Unit 6
Modeling and Validation Processes

6.1 Introduction to machine learning

6.2 Introduction to supervised, unsupervised and reinforcement learning

6.3 Modeling process, training /validating model, cross validation methods, predicting new observations interpretation

6.4 Measures for model performance and evaluation: Classification accuracy, confusion matrix, sensitivity, specificity, precision, recall, F-score, ROC curve, clustering performance measures, other measures

Introduction to Machine Learning

It is said that the term machine learning was first coined by Arthur Lee Samuel, a pioneer in the AI field, in 1959 .

Machine Learning (ML) is a subset of artificial intelligence (AI) that enables systems to learn and make decisions without being explicitly programmed. It involves using algorithms to identify patterns, make predictions, or automate processes based on data. As data continues to grow exponentially, ML has become a crucial technology across industries, driving innovation and efficiency.

So what do explicitly progammed mean? Traditionally, when it comes to programming, it was a process in which a person specifically sets the rules, saying, "Do the B command in condition A, and perform the D command in the C condition." However, there is a limit to this explicit programming. For example, let's say you make a filter to filter spam. There are not many cases where the phrase "advertisement" is hung in the title. They intentionally make typos or use special characters to avoid being caught in the spam filter. If we want to filter out these cases one by one, we have to set too many rules. Also, let's consider the case of developing autonomous vehicles. Many emergencies occur while driving, so you have to be prepared for many unexpected anomalies. In order to solve problems that cannot be solved by existing explicit programming, machine learning, that is, an approach that allows machines to 'learn' certain patterns by themselves, has emerged.

Although it has been in the spotlight recently, machine learning itself is a very old concept. Arthur Samuel defined machine learning as follows in 1959.

"Field of study that gives computers the ability to learn without being explicitly programmed."

In short, machine learning is a way for the program itself to learn on its own through data, instead of pre-determining numerous rules directly by the programmer. As AI research progressed, machine learning techniques also developed dramatically, and are now being accepted as a major feature of electronics.

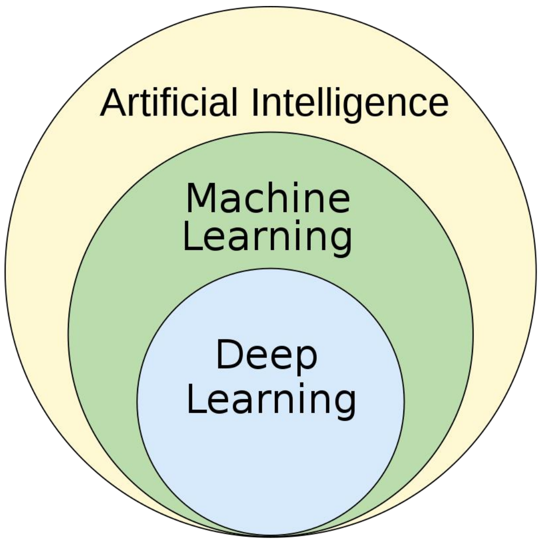
Formal Definition

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

✓ Handwriting recognition learning problem
- Task T: Recognising and classifying handwritten words within images
- Performance P: Percent of words correctly classified
- Training experience E: A dataset of handwritten words with given classifications

✓ A robot driving learning problem
- Task T: Driving on highways using vision sensors
- Performance measure P: Average distance traveled before an error
- training experience: A sequence of images and steering commands recorded while observing a human driver

Machine learning as subfield of AI



Goals of Machine Learning:

- Automation: Enable systems to perform tasks with minimal human intervention.
- Prediction: Make accurate forecasts or classifications based on data.
- Pattern Recognition: Identify complex patterns or structures in data.
- Adaptation: Learn and adapt to changes in the environment or data.
- Generalization: Develop models that perform well on unseen or new data (beyond the training set).
- Efficiency: Reduce computational complexity and optimize resource usage.
- Continuous Learning: Improve performance over time with more data or interaction.

Applications of Machine Learning:

Popular applications of machine learning include the following:

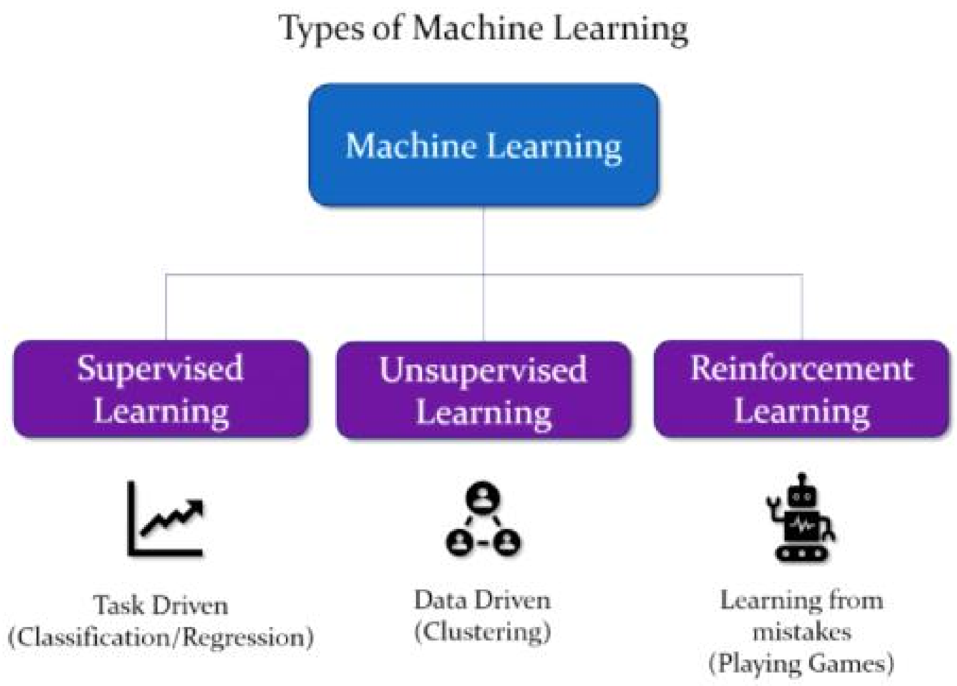
- Email spam detection
- Face detection and matching (e.g., iPhone X)
- Web search (e.g., DuckDuckGo, Bing, Google)
- Sports predictions
- Post office (e.g., sorting letters by zip codes)
- ATMs (e.g., reading checks)
- Credit card fraud
- Stock predictions
- Smart assistants (Apple Siri, Amazon Alexa, . . .)
- Product recommendations (e.g., Netflix, Amazon)
- Self-driving cars (e.g., Uber, Tesla)
- Language translation (Google translate)
- Sentiment analysis
- Drug design
- Medical diagnoses

Rule-Based Programming vs. Machine Learning

Rule-Based Programming	Machine Learning
Programming with explicitly defined rules and logic.	Systems learn patterns from data to make decisions.
Flexibility Limited; hard to adjust to new scenarios.	Flexible; adapts to new data and conditions.
Performance Depends on the programmer's ability to anticipate all scenarios.	Learns from data; can outperform humans in complex tasks.
E.g. Spam filter with keyword lists, tax calculation.	Spam filter using supervised learning, fraud detection.
Becomes complex and unwieldy with many rules. So, less scalable.	Scales well with large datasets and complexity.

Overview of the Categories of Machine Learning

At a broad level, machine learning can be classified into three types:


Types of Machine Learning
Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Task Driven (Classification/Regression)
Data Driven (Clustering)
Learning from mistakes (Playing Games)

1. Supervised Machine Learning:

As its name suggests, supervised machine learning is based on supervision.

• It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output.

• The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are Risk Assessment, Fraud Detection, Spam filtering, etc.

Categories of Supervised Machine Learning:

Supervised machine learning can be classified into two types of problems, which are given below:

✓ Classification

✓ Regression

Classification:

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "Yes" or No, Male or Female, Red or Blue, etc.

• The classification algorithms predict the categories present in the dataset.

• Some real-world examples of classification algorithms are Spam Detection, Email filtering, etc.
• Some popular classification algorithms are given below:
- ✓ Random Forest Algorithm
- ✓ Decision Tree Algorithm
- ✓ Logistic Regression Algorithm
- ✓ Support Vector Machine Algorithm

Regression:
Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables.
These are used to predict continuous output variables, such as market trends, weather prediction, etc.
• Some popular Regression algorithms are given below:
- ✓ Simple Linear Regression Algorithm
- ✓ Multivariate Regression Algorithm
- ✓ Decision Tree Algorithm
- ✓ Lasso Regression

Advantages and Disadvantages of Supervised Learning:

Advantages:
- ✓ Since supervised learning work with the labelled dataset so we can have an exact idea about the classes of objects.
- ✓ These algorithms are helpful in predicting the output on the basis of prior experience.

Disadvantages:
- ✓ These algorithms are not able to solve complex tasks.
- ✓ It may predict the wrong output if the test data is different from the training data.
- ✓ It requires lots of computational time to train the algorithm.

2. Unsupervised Machine Learning:
Unsupervised learning is different from the supervised learning technique; as its name suggests, there is no need for supervision.
It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output (w)

- ✓ The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences.
- ✓ Machines are instructed to find the hidden patterns from the input dataset.

Categories of Unsupervised Machine Learning:

Unsupervised Learning can be further classified into two types, which are given below:

- ✓ Clustering
- ✓ Association

1) Clustering:

The clustering technique is used when we want to find the inherent groups from the data.
It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups.
An example of the clustering algorithm is grouping the customers by their purchasing behavior.
Some of the popular clustering algorithms are given below:

- ✓ K-Means Clustering algorithm
- ✓ Mean-shift algorithm
- ✓ DBSCAN Algorithm
- ✓ Principal Component Analysis
- ✓ Independent Component Analysis

2) Association:

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.
Some popular algorithms of Association rule learning are:

- ✓ Apriori Algorithm,
- ✓ Eclat,
- ✓ FP-growth algorithm.

Advantages and Disadvantages of Unsupervised Learning Algorithm:

Advantages:

- ✓ These algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabeled dataset.
- ✓ Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is easier as compared to the labelled dataset.

Disadvantages:
- ✓ The output of an unsupervised algorithm can be less accurate as the dataset is not labelled, and algorithms are not trained with the exact output in prior.
- ✓ Working with Unsupervised learning is more difficult as it works with the unlabeled dataset that does not map with the output.

3. Semi-Supervised Learning:
Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning.

It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced.
- ✓ We can imagine these algorithms with an example. Supervised learning is where a student is under the supervision of an instructor at home and college.
- ✓ Further, if that student is self- analyzing the same concept without any help from the instructor, it comes under unsupervised learning.
- ✓ Under semi-supervised learning, the student has to revise himself after analyzing the same concept under the guidance of an instructor at college.

Advantages:
- ✓ It is simple and easy to understand the algorithm.
- ✓ It is highly efficient.
- ✓ It is used to solve drawbacks of Supervised and Unsupervised Learning algorithms.

