# 3 Modeling and Evaluation



**Model:** Structured representation of raw input data to the meaningful pattern is called a model.

**Model Training:** The process of fitting a specific model to a data set is called model training.

**Target Function:** Target function of a model is the function defining the relationship between the input (also called predictor or independent) variables and the output (also called response or dependent or target) variable.

It is represented in the general form: $Y = f(X) + e$, where Y is the output variable, X represents the input variables and 'e' is a random error term.

## 3.1 Selecting a Model (Predictive/ Descriptive)

Machine learning algorithms are broadly of two types:

1. Models for supervised learning, which primarily focus on solving predictive problems
2. Models for unsupervised learning, which solve descriptive problems.

### 3.1.1 Predictive Models

Models for supervised learning approach are Predictive models. They try to predict certain value using the values in an input data set.

The predictive models have a clear focus on what they want to learn and how they want to learn.

Predictive models may need to predict the value of a category or class to which a data instance belongs to.

Examples :

1. Predicting win/loss in a basketball match
2. Predicting whether a bank transaction is fraud

The models which are used for prediction of target features of categorical value are known as **classification models**. The target feature is known as a class and the categories to which classes are divided into are called levels. Examples of classification models are k-Nearest Neighbor (kNN), Naïve Bayes, and Decision Tree.

Predictive models may also be used to predict numerical values of the target feature based on the predictor features.

Example :

1. Prediction of financial growth in the next coming year
2. Prediction of heat wave in the summer

The models which are used for prediction of the numerical value of the target feature of a data instance are known as **regression models**. Linear Regression and Logistic Regression models are examples of regression models.

### 3.1.2 Descriptive Models

Models for unsupervised learning approach are Descriptive models. They are used to describe a data set or gain insight from a data set.

There is no target feature or single feature of interest in case of unsupervised learning. Based on the value of all features, interesting patterns or insights are derived about the data set.

Descriptive models which group together similar data instances are called clustering models. The most popular model for clustering is k-Means.

Examples:

1. Customer grouping based on society
2. Movie grouping based on language and age

Descriptive models related to pattern discovery is used for market basket analysis of transactional data.

**Market Basket Analysis** is a modeling technique based upon the theory that if you buy a certain group of items, you are more (or less) likely to buy another group of items.

For example, if you are in a store and you buy milk and don't buy bread, you are more likely to buy fruits at the same time than somebody who didn't buy bread.

The set of items a customer buys is referred to as an item set, and market basket analysis seeks to find relationships between purchases.

Typically, the relationship will be in the form of a rule like:

IF {milk, butter} THEN {bread}.

The probability that a customer will buy milk without an butter (i.e. that the antecedent is true) is referred to as the **support for the rule**.

The conditional probability that a customer will purchase bread is referred to as the **confidence**.

## 3.2 Training a model for supervised learning

### 3.2.1 Holdout Method

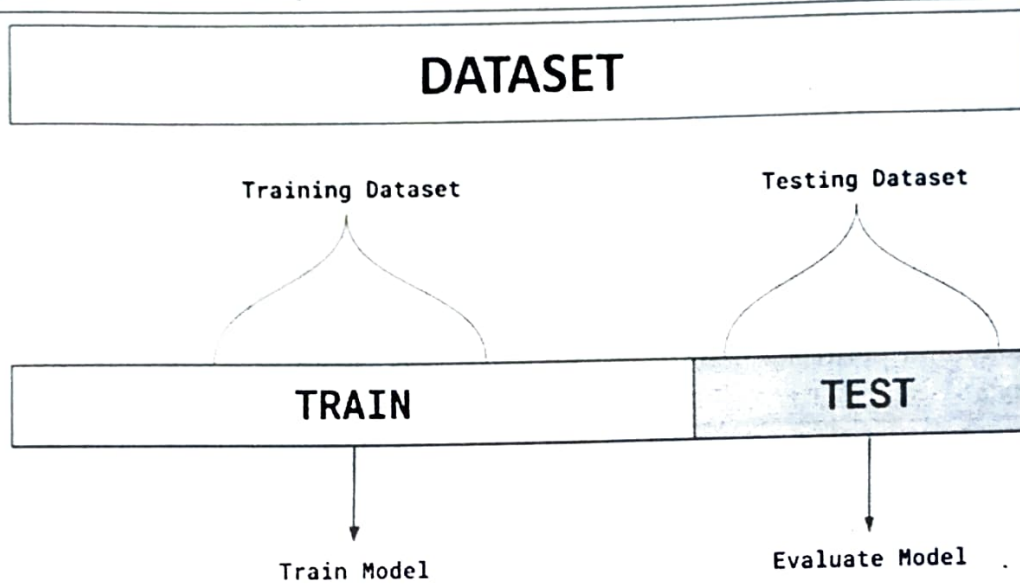
The hold-out method involves splitting the data into multiple parts and using one part for training the model and the rest for

validating and testing it. It can be used for both model evaluation and selection.

Model evaluation using the hold-out method entails splitting the dataset into training and test datasets, evaluating model performance, and determining the most optimal model.

The objective of this technique is to select the best model based on its accuracy on the testing dataset and compare it with other models. Models are trained to improve model accuracy on test datasets based on the assumption that the test dataset represents the population.

This diagram illustrates the hold-out method for model evaluation.




**Holdout method**

There are two parts to the dataset in the diagram above.

