Jntuk R20 ML UNIT-I Final

machine learning (Jawaharlal Nehru Technological University, Kakinada)

Unit I:

Introduction- Artificial Intelligence, Machine Learning, Deep learning, Types of Machine Learning Systems, Main Challenges of Machine Learning.

Statistical Learning: Introduction, Supervised and Unsupervised Learning, Training and Test Loss, Tradeoffs in Statistical Learning, Estimating Risk Statistics, Sampling distribution of an estimator, Empirical Risk Minimization

.....

Need For Machine Learning

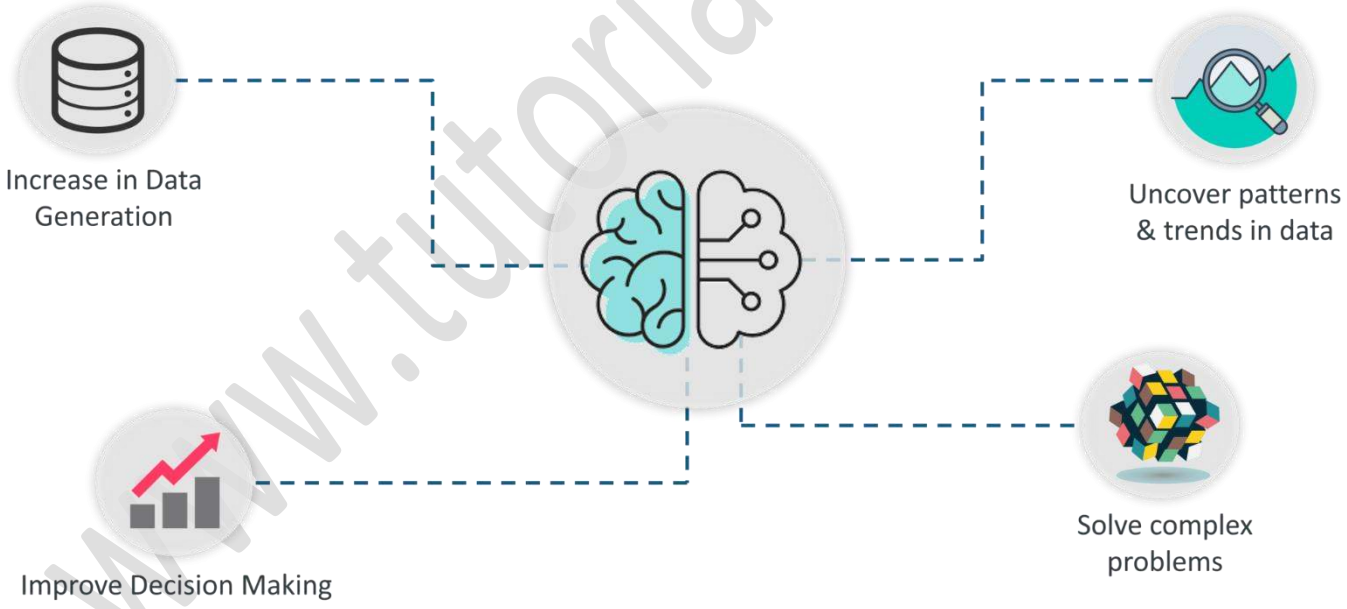
Ever since the technical revolution, we've been generating an immeasurable amount of data. As per research, we generate around 2.5 quintillion bytes of data every single day! It is estimated that by 2020, 1.7MB of data will be created every second for every person on earth.

With the availability of so much data, it is finally possible to build predictive models that can study and analyze complex data to find useful insights and deliver more accurate results.

Top Tier companies such as Netflix and Amazon build such Machine Learning models by using tons of data in order to identify profitable opportunities and avoid unwanted risks.

Here's a list of reasons why Machine Learning is so important:

- **Increase in Data Generation:** Due to excessive production of data, we need a method that can be used to structure, analyze and draw useful insights from data. This is where Machine Learning comes in. It uses data to solve problems and find solutions to the most complex tasks faced by organizations.
- **Improve Decision Making:** By making use of various algorithms, Machine Learning can be used to make better business decisions. For example, Machine Learning is used to forecast sales, predict downfalls in the stock market, identify risks and anomalies, etc.



- **Uncover patterns & trends in data**: Finding hidden patterns and extracting key insights from data is the most essential part of Machine Learning. By building predictive models and using statistical techniques, Machine Learning allows you to dig beneath the surface and explore the data at a minute scale. Understanding data and extracting patterns manually will take days, whereas Machine Learning algorithms can perform such computations in less than a second.
- **Solve complex problems:** From detecting the genes linked to the deadly ALS disease to building self-driving cars, Machine Learning can be used to solve the most complex problems.

To give you a better understanding of how important Machine Learning is, let's list down a couple of Machine Learning Applications:

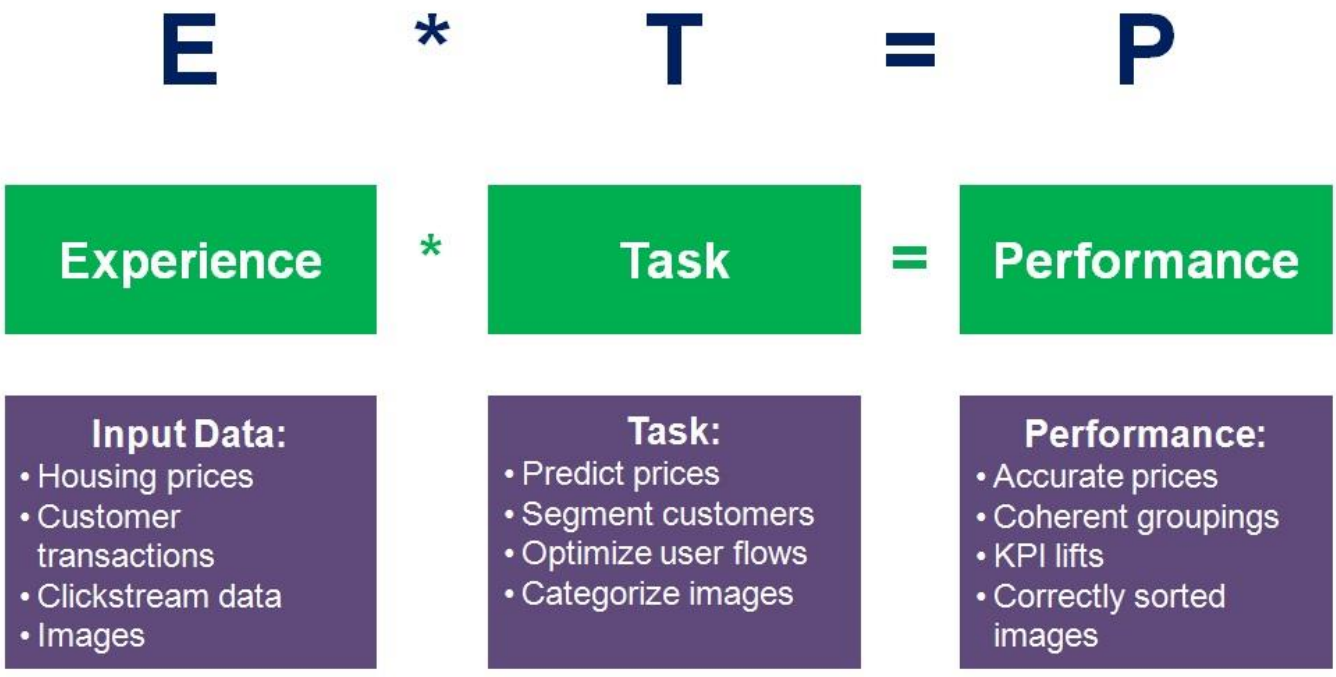
- *Netflix's Recommendation Engine*: The core of Netflix is its infamous recommendation engine. Over 75% of what you watch is recommended by Netflix and these recommendations are made by implementing Machine Learning.
- *Facebook's Auto-tagging feature:* The logic behind Facebook's DeepMind face verification system is Machine Learning and Neural Networks. DeepMind studies the facial features in an image to tag your friends and family.
- *Amazon's Alexa:* The infamous Alexa, which is based on Natural Language Processing and Machine Learning is an advanced level Virtual Assistant that does more than just play songs on your playlist. It can book you an Uber, connect with the other IoT devices at home, track your health, etc.
- *Google's Spam Filter:* Gmail makes use of Machine Learning to filter out spam messages. It uses Machine Learning algorithms and Natural Language Processing to analyze emails in real-time and classify them as either spam or non-spam.

Introduction To Machine Learning

The term Machine Learning was first coined by Arthur Samuel in the year 1959. Looking back, that year was probably the most significant in terms of technological advancements.

If you browse through the net about 'what is Machine Learning', you'll get at least 100 different definitions. However, the very first formal definition was given by Tom M. Mitchell:

"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."



In simple terms, Machine learning is a subset of Artificial Intelligence (AI) which provides machines the ability to learn automatically & improve from experience without being explicitly programmed to do so. In the sense, it is the practice of getting Machines to solve problems by gaining the ability to think.

But wait, can a machine think or make decisions? Well, if you feed a machine a good amount of data, it will learn how to interpret, process and analyze this data by using Machine Learning Algorithms, in order to solve real-world problems.

Machine Learning Definitions

Algorithm: A Machine Learning algorithm is a set of rules and statistical techniques used to learn patterns from data and draw significant information from it. It is the logic behind a Machine Learning model. An example of a Machine Learning algorithm is the Linear Regression algorithm.

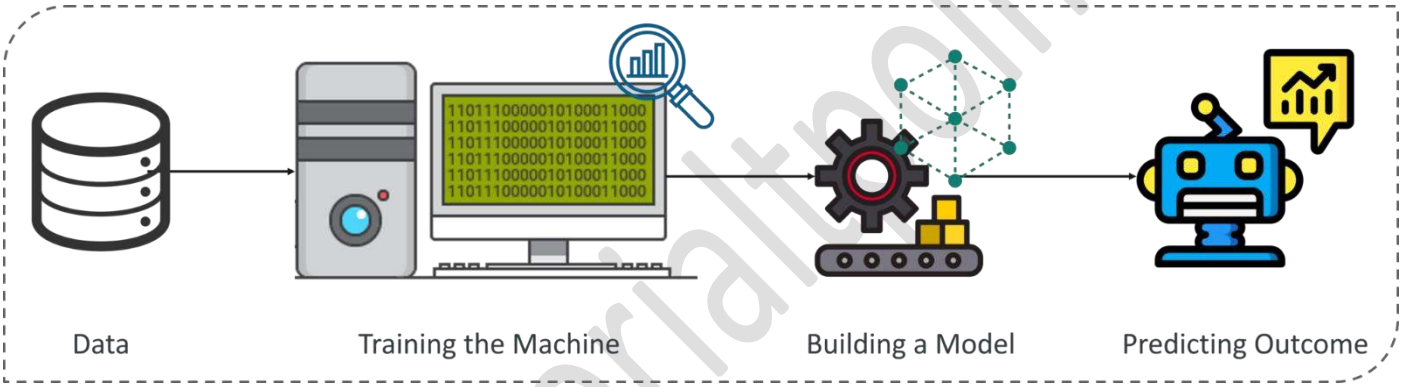
Model: A model is the main component of Machine Learning. A model is trained by using a Machine Learning Algorithm. An algorithm maps all the decisions that a model is supposed to take based on the given input, in order to get the correct output.

Predictor Variable: It is a feature(s) of the data that can be used to predict the output.

Response Variable: It is the feature or the output variable that needs to be predicted by using the predictor variable(s).

Training Data: The Machine Learning model is built using the training data. The training data helps the model to identify key trends and patterns essential to predict the output.

Testing Data: After the model is trained, it must be tested to evaluate how accurately it can predict an outcome. This is done by the testing data set.



To sum it up, take a look at the above figure. *A Machine Learning process begins by feeding the machine lots of data, by using this data the machine is trained to detect hidden insights and trends. These insights are then used to build a Machine Learning Model by using an algorithm in order to solve a problem.*

The next topic in this Introduction to Machine Learning blog is the Machine Learning Process.

Machine Learning Process

The Machine Learning process involves building a Predictive model that can be used to find a solution for a Problem Statement. To understand the Machine Learning process let's assume that you have been given a problem that needs to be solved by using Machine Learning.

The problem is to predict the occurrence of rain in your local area by using Machine Learning.

The below steps are followed in a Machine Learning process:

Step 1: Define the objective of the Problem Statement

At this step, we must understand what exactly needs to be predicted. In our case, the objective is to predict the possibility of rain by studying weather conditions. At this stage, it is also essential to take mental notes on what kind of data can be used to solve this problem or the type of approach you must follow to get to the solution.