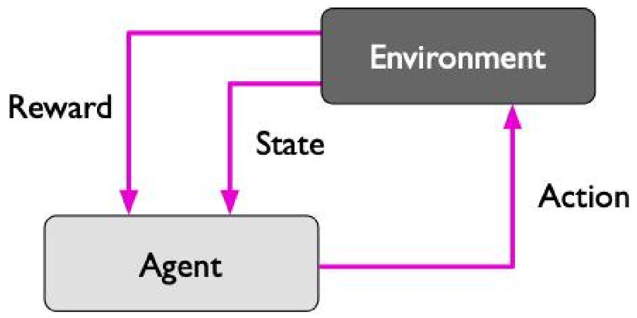
Disadvantages:
- ✓ Iterations results may not be stable.
- ✓ We cannot apply these algorithms to network-level data.
- ✓ Accuracy is low.

4. Reinforcement Learning:

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance.

Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.



The reinforcement learning process is similar to a human being; for example, a child learns various things by experiences in his day-to-day life.

- ✓ An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score.
- ✓ Agent receives feedback in terms of punishment and rewards.
- ✓ Due to its way of working, reinforcement learning is employed in different fields such as Game theory, Operation Research, Information theory, multi-agent systems.

Agents are the decision-maker that interacts with the environment to learn a strategy or behavior.

Environment is the external system or world within which the agent operates and interacts.

State is a snapshot of the environment that provides the agent with all the information it needs at a given moment.

Action is a decision or move made by the agent to interact with the environment.

Reward is a scalar feedback signal from the environment that indicates the immediate benefit or penalty of an action.

Policy (π) is a mapping from states to actions that defines the agent's behavior.

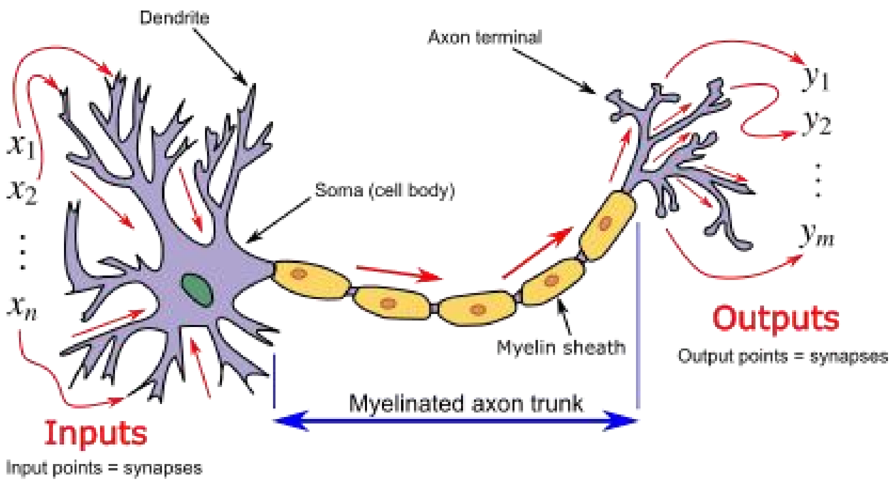
Real-world Use cases of Reinforcement Learning

- ✓ Video Games
- ✓ Robotics
- ✓ Autonomous Vehicles

Assignment: Difference Between Supervised and Unsupervised Machine Learning:

Details of ANN

The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



The given figure above illustrates the typical diagram of Biological Neural Network.

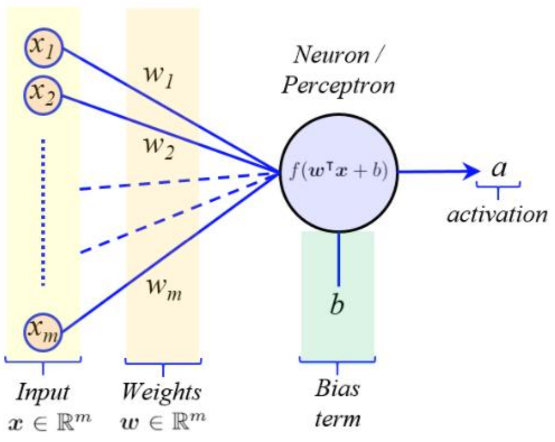
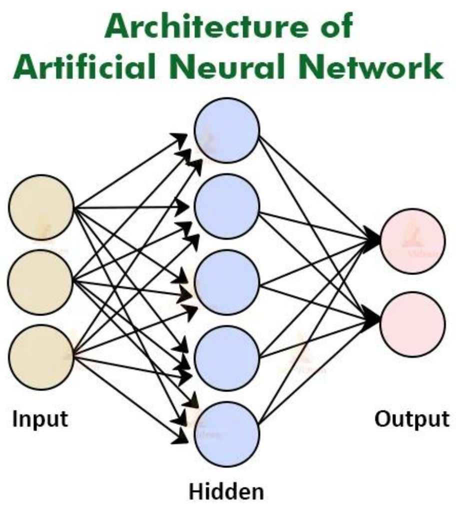


Illustration of a single neuron/perceptron in a standard ANN

Biological Neural Network	Artificial Neural Network
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

Architecture of ANN



A standard Neural Network model (also known as Artificial Neural Network) is organized into three parts: an input layer, an output layer and hidden layers as explained below

• An input layer receives the initial data or features that need to be processed. Each neuron in the input layer corresponds to features of the data. Depending on the specific application, the input layer may preprocess the input data to make it suitable for neural net training.

• An output layer provides the final result of the neural network's computation. The number of neurons in the output layer depends on the task at hand. For example, in the case of multi-class classification, the output layer might contain as many nodes as classes.

• Hidden layers are intermediate layers between the input and output layers. They perform computations on the input data through weighted connections between neurons and transformed using an activation function. The hidden layers, in essence, are responsible for extracting features and patterns from the data.

Working:

The operation of Artificial Neural Networks (ANNs) is inspired by the way biological neurons work in the human brain. ANNs consist of layers of interconnected nodes, or artificial neurons, each with a set of weights and biases. The working of ANNs involves the following steps:

✓ Input Layer: The input layer receives raw data or features from the input source. Each input is associated with a weight, indicating its importance.

- ✓ Hidden Layers: In the hidden layers, each neuron processes the weighted sum of inputs and biases using an activation function(Sigmoid, ReLU, Tanh, Leaky ReLU). The activation function introduces non-linearity, allowing the network to capture complex patterns.
- ✓ Weights and Biases Adjustment: During the learning phase, the network adjusts weights and biases through a process called back propagation. It compares the network's output to the desired output, calculates the error, and adjusts weights to minimize this error.
- ✓ Output Layer: The processed information propagates through the hidden layers to the output layer. The output layer provides the final prediction or classification result.
- ✓ Training: The network iteratively updates weights and biases using training data to minimize the prediction error. This process fine-tunes the network's ability to make accurate predictions.

Details on Naive Bayes:

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
Some popular application of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Naïve Bayes is a classification algorithm that gets its name from two key aspects:

Naïve: This refers to the core assumption of the algorithm: feature independence. It assumes that the presence or absence of a particular feature in a class is unrelated to the presence or absence of any other feature.

Example: If you're classifying fruits based on color, shape, and taste, the Naïve Bayes assumption would be that the color of a fruit is independent of its shape and taste. While this is often not strictly true in reality, it simplifies the calculations and makes the algorithm efficient.

Bayes: This acknowledges that the algorithm is founded on Bayes' theorem. Bayes' theorem provides a way to calculate the probability of an event (e.g., the class of an object) given prior knowledge or evidence (the observed features).

Math behind Naive Bays Algorithm

Given a features vector X=(x1,x2,⋯,xn) and a class variable y, Bayes Theorem states that:

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

We’re interested in calculating the posterior probability P(y ∣ X) from the likelihood P(X ∣ y) and prior probabilities P(y),P(X).

Using the chain rule, the likelihood P(X ∣ y) can be decomposed as:

$$P(X|y) = P(x_1, x_2, \ldots, x_n|y)$$

$$= P(x_1|x_2, \ldots, x_n, y) * P(x_2|x_3, \ldots, x_n, y) \ldots P(x_n|y)$$

but because of the Naive's conditional independence assumption, the conditional probabilities are independent of each other.

$$P(X|y) = P(x_1|y) * P(x_2|y) \ldots\ldots P(x_n|y)$$

Thus, by conditional independence, we have:

$$P(y|X) = \frac{P(x_1|y) * P(x_2|y) \ldots\ldots P(x_n|y) * P(y)}{P(x_1) * P(x_2) \ldots P(x_n)}$$

And as denominator remains constant for all values, the posterior probability can then be:

$$P(y|x_1, x_2, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

The Naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that's most probable; this is known as the maximum a posteriori or MAP decision rule.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Advantages of Naive Bayes: The following are some of the benefits of the Naive Bayes classifier:

- ✓ It is simple and easy to implement
- ✓ It doesn't require as much training data

- ✓ It handles both continuous and discrete data
- ✓ It is highly scalable with the number of predictors and data points
- ✓ It is fast and can be used to make real-time predictions
- ✓ It is not sensitive to irrelevant features

Disadvantages of Naive Bayes:

- ✓ Naive Bayes assumes that all predictors (or features) are independent, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases.
- ✓ This algorithm faces the 'zero-frequency problem' where it assigns zero probability to a categorical variable whose category in the test data set wasn't available in the training dataset. It would be best if you used a smoothing technique to overcome this issue.
- ✓ Its estimations can be wrong in some cases, so you shouldn't take its probability outputs very seriously.

Details of K means clustering and density-based clustering

K-Means Clustering

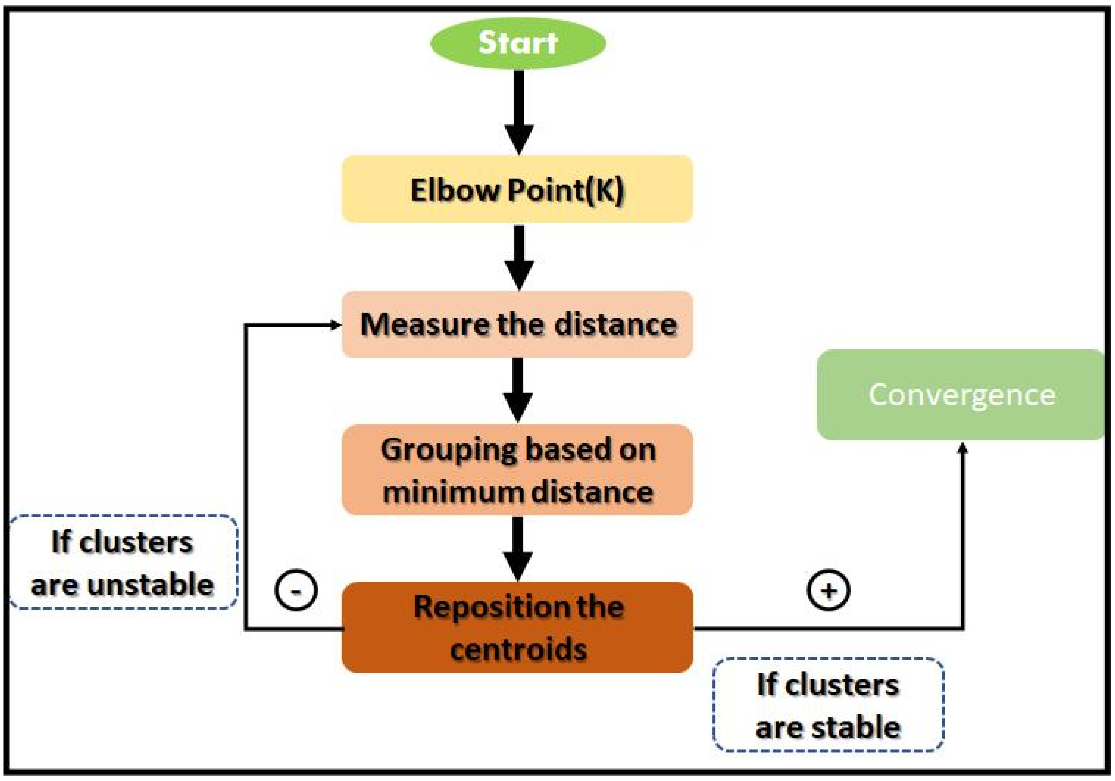
K-means clustering is a way of grouping data based on how similar or close the data points are to each other. Imagine you have a bunch of points, and you want to group them into clusters. The algorithm works by first randomly picking some central points (called centroids) and then assigning every data point to the nearest centroid.

