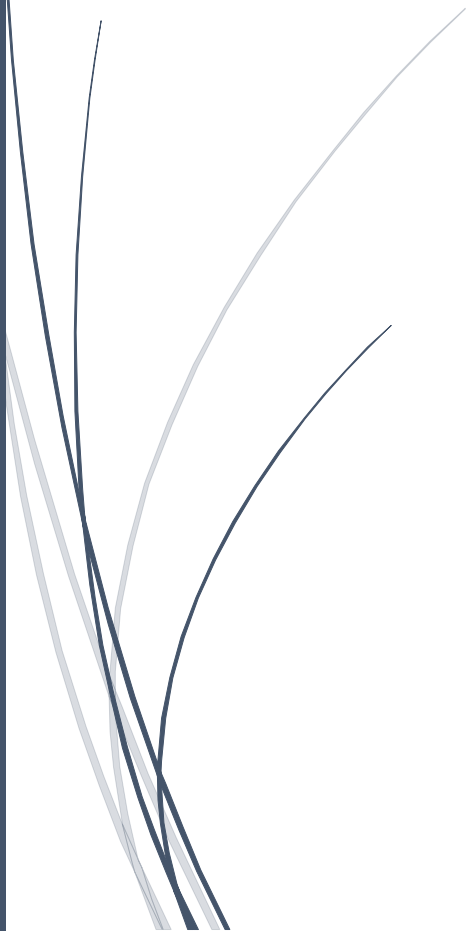
Basics of Machine Learning

Notes



- Puneet Sulodhia

Machine Learning

Unit-1

Introduction Machine Learning: Definition, History, Need, Features, Block diagrammatic representation of learning machines, Classification of Machine Learning: Supervised learning, Unsupervised learning, Reinforcement Learning, Machine Learning life cycle, Applications of Machine Learning.

Unit-2

Dimensionality Reduction Dimensionality reduction: Definition, Row vector and Column vector, how to represent a dataset, how to represent a dataset as a Matrix, Data preprocessing in Machine Learning: Feature Normalization, Mean of a data matrix, Column Standardization, Co-variance of a Data Matrix, Principal Component Analysis for Dimensionality reduction.

Unit-3

Supervised Learning Supervised Learning: Definition, how it works. Types of Supervised learning algorithms k - Nearest Neighbours, Naïve Bayes, Decision Trees, Naive Bayes, Linear Regression, Logistic Regression, Support Vector Machines.

Unit-4

Unsupervised Learning: Clustering: K-means. Ensemble Methods: Boosting, Bagging, Random Forests. Evaluation: Performance measurement of models in terms of accuracy, confusion matrix, precision & recall, F1-score, receiver Operating Characteristic Curve (ROC) curve and AUC, Median absolute deviation (MAD), Distribution of errors.

Unit – 1

What is machine learning?

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance.

Why is machine learning important?

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

What are the different types of machine learning?

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches:supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. The type of algorithm data scientists choose to use depends on what type of data they want to predict.

- **Supervised learning:** In this type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.
- **Unsupervised learning:** This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.
- **Semi-supervised learning:** This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.

- **Reinforcement learning:** Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

How does supervised machine learning work?

Supervised machine learning requires the data scientist to train the algorithm with both labeled inputs and desired outputs. Supervised learning algorithms are good for the following tasks:

- **Binary classification:** Dividing data into two categories.
- **Multi-class classification:** Choosing between more than two types of answers.
- **Regression modeling:** Predicting continuous values.
- **Ensembling:** Combining the predictions of multiple machine learning models to produce an accurate prediction.

How does unsupervised machine learning work?

Unsupervised machine learning algorithms do not require data to be labeled. They sift through unlabeled data to look for patterns that can be used to group data points into subsets. Most types of deep learning, including neural networks, are unsupervised algorithms. Unsupervised learning algorithms are good for the following tasks:

- **Clustering:** Splitting the dataset into groups based on similarity.
- **Anomaly detection:** Identifying unusual data points in a data set.
- **Association mining:** Identifying sets of items in a data set that frequently occur together.
- **Dimensionality reduction:** Reducing the number of variables in a data set.