Step 2: Data Gathering

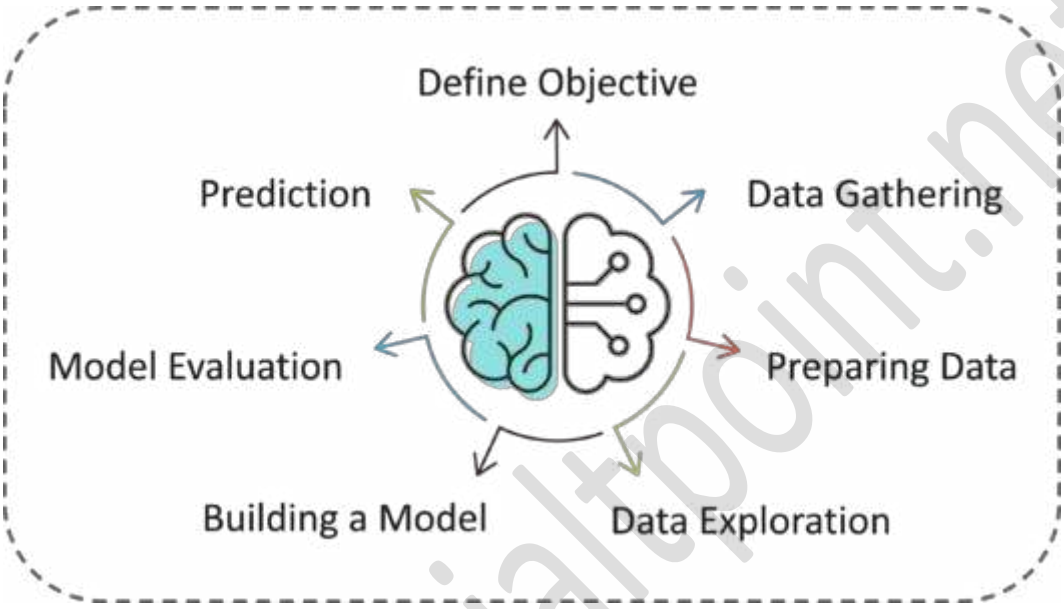
At this stage, you must be asking questions such as,

- What kind of data is needed to solve this problem?

- Is the data available?
- How can I get the data?

Once you know the types of data that is required, you must understand how you can derive this data. Data collection can be done manually or by web scraping. However, if you're a beginner and you're just looking to learn Machine Learning you don't have to worry about getting the data. There are 1000s of data resources on the web, you can just download the data set and get going.

Coming back to the problem at hand, the data needed for weather forecasting includes measures such as humidity level, temperature, pressure, locality, whether or not you live in a hill station, etc. Such data must be collected and stored for analysis.



Step 3: Data Preparation

The data you collected is almost never in the right format. You will encounter a lot of inconsistencies in the data set such as missing values, redundant variables, duplicate values, etc. Removing such inconsistencies is very essential because they might lead to wrongful computations and predictions. Therefore, at this stage, you scan the data set for any inconsistencies and you fix them then and there.

Step 4: Exploratory Data Analysis

Grab your detective glasses because this stage is all about diving deep into data and finding all the hidden data mysteries. EDA or Exploratory Data Analysis is the brainstorming stage of Machine Learning. Data Exploration involves understanding the patterns and trends in the data. At this stage, all the useful insights are drawn and correlations between the variables are understood.

For example, in the case of predicting rainfall, we know that there is a strong possibility of rain if the temperature has fallen low. Such correlations must be understood and mapped at this stage.

Step 5: Building a Machine Learning Model

All the insights and patterns derived during Data Exploration are used to build the Machine Learning Model. This stage always begins by splitting the data set into two parts, training data, and testing data. The training data will be used to build and analyze the model. The logic of the model is based on the Machine Learning Algorithm that is being implemented.

In the case of predicting rainfall, since the output will be in the form of True (if it will rain tomorrow) or False (no rain tomorrow), we can use a Classification Algorithm such as Logistic Regression.

Choosing the right algorithm depends on the type of problem you're trying to solve, the data set and the level of complexity of the problem. In the upcoming sections, we will discuss the different types of problems that can be solved by using Machine Learning.

Step 6: Model Evaluation & Optimization

After building a model by using the training data set, it is finally time to put the model to a test. The testing data set is used to check the efficiency of the model and how accurately it can predict the outcome. Once the accuracy is calculated, any further improvements in the model can be implemented at this stage. Methods like parameter tuning and cross-validation can be used to improve the performance of the model.

Step 7: Predictions

Once the model is evaluated and improved, it is finally used to make predictions. The final output can be a Categorical variable (eg. True or False) or it can be a Continuous Quantity (eg. the predicted value of a stock).

In our case, for predicting the occurrence of rainfall, the output will be a categorical variable.

So that was the entire Machine Learning process. Now it's time to learn about the different ways in which Machines can learn.

2. Machine Learning Types

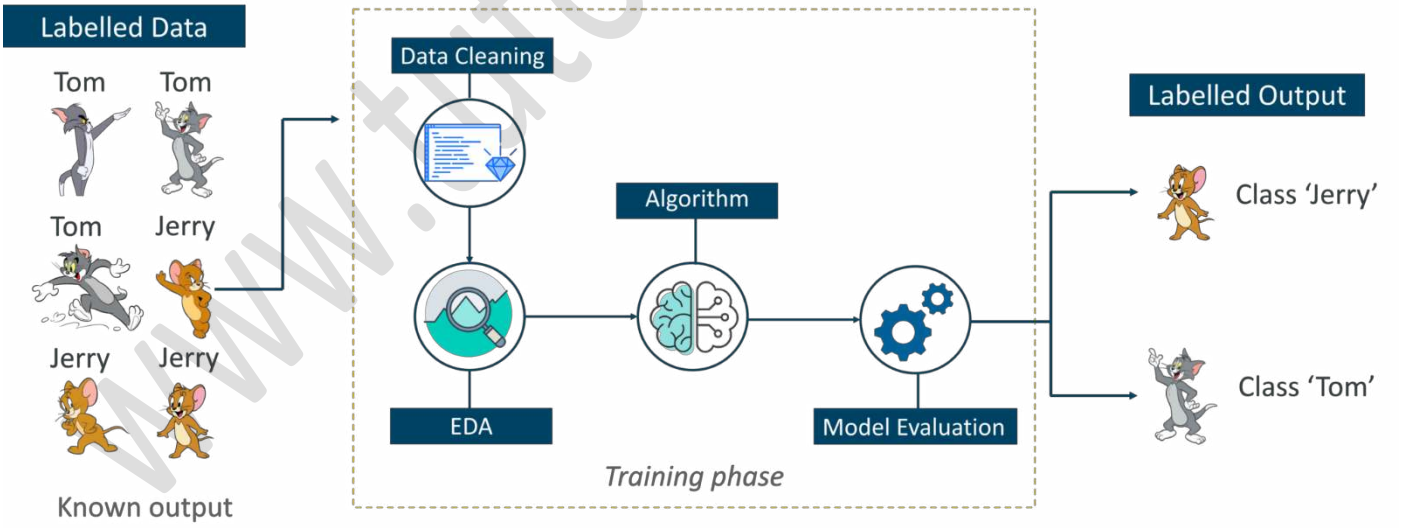
A machine can learn to solve a problem by following any one of the following three approaches. These are the ways in which a machine can learn:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Supervised Learning

