Challenges of Machine Learning

Some of the major challenges that you might face while developing your machine learning model.

During the development phase our focus is to select a learning algorithm and train it on some data, the two things that might be a problem are a **bad algorithm** or **bad data,** or perhaps **both of them**.

1. Collection of Good Quality Data
2.Inadequate Training Data
3. Non-representative training data
4. Overfitting and Underfitting
5. Monitoring and maintenance
6. Lack of skilled resources
7. Slow implementations and results
8.Irrelevant features

1. Collection of Good Quality Data

- Data plays a significant role in machine learning, and it must be of good quality as well. Noisy data, incomplete data, inaccurate data, and unclean data lead to less accuracy in classification and low-quality results. Hence, data quality can also be considered as a major common problem while processing machine learning algorithms.

2.Inadequate Training Data

- The major issue that comes while using machine learning algorithms is the lack of quality as well as quantity of data. Although data plays a vital role in the processing of machine learning algorithms, many data scientists claim that inadequate data, noisy data, and unclean data are extremely exhausting the machine learning algorithms.
- Noisy Data- It is responsible for an inaccurate prediction that affects the decision as well as accuracy in classification tasks.
- Incorrect data- It is also responsible for faulty programming and results obtained in machine learning models. Hence, incorrect data may affect the accuracy of the results also.
- Generalizing of output data- Sometimes, it is also found that generalizing output data becomes complex, which results in comparatively poor future actions.

3. Non-representative training data

If we are using non-representative training data in the model, it results in less accurate predictions. A machine learning model is said to be ideal if it predicts well for generalized cases and provides accurate decisions. If there is less training data, then there will be a sampling noise in the model, called the non-representative training set. It won't be accurate in predictions. To overcome this, it will be biased against one class or a group.

4. Overfitting and Underfitting

Overfitting: Overfitting is one of the most common issues faced by Machine Learning engineers and data scientists. Whenever a machine learning model is trained with a huge amount of data, it starts capturing noise and inaccurate data into the training data set. It negatively affects the performance of the model. Let's understand with a simple example where we have a few training data sets such as 1000 mangoes, 1000 apples, 1000 bananas, and 5000 papayas. Then there is a considerable probability of identification of an apple as papaya because we have a massive amount of biased data in the training data set; hence prediction got negatively affected.

5. Monitoring and maintenance

As we know that generalized output data is mandatory for any machine learning model; hence, regular monitoring and maintenance become compulsory for the same. Different results for different actions require data change; hence editing of codes as well as resources for monitoring them also become necessary.

6. Lack of skilled resources

Although Machine Learning and Artificial Intelligence are continuously growing in the market, still these industries are fresher in comparison to others. The absence of skilled resources in the form of manpower is also an issue. Hence, we need manpower having in-depth knowledge of mathematics, science, and technologies for developing and managing scientific substances for machine learning.

7. Slow implementations and results

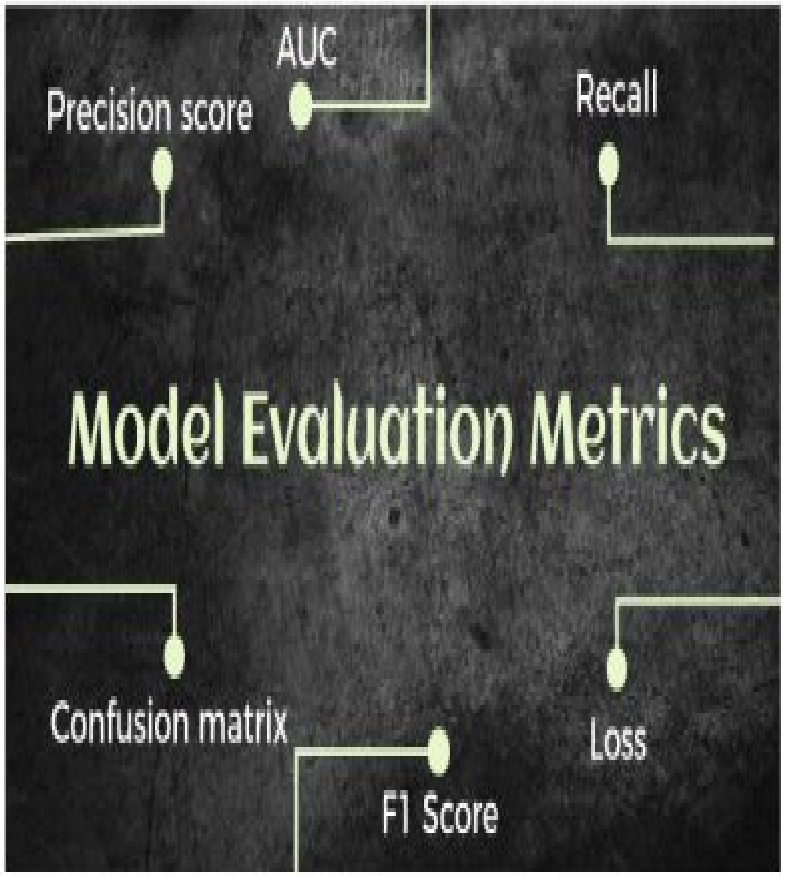
This issue is also very commonly seen in machine learning models. However, machine learning models are highly efficient in producing accurate results but are time-consuming. Slow programming, excessive requirements' and overloaded data take more time to provide accurate results than expected. This needs continuous maintenance and monitoring of the model for delivering accurate results