How does semi-supervised learning work?

Semi-supervised learning works by data scientists feeding a small amount of labeled training data to an algorithm. From this, the algorithm learns the dimensions of the data set, which it can then apply to new, unlabeled data. The performance of algorithms typically improves when they train on labeled data sets. But labeling data can be time consuming and expensive. Semi-supervised learning strikes a middle ground between the performance of supervised learning and the efficiency of unsupervised learning. Some areas where semi-supervised learning is used include:

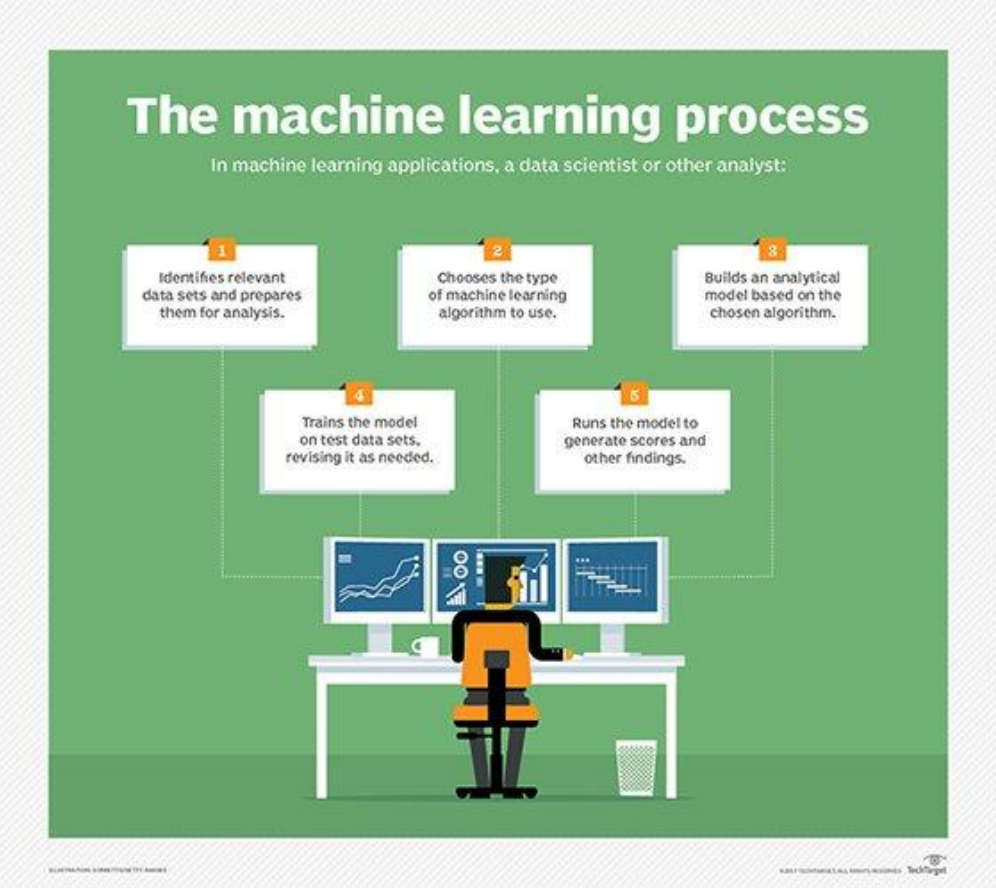
- **Machine translation:** Teaching algorithms to translate language based on less than a full dictionary of words.

- **Fraud detection:** Identifying cases of fraud when you only have a few positive examples.
- **Labelling data:** Algorithms trained on small data sets can learn to apply data labels to larger sets automatically.

How does reinforcement learning work?