Supervised learning is a technique in which we teach or train the machine using data which is well labeled.

To understand Supervised Learning let's consider an analogy. As kids we all needed guidance to solve math problems. Our teachers helped us understand what addition is and how it is done. Similarly, you can think of supervised learning as a type of Machine Learning that involves a guide. The labeled data set is the teacher that will train you to understand patterns in the data. The labeled data set is nothing but the training data set.

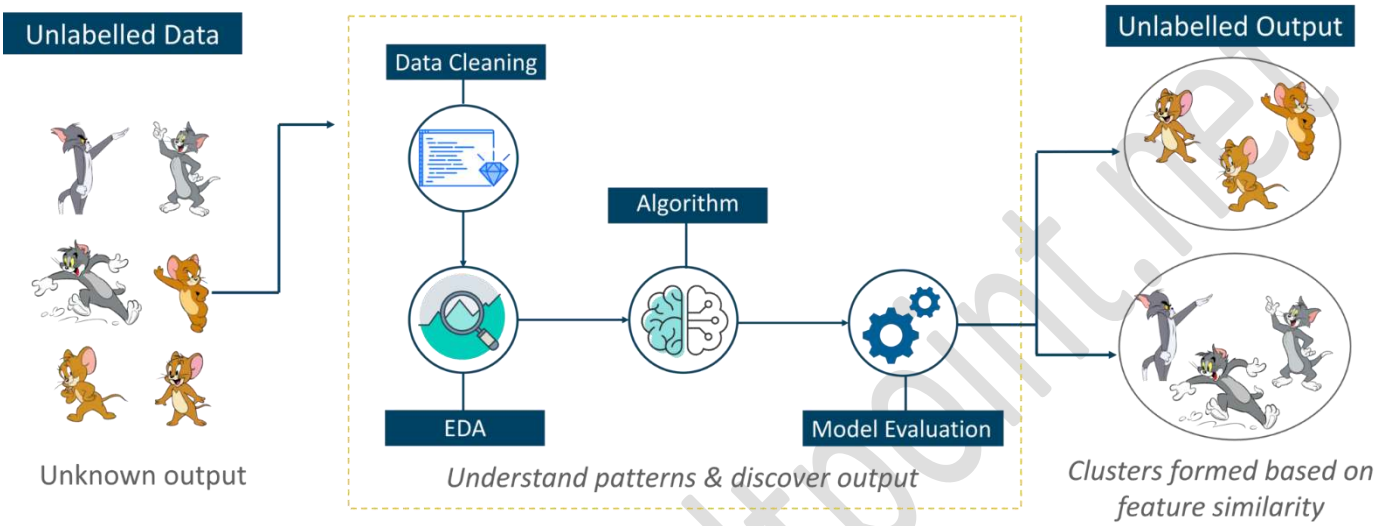


Consider the above figure. Here we're feeding the machine images of Tom and Jerry and the goal is for the machine to identify and classify the images into two groups (Tom images and Jerry images). The training data set that is fed to the model is labeled, as in, we're telling the machine, 'this is how Tom looks and this is Jerry'. By doing so you're training the machine by using labeled data. In Supervised Learning, there is a well-defined training phase done with the help of labeled data.

Unsupervised Learning

Unsupervised learning involves training by using unlabeled data and allowing the model to act on that information without guidance.

Think of unsupervised learning as a smart kid that learns without any guidance. In this type of Machine Learning, the model is not fed with labeled data, as in the model has no clue that 'this image is Tom and this is Jerry', it figures out patterns and the differences between Tom and Jerry on its own by taking in tons of data.



For example, it identifies prominent features of Tom such as pointy ears, bigger size, etc, to understand that this image is of type 1. Similarly, it finds such features in Jerry and knows that this image is of type 2. Therefore, it classifies the images into two different classes without knowing who Tom is or Jerry is.

Reinforcement Learning

Reinforcement Learning is a part of Machine learning where an agent is put in an environment and he learns to behave in this environment by performing certain actions and observing the rewards which it gets from those actions.
This type of Machine Learning is comparatively different.

Imagine that you were dropped off at an isolated island! What would you do?

Panic? Yes, of course, initially we all would. But as time passes by, you will learn how to live on the island. You will explore the environment, understand the climate condition, the type of food that grows there, the dangers of the island, etc. This is exactly how Reinforcement Learning works, it involves an Agent (you, stuck on the island) that is put in an unknown environment (island), where he must learn by observing and performing actions that result in rewards.

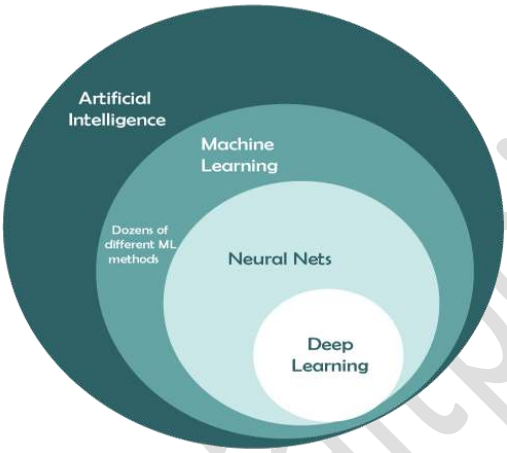
Reinforcement Learning is mainly used in advanced Machine Learning areas such as self-driving cars, AplhaGo, etc.