Once that's done, it recalculates the centroids based on the new groupings and repeats the process until the clusters make sense. It's a pretty fast and efficient method, but it works best when the clusters are distinct and not too mixed up. One challenge, though, is figuring out the right number of clusters (K) beforehand. Plus, if there's a lot of noise or overlap in the data, K Means might not perform as well.

Objective of K-Means Clustering

- ✓ Grouping Similar Data Points
- ✓ Minimizing Within-Cluster Distance
- ✓ Maximizing Between-Cluster Distance

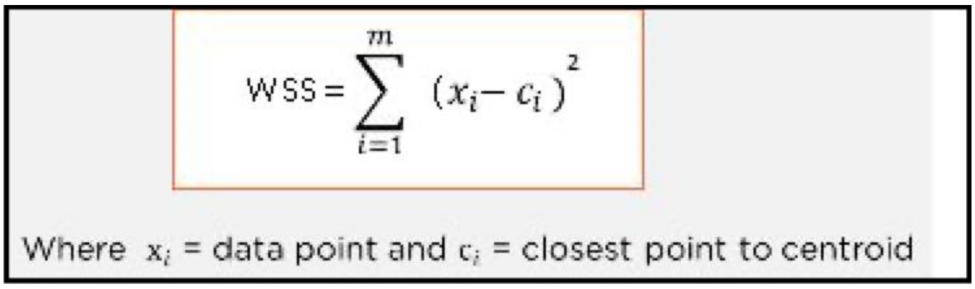
Working:



Step 1: Finding Optimal value of K

The Elbow method is the best way to find the number of clusters.

Next, we use within-sum-of-squares as a measure to find the optimum number of clusters that can be formed for a given data set. Within the sum of squares (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid.

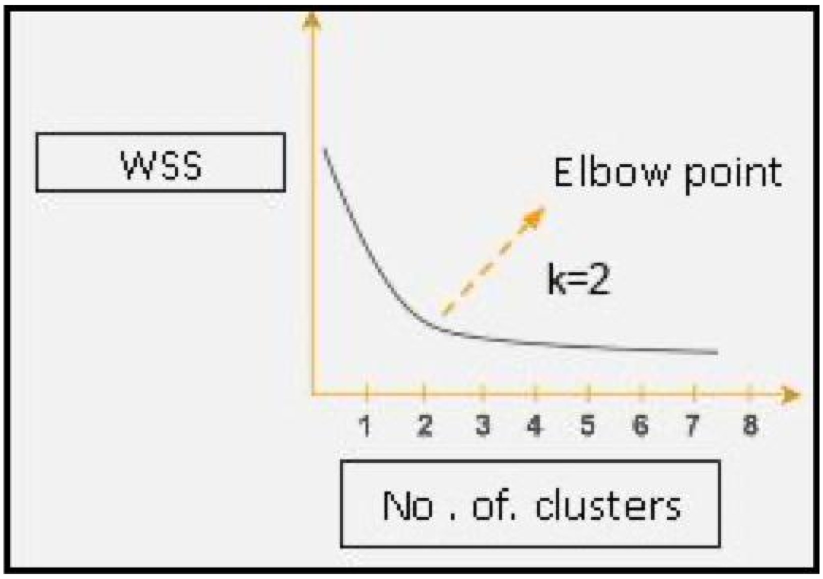


$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$

Where x_i = data point and c_i = closest point to centroid

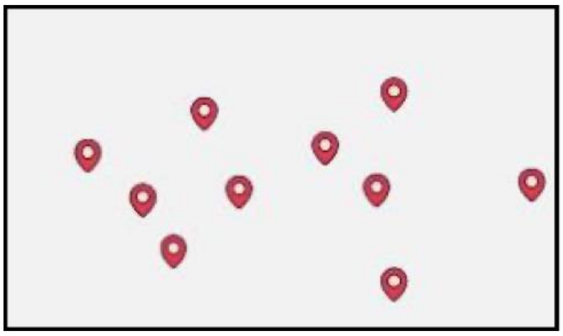
The WSS is measured for each value of K. The value of K, which has the least amount of WSS, is taken as the optimum value.

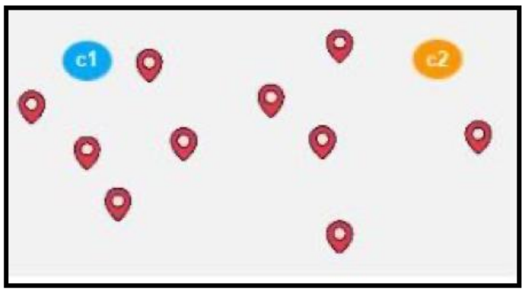
Now, we draw a curve between WSS and the number of clusters.



You can see that there is a very gradual change in the value of WSS as the K value increases from 2.
So, you can take the elbow point value as the optimal value of K. It should be either two, three, or at most four. But, beyond that, increasing the number of clusters does not dramatically change the value in WSS, it gets stabilized.

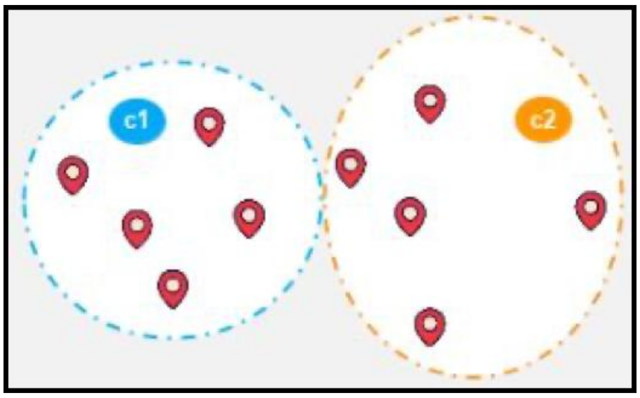
Step 2:
Let's assume that these are our delivery points:





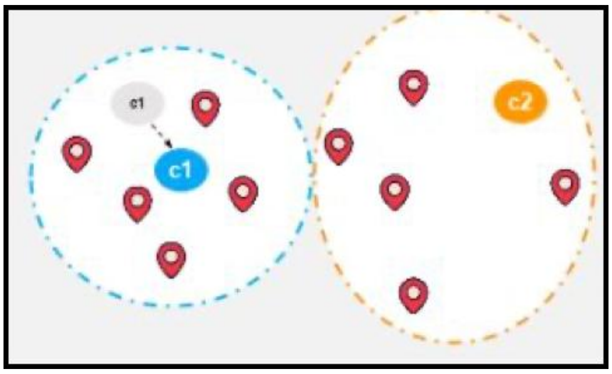
We can randomly initialize two points called the cluster centroids. Here, C1 and C2 are the centroids assigned randomly.

Now the distance of each location from the centroid is measured, and each data point is assigned to the centroid, which is closest to it.
This is how the initial grouping is done:



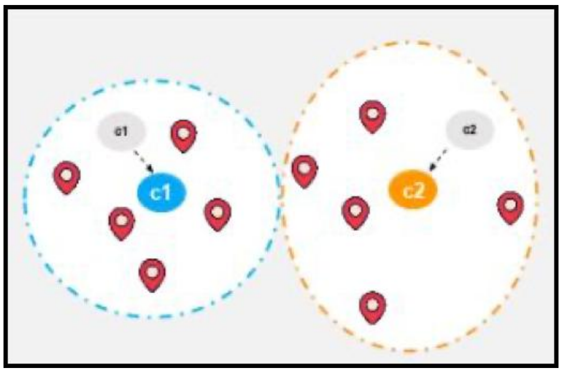
Step 4:
Compute the actual centroid of data points for the first group.

Step 5:
Reposition the random centroid to the actual centroid.



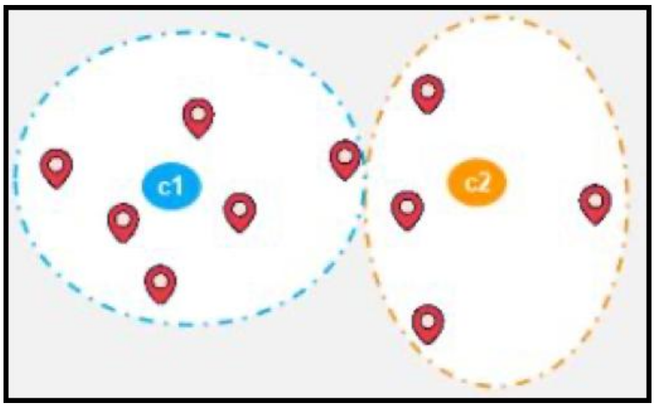
Step 6:
Compute the actual centroid of data points for the second group.

Step 7:
Reposition the random centroid to the actual centroid.



Step 8:
Once the cluster becomes static, the k-means algorithm is said to be converged.

The final cluster with centroids c1 and c2 is as shown below:



Advantages of K-means:
- Simple and easy to implement,
- Fast and efficient,
- Scalability(scaled to handle even larger datasets),
- Flexibility(used with varying metrics of distance and initialization methods).

Disadvantages of K-means:
- Sensitivity to initial centroids: K-means is sensitive to the initial selection of centroids and can converge to a suboptimal solution.
- Requires specifying the number of clusters: The number of clusters k needs to be specified before running the algorithm, which can be challenging in some applications.
- Sensitive to outliers: K-means is sensitive to outliers, which can have a significant impact on the resulting clusters.

