduction error for the consecutive epochs, then it indicates to stop increasing the number of epochs.

### Hyperparameter for Specific Models

Hyperparameters that are involved in the structure of the model are known as hyperparameters for specific models. These are given below:

* **A number of Hidden Units:** Hidden units are part of neural networks, which refer to the components comprising the layers of processors between input and output units in a neural network.

It is important to specify the number of hidden units hyperparameter for the neural network. It should be between the size of the input layer and the size of the output layer. More specifically, the number of hidden units should be 2/3 of the size of the input layer, plus the size of the output layer.

For complex functions, it is necessary to specify the number of hidden units, but it should not overfit the model.

* **Number of Layers:** A neural network is made up of vertically arranged components, which are called layers. There are mainly **input layers, hidden layers, and output layers**. A 3-layered neural network gives a better performance than a 2-layered network. For a Convolutional Neural network, a greater number of layers make a better model.