8.Irrelevant features

Although machine learning models are intended to give the best possible outcome, if we feed garbage data as input, then the result will also be garbage. Hence, we should use relevant features in our training sample. A machine learning model is said to be good if training data has a good set of features or less to no irrelevant features.

Performance Metrics in Machine Learning

- Evaluating the performance of a ML model is one of the important steps while building an effective ML model. **To evaluate the performance or quality of the model, different metrics are used**, and these metrics are known as **performance metrics or evaluation metrics.**
- we can improve the model's performance by tuning the hyper-parameters. Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset.



- In machine learning, each task or problem is divided into classification and Regression. Not all metrics can be used for all types of problems; hence, it is important to know and understand which metrics should be used.
- **1. Performance Metrics for Classification**
- In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based on the training. It predicts class labels as the output, such as Yes or No, 0 or 1, Spam or Not Spam, etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:

- Accuracy
- Confusion Matrix
- Precision
- Recall
- F-Score
- AUC(Area Under the Curve)-ROC

I. Accuracy

- The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ predictions}$$

To implement an accuracy metric, we can compare ground truth and predicted values in a loop, or we can also use the scikit-learn module for this.

Firstly, we need to import the *accuracy_score* function of the scikit-learn library as follows:

```
from sklearn.metrics import accuracy_score
```

Here, metrics is a class of sklearn.

Then we need to pass the ground truth and predicted values in the function to calculate the accuracy.

```
print(f'Accuracy Score is {accuracy_score(y_test,y_hat)}')
```

II. Confusion Matrix

A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of test data when true values are known.

The confusion matrix is simple to implement, but the terminologies used in this matrix might be confusing for beginners.

A typical confusion matrix for a binary classifier looks like the below image(However, it can be extended to use for classifiers with more than two classes).

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

II. Confusion Matrix

We can determine the following from the above matrix:

- In the matrix, columns are for the prediction values, and rows specify the Actual values. Here Actual and prediction give two possible classes, Yes or No. So, if we are predicting the presence of a disease in a patient, the Prediction column with Yes means, Patient has the disease, and for NO, the Patient doesn't have the disease.
- In this example, the total number of predictions are 165, out of which 110 time predicted yes, whereas 55 times predicted No.
- However, in reality, 60 cases in which patients don't have the disease, whereas 105 cases in which patients have the disease.

In general, the table is divided into four terminologies, which are as follows:

1. **True Positive(TP):** In this case, the prediction outcome is true, and it is true in reality, also.
2. True Negative(TN): in this case, the prediction outcome is false, and it is false in reality, also.
3. False Positive(FP): In this case, prediction outcomes are true, but they are false in actuality.
4. False Negative(FN): In this case, predictions are false, and they are true in actuality.

III. Precision

The precision metric is used to overcome the limitation of Accuracy. The precision determines the proportion of positive prediction that was actually correct. It can be calculated as the True Positive or predictions that are actually true to the total positive predictions (True Positive and False Positive).

$$Precision = \frac{TP}{(TP + FP)}$$

IV. Recall or Sensitivity

It is also similar to the Precision metric; however, it aims to calculate the proportion of actual positive that was identified incorrectly. It can be calculated as True Positive or predictions that are actually true to the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (true Positive and false negative).

The formula for calculating Recall is given below:

$$\text{Recall} = \frac{TP}{TP+FN}$$

V. F-Scores

F-score or F1 Score is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, ***the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them.***

The formula for calculating the F1 score is given below:

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

When to use F-Score?

As F-score make use of both precision and recall, so it should be used if both of them are important for evaluation, but one (precision or recall) is slightly more important to consider than the other. For example, when False negatives are comparatively more important than false positives, or vice versa.