Density-based clustering (DBSCAN)

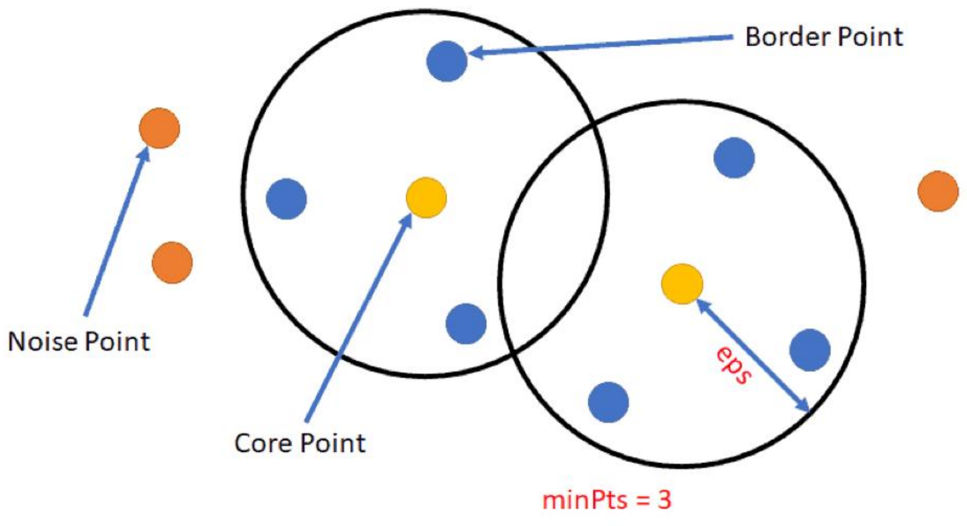
DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise.

Density-based clustering means that the algorithm looks for areas in the data where points are densely packed (i.e., many data points are close to each other).
These dense areas are considered clusters, and the empty or less dense areas are considered as separation between clusters.

The main idea is that points that are close to each other (in dense regions) belong to the same cluster, while points far from others (in low-density regions) are considered outliers or noise.

It also does not require the number of clusters to be told beforehand.

The DBSCAN algorithm requires only 2 parameters: epsilon and minPoints.
Epsilon is the radius of the circle to be created around each data point to check its density, and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point.



Algorithm:

DBSCAN operates by examining the neighborhood of each point in the dataset. The algorithm follows a step-by-step process to identify clusters based on the density of data points.

1. Parameter Selection
Choose ε (epsilon): The maximum distance between two points for them to be considered as neighbors.
Choose MinPts: The minimum number of points required to form a dense region.

2. Select a Starting Point
The algorithm starts with an arbitrary unvisited point in the dataset.

3. Examine the Neighborhood
- It retrieves all points within the ε distance of the starting point.
- If the number of neighboring points is less than MinPts, the point is labeled as noise (for now).
- If there are at least MinPts points within ε distance, the point is marked as a core point, and a new cluster is formed.

4. Expand the Cluster
-All the neighbors of the core point are added to the cluster.
- For each of these neighbors:
    - If it's a core point, its neighbors are added to the cluster recursively.
    - If it's not a core point, it's marked as a border point, and the expansion stops.

5. Repeat the Process
- The algorithm moves to the next unvisited point in the dataset.
- Steps 3-4 are repeated until all points have been visited.

6. Finalize Clusters
- After all points have been processed, the algorithm identifies all clusters.
- Points initially labeled as noise might now be border points if they're within ε distance of a core point.

7. Handling Noise
- Any points not belonging to any cluster remain classified as noise.

This process allows DBSCAN to form clusters of arbitrary shapes and identify outliers effectively. The algorithm's ability to find clusters without specifying the number of clusters beforehand is one of its key strengths.

It's important to note that the choice of ε and MinPts can significantly affect the clustering results.

Advantages of DBSCAN:

- ✓ Can find clusters of arbitrary shapes.
- ✓ Robust to noise and outliers.
- ✓ No need to specify the number of clusters in advance.

Disadvantages of DBSCAN:

- ✓ The choice of ε and MinPts can significantly affect the results.
- ✓ Struggles with clusters of varying density.
- ✓ Sensitive to the scale of the data, especially in high-dimensional spaces.

Modeling process:

Modeling is a multi-stage methodology for creating trained and tested Machine Learning and AI models.
The Modeling Process is essentially a scientific experiment which includes:

- ✓ Development of a Hypothesis - e.g., data collected about a specific previous consumer behavior can be used to predict future behavior
- ✓ Design of the Experiment - e.g., model/algorithm selection
- ✓ Execution of the Experiment - e.g., model training and testing
- ✓ Evaluation and Explanation of Results - e.g., is the hypothesis true or false, what is the accuracy

Modeling process, which can be highly recursive/iterative, generally include:

- ✓ Type Identification
- ✓ Platform Selection
- ✓ Data Collection
- ✓ Model/Algorithm Selection
- ✓ Model Hyperparameters Setting
- ✓ Model Training
- ✓ Model Testing
- ✓ Model Evaluation
- ✓ Model Deployment

1. Type Identification:

The type of ML/AI needed can have a significant influence on the details of modelng process phases. Type identification can be driven by:

- ✓ Applications Needed
- ✓ Areas of Interest
- ✓ Educational Needs
- ✓ Research and Development

Major category types include:

- ✓ Computer Vision (e.g., object recognition, facial recognition, handwriting recognition)
- ✓ Natural Language Processing (e.g., speech to text, translation, understanding)
- ✓ Pattern Recognition (e.g., event prediction, medical diagnosis)
- ✓ Generative AI (e.g. Large Language Models)

Platform Selection:

Choosing the tools, environment, and hardware for the project, such as:

- ✓ Programming languages (e.g., Python, R).
- ✓ Machine learning libraries (e.g., TensorFlow, Scikit-learn, PyTorch).
- ✓ Cloud services or infrastructure (e.g., AWS, Azure, Google Cloud).
- ✓ Hardware (e.g., GPUs for deep learning).

2. Data Collection:

Data Collection is the processing of finding, organizing, cleaning, and storing data in a form that can be fed into model training and prediction processing. Gathering the data required for the model from various sources such as:

- ✓ Public datasets.
- ✓ Company databases.
- ✓ APIs or web scraping.
- ✓ Sensors or other devices.

Data Collection can involve:

- ✓ Data Discovery
- ✓ Data Flow
- ✓ Data Pre-processing and transformation.
- ✓ Data Cleaning
- ✓ Data ETL (Extract, Transform, Load)
- ✓ Databases

3. Model/Algorithm Selection

Several alogithm like ANN, SVM, Linear Regression, Logistic Regression, etc, exits.

But Methods of selecting an algorithm include:

- Identifying Project Key Criteria - often include model application, need for model explainability and interpretability, training data availability
- Review Model Categories: Look at different types of models (e.g., decision trees, neural networks) and choose based on your problem and data.
- Research Latest Advancements: Stay updated on new methods and tools.
- Experiment with Options: Test different algorithms on your data.
- Compare Models: Track performance (e.g., accuracy)

4. Model Hyperparameters Setting :

Hyperparameters control aspects of model instantiation and training and can include factors, depending on the model algorithm being used, such as: activation_function, batch_size, learning_rate, etc.

5. Model Training:

Split the data into training and test sets (typically an 80/20 or 70/30 split). Then Training Data is iteratively processed through the model to adjust the weights and biases applied to data array links to produced increasingly more accurate output results. Typically the training process is performed iteratively while monitoring for factors such as best accuracy results.

6. Model Testing:

After training, the model is evaluated on a separate test dataset that it hasn't seen during training. This helps assess how well the model can generalize to new, unseen data. Common testing methods include using a train-test split or cross-validation.

7. Model Evaluation:

Assessing the model's performance using relevant metrics. These metrics depend on the problem type, such as:

- ✓ Accuracy, Precision, Recall, F1-Score for classification.
- ✓ Mean Squared Error (MSE) or R-squared for regression.
- ✓ Confusion Matrix, ROC-AUC, etc., for model diagnostics.

This evaluation helps you understand the strengths and weaknesses of the model.

8. Model Deployment and Improvement:

Deploying the trained model to production where it can be used to make predictions on new data.

Common methods include:

- ✓ Exposing the model via an API (using Flask, FastAPI, or similar).
- ✓ Deploying to cloud services like AWS Lambda, Google Cloud AI, or Azure ML.
- ✓ Monitoring the model in production to ensure it continues to perform well and retraining it as needed with new data.

Reference:

https://www.ml-science.com/modeling-process

Training and Validating the Model

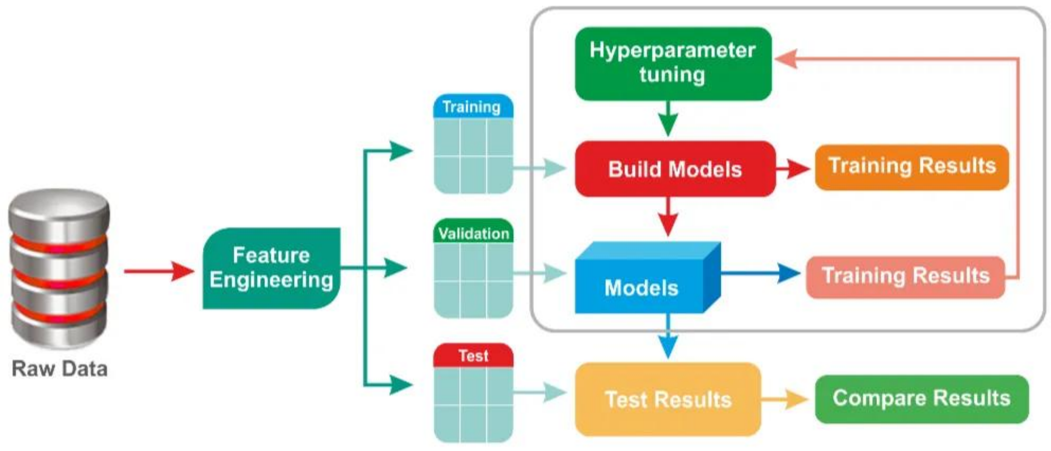
The model is trained by using an optimization algorithm to minimize a loss function (such as mean squared error for regression or cross-entropy for classification).
During training, the model's parameters (e.g., coefficients, weights) are updated to reduce the error in predictions.

Validation is crucial for evaluating how well the model generalizes to unseen data. If the model performs well on the training set but poorly on the validation set, it may have overfit to the training data. On the other hand, it may underfit, failing to capture the necessary patterns.

Validation metrics vary based on the task:

✓ Regression: Metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared are common.

✓ Classification: Metrics like accuracy, precision, recall, F1-score, and AUC-ROC are used.



Training Loss and Validation Loss: A Crucial Duo

In machine learning, particularly deep learning, understanding the difference between training loss and validation loss is paramount. These two metrics provide crucial insights into how well your model is learning and generalizing.