One split is held aside as a training set. Another set is held back for testing or evaluation of the model. The percentage of the split is determined based on the amount of training data available. A typical split of 70–30% is used in which 70% of the dataset is used for training and 30% is used for testing the model.

Follow the steps below for using the hold-out method for model evaluation:

1. Split the dataset in two (preferably 70–30%; however, the split percentage can vary and should be random).
2. Now, we train the model on the training dataset by selecting some fixed set of hyper-parameters while training the model.
3. Use the hold-out test dataset to evaluate the model.
4. Use the entire dataset to train the final model so that it can generalize better on future datasets.

In this process, the dataset is split into training and test sets, and a fixed set of hyper-parameters is used to evaluate the model.

### 3.2.1 K-fold cross validation method

In machine learning, we couldn't fit the model on the training data and can't say that the model will work accurately for the real data.

For this, we have to assure that our model got the correct patterns from the data, and it is not getting up too much noise. For this purpose, we use the cross-validation technique.

Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds.

This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance.

The main purpose of cross validation is to overcome overfitting, which occurs when a model is trained too well on the training data and performs poorly on new, unseen data.

In **K-Fold Cross Validation** method, we split the data-set into k number of subsets (known as folds) then we perform training on the all the subsets but leave one(k-1) subset for the evaluation of the trained model. In this method, we iterate k times with a different subset reserved for testing purpose each time.

The general steps to implement K-Fold cross validation are as follows:

Step 1: Shuffle the dataset randomly.

Step 2: Split the dataset into k groups

Step 3: For each unique group:

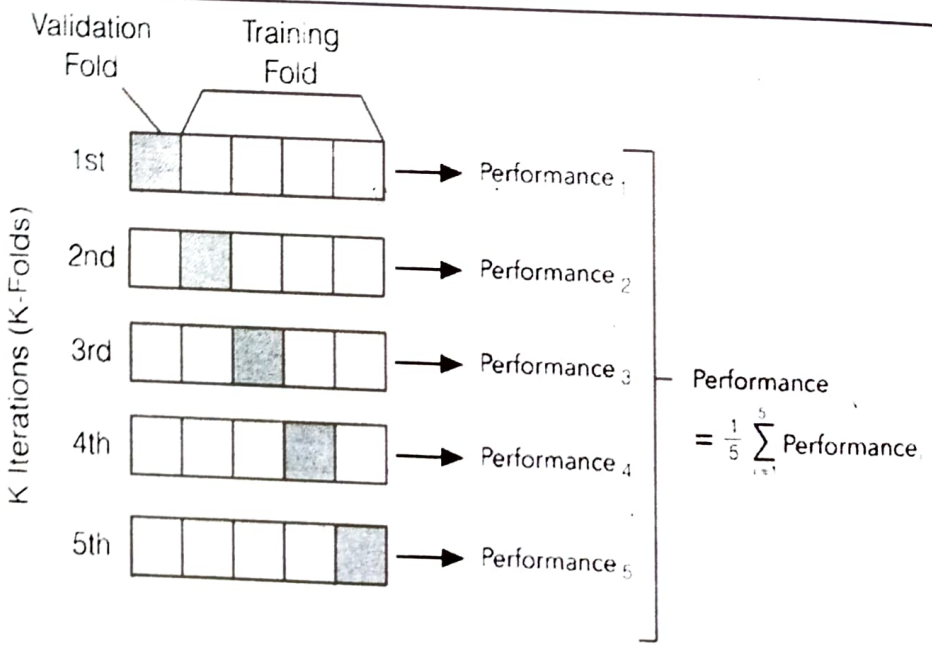
a. Take the group as a hold out or test data set

b. Take the remaining groups as a training data set

c. Fit a model on the training set and evaluate it on the test set

d. Retain the evaluation score and discard the model

Step 4 : Summarize the skill of the model using the sample of model evaluation scores




**K-Fold cross validation**

Advantages of cross-validation:

- More accurate estimate of out-of-sample accuracy.
- More "efficient" use of data as every observation is used for both training and testing.
- Overcome the issue of overfitting

Three common tactics for choosing a value for k are as follows:

**Representative:** The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.

**k=5 or 10:** The value for k is fixed to 5 or 10, a value that has been found through experimentation to generally result in a model skill estimate with low bias a modest variance.