VI. AUC-ROC

Sometimes we need to visualize the performance of the classification model on charts; then, we can use the AUC-ROC curve. It is one of the popular and important metrics for evaluating the performance of the classification model.

Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve. ***ROC represents a graph to show the performance of a classification model at different threshold levels***. The curve is plotted between two parameters, which are:

- **True Positive Rate**
- **False Positive Rate**

TPR or true Positive rate is a synonym for Recall, hence can be calculated as:

$$TPR = \frac{TP}{TP + FN}$$

FPR or False Positive Rate can be calculated as:

$$TPR = \frac{FP}{FP + TN}$$

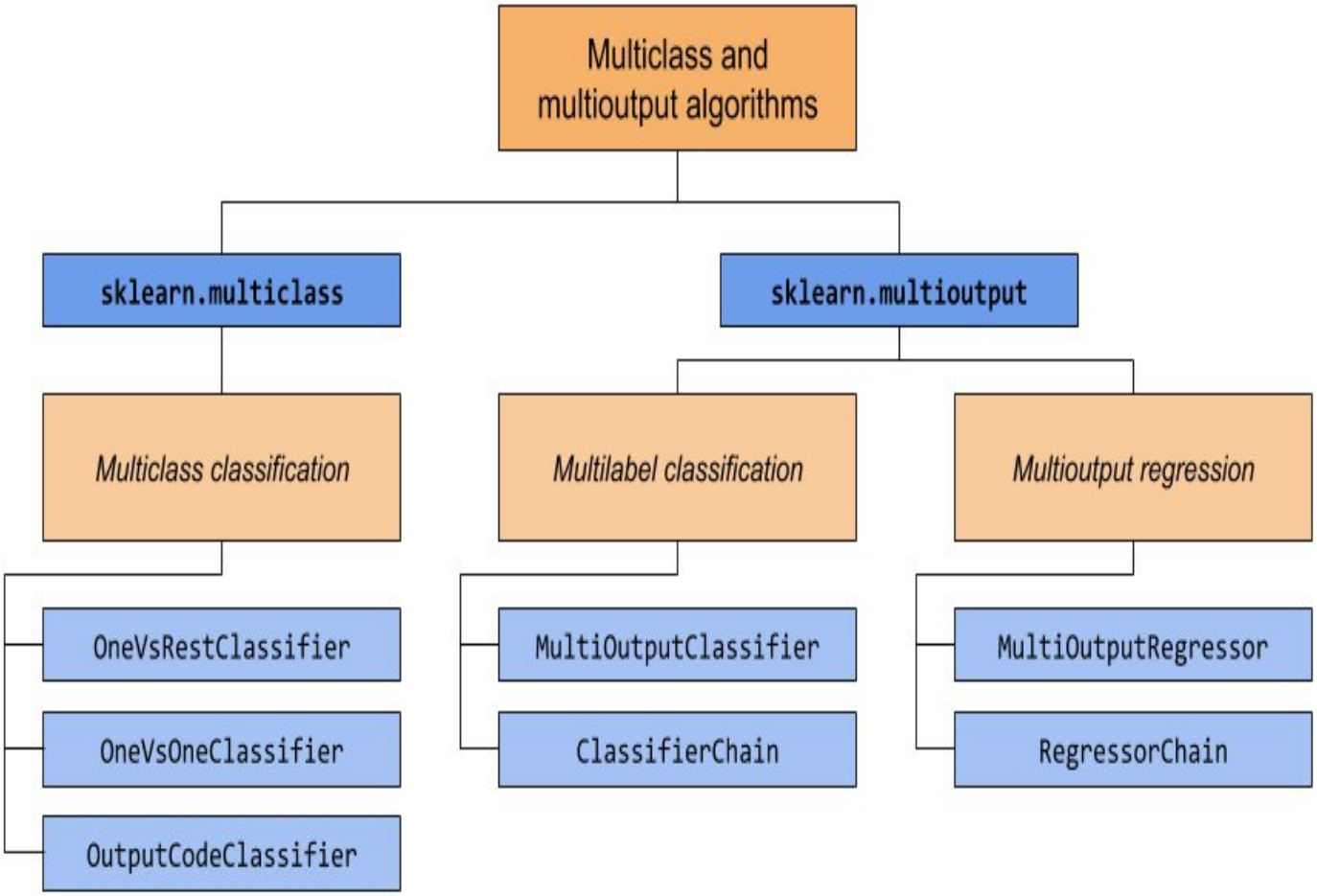
To calculate value at any point in a ROC curve, we can evaluate a logistic regression model multiple times with different classification thresholds, but this would not be much efficient. So, for this, one efficient method is used, which is known as AUC.