1. Artificial Intelligence, Machine Learning, Deep learning

Deep Learning, Machine Learning, and Artificial Intelligence are the most used terms on the internet for IT folks. However, all these three technologies are connected with each other. ***Artificial Intelligence (AI) can be understood as an umbrella that consists of both Machine learning and deep learning.*** Or We can say deep learning and machine learning both are subsets of artificial intelligence.

As these technologies look similar, most of the persons have misconceptions about 'Deep Learning, Machine learning, and Artificial Intelligence' that all three are similar to each other. But in reality, although all these technologies are used to build intelligent machines or applications that behave like a human, still, they differ by their functionalities and scope.

It means these three terms are often used interchangeably, but they do not quite refer to the same things. Let's understand the fundamental difference between deep learning, machine learning, and Artificial Intelligence with the below image.



With the above image, you can understand Artificial Intelligence is a branch of computer science that helps us to create smart, intelligent machines. Further, ML is a subfield of AI that helps to teach machines and build AI-driven applications. On the other hand, Deep learning is the sub-branch of ML that helps to train ML models with a huge amount of input and complex algorithms and mainly works with neural networks.

What is Artificial Intelligence (AI)?

Artificial Intelligence is defined as a field of science and engineering that deals with making intelligent machines or computers to perform human-like activities.

Mr. **John McCarthy** is known as the godfather of this amazing invention. There are some popular definitions of AI, which are as follows:

"AI is defined as the capability of machines to imitate intelligent human behavior."

"A computer system able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages."

What is Deep Learning?

"Deep learning is defined as the subset of machine learning and artificial intelligence that is based on artificial neural networks". In deep learning, **the deep word refers to the number of layers in a neural network.**

Deep Learning is a set of algorithms inspired by the structure and function of the human brain. It uses a huge amount of structured as well as unstructured data to teach computers and predicts accurate results. The main difference between machine learning and deep learning technologies is of presentation of data. Machine

learning uses structured/unstructured data for learning, while deep learning uses neural networks for learning models.

3. Main Challenges of Machine Learning

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

The following are some of the challenges of ML

1. Not enough training data.

Machine Learning is not quite there yet; it takes a lot of data for most Machine Learning algorithms to work properly. Even for very simple problems you typically need thousands of examples, and for complex problems such as image or speech recognition you may need millions of examples.

2. **Poor Quality of data:**

Obviously, if your training data has lots of errors, outliers, and noise, it will make it impossible for your machine learning model to detect a proper underlying pattern. Hence, it will not perform well.

So put in **every ounce of effort** in cleaning up your training data. No matter how good you are in selecting and hyper tuning the model, this part plays a major role in helping us make an accurate machine learning model.

"Most Data Scientists spend a significant part of their time in cleaning data".

There are a couple of examples when you'd want to clean up the data :

- If you see some of the instances are clear outliers just discard them or fix them manually.
- If some of the instances are missing a feature like (E.g., 2% of user did not specify their age), you can either ignore these instances, or fill the missing values by median age, or train one model with the feature and train one without it to come up with a conclusion.

3. Irrelevant Features:

"Garbage in, garbage out (GIGO)."

In the above image, we can see that even if our model is "AWESOME" and we feed it with garbage data, the result will also be garbage(output). Our training data must always contain **more relevant** and **less to none irrelevant features.**

The credit for a successful machine learning project goes to coming up with a good set of features on which it has been trained (often referred to as **feature engineering**), which includes feature selection, extraction, and creating new features which are other interesting topics to be covered in upcoming blogs.

4. Nonrepresentative training data:

To make sure that our model generalizes well, we have to make sure that our training data should be representative of the new cases that we want to generalize to.

If train our model by using a nonrepresentative training set, it won't be accurate in predictions it will be **biased against one** class or a group.

For E.G., Let us say you are trying to build a model that recognizes the genre of music. One way to build your training set is to search it on youtube and use the resulting data. Here we assume that youtube's search engine is providing representative data but in reality, the search will be biased towards popular artists and maybe even the artists that are popular in your location(if you live in India you will be getting the music of Arijit Singh, Sonu Nigam or etc).

So use representative data during training, so your model won't be biased among one or two classes when it works on testing data.

5. Overfitting the Training Data

Overfitting happens when the model is too complex relative to the amount and noisiness of the training data. The possible solutions are:

To simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model), by reducing the number of attributes in the training data or by constraining the model

- To gather more training data
- To reduce the noise in the training data (e.g., fix data errors and remove outliers)

6. Underfitting the Training Data

Underfitting is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data. For example, a linear model of life satisfaction is prone to underfit; reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training examples.

The main options to fix this problem are:

- Selecting a more powerful model, with more parameters
- Feeding better features to the learning algorithm (feature engineering)
- Reducing the constraints on the model (e.g., reducing the regularization hyperparameter)

CHAPTER-II

Q) Sampling distribution of an estimator

In statistics, an estimator is a function of the data that is used to estimate an unknown parameter of the population. The sampling distribution of an estimator refers to the distribution of the estimator over many different samples drawn from the population.

More specifically, the sampling distribution of an estimator is the probability distribution of the estimator's values when computed from a large number of random samples of fixed size, taken from the same population.

For example, suppose we are interested in estimating the mean weight of all the students in a school. We take a random sample of 100 students from the school and compute the sample mean weight. We repeat this process many times, each time taking a different random sample of 100 students. The sampling distribution of the sample mean weight is the distribution of the mean weight computed from all these different samples.

In machine learning, the concept of an estimator is related to the idea of a model. An estimator in machine learning is a function or algorithm that is used to estimate the value of some unknown parameter or function based on a set of observed data. For example, in linear regression, the estimator is a linear function that estimates the relationship between the input variables and the output variable.

The sampling distribution of an estimator in machine learning can be thought of as the distribution of the estimator's performance over many different samples drawn from the same population. In this context, the "performance" of the estimator refers to its ability to accurately predict the unknown parameter or function.

For example, suppose we are using a linear regression model to predict housing prices based on various features of the house (e.g. size, location, number of rooms, etc.). We can generate multiple random samples of houses from the same population and use each sample to train and test our model. The sampling distribution of our estimator (i.e. the linear regression model) would be the distribution of the model's performance (e.g. mean squared error, R-squared, etc.) over all of these different samples.