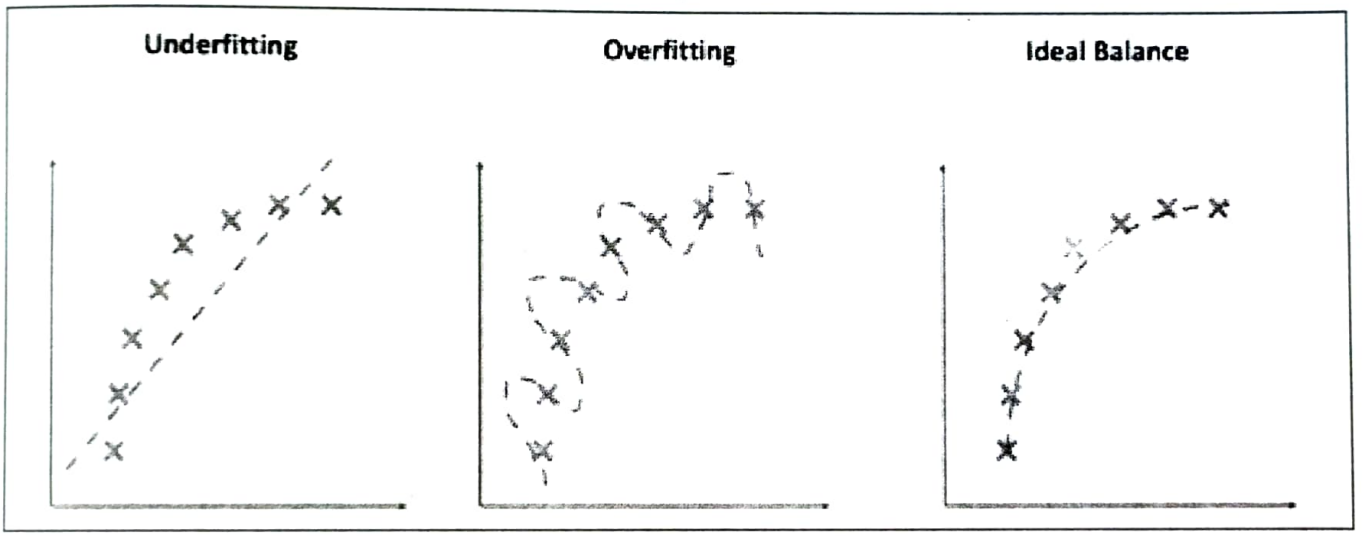
**k=n:** The value for k is fixed to n, where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called leave-one-out cross-validation.

## 3.3 Model representation and interpretability

Interpretability focuses on the *how*.

Fitness of a target function approximated by a learning algorithm determines how correctly it is able to classify a set of data it has never seen.

In Machine Learning, if the model is able to fit on training data doesn't mean that it will perform well on testing data. This disparity between the performance on the training and test data is called Generalization Gap. It is common in machine learning problems to observe a gap between the training and testing performance.




### 3.3.1 OVERFITTING

Overfitting is a scenario where the machine learning model tries to learn from the data along with the noise present in the data and tries to fit each data point on the curve.

Overfitting is a fundamental issue in supervised machine learning which prevents us from perfectly generalizing the models to well fit observed data on training data, as well as unseen data on the testing sets. When a model is high on variance, it is then said to as Overfitting of Data.

**REASONS FOR OVERFITTING**

- Data used for training is not cleaned and contains noise (garbage values) in it
- The model has a high variance
- The size of training data used is not enough
- The model is too complex

**METHODS TO AVOID OVERFITTING**

- Use re-sampling techniques like k-fold Cross-validation

- Hold back of a validation data set
- Remove the nodes which have no. predictive power

### 3.3.2 UNDERFITTING

**Underfitting is a scenario where a machine learning model can neither learn the relationship between variables in the data nor predict or classify a new data point.**

In order to avoid overfitting, we could stop the training at an earlier stage. But it might also lead to the model not being able to learn enough from training data, which may find it difficult to capture the dominant trend. This is known as underfitting.

As the model doesn't fully learn the patterns, it accepts every new data point during the prediction. An underfitting model has low variance and high bias.

**Reasons for Underfitting**

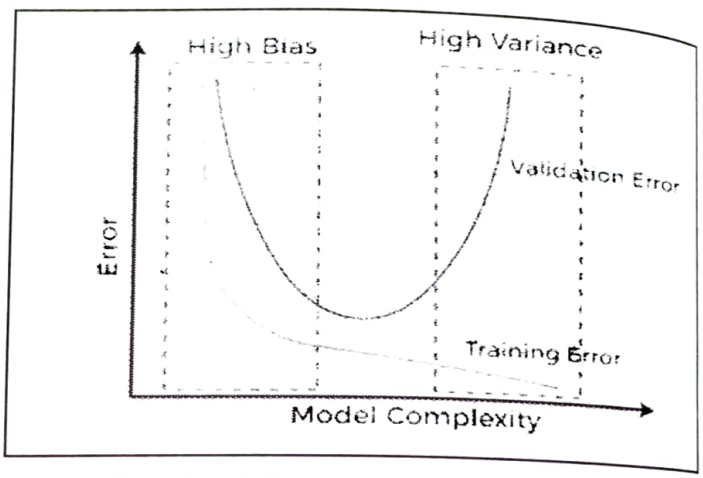
- Data used for training is not cleaned and contains noise (garbage values) in it
- The model has a high bias
- The size of training data used is not enough
- The model is too simple

**Methods to avoid Underfitting**

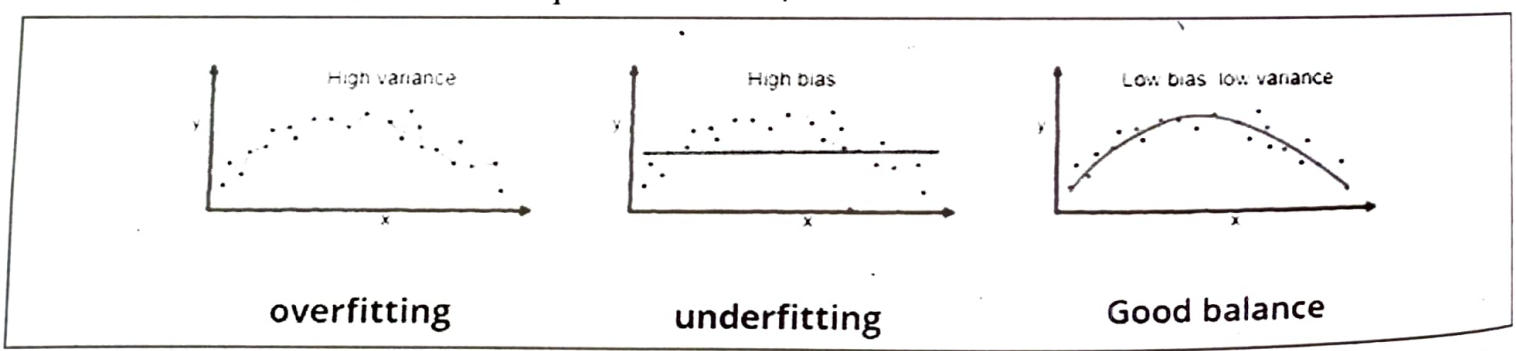
- Use more training data
- Reduce features by effective feature selection method

### 3.3.3 Bias-variance trade-off

It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine learning algorithm.