Performance Metrics for Regression

- Regression is a supervised learning technique that aims to find the relationships between the dependent and independent variables. A predictive regression model predicts a numeric or discrete value. The metrics used for regression are different from the classification metrics. It means we cannot use the Accuracy metric (explained above) to evaluate a regression model; instead, the performance of a Regression model is reported as errors in the prediction. Following are the popular metrics that are used to evaluate the performance of Regression models.

- Mean Absolute Error
- Mean Squared Error
- R2 Score
- Adjusted R2

Classifications:



Multiclass classification

- There are many different types of classification tasks that you may encounter in machine learning and specialized approaches to modeling that may be used for each.
- This is divided into five parts; they are:
- Binary Classification
- Multi-Class Classification
- Multi-Label Classification
- Imbalanced Classification

Binary Classification

- Typically, binary classification tasks involve one class that is the normal state and another class that is the abnormal state.

- For example “not spam” is the normal state and “spam” is the abnormal state. Another example is “cancer not detected” is the normal state of a task that involves a medical test and “cancer detected” is the abnormal state.
- Popular algorithms that can be used for binary classification include:

- Logistic Regression
- k-Nearest Neighbors
- Decision Trees
- Support Vector Machine
- Naive Bayes
- Some algorithms are specifically designed for binary classification and do not natively support more than two classes; examples include Logistic Regression and Support Vector Machines.

Multi-Class Classification

- Multi-class labels are used in classification tasks referred to as multi-class classification.
- Examples comprise -
- Categorization of faces.
- Classifying plant species.
- Character recognition using optical.
- The multi-class classification does not have the idea of normal and abnormal outcomes, in contrast to binary classification. Instead, instances are grouped into one of several well-known classes.
- In some cases, the number of class labels could be rather high. In a facial recognition system, for instance, a model might predict that a shot belongs to one of thousands or tens of thousands of faces.

Multi-Class Classification

- Multi-class problems can be solved using algorithms created for binary classification.
- In order to do this, a method is known as "one-vs-rest" or "one model for each pair of classes" is used, which includes fitting multiple binary classification models with each class versus all other classes (called one-vs-one).
- One-vs-One: For each pair of classes, fit a single binary classification model.
- The following binary classification algorithms can apply these multi-class classification techniques:
- One-vs-Rest: Fit a single binary classification model for each class versus all other classes.
- The following binary classification algorithms can apply these multi-class classification techniques:
- Support vector Machine
- Logistic Regression

Multi-Label Classification

Multilabel classification (closely related to **multioutput classification**) is a classification task labeling each sample with m labels from n_classes possible classes, where m can be 0 to n_classes inclusive. This can be thought of as predicting properties of a sample that are not mutually exclusive. Formally, a binary output is assigned to each class, for every sample. Positive classes are indicated with 1 and negative classes with 0 or -1. It is thus comparable to running n_classes binary classification tasks, for example with MultiOutputClassifier. This approach treats each label independently whereas multilabel classifiers *may* treat the multiple classes simultaneously, accounting for correlated behavior among them.

For example, prediction of the topics relevant to a text document or video. The document or video may be about one of 'religion', 'politics', 'finance' or 'education', several of the topic classes or all of the topic classes.

MultiOutputClassifier

Multilabel classification support can be added to any classifier with MultiOutputClassifier. This strategy consists of fitting one classifier per target. This allows multiple target variable classifications. The purpose of this class is to extend estimators to be able to estimate a series of target functions (f1,f2,f3…,fn) that are trained on a single X predictor matrix to predict a series of responses (y1,y2,y3…,yn).

You can find a usage example for MultiOutputClassifier as part of the section on Multiclass-multioutput classification since it is a generalization of multilabel classification to multiclass outputs instead of binary outputs

Training vs Testing vs Validation Sets

- 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.