Understanding the sampling distribution of an estimator in machine learning is important because it allows us to make statements about the model's expected performance on new, unseen data. For example, we can use the sampling distribution to construct confidence intervals for the model's predictions, or to perform hypothesis tests to determine whether the model is significantly better than a baseline or alternative model.

1. Bootstrap

Bootstrap is a statistical resampling technique that can be used to estimate the sampling distribution of an estimator in machine learning. It involves repeatedly sampling the original dataset with replacement to generate a large number of "bootstrap samples", which are then used to estimate the sampling distribution of the estimator.

To use bootstrap in machine learning, we first train our estimator (e.g. a machine learning model) on the original dataset. We then generate a large number of bootstrap samples by randomly sampling the original dataset with replacement. For each bootstrap sample, we train a new instance of the estimator on the sample, and use it to estimate the parameter or function of interest. We repeat this process many times to generate a large number of estimates, which can be used to estimate the sampling distribution of the estimator.

For example, suppose we are using a decision tree to predict whether a customer will buy a product based on their age, income, and other demographic information. We can use bootstrap to estimate the sampling distribution of the decision tree's accuracy on new, unseen data. We first train the decision tree on the original dataset. We then generate a large number of bootstrap samples by randomly sampling the original dataset with replacement. For each bootstrap sample, we train a new instance of the decision tree on the sample, and use it to predict the outcomes of a test set. We repeat this process many times to generate a large number of estimates of the decision tree's accuracy on new, unseen data. We can then use these estimates to construct a confidence interval or perform hypothesis testing to make statements about the decision tree's expected accuracy on new, unseen data.

Bootstrap can be a powerful technique for estimating the sampling distribution of an estimator, especially when the distribution is difficult or impossible to calculate analytically. However, it can be computationally intensive, especially for large datasets or complex models.

2. Large sample theory for the MLE

In machine learning, the maximum likelihood estimator (MLE) is a commonly used method for estimating the parameters of a probabilistic model. Large sample theory is a branch of statistics that studies the behavior of estimators as the sample size becomes very large.

The large sample theory for the MLE in machine learning is based on the idea that, as the sample size increases, the distribution of the MLE becomes approximately normal (i.e. follows a normal distribution). This result is known as the central limit theorem.

More specifically, under certain assumptions, as the sample size n approaches infinity, the distribution of the MLE becomes approximately normal with mean equal to the true parameter value and variance equal to the inverse of the Fisher information matrix evaluated at the true parameter value. The Fisher information matrix is a measure of how much information the data contains about the parameter.

This result has important implications for machine learning, as it allows us to make statements about the expected performance of the MLE as the sample size increases. For example, we can use the central limit theorem to construct confidence intervals for the MLE's estimates, or to perform hypothesis tests to determine whether the estimates are significantly different from a hypothesized value.

However, it is important to note that the large sample theory for the MLE assumes that certain conditions are met, such as that the model is correctly specified and that the data are independent and identically distributed. Violations of these assumptions can lead to biased or inconsistent estimates, even in large samples. Therefore, it is important to carefully consider the assumptions underlying the MLE and the large sample theory before applying them in practice.

Q) Empirical Risk Minimization

Empirical Risk Minimization (ERM) is a fundamental principle in machine learning that underlies many common approaches to training models. It is based on the idea that the best model is the one that minimizes the expected risk on unseen data, and that we can estimate this risk by minimizing the empirical risk on the available training data.

The empirical risk of a model is defined as the average loss over the training data, where the loss measures the discrepancy between the model's predictions and the true outcomes. Formally, if we have a dataset consisting of input-output pairs (x,y), the empirical risk of a model f(x) is given by:

$R\_emp(f) = (1/n) * sum_i L(y_i, f(x_i))$

where n is the size of the dataset, $L(y_i, f(x_i))$ is the loss function that measures the discrepancy between the model's prediction $f(x_i)$ and the true output y_i for each example i.

ERM involves finding the model that minimizes the empirical risk over the training data. This is typically done by choosing a parametric form for the model (e.g. a neural network, decision tree, linear regression, etc.) and then searching for the values of the parameters that minimize the empirical risk. This process is often called "training" the model, and typically involves using an optimization algorithm (e.g. stochastic gradient descent) to iteratively update the parameters to reduce the empirical risk.

One important consideration when using ERM is overfitting, which occurs when the model becomes too complex and begins to fit the noise in the training data rather than the underlying signal. Regularization techniques, such as L1 or L2 regularization, are often used to prevent overfitting by adding a penalty term to the empirical risk that discourages the model from using overly complex parameter values.

Overall, ERM is a powerful and widely used approach for training machine learning models, but it is important to carefully consider the choice of loss function, regularization, and other hyperparameters to avoid overfitting and ensure good generalization performance on unseen data.

1. Regularized risk minimization

Regularized Risk Minimization (RRM) is a principle in machine learning that involves finding a model that minimizes a regularized version of the expected risk on unseen data. It is similar to Empirical Risk Minimization (ERM), but with the addition of a regularization term that penalizes complex models and encourages simpler ones.

The regularized risk of a model is defined as:

$R\_risk(f) = E[L(Y, f(X))] + \lambda\Omega(f)$

where $E[L(Y, f(X))]$ is the expected loss of the model on unseen data, $\Omega(f)$ is a complexity measure of the model, and λ is a regularization parameter that controls the trade-off between the loss and complexity terms.

The regularized risk can be minimized by finding the model that achieves the optimal trade-off between the loss and complexity terms.

RRM is a powerful approach for training machine learning models that can help prevent overfitting and improve generalization performance on unseen data. However, it is important to choose an appropriate regularization technique and regularization parameter to balance the trade-off between model complexity and performance.

2. Structural risk minimization

Structural Risk Minimization (SRM) is a principle in machine learning that aims to find a model that minimizes the expected risk on unseen data while also controlling the complexity of the model. It is similar to Empirical Risk Minimization (ERM) and Regularized Risk Minimization (RRM), but with a focus on model selection and choosing the optimal level of complexity.

The basic idea behind SRM is to balance the bias-variance trade-off in model selection. A model with low complexity (e.g. a linear model) may have low variance but high bias, while a complex model (e.g. a deep neural network) may have low bias but high variance. SRM aims to find the optimal level of complexity that minimizes both the bias and variance of the model.