If the algorithm is too simple then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as **Bias Variance Trade-off.**




This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time. To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.

## 3.4 Evaluating Performance of a Model

Classification is one of the major tasks in Supervised Learning. In classification, the number of correct and incorrect predictions made by the model is evaluated and accuracy of the model calculated based on that evaluation.

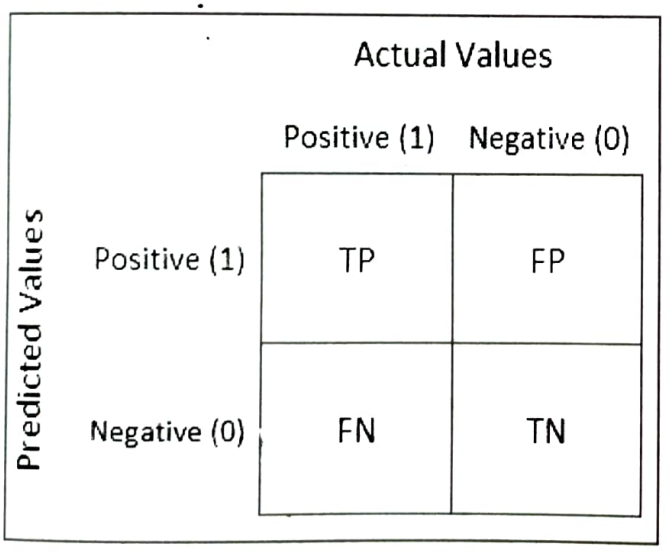
### 3.4.1 Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a machine learning model by comparing its predictions to the true values of the dataset. It is used in classification.

It consists of four possible outcomes: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

The rows of the confusion matrix represent the actual values of the target variable, while the columns represent the predicted values of the target variable. Each cell in the matrix represents the number of instances that fall into a particular combination of actual and predicted values.

The structure of confusion matrix is given below :




**True positives (TP) :** These are cases where the model predicts a positive class, and the actual class is also positive. For example, if we're trying to classify whether an email is spam or not, a true positive would be when the model correctly predicts that an email is spam and the email is actually spam.

**True negatives (TN) :** These are cases where the model predicts a negative class, and the actual class is also negative. For example, using the same email classification problem, a true negative would be when the model correctly predicts that an email is not spam, and the email is actually not spam.

**False positives (FP) :** These are cases where the model predicts a positive class, but the actual class is negative. For example, in the email classification problem, a false positive would be when the model incorrectly predicts that an email is spam, but the email is actually not spam.

**False negatives (FN) :** These are cases where the model predicts a negative class, but the actual class is positive. For example, in the email classification problem, a false negative would be when the model incorrectly predicts that an email is not spam, but the email is actually spam.

In general, we want to minimize false positives and false negatives and maximize true positives and true negatives to have a good performance of a machine learning model.

The confusion matrix summarizes these four possible outcomes and helps us to calculate various performance metrics such as accuracy, error rate, sensitivity, specificity, precision, recall, and F1-score.

- **Accuracy :** This measures the proportion of correctly classified instances out of all the instances in the dataset. It is calculated

by dividing the number of correct predictions by the total number of predictions. The formula is :

- **Accuracy = (TP + TN) / (TP + TN + FP + FN)**

**Error rate**: This indicates the percentage of misclassification. The formula is :

**Error rate = (FP + FN) / (TP + TN + FP + FN)**

In other words,

**Error rate = 1 – Model Accuracy**

- **Sensitivity**: The sensitivity of a model measures the proportion of positive examples which were correctly classified. The formula is:

**Sensitivity = TP / (TP + FN)**

- **Specificity**: The specificity of a model measures the proportion of negative examples which were correctly classified. The formula is:

**Specificity = TN / (TN + FP)**

- **Precision**: This measures the proportion of true positive predictions out of all the positive predictions made by the model. It is calculated by dividing the number of true positives by the total number of positive predictions. The formula is:

**Precision = TP / (TP + FP)**

- **Recall**: This measures the proportion of true positive predictions out of all the actual positive instances in the dataset. It is calculated by dividing the number of true positives by the total number of positive instances. The formula is:

**Recall = TP / (TP + FN)**

- **F1 score**: This is a harmonic mean of precision and recall, and provides a balanced measure of a model's performance. It is calculated by taking the harmonic mean of precision and recall. The formula is:

**F1 score = 2 * (Precision * Recall) / (Precision + Recall)**

**Example 1** :

Suppose we have a binary classification problem where we want to predict if a person has a disease or not based on their medical test results. We have a dataset of 100 patients, where 60 do not have the disease and 40 have the disease. We train a machine learning model on this data to predict the disease status based on test results.