Reinforcement learning works by programming an algorithm with a distinct goal and a prescribed set of rules for accomplishing that goal. Data scientists also program the algorithm to seek positive rewards -- which it receives when it performs an action that is beneficial toward the ultimate goal -- and avoid punishments -- which it receives when it performs an action that gets it farther away from its ultimate goal. Reinforcement learning is often used in areas such as:

- **Robotics:** Robots can learn to perform tasks the physical world using this technique.
- **Video gameplay:** Reinforcement learning has been used to teach bots to play a number of video games.
- **Resource management:** Given finite resources and a defined goal, reinforcement learning can help enterprises plan out how to allocate resources.



Machine learning is like statistics on steroids.

Who's using machine learning and what's it used for?

Today, machine learning is used in a wide range of applications. Perhaps one of the most well-known examples of machine learning in action is the recommendation engine that powers Facebook's news feed.

Facebook uses machine learning to personalize how each member's feed is delivered. If a member frequently stops to read a particular group's posts, the recommendation engine will start to show more of that group's activity earlier in the feed.

Behind the scenes, the engine is attempting to reinforce known patterns in the member's online behavior. Should the member change patterns and fail to read posts from that group in the coming weeks, the news feed will adjust accordingly.

In addition to recommendation engines, other uses for machine learning include the following:

- **Customer relationship management.** CRM software can use machine learning models to analyze email and prompt sales team members to respond to the most important messages first. More advanced systems can even recommend potentially effective responses.
- **Business intelligence.** BI and analytics vendors use machine learning in their software to identify potentially important data points, patterns of data points and anomalies.
- **Human resource information systems.** HRIS systems can use machine learning models to filter through applications and identify the best candidates for an open position.
- **Self-driving cars.** Machine learning algorithms can even make it possible for a semi-autonomous car to recognize a partially visible object and alert the driver.
- **Virtual assistants.** Smart assistants typically combine supervised and unsupervised machine learning models to interpret natural speech and supply context.

What are the advantages and disadvantages of machine learning?

Machine learning has seen use cases ranging from predicting customer behavior to forming the operating system for self-driving cars.

When it comes to advantages, machine learning can help enterprises understand their customers at a deeper level. By collecting customer data and correlating it with behaviors over time, machine learning algorithms can learn associations and help teams tailor product development and marketing initiatives to customer demand.

Some companies use machine learning as a primary driver in their business models. Uber, for example, uses algorithms to match drivers with riders. Google uses machine learning to surface the ride advertisements in searches.

But machine learning comes with disadvantages. First and foremost, it can be expensive. Machine learning projects are typically driven by data scientists, who command high salaries. These projects also require software infrastructure that can be expensive.

There is also the problem of machine learning bias. Algorithms trained on data sets that exclude certain populations or contain errors can lead to inaccurate models of the world that, at best, fail and, at worst, are discriminatory. When an enterprise bases core business processes on biased models it can run into regulatory and reputational harm.

How to choose the right machine learning model

The process of choosing the right machine learning model to solve a problem can be time consuming if not approached strategically.

Step 1: Align the problem with potential data inputs that should be considered for the solution. This step requires help from data scientists and experts who have a deep understanding of the problem.

Step 2: Collect data, format it and label the data if necessary. This step is typically led by data scientists, with help from data wranglers.

Step 3: Chose which algorithm(s) to use and test to see how well they perform. This step is usually carried out by data scientists.

Step 4: Continue to fine tune outputs until they reach an acceptable level of accuracy. This step is usually carried out by data scientists with feedback from experts who have a deep understanding of the problem.

Importance of human interpretable machine learning

Explaining how a specific ML model works can be challenging when the model is complex. There are some vertical industries where data scientists have to use simple machine learning models because it's important for the business to explain how every decision was made. This is especially true in industries with heavy compliance burdens such as banking and insurance.

Complex models can produce accurate predictions, but explaining to a lay person how an output was determined can be difficult.