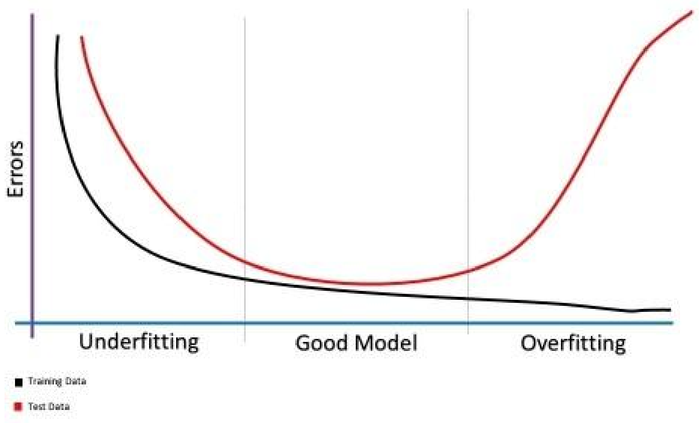
Training Loss is the error your model makes on the training data. It's calculated after each batch of training examples.
Validation Loss is the error your model makes on a separate dataset (the validation set) that it hasn't seen during training.

By monitoring both training and validation loss, you can:

- ✓ Detect overfitting: Take steps to prevent it, such as early stopping or regularization.
- ✓ Choose the optimal model: Select the model with the lowest validation loss.
- ✓ Tune hyperparameters: Adjust hyperparameters to improve both training and validation loss.

Key Points:

- ✓ Overfitting: When the training loss continues to decrease, but the validation loss starts to increase, it's a sign of overfitting. The model is memorizing the training data too well and performs poorly on new data.
- ✓ Underfitting: If both the training and validation loss are high and don't improve significantly, it's a sign of underfitting. The model is too simple to capture the underlying patterns in the data.
- ✓ Optimal Model: The point where the validation loss is at its minimum is often considered the optimal point for the model.

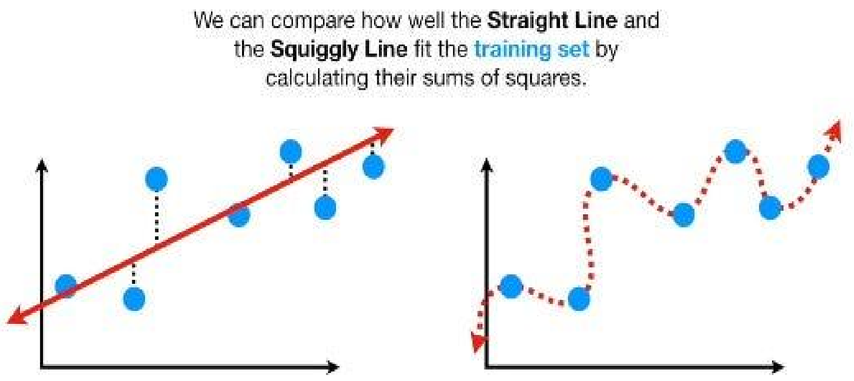


Bias-Variance Trade off

It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine learning algorithm. There is a tradeoff between a model's ability to minimize bias and variance which is referred to as the best solution for selecting a value of Regularization constant. Proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

Bias

The bias is known as the difference between the prediction of the values by the ML model and the correct value. Being high in biasing gives a large error in training as well as testing data. Its recommended that an algorithm should always be low biased to avoid the problem of underfitting.



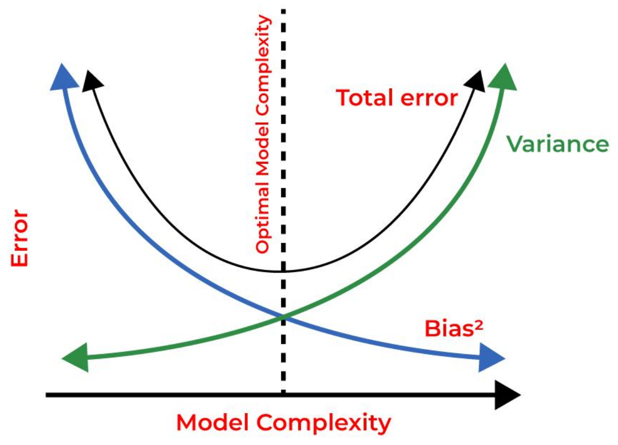
When the model tries to reduce bias it tends to overfit the data. Left fig have high bias & right fig have low bias.

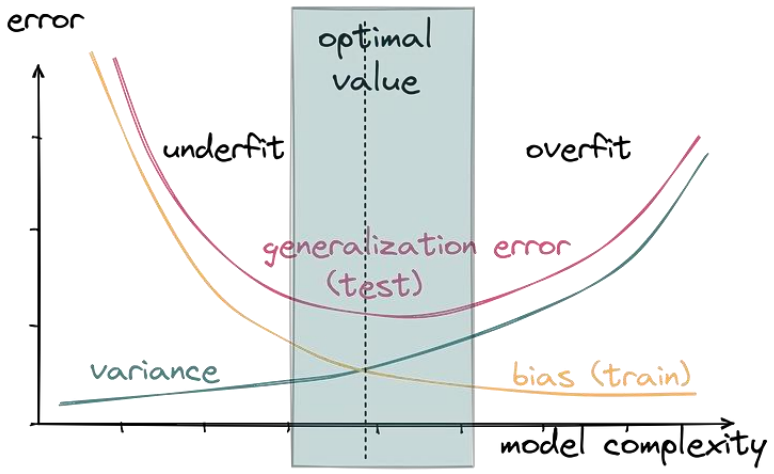
Variance

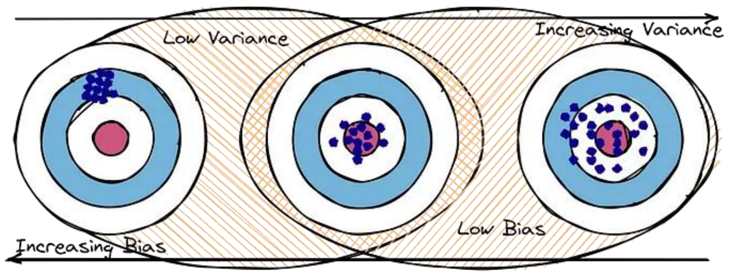
The variability of model prediction for a given data point which tells us spread of our data is called the variance of the model. The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.When a model is high on variance, it is then said to as Overfitting of Data.

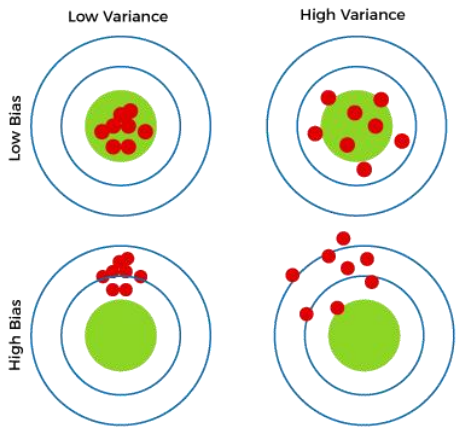
It is impossible to completely get rid of both bias and variance and have a perfect fit of the training and validation curve. There will always be some error. So how to minimize the error and achieve a good accuracy of the model is an important question. The answer is a tradeoff between bias and variance. Instead of completely getting rid of both bias and variance it is wise to balance them both to minimize the error. It is called a bias vs variance tradeoff.

The error to complexity graph to show trade-off is given as –


Optimal Model Complexity
Total error
Variance
Error
Bias²
Model Complexity


error
optimal value
underfit
overfit
generalization error (test)
variance
bias (train)
model complexity


Low Variance
Increasing Variance
Low Bias
Increasing Bias


Low Variance
High Variance
Low Bias
High Bias

Prediction on New Observations and Interpretation of Predictions:

Once the model is trained, we need to test the model on the test dataset to evaluate its performance on unseen data.

✓ Prediction Output:

In regression, the model will output a continuous value (e.g., a price, a temperature).
In classification, the output will be a discrete class label or probability distribution over possible classes.

✓ Model deployment:

Key Steps:

1. Model Selection: Choose the best-performing model based on validation results.
2. Model Serialization: Save the model's parameters and architecture in a suitable format (e.g., pickle, joblib, ONNX) for later use.
3. Deployment Platform: Select a deployment platform (e.g., cloud services like AWS SageMaker, Azure ML, or Google AI Platform; containerization with Docker and Kubernetes; or local deployment).
4. API Creation: Create an API endpoint that allows other applications to interact with the deployed model.
5. Monitoring and Maintenance: Continuously monitor the model's performance and retrain it as needed to maintain accuracy.

✓ Model inference: It is the process of using a deployed model to make predictions on new, unseen data.

Key Steps:

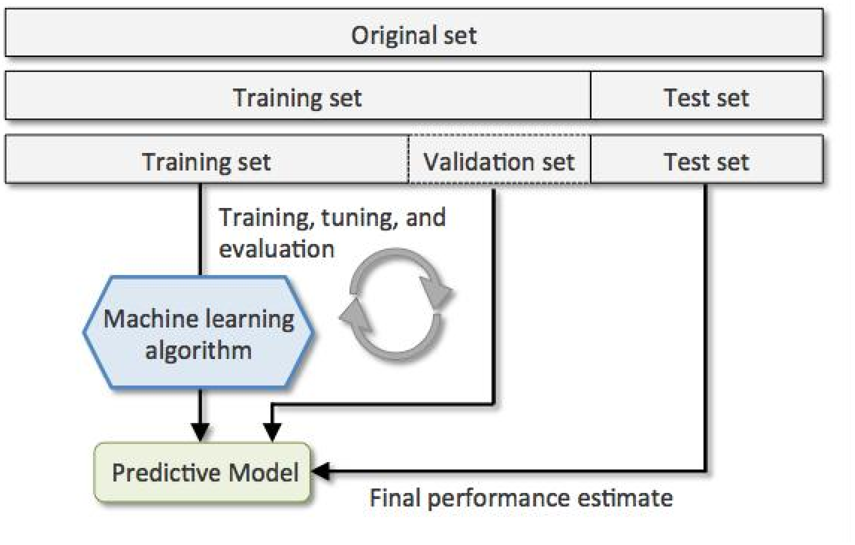
1. Data Preprocessing: Prepare the new data in the same format as the training data (e.g., scaling, encoding categorical variables).
2. Model Loading: Load the serialized model from the deployment platform.
3. Prediction: Use the loaded model to make predictions on the new data.
4. Post-processing: Convert the model's output into a human-readable format or take further actions based on the prediction.

Cross-Validation Techniques

Whenever we build a machine learning model, we typically split the original dataset into a Training set and a Test set. The training set is further divided into two parts: a Training set and a Validation set. The training set is used to train the model, while the validation set is used to evaluate the model's performance during training. If the model's performance is not satisfactory, hyperparameter tuning or other adjustments are made to improve it. After training and tuning, the model is tested on the Test set, which contains unseen data. The performance on this test set indicates how well the model generalizes to new, unseen data.

If the model performs well on the test set, it indicates that the model is stable and capable of making accurate predictions.