After training the model, we test it on a separate dataset of 20 patients and get the following results:

True positives (TP) : The model correctly predicted that 7 patients have the disease.

True negatives (TN) : The model correctly predicted that 10 patients do not have the disease.

False positives (FP) : The model predicted that 2 patients have the disease, but they do not actually have the disease.

False negatives (FN) : The model predicted that 1 patient does not have the disease, but they actually have the disease.

We can use these results to create a confusion matrix, which would look like this:

| | Predicted No Disease | Predicted Disease |
|---|---|---|
| Actual No Disease | TN = 10 | FP = 2 |
| Actual Disease | FN = 1 | TP = 7 |

From the confusion matrix, we can calculate several performance metrics of the model, including accuracy, error rate, sensitivity, specificity, precision, recall, and F1 score.

For **example :**

Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (7 + 10) / (7 + 10 + 2 + 1)
= 0.85 (85% Accuracy of the model)

Error rate = 1 – Accuracy
= 1 – 0,85 = 0.15

Sensitivity = TP / (TP + FN)
= 7 / (7+1) = 0.875

Specificity = TN / (TN + FP)
= 10 / (10 + 2) = 0.8333

Precision = TP / (TP + FP)
= 7 / (7 + 2) = 0.78

Recall = TP / (TP + FN)
= 7 / (7 + 1) = 0.88

F1 score = 2 * (precision * recall) / (precision + recall)
= 2 * (0.78 * 0.88) / (0.78 + 0.88) = 0.82

These metrics provide insight into how well the model is performing and can help us make improvements or adjustments as necessary.

**Example 2 :**

Let's consider the example of a binary classification problem where we want to predict whether an email is spam or not spam. We have a dataset of 100 emails, where 60 are not spam (negative class) and 40 are spam (positive class). We build a machine learning model to predict whether each email is spam or not, and the results are as follows :

| | Predicted Not Spam | Predicted Spam |
|---|---|---|
| Actual Not Spam | 50 | 10 |
| Actual Spam | 5 | 35 |

This confusion matrix tells us that out of the 60 actual not spam emails, the model correctly predicted 50 as not spam (True Negatives, TN) but incorrectly predicted 10 as spam (False Positives, FP). Out of the 40 actual spam emails, the model correctly predicted 35 as spam (True Positives, TP) but incorrectly predicted 5 as not spam (False Negatives, FN).

Using this confusion matrix, we can calculate several performance metrics of the model.

For the given example, accuracy is (50+35)/ 100 = 85%,

precision is 35/(10+35) = 77.8%,

recall is 35/(5+35) = 87.5%, and

F1 score is 2*(precision*recall)/ (precision+recall) = 82.4%.

These metrics provide insight into how well the model is performing and can help us make improvements or adjustments as necessary.

## 3.5 Improving performance of a model

Let's see the different avenues to improve the performance of models.

1. Feature engineering: Feature engineering is the process of selecting and transforming the input variables used in the model. By selecting the most important features and transforming them into a more suitable form, we can improve the performance of the model.
2. Model Parameter tuning: Most machine learning algorithms have a set of model parameters that can be adjusted to improve performance. Model parameters control the learning rate, regularization, number of trees in a random forest, and so on. Tuning these model parameters can lead to significant performance improvements.

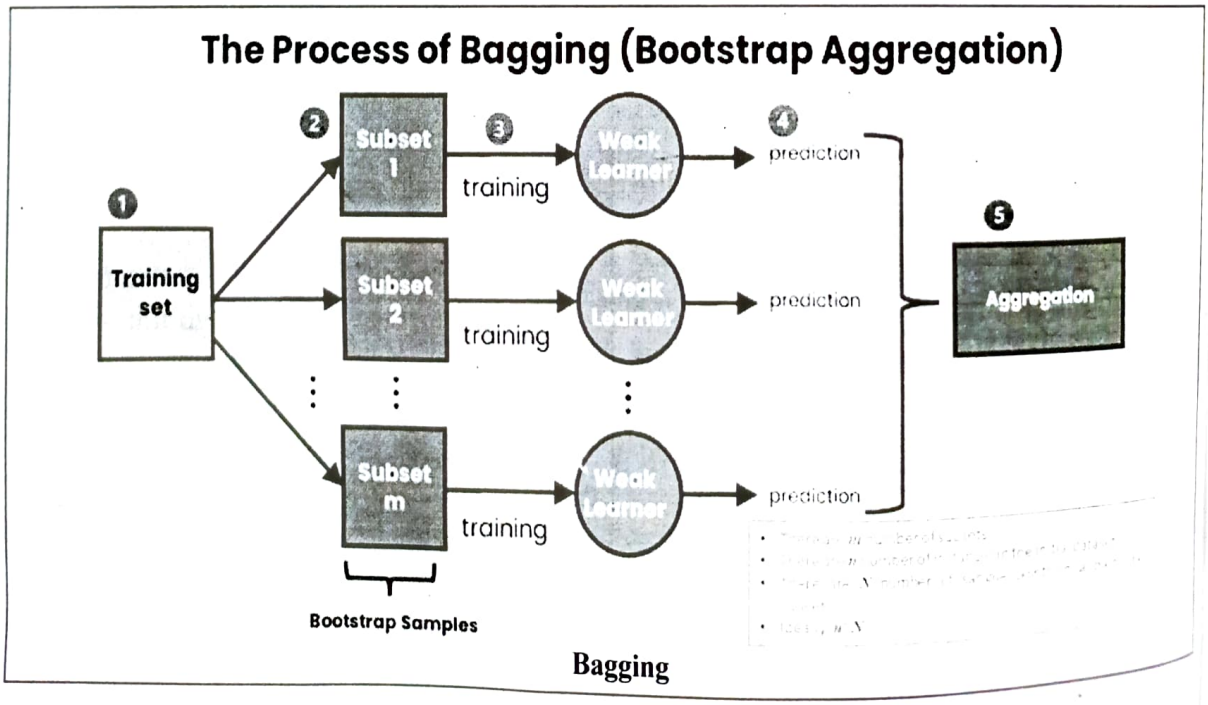
3. Data cleaning and preprocessing: It is important to clean and preprocess the data before training the model. This includes removing missing values, scaling the features, encoding categorical variables, and so on. By carefully preprocessing the data, we can improve the performance of the model.
4. Ensemble methods: Ensemble methods combine multiple models to improve performance. This can include bagging, boosting, and stacking methods.
5. Larger training set: In some cases, the model may be underfitting because the training set is too small. By increasing the size of the training set, we can improve the performance of the model.
6. Regularization: Regularization methods such as L1, L2, and dropout can help prevent overfitting and improve the generalization ability of the model.
7. Different algorithms: If one machine learning algorithm is not performing well, it may be worth trying a different algorithm. There are many different algorithms to choose from, and some may be better suited to the particular problem at hand.