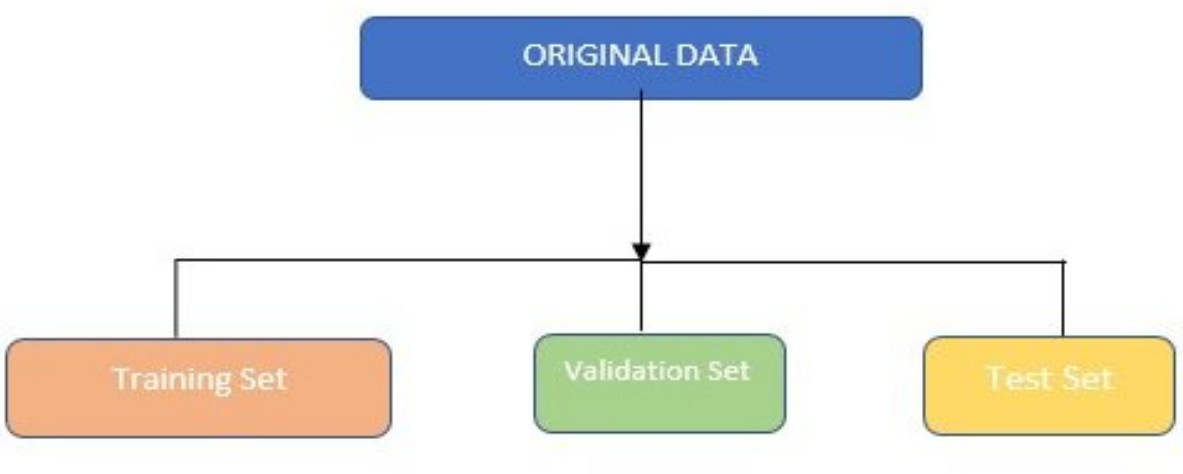


Fig. General representation of training, validation, and testing datasets

Training vs Testing vs Validation Sets

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 Training Set

Applications of Training 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.

TestingSet

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

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

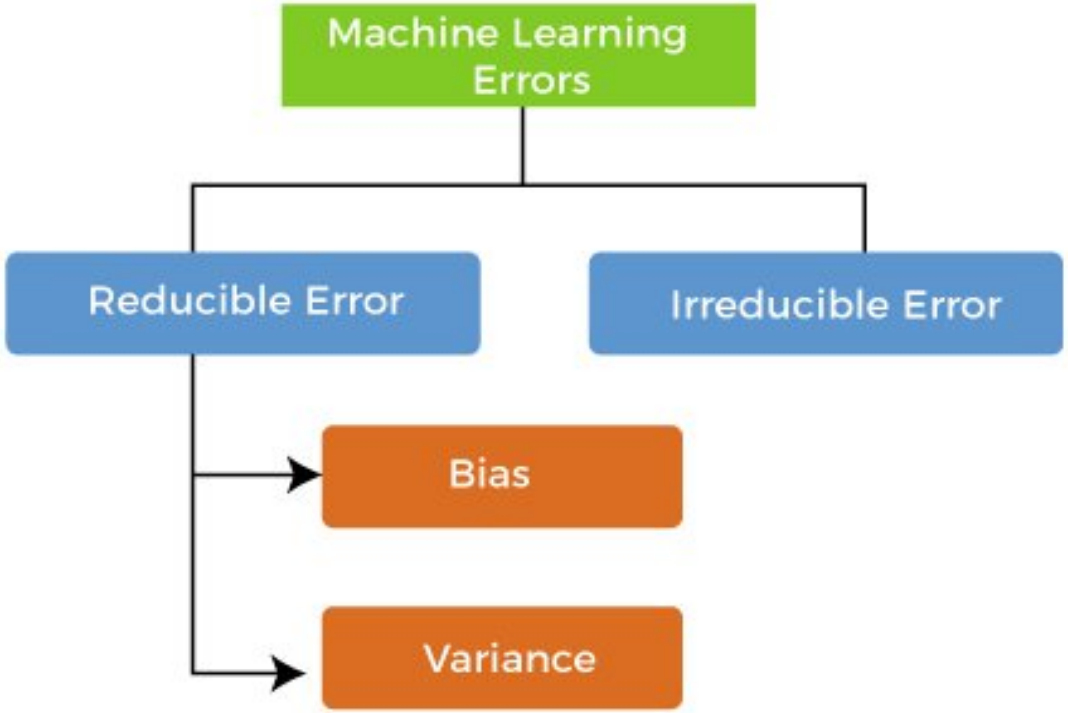
The below table summarizes Training, Validation, and Testing sets.

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

Errors in Machine Learning

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



Errors in Machine Learning

- What is Bias?
- In general, a machine learning model analyses the data, find patterns in it and make predictions. While training, the model learns these patterns in the dataset and applies them to test data for prediction. ***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***.
- What is Variance?
- Variance is the very opposite of Bias. During training, it allows our model to 'see' the data a certain number of times to find patterns in it. If it does not work on the data for long enough, it will not find patterns and bias occurs. On the other hand, if our model is allowed to view the data too many times, it will learn very well for only that data. It will capture most patterns in the data, but it will also learn from the unnecessary data present, or from the noise