However, this process does not always guarantee a stable model. Machine learning models may not always generalize well, leading to issues like overfitting (where the model learns too much from the training data, capturing noise rather than general patterns). This is where Cross-Validation comes into play.


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

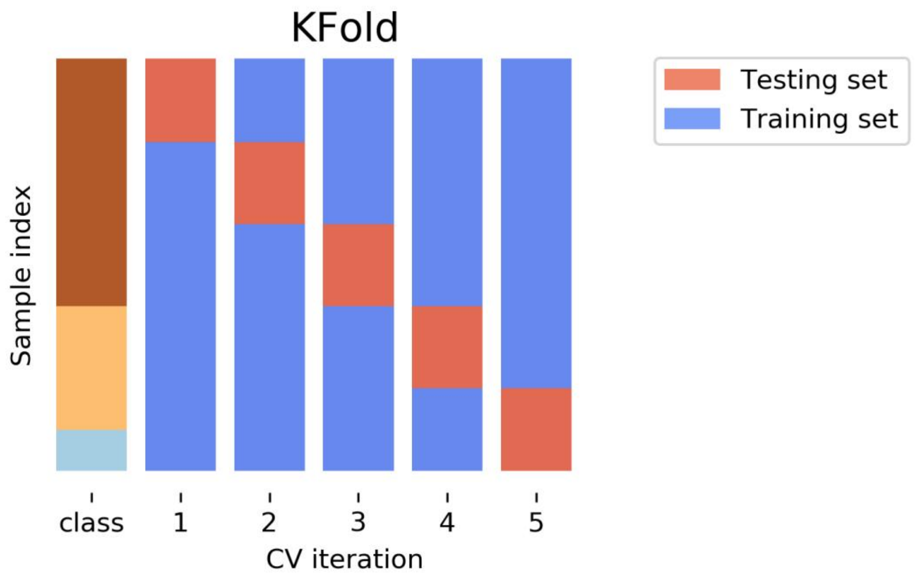
Note: The method explained above is also called HoldOut Method.

Cross-validation involves splitting the data into multiple subsets (or folds) and training/testing the model on different combinations of these subsets. This ensures that the model is evaluated on multiple different training and validation sets, providing a more reliable estimate of its performance and helping to identify overfitting. By using cross-validation, the model is trained and tested on different portions of the data, giving a more comprehensive view of its stability and performance across different datasets.

There are some common methods that are used for cross-validation. These methods are given below:

1. K-folds
2. Stratified K-folds
3. Leave-one-out
4. Leave-p-out

1. K-folds:



In k-fold cross-validation, the data is divided into k subsets, and the model is trained and validated k times. Each time, one of the k subsets is used as the test set, and the remaining k-1 subsets form the training set. The final performance is the average of the k trials.

How to set the Value of K in K-fold cross-validation?

Choose a value for 'k' so the model can avoid large variance and significant distortion. In most cases, the choice of k is usually 5 or 10, but there is no formal rule. However, the Value of k depends on the size of the dataset.

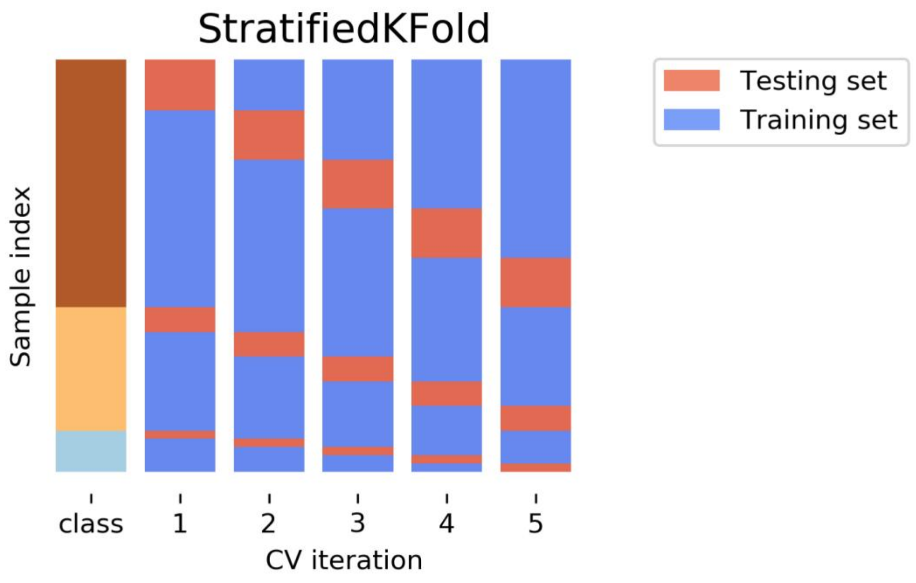
Advantages of K-fold cross validation:

- ✓ Overfitting: This prevents the overfitting of the training data set.
- ✓ Better estimate of model performance: Since each data point is used for both training and validation, K-fold cross-validation provides a more reliable and stable estimate of model performance compared to using a single training-validation split.

Disadvantages of K-fold cross validation:

✓ Don't work on an imbalanced dataset: If your data is imbalanced (if you have class "A" and class "B," the training set has class "A" and the test set has class "B"), it doesn't work well.
✓ Increased training time: Cross-validation requires training the model on multiple training sets.
✓ Computationally expensive: Cross-validation is computationally expensive as it needs to be trained on multiple training sets.

2. Stratified K-folds:



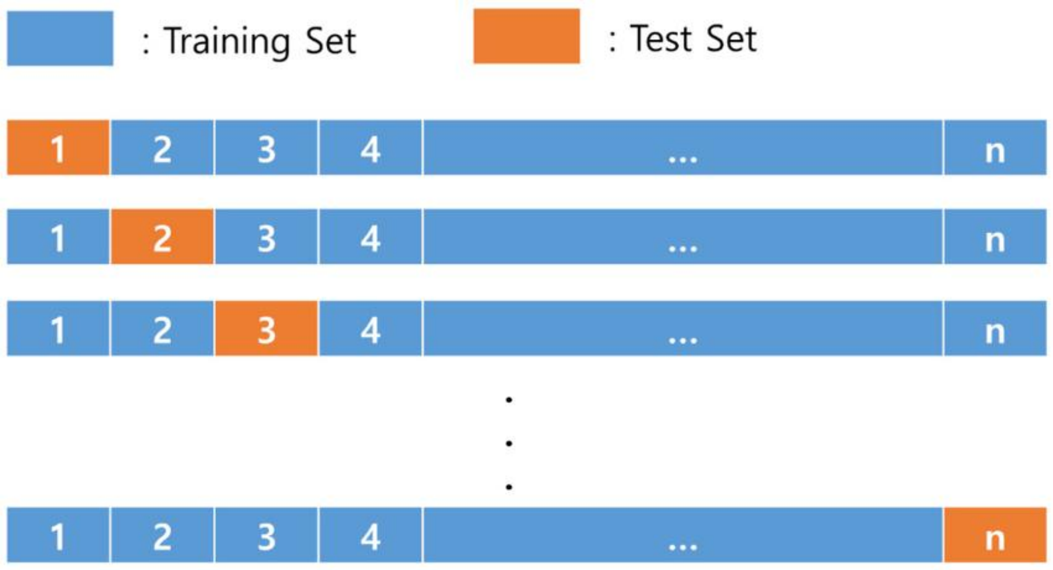
This technique is similar to k-fold cross-validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.
It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k-fold cross-validation technique is useful.

Advantages of Stratified K-fold cross validation:
✓ Effective to data imbalance.
✓ Helps in improving generalization by ensuring the model sees a variety of samples from different classes.

Disadvantages of Stratified K-fold cross validation:
✓ More computationally intensive than standard k-fold cross-validation.
✓ May not provide additional benefits if the dataset is already balanced.

3. Leave-one-out:(LOOCV)

In this approach, for each learning set, only one data point is reserved, and the remaining dataset is used to train the model. This process repeats for each data point. Hence for n samples, we get n different training set and n test set.



Advantages of Leave-one-out cross validation:

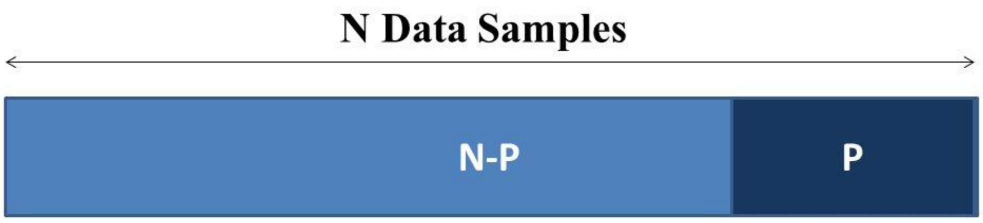
- ✓ All data points are taken as test data once hence it has a low bias. Low bias here mean a model has minimal systematic error in its predictions. i.e the model is capable of capturing the underlying patterns in the data well.
- ✓ Useful for small datasets where maximizing the use of available data is critical.

Disadvantages of Leave-one-out cross validation:

- ✓ Computationally expensive for large datasets due to n iterations.
- ✓ Testing against a single data point at a time can lead to unstable performance metrics.i.e High variance
- ✓ High execution time

4. Leave-p-out cross-validation

This approach separates 'P' data points out of training data, i.e.



Where,
N-P= training data samples
P=testing samples
N=Data samples

Advantages of Leave-p-out cross validation:

- ✓ All data points are used multiple times for both training and testing, ensuring no data is wasted. It evaluates the model on all possible subsets of the data.
- ✓ Flexibility Testing Proportion: By adjusting p, you can control the size of the test set to match the specific requirements of your problem.
- ✓ LpOCV offers a better understanding of how the model performs across various data combinations, helping to identify overfitting or underfitting.

Disadvantages of Leave-p-out cross validation:

- ✓ Computationally Expensive: For a dataset with n samples, the number of possible p-sized subsets is $\binom{n}{p}$, which grows exponentially as n and p increase.
- ✓ High Variance: Testing on small subsets (especially for low p) can lead to high variance in performance metrics
- ✓ Setting up LpOCV can be more challenging compared to simpler cross-validation methods, especially for datasets where n is large, and combinations need to be generated efficiently.