Let's discuss two avenues in detail.

### 3.5.1 Ensemble Approach

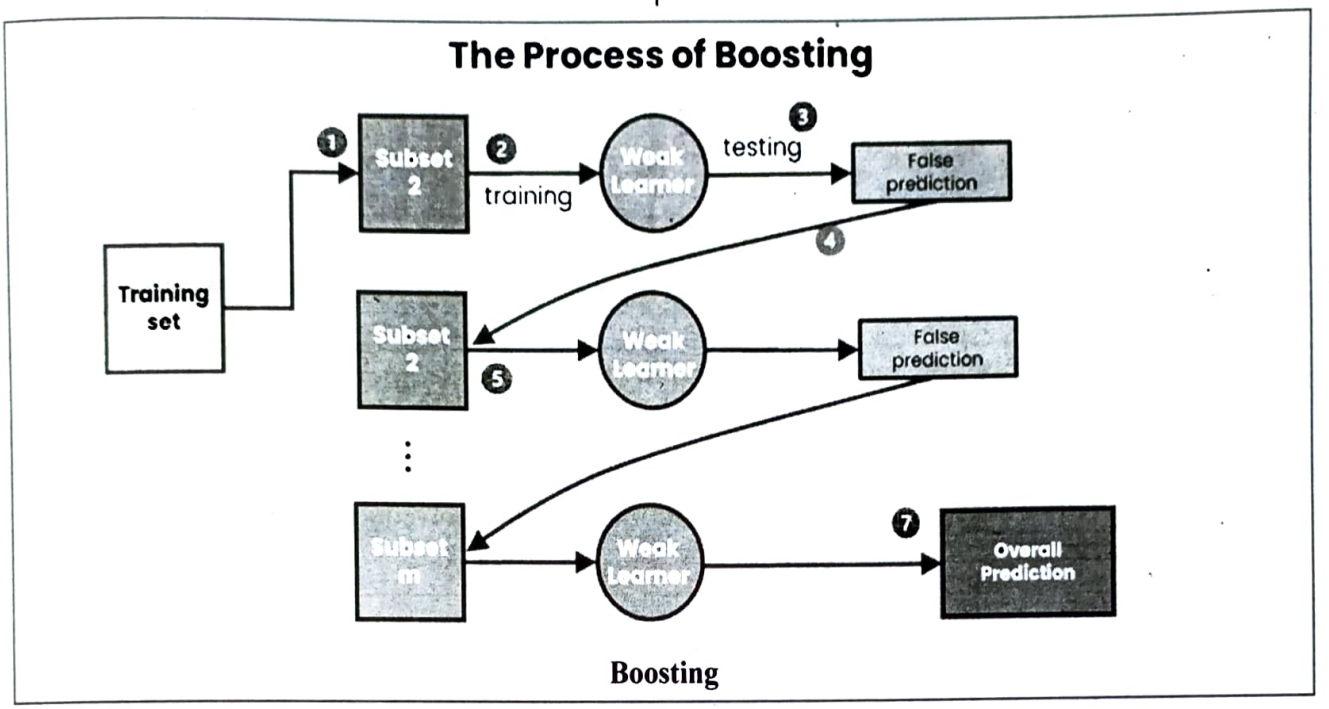
Ensemble learning is a machine learning technique that combines multiple models to improve the overall performance. The basic idea behind ensemble learning is that by combining multiple models, we can reduce the error caused by individual models and improve the overall accuracy. There are several different ways to create an ensemble of models, but some of the most common methods include:

**Bagging:** Bagging stands for Bootstrap Aggregating. In bagging, we create multiple samples of the training data by sampling with replacement. Then we train a separate model on each of these samples. Finally, we combine the results of all models to make the final prediction.