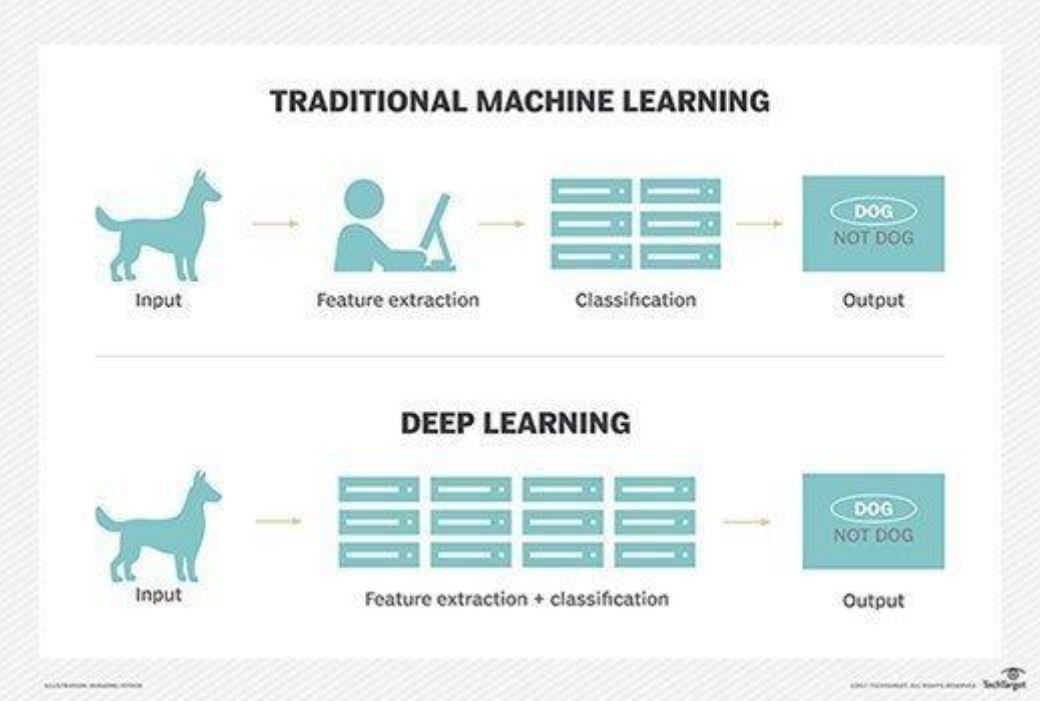
What is the future of machine learning?

While machine learning algorithms have been around for decades, they've attained new popularity as artificial intelligence has grown in prominence. Deep learning models, in particular, power today's most advanced AI applications.

Machine learning platforms are among enterprise technology's most competitive realms, with most major vendors, including Amazon, Google, Microsoft, IBM and others, racing to sign customers up for platform services that cover the spectrum of machine learning activities, including data collection, data preparation, data classification, model building, training and application deployment.

As machine learning continues to increase in importance to business operations and AI becomes more practical in enterprise settings, the machine learning platform wars will only intensify.

Continued research into deep learning and AI is increasingly focused on developing more general applications. Today's AI models require extensive training in order to produce an algorithm that is highly optimized to perform one task. But some researchers are exploring ways to make models more flexible and are seeking techniques that allow a machine to apply context learned from one task to future, different tasks.



Deep learning works in very different ways than traditional machine learning.

How has machine learning evolved?

1642 - Blaise Pascal invents a mechanical machine that can add, subtract, multiply and divide.

1679 - Gottfried Wilhelm Leibniz devises the system of binary code.

1834 - Charles Babbage conceives the idea for a general all-purpose device that could be programmed with punched cards.

1842 - Ada Lovelace describes a sequence of operations for solving mathematical problems using Charles Babbage's theoretical punch-card machine and becomes the first programmer.

1847 - George Boole creates Boolean logic, a form of algebra in which all values can be reduced to the binary values of true or false.

1936 - English logician and cryptanalyst Alan Turing proposes a universal machine that could decipher and execute a set of instructions. His published proof is considered the basis of computer science.

1952 - Arthur Samuel creates a program to help an IBM computer get better at checkers the more it plays.