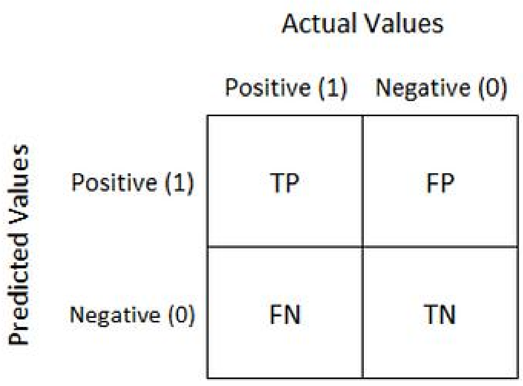
Measures for model performance and evaluation

Evaluating the performance of a Machine learning 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. These performance metrics help us understand how well our model has performed for the given data. In this way, 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.
Below is an overview of the key performance measures for classification, regression, clustering, and other tasks:

1. Performance Metrics for Classification:

Confusion matrix

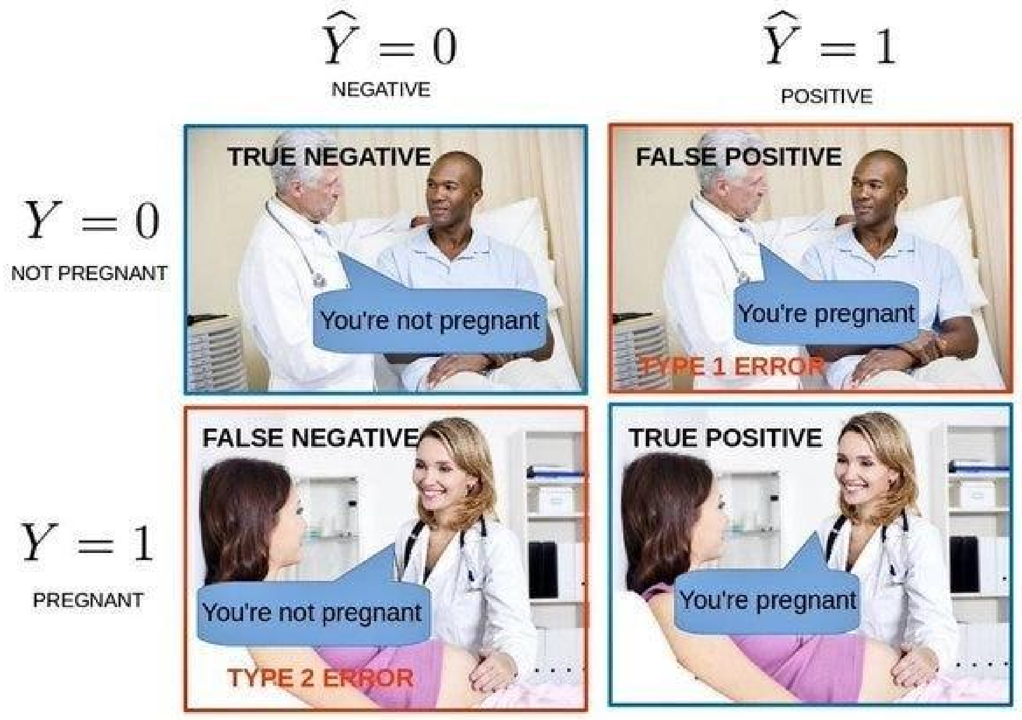
The confusion matrix, also known as the error matrix or contingency table, is a fundamental tool in the evaluation of the performance of classification algorithms. A confusion matrix is a table that outlines the performance of a classification model by displaying the actual versus predicted classifications. It comprises four components: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC Curve.

Let's try to understand TP, FP, FN, TN with an example of pregnancy analogy.



- ✓ True Positive: We predicted positive and it's true. In the image, we predicted that a woman is pregnant and she actually is.

- ✓ True Negative: We predicted negative and it's true. In the image, we predicted that a man is not pregnant and he actually is not.
- ✓ False Positive (Type 1 Error): We predicted positive and it's false. In the image, we predicted that a man is pregnant but he actually is not.

- ✓ False Negative (Type 2 Error): We predicted negative and it's false. In the image, we predicted that a woman is not pregnant but she actually is.

Classification accuracy : It is one of the important parameters to determine the accuracy of the classification problems. It defines how oftern the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifer to all number of predictions made by the classifers.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

When to Use Accuracy? It is good to use the Accuracy metric when the target variable classes in data are approximately balanced. For example, if 60% of classes in a fruit image dataset are of Apple, 40% are Mango. In this case, if the model is asked to predict whether the image is of Apple or Mango, it will give a prediction with 97% of accuracy.

When not to use Accuracy?
It is recommended not to use the Accuracy measure when the target variable majorly belongs to one class. For example, Suppose there is a model for a disease prediction in which, out of 100 people, only five people have a disease, and 95 people don't have one. In this case, if our model predicts every person with no disease (which means a bad prediction), the Accuracy measure will be 95%, which is not correct.

Precision

The precision metric is used to overcome the limitation of Accuracy. Precision measures the ratio of correctly predicted positive observations to the total predicted positives. It explains how many of the correctly predicted cases actually turned out to be positive.

Precision is useful in the cases where False Positive is a higher concern than False Negatives. The importance of Precision is in music or video recommendation systems, e-commerce websites, etc. where wrong results could lead to customer churn and this could be harmful to the business.
Like with accuracy, precision is measured on a range of 0 to 1, where higher values indicate a more precise model and is calculated using the following equation:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall or Sensitivity

Recall for a label is defined as the number of true positives divided by the total number of actual positives.

It explains how many of the actual positive cases we were able to predict correctly with our model. Recall is a useful metric in cases where False Negative is of higher concern than False Positive. It is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected!

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Trade-Off

Precision and Recall often conflict with each other:

Maximizing Precision: The model becomes more cautious about calling something "positive." This means fewer False Positives, but it might miss some true positives (low Recall).

Maximizing Recall: The model becomes more aggressive in labeling things as "positive." This means it catches more true positives, but it might incorrectly label some negatives as positive (low Precision).

Specificity (True Negative Rate):

Specificity, or True Negative Rate, quantifies the ratio of correctly classified negative instances to the total actual negative instances:

It measures the ability to correctly identify negative instances.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

F1 Score:

It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall. F1 Score is the harmonic mean of precision and recall.

$$F1 = 2.\frac{Precision \times Recall}{Precision + Recall}$$

F1 Score could be an effective evaluation metric in the following cases:

- ✓ When FP and FN are equally costly.
- ✓ Adding more data doesn't effectively change the outcome
- ✓ True Negative is high
- ✓ It Does not take True Negatives into account.
- ✓ The F-Score (or F1-Score) is particularly useful when you need to strike a balance between Precision and Recall, especially in situations where both metrics matter, but you might prioritize one slightly more.

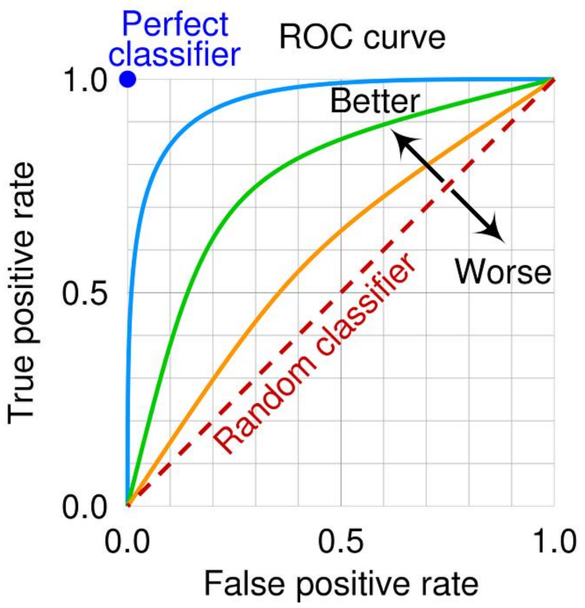
Receiver Operating Characteristic (ROC) Curve

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(Recall) and False Positive Rate.

$$\text{TPR} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

$$\text{FPR} = \frac{\text{False Positives (FP)}}{\text{False Positives (FP)} + \text{True Negatives (TN)}}$$

AUC is known for Area Under the ROC curve. As its name suggests, AUC calculates the two-dimensional area under the entire ROC curve, as shown below image:



AUC calculates the performance across all the thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1. It means a model with 100% wrong prediction will have an AUC of 0.0, whereas models with 100% correct predictions will have an AUC of 1.0.

How to interpret it:

- ✓ Top-Left Corner is Best: If the curve is closer to the top-left corner, that's great! It means we're correctly identifying spam without mislabeling many genuine emails. (Maximum True Positives, minimum false positives)
- ✓ Above the Diagonal Line: If the curve is above the diagonal line (from bottom-left to top-right), our model is better than just random guessing because the diagonal line means where TPR = FPR.
- ✓ Area Under Curve (AUC): The bigger the area under the curve (closer to 1), the better our model is.

2. Performance Metrics for Regression:

There are three error metrics that commonly used for evaluating and reporting the performance of a regression model; they are:
- Mean Squared Error (MSE).
- Root Mean Squared Error (RMSE).
- Mean Absolute Error (MAE)

✓ Mean Absolute Error: The MAE is calculated as the mean or average of the differences between predicted and expected target values in a dataset.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Advantages:
- MAE does not penalize large errors as heavily, making it more robust to outliers. This can be useful when outliers are not of primary concern.
- The MAE you get is in the same unit as the output variable.

Disadvantages:
- In optimization algorithms that rely on gradient-based methods, such as gradient descent, a key assumption is that the function being minimized is differentiable everywhere. But, the function refering to MAE is not differentiable at zero which complicates this process. Now to overcome the disadvantage of MAE next metric came as MSE.

- ✓ Mean Square Error: The MSE is calculated as the mean or average of the square differences between predicted and expected target values in a dataset.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Advantages:

- The graph of MSE is differentiable, so you can easily use it as a loss function.

Disadvantages:

- The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.
- If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

- ✓ Root Mean Squared Error(RMSE)

As RMSE is clear by the name itself, that it is a simple square root of mean squared error.

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$$

Advantages:

- The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.

Disadvantages:

- It is not that robust to outliers as compared to MAE.

- ✓ Root Mean Squared Log Error(RMSLE)
- ✓ R Squared (R^2)
- ✓ Adjusted R Squared
- ✓ Huber Loss

3. Performance Metrics for Clustering

Since in every classification evaluation metric either it is a confusion matrix or log loss there is a need for a dependent variable or target variable. As these evaluation techniques calculate the metrics based on observed and predicted values. This is the reason why any classification evaluation metrics can not be used to evaluate the performance of any clustering algorithms.

In clustering, there are two main types of measures used to evaluate the quality of the clustering results: internal measures and external measures.

Internal Measures:

These measures assess the quality of the clustering without requiring any external information or ground truth labels. They focus purely on the data and how well the algorithm has grouped it. E.g Silhouette Score, Dunn Index, Within-Cluster Sum of Squares (WCSS) etc.

External Measures:

These measures compare the clustering results to external ground truth labels to assess how accurately the clustering algorithm has grouped the data according to predefined classes. e.g Rand Index, Adjusted Rand Index (ARI), Normalized Mutual Information (NMI)

Silhouette coefficient /score:

The Silhouette Score is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

The Silhouette Score for each point is calculated using the following formula:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

a(i): The average distance from the i^{th} point to the other points in the same cluster.
b(i): The smallest average distance from the i^{th} point and all other point in the different cluster.