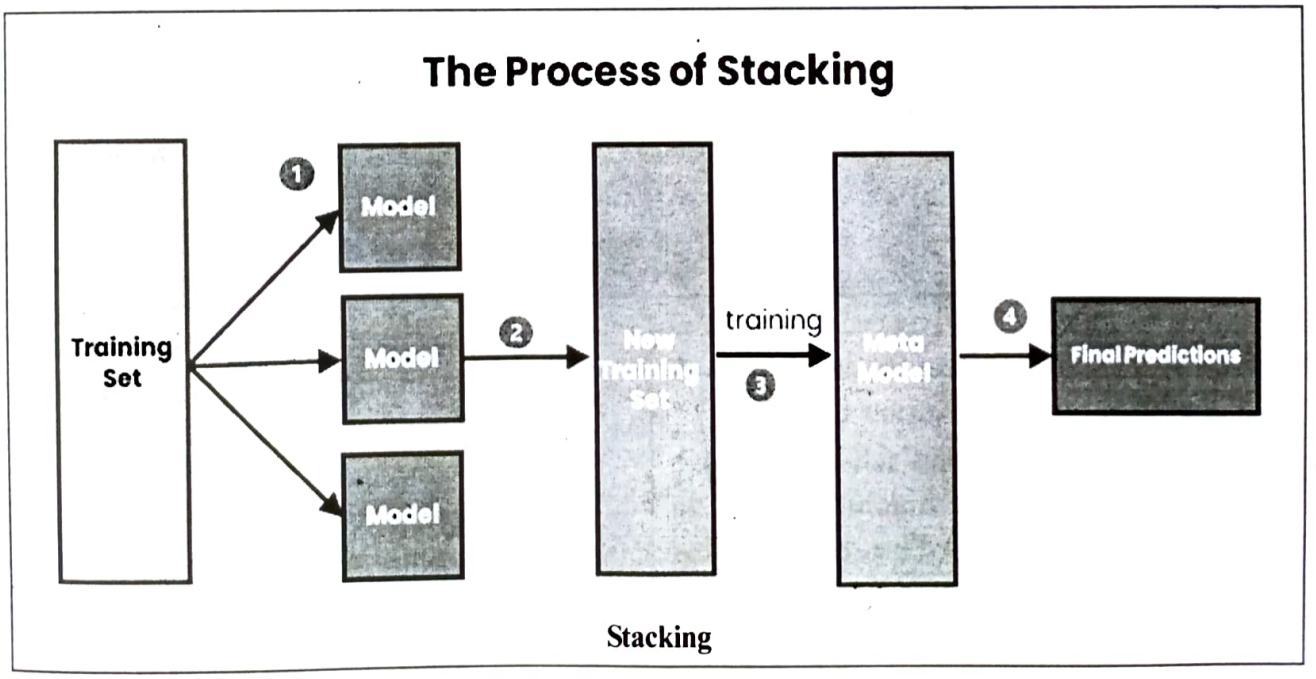

Bagging

**Boosting:** Boosting is a sequential process that creates an ensemble of models by iteratively improving the performance of a single model. In boosting, we train a base model on the entire training set. Then, we assign weights to each sample in the training set and train a new model on the same dataset. In each iteration, we give more weight to the misclassified samples and less weight to the correctly classified samples. Finally, we combine the results of all models to make the final prediction.




Boosting

**Stacking:** Stacking is a process that involves combining multiple models that use different algorithms or different representations of the data. In stacking, we train multiple models on the same dataset. Then, we train a meta-model on the output of these models. The meta-model learns how to combine the outputs of the models to make the final prediction.

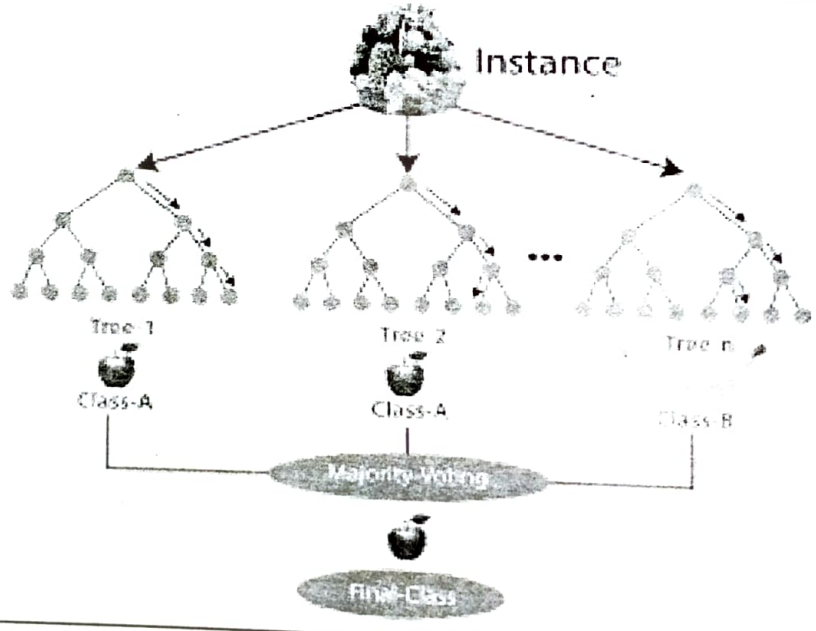



Stacking

| | Bagging | Boosting | Stacking |
|---|---|---|---|
| Purpose | Reduce Variance | Reduce Bias | Improve Accuracy |
| Base Learner Types | Homogeneous | Homogeneous | Heterogeneous |
| Base Learner Training | Parallel | Sequential | Meta Model |
| Aggregation | Max Voting, Averaging | Weighted Averaging | Weighted Averaging |

**One powerful way for ensemble based technique is Random forest:**

In Random Forest, a large number of decision trees are trained on randomly selected subsets of the data. Each tree is trained on a different subset of the data, using a different set of randomly selected features. This randomness helps to reduce overfitting and improve the generalization ability of the model.