1959 - MADALINE becomes the first artificial neural network applied to a real-world problem: removing echoes from phone lines.

1985 - Terry Sejnowski's and Charles Rosenberg's artificial neural network taught itself how to correctly pronounce 20,000 words in one week.

1997 - IBM's Deep Blue beat chess grandmaster Garry Kasparov.

1999 - A CAD prototype intelligent workstation reviewed 22,000 mammograms and detected cancer 52% more accurately than radiologists did.

2006 - Computer scientist Geoffrey Hinton invents the term deep learning to describe neural net research.

2012 - An unsupervised neural network created by Google learned to recognize cats in YouTube videos with 74.8% accuracy.

2014 - A chatbot passes the Turing Test by convincing 33% of human judges that it was a Ukrainian teen named Eugene Goostman.

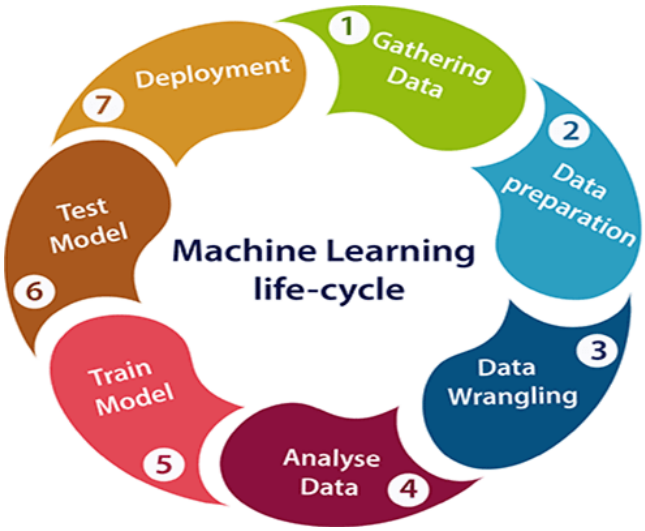
2014 - Google's AlphaGo defeats the human champion in Go, the most difficult board game in the world.

2016 - LipNet, DeepMind's artificial intelligence system, identifies lip-read words in video with an accuracy of 93.4%.

2019 - Amazon controls 70% of the market share for virtual assistants in the U.S.

- **Machine learning life cycle** involves seven major steps, which are given below:

  - **Gathering Data**
  - **Data preparation**
  - **Data Wrangling**
  - **Analyse Data**
  - **Train the model**
  - **Test the model**
  - **Deployment**



The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

1. Gathering Data:

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as **files**, **database**, **internet**, or **mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

- **Identify various data sources**
- **Collect data**
- **Integrate the data obtained from different sources**

By performing the above task, we get a coherent set of data, also called as a **dataset**. It will be used in further steps.

---

2. Data preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.

In this step, first, we put all data together, and then randomize the ordering of data.

This step can be further divided into two processes:

- **Data exploration:**
  It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
- **Data pre-processing:**
  Now the next step is preprocessing of data for its analysis.

---

3. Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- **Missing Values**
- **Duplicate data**
- **Invalid data**
- **Noise**

So, we use various filtering techniques to clean the data.

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

---

4. Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- **Selection of analytical techniques**
- **Building models**
- **Review the result**

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as **Classification**, **Regression**, **Cluster analysis**, **Association**, etc. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

---

5. Train Model

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

---

6. Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

---

7. Deployment

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

Unit – 2

What is Predictive Modeling

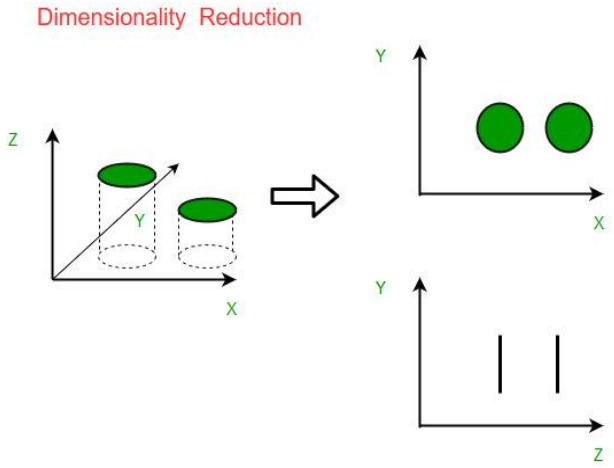
Predictive modeling is a probabilistic process that allows us to forecast outcomes, on the basis of some predictors. These predictors are basically features that come into play when deciding the final result, i.e. the outcome of the model.