One way to achieve this is to use a two-stage approach for model selection. In the first stage, a family of models with varying complexity is generated, such as a set of neural networks with different numbers of layers. In the second stage, the optimal model is chosen from the family based on its performance on a validation set or through cross-validation.

Another approach is to use a model selection criterion that penalizes both the loss and complexity terms, such as the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC). These criteria balance the trade-off between goodness of fit and model complexity and can help to avoid overfitting.

SRM is an important principle in machine learning that can help to improve the generalization performance of models by controlling their complexity. However, it can be computationally expensive to search over a large family of models, and there is often a trade-off between model complexity and performance that must be carefully balanced.

3. Estimating the risk using cross validation

Cross-validation is a widely used technique in machine learning for estimating the generalization performance of a model and selecting its hyperparameters. One common application of cross-validation is to estimate the risk of a model, which is the expected loss on unseen data.

The basic idea behind cross-validation is to partition the available data into several subsets or "folds". For example, in k-fold cross-validation, the data is divided into k non-overlapping folds of equal size. Then, the model is trained on k-1 folds and evaluated on the remaining fold, with the process repeated k times so that

each fold is used for evaluation once. The average performance of the model across the k folds is then used as an estimate of its generalization performance.

To estimate the risk of a model using cross-validation, the data is first randomly divided into training and testing sets. The training set is used to train the model with a specific set of hyperparameters, while the testing set is used to evaluate its performance. However, since there is only one testing set, the estimate of the risk may be biased and have high variance.

Cross-validation helps to mitigate this issue by using multiple testing sets and averaging the results. By repeatedly training and evaluating the model on different subsets of the data, cross-validation provides a more robust estimate of the model's performance and helps to reduce the variance of the estimate.

The choice of the number of folds (k) and the method for partitioning the data can affect the performance and computational efficiency of cross-validation. For example, leave-one-out cross-validation (LOOCV) is a special case of k-fold cross-validation where k is equal to the number of samples, which can be computationally expensive but provides an unbiased estimate of the risk. On the other hand, stratified sampling can be used to ensure that the partitions of the data are representative of the class distributions.

Overall, cross-validation is a powerful technique for estimating the risk of a model and selecting its hyperparameters, and is widely used in machine learning for evaluating and comparing different models.

4. Upper bounding the risk using statistical learning theory

Statistical learning theory is a framework in machine learning that provides theoretical bounds on the generalization performance of models. These bounds can be used to estimate the risk of a model and to guide the selection of hyperparameters.

The main idea behind statistical learning theory is to use the concept of empirical risk minimization (ERM) to bound the expected risk of a model. ERM is a principle in machine learning that aims to find a model that minimizes the empirical risk on the training data, which is the average loss over the samples in the training set. The expected risk, on the other hand, is the average loss over all possible data sets drawn from the underlying distribution.

The key insight of statistical learning theory is that the expected risk can be upper-bounded by the empirical risk plus a term that depends on the complexity of the model and the size of the data set. This term, known as the "generalization error", quantifies the difference between the expected risk and the empirical risk and is often used as a measure of the model's generalization performance.

There are different types of bounds in statistical learning theory, such as the Rademacher complexity bound, the VC-dimension bound, and the uniform convergence bound. These bounds provide different levels of generality and tightness, depending on the assumptions made about the underlying distribution and the complexity of the model.

In practice, statistical learning theory can be used to estimate the risk of a model by evaluating its empirical risk on the training data and using the generalization error term to compute an upper bound on the expected risk. This can help to guide the selection of hyperparameters and to avoid overfitting, which occurs when the model fits the noise in the training data and fails to generalize to new data.

Overall, statistical learning theory provides a powerful tool for understanding the generalization performance of models and for guiding the design and evaluation of machine learning algorithms.

5. Surrogate loss functions

A surrogate loss function in machine learning is a differentiable function that is used in place of the true loss function when it is difficult or impossible to optimize directly. The surrogate loss function is designed to approximate the behavior of the true loss function in a way that makes it easier to optimize the model parameters.

In many machine learning tasks, the true loss function is non-differentiable or has other properties that make it difficult to optimize directly. For example, the 0-1 loss function, which measures the number of

misclassifications, is non-differentiable and discontinuous. Similarly, the hinge loss function, which is commonly used in support vector machines, is non-differentiable at the origin.

To overcome these challenges, surrogate loss functions are used to approximate the true loss function while preserving desirable properties, such as differentiability and convexity. Surrogate loss functions can be derived by making certain assumptions about the relationship between the model outputs and the true labels, and by designing a function that reflects these assumptions.

One common example of a surrogate loss function is the cross-entropy loss function, which is used for binary classification and measures the distance between the predicted probabilities and the true labels. The cross-entropy loss function is differentiable and convex, and is often used as a surrogate for the 0-1 loss function, which is non-differentiable.

Another example is the hinge loss function, which is used for binary classification with support vector machines. The hinge loss function is non-differentiable at the origin, but can be approximated by a differentiable function, such as the smoothed hinge loss function, which is convex and easier to optimize.

Surrogate loss functions can also be used in multi-class classification, regression, and other machine learning tasks, where the true loss function may be difficult to optimize directly. The choice of the surrogate loss function can have a significant impact on the performance of the model and the speed of convergence, and is an important consideration in the design and evaluation of machine learning algorithms.

Chapter-II

Statistical Learning

Introduction, supervised and unsupervised learning, Training and Testing loss, Tradeoffs in statistical learning, Risk statistics, Sampling distribution of an estimator, Empirical Risk Minimization.

Introduction

→ structuring and visualizing data are important aspects of data science.

→ When the goal is to interpret the model and quantify the uncertainty in the data, this analysis is usually refered to as statistical learning.

→ There are two major goals for modeling data:

(1) To accurately predict some future quantity of interest, given some observed data

(2) To discover usual (or) interesting patterns in the data.

To achive these goals, one must rely on knowledge from three important pillars of the mathematical sciences.

(1) Function approximation

(2) Optimization

(3) probability and statistics.

Function approximation:- A mathematical function is used to represent the relationships between the variables. As a data scientist, you need to understand how best to approximate and represent function using least amount

of computer processing and memory.

Optimization:-

→ Given a set of mathematical models, we wish to find best possible model in that set.

→ This step usually requires knowledge of optimization algorithms and efficient computer coding or programming.

probability and statistics

The knowledge of probability and statistics is needed to fit (or) train algorithm and generate a model.