Random Forest is a powerful ensemble-based technique that can be used for both classification and regression problems.

Ensemble learning approach can improve the performance of machine learning models in several ways. First, by combining multiple models, we can reduce the variance caused by individual models and improve the generalization ability of the model. Second, by using different algorithms or representations of the data, we can capture different aspects of the data and improve the accuracy of the model. Finally, ensemble learning can be used to create more robust models that are less sensitive to changes in the training data.

However, ensemble learning approach also has some drawbacks. It can be computationally expensive and may require a large amount of memory. In addition, the performance of the ensemble may be highly dependent on the performance of the individual models. Therefore, it is important to carefully select the models and ensure that they are diverse and complementary to each other.

### 3.5.2 Model Parameter Tuning

Model parameter tuning is the process of adjusting the hyperparameters of a machine learning model in order to improve its performance. Hyperparameters are the parameters that are set before the training of the model and cannot be learned from the data. Examples of hyperparameters include the learning rate, the number of hidden layers in a neural network, and the regularization parameter.

The model parameter tuning approach in machine learning involves the following steps:

Step 1 : Define the hyperparameters: The first step is to define the hyperparameters that need to be tuned. This can be done by looking at the model architecture and understanding how the hyperparameters affect the performance of the model.

Step 2 : Define the search space: The next step is to define the search space for the hyperparameters. The search space is a range of possible values for each hyperparameter. The search space can be defined using a grid search or a random search.

Step 3 : Select a performance metric: The performance metric is the metric that will be used to evaluate the performance of the model. The most common performance metrics are accuracy, precision, recall, and F1-score.

Step 4 : Evaluate the model: The model is trained on the training data using different combinations of hyperparameters. The performance of the model is evaluated on the validation data using the selected performance metric.

Step 5 : Select the best hyperparameters: The hyperparameters that give the best performance on the validation data are selected as the final hyperparameters. These hyperparameters are then used to train the model on the entire training dataset.

Step 6 : Evaluate the final model: The final model is evaluated on the test data using the selected performance metric to determine the generalization performance of the model.

The model parameter tuning approach is an iterative process that involves repeating the above steps until the desired level of performance is achieved.

It is important to note that model parameter tuning can be time-consuming and computationally expensive, especially for large datasets and complex models.

## Question Bank

■ **Short-Answer Questions** **(3 or 4 marks) :**

1. Define model. How can you train a model ?
2. Give the difference between predictive model and descriptive model.
3. State any four real-world problems solved by predictive models. Explain any one in brief.

4. State any four real-world problems solved by descriptive models. Explain any one in brief.
5. Write down steps to use holdout method for model training.
6. Draw a detailed diagram to show the approach of 10-fold cross validation.
7. Give the difference between Bagging and Boosting.
8. Define overfitting. When does it happen?
9. Define underfitting. When does it happen?
10. Write a short note on bias-variance trade-off in context of model fitting.
11. Explain structure of confusion matrix.
12. Define following terms: Sensitivity, Specificity
13. Write a brief note on stacking.
14. State various ways to improve performance of a model. **(7 marks)**

■ **Long-Answer Questions**

1. Explain different types of model.
2. Explain Holdout method in detail.
3. Describe k-fold cross validation in detail.
4. Explain bagging, boosting and stacking in detail.
5. Describe Ensemble learning approach in detail.
6. Consider the following confusion matrix of the win/loss prediction of cricket match, calculate the accuracy, error rate, sensitivity, specificity, precision, recall and F-measure of the model.

| | Actual Win | Actual Loss |
|---|---|---|
| Predicted Win | 82 | 7 |
| Predicted Loss | 3 | 8 |

7. While predicting malignancy of tumour of a set of patients using a classification model, following are the data recorded:

   1. Correct predictions – 20 malignant, 70 benign
   2. Incorrect predictions – 4 malignant, 6 benign

   Create confusion matrix for the same. And, calculate the accuracy, error rate, sensitivity, specificity, precision, recall and F-measure of the model.

■ **Check your knowledge :**

1. What is the difference between bagging and boosting?
   - **a) Bagging reduces the variance of a model, while boosting reduces the bias of a model.**
   - b) Bagging reduces the bias of a model, while boosting reduces the variance of a model.
   - c) Bagging increases the complexity of a model, while boosting reduces the complexity of a model.
   - d) Bagging reduces the complexity of a model, while boosting increases the complexity of a model.
2. Which of the following is not a metric that can be calculated using a confusion matrix?
   - a) Precision
   - b) Recall
   - c) F1 score
   - **d) R-squared**
3. Which of the following is a common technique used to prevent overfitting in a machine learning model?
   - a) Adding more features to the model
   - b) Increasing the number of epochs during training
   - **c) Reducing the size of the model**
   - d) Decreasing the learning rate during
4. What is the holdout method used for?
   - **a) To evaluate the performance of a machine learning model on a dataset that it hasn't seen during training**
   - b) To train a machine learning model on a subset of a dataset
   - c) To generate new data for a machine learning model
   - d) To reduce the complexity of a machine learning model
5. Which of the following is a common technique used to prevent underfitting in a machine learning model?
   - **a) Adding more features to the model**
   - b) Increasing the number of epochs during training
   - c) Reducing the size of the model
   - d) Decreasing the learning rate during training

■