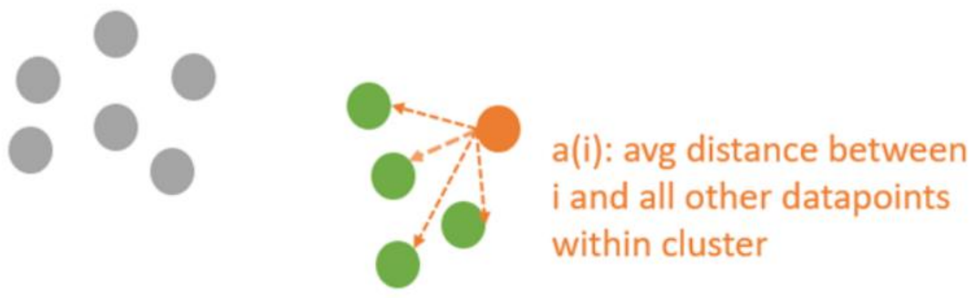
The Silhouette Score is particularly useful in the following scenarios:

- ✓ When you want to validate the consistency within clusters of data.
- ✓ To determine the optimal number of clusters.
- ✓ To visualize the quality and separation distance of the formed clusters.

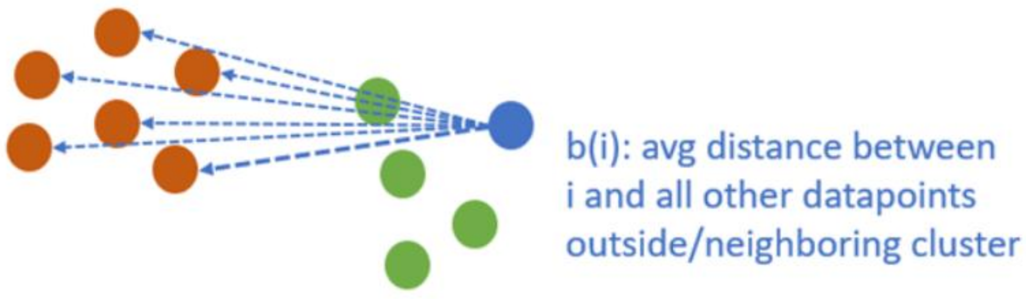
Despite its usefulness, the silhouette score has limitations:

- ✓ It may not perform well with clusters of varying densities.
- ✓ High dimensional data can reduce its effectiveness.

Assignment:
Cluster 1: Point A1: (2, 5) , Point A2: (3, 4), Point A3: (4, 6)
Cluster 2: Point B1: (8, 3) , Point B2: (9, 2) , Point B3: (10, 5)
Cluster 3: Point C1: (6, 10), Point C2: (7, 8), Point C3: (8, 9)
Find the Silhouette Score for point A1.



a(i)


b(i): avg distance between
i and all other datapoints
outside/neighboring cluster

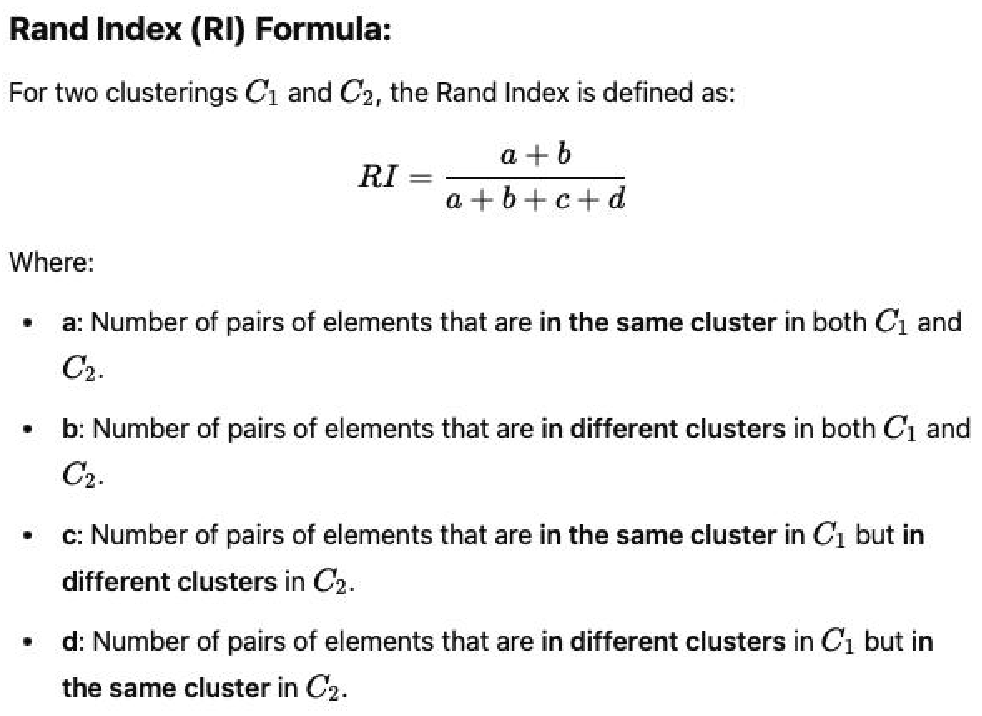
b(i)

Assignment: A dataset has been clustered into 3 clusters. For a particular data point i, the average intra-cluster distance (distance to another point in the

same cluster) is a=2.0 and average nearest-cluster distance (distance to the closest point in another cluster) is b=4.0.
A. Compute the silhouette for this point.
B. Is clustering for this point well-defined or ambiguous?

Rand Index:

The Rand Index (RI) is a measure used to evaluate the similarity between two data clusterings. It compares the predicted clustering with a ground truth clustering (the actual class labels) by considering all possible pairs of elements and counting how often the pairings are either consistent or inconsistent between the two clusterings.



Rand Index (RI) Formula:

For two clusterings C_1 and C_2, the Rand Index is defined as:

$$RI = \frac{a+b}{a+b+c+d}$$

Where:

- **a**: Number of pairs of elements that are **in the same cluster** in both C_1 and C_2.
- **b**: Number of pairs of elements that are **in different clusters** in both C_1 and C_2.
- **c**: Number of pairs of elements that are **in the same cluster** in C_1 **but in different clusters** in C_2.
- **d**: Number of pairs of elements that are **in different clusters** in C_1 **but in the same cluster** in C_2.

Interpretation of the Rand Index:

✓ 0: No similarity between the two clusterings (completely different).
✓ 1: Perfect match between the two clusterings (identical).
✓ Between 0 and 1: A value closer to 1 means the two clusterings are more similar, while a value closer to 0 means the clusterings are more different.

Numerical:
We have 4 data points: A,B,C,D.
Ground Truth Clusters (C1):
Cluster 1: {A,B}
Cluster 2: {C,D}

Predicted Clusters (C2):
Cluster 1: {A,C}
Cluster 2: {B,D}

Solution,
Form Give data points, the pairs are:
(A,B),(A,C),(A,D),(B,C),(B,D),(C,D)

We classify each pair into the four categories (a,b,c,d) based on the clustering:
a: Pairs that are in the same cluster in both C1(Ground Truth) and C2.
b: Pairs that are in different clusters in both C1 and C2.
c: Pairs that are in the same cluster in C1, but in different clusters in C2.
d: Pairs that are in different clusters in C1, but in the same cluster in C2.

Pair	C1 relation	C2 Realtion	Classify
(A,B)	Same cluster	Different Cluster	c
(A,C)	Different Cluster	Same cluster	d
(A,D)	Different Cluster	Different Cluster	b
(B,C)	Different Cluster	Different Cluster	b
(B,D)	Different Cluster	Same cluster	d
(C,D)	Same cluster	Different Cluster	c

ie. a=0, b=2, c=2 , d=2

The Rand Index is given by:

$$RI = \frac{a+b}{a+b+c+d}$$

Substitute the values:

$$RI = \frac{0+2}{0+2+2+2} = \frac{2}{6} = 0.333$$

This value indicates a low similarity between the predicted clusters (C2) and the ground truth clusters (C1).

Eg. A binary classifier is used to predict whether a patient has a disease (positive) or not (negative). The confusion matrix for the classifier's prediction is as follows.

	Precicted Positive	Predicted Negative
Actual Positive	50	10
Actual Negative	2	35

- ✓ Calculate the classification accuracy.
- ✓ Compute sensitivity (recall) and specificity.
- ✓ Determine precision andF1-score.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$= \frac{50 + 35}{50 + 35 + 2 + 10}$$

$$= 0.876 = 87.6\%$$

$$\text{Sensitivity (recall)} = \frac{TP}{TP + FN}$$

$$= \frac{50}{50 + 10}$$

$$= 0.833 = 83.3\%$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$= \frac{35}{35 + 2}$$

$$= 0.946 = 94.6\%$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$= \frac{50}{50 + 2}$$

$$= 0.962 = 96.2\%$$

$$F1-\text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$= 2 * \frac{0.962 * 0.833}{0.962 + 0.833}$$

$$= 0.892 = 89.2\%$$

Thus, while the accuracy is high, it's important to consider recall &precision, especially in contexts like disease prediction, where false negatives (missing positive cases) can have serious consequences.

- The classifier has high precision (96.2%), meaning that when it predicts a positive case, it is very likely to be correct.

- The recall (sensitivity) is a bit lower (83.3%), indicating that the model misses about 16.7% of the actual positive cases, which could be critical in medical applications.

- Specificity (94.6%) is also high, meaning the classifier correctly identifies most of the actual negative cases (non-disease).

- The F1-score of 89.2% indicates a good balance between precision and recall, considering both false positives and false negatives.

Example 2: Discuss the different approaches for validating a classifier with calculating the accuracy of this Covid case test data. A confusion matrix for covid testing classifier is as follows:

		Predicted Covid Cases	
		True	False
Actual COVID cases	Actual Positive	456	52
	Actual Negative	78	11569

Is accuracy sufficient to indicate the performance of this classifier? Justify with calculations and comparison of other parameters like precision, recall and F-1 scores.

Solution,

Accuracy = 98.93% (calculate by yourself)

Although accuracy is very high, it may not be sufficient in this context, especially in cases like medical testing where False Negatives (FN) and False Positives (FP) have serious implications. Let us calculate additional metrics to better understand the classifier's performance.

Precision = 85.4% (calculate by yourself)
Recall = 89.8% (calculate by yourself)
Specificity = 99.3% (calculate by yourself)
F1-Score = 87.5% (calculate by yourself)

- ✓ While accuracy is very high (98.93%), it does not provide a full picture of the classifier's performance, particularly in imbalanced datasets (e.g., many more negative cases than positive cases).

- ✓ In this scenario, high accuracy is influenced heavily by the large number of True Negatives (11569).

- ✓ Precision (85.4%) tells us how often the predicted positive cases are correct. This is important in reducing False Positives (e.g., diagnosing healthy individuals as Covid-positive).

- ✓ Recall (89.8%) highlights how many of the actual positive cases are correctly identified. This is critical in minimizing False Negatives, which is crucial for disease detection.

- ✓ Specificity (99.3%) demonstrates the classifier's ability to correctly identify negative cases, meaning it minimizes the False Positives.

- ✓ The F1-Score (87.5%) balances precision and recall, providing a more comprehensive metric for evaluating the classifier’s ability to handle both False Positives and False Negatives.