1. Supervised and Unsupervised Learning

→ Given an input x, one of the main goals of ML is to predict an output variable y.

Examples:

1) x: digitized signature

y: whether the signature is genuine (or) false.

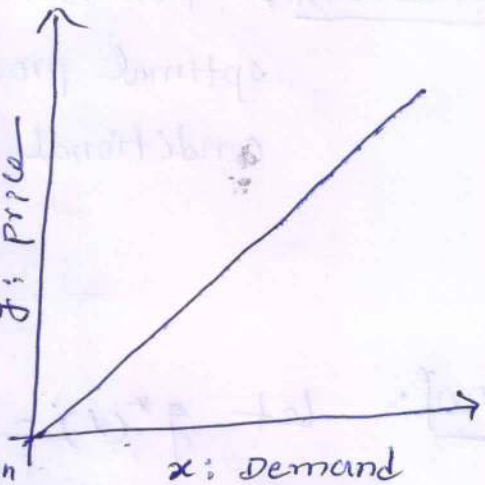
2) x: weight and smoking habits of an expecting mother.

y: The birth weight of the baby.

→ The data scientist attempt at this prediction is encoded in a mathematical function g, called the prediction function, which takes an input x and outputs a guess $g(x)$ for y denoted by $\hat{y}$.

→ In regression problems, the response variable 'y' can take any real value.

→ In contract, when 'y' can only lie in a finite set say $y \in \{0, \ldots, C-1\}$ then predicting 'y' is conceptually the same as classifying the input 'x' into one of 'c' categories, and so prediction becomes a classification problem.



→ We measure the accuracy of a prediction $\hat{y}$ with respect to a given response 'y' by using some loss function $\text{loss}(y, \hat{y})$

→ It is unlikely that any mathematical function 'g' will be able to make predictions for all possible pairs (x, y) one may encounter in nature.

→ One reason for this is that, even with the same input x, the output y may be different, depending on chance circumstances or randomness.

→ For this reason, we adopt a probabilistic approach and assume that each pair (x, y) is the outcome of a random pair (X, Y) that has some joint probability density $f(x, y)$.

→ We then asses the predictive performance via the expected loss, usually called the risk, for g:

$$l(g) = E\,\text{Loss}(Y, g(X))$$

Optimal prediction function for Squared-Error Loss

Statement:- For the squared-error loss $(y, \hat{y}) = (y-\hat{y})^2$, the optimal prediction function g^* is equal to the conditional expectation of Y given $X = x$:

$$g^*(x) = E[Y|X=x]$$

Proof:- Let $g^*(x) = E[Y|X=x]$. For any function g, the squared-error risk satisfies

$$E(Y-g(x))^2 = E[(Y-g^*(x)+g^*(x)-g(x))^2]$$

$$= E(Y-g^*(x))^2 + 2E[(Y-g^*(x))(g^*(x)-g(x))] + E[(g^*(x)-g(x))^2]$$

$$\geq E[Y-g^*(x)]^2 + 2E[(Y-g^*(x))(g^*(x)-g(x))]$$

$$= E[Y-g^*(x)]^2 + 2E[(g^*(x)-g(x))E[Y-g^*(x)|X]]$$

In the last equation we used the tower property. By the definition of the conditional expectation, we have

$$E[Y-g^*(x)|X] = 0$$

It follows that $E[Y-g(x)]^2 \geq E[Y-g^*(x)]^2$, showing that g^* yields the smallest squared-error risk.

From the above calculation, conditional $X = x$ the (random) response Y can be written as

$$Y = g^*(x) + \varepsilon_e(x)$$

where $g^*(x)$ is optimal prediction function.

$\varepsilon_e(x)$ is random deviation of the response from its conditional mean.

→ our goal is to "learn" the unknown g^* using 'n' examples in the Traing set T. let us denote this as g_T the best approximation for g^* that we can construct from T.

→ A particular outcome is denoted by g_T.

→ let us take a relationship $g^* : x \rightarrow y$

for the above g_T is a learn, who learns the relation between the input x and the output y with 'n' examples.

→ Since the learner learns the relationship with examples this type of learning is called Supervised learning. In this 'x' is a vector of explanator variable.

→ In contranst, unsupervised learning makes no distinction between response and explanatory variables, and the objective is simply to learn the structure of the unknown distribution of the data.

2. Training and Testing loss

In machine learning, training and testing loss are used to evaluate the performance of a model.

Training loss is the error rate on the training data during the training process. The goal of the training process is to minimize the loss, meaning the model is learning to make accurate predictions on the training data.

Testing loss is the error rate on the testing data, which is a separate dataset that is not used during during the training process. This is used to evaluate the generalization performance of the model, meaning how well it can make accurate predictions on new, unseen data.

The training loss typically decreases during the training process, as the model becomes better at making predictions the training data. However, if the model is overfitting to the training data, the testing loss may increase even as the training loss continues to decrease. This indicates that the model is not able to generalize well on to new data.

The goal of ML is to minimize both the training loss and testing loss, while also preventing overfitting. This is achieved by using techniques such as regularization, early stopping, and cross validation.

3. Risk statistics

Risk statistics are used in machine learning to evaluate the performance and generalization of a model. Some commonly used risk statistics include:

(1) **Training Error:-** This is the error rate of the model on the training data. The training error is used to evaluate how well the model fits the training data.

(2) **Testing Error:-** This is the error rate of the model on the test data. The test error is used to evaluate how well the model generalizes to new, unseen data

(3) **cross-validation Error:-** This the error rate of the model on a validation set that is created by partitioning the data into multiple subsets.

cross validation is used to estimate the generalization performance of the model and to prevent overfitting.

(4) **Bias:-** This the difference between the expected value of the predictions made by the model and the true value of the target variable. Bias measures how well the model captures the true relationship between the input features and the target variable.

(5) Variance : This is the variability of the model's predictions for different training sets. Variance measures how sensitive the model is to small changes in training data.

(6) Mean Sure Error (MSE) :-

This is the average of the square differences between the predicted value and the true value. MSE is used to evaluate the overall performance of the model.

7. Root Mean Squared Error :-

This is the square root of the MSE. RMSE is used to measure the average magnitude of the error made by the model.

In summary, risk stastics are used to evaluate the performance and generalization of ML model. By using multiple risk stastistics, we can gain a more comprehensive understanding of the model's strengths and weaknesses.