What is Dimensionality Reduction?

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

Why is Dimensionality Reduction important in Machine Learning and Predictive Modeling?

An intuitive example of dimensionality reduction can be discussed through a simple e-mail classification problem, where we need to classify whether the e-mail is spam or not. This can involve a large number of features, such as whether or not the e-mail has a generic title, the content of the e-mail, whether the e-mail uses a template, etc. However, some of these features may overlap. In another condition, a classification problem that relies on both humidity and rainfall can be collapsed into just one underlying feature, since both of the aforementioned are correlated to a high degree. Hence, we can reduce the number of features in such problems. A 3-D classification problem can be hard to visualize, whereas a 2-D one can be mapped to a simple 2 dimensional space, and a 1-D problem to a simple line. The below figure illustrates this concept, where a 3-D feature space is split into two 1-D feature spaces, and later, if found to be correlated, the number of features can be reduced even further.



Components of Dimensionality Reduction

There are two components of dimensionality reduction:

- **Feature selection:** In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. It usually involves three ways:
  1. Filter
  2. Wrapper
  3. Embedded

- **Feature extraction:** This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

Methods of Dimensionality Reduction

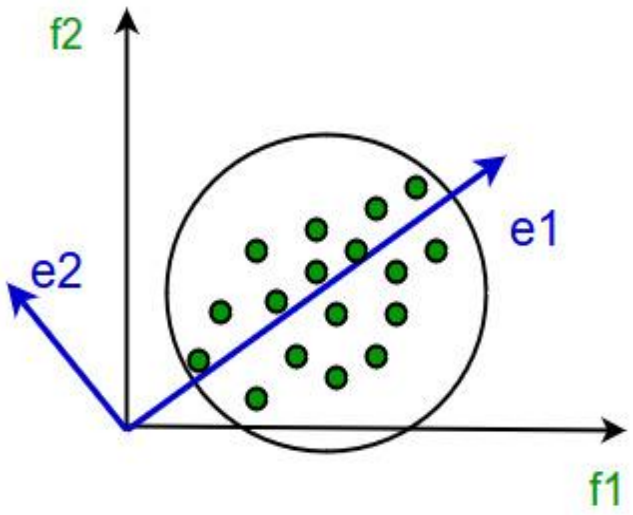
The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

Dimensionality reduction may be both linear or non-linear, depending upon the method used. The prime linear method, called Principal Component Analysis, or PCA, is discussed below.

Principal Component Analysis

This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.



It involves the following steps:

- Construct the covariance matrix of the data.
- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.

Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.

Advantages of Dimensionality Reduction

- It helps in data compression, and hence reduced storage space.
- It reduces computation time.
- It also helps remove redundant features, if any.

Disadvantages of Dimensionality Reduction

- It may lead to some amount of data loss.
- PCA tends to find linear correlations between variables, which is sometimes undesirable.
- PCA fails in cases where mean and covariance are not enough to define datasets.
- We may not know how many principal components to keep- in practice, some thumb rules are applied.

What is a vector?

A **vector** is an object that has both a magnitude and a direction. Geometrically, we can picture a **vector** as a directed line segment, whose length is the magnitude of the **vector** and with an arrow indicating the direction. The direction of the **vector** is from its tail to its head.

✓ **Row-Vector Representation:**

$$\begin{bmatrix} a_1 & a_2 & a_3 \ldots a_n \end{bmatrix}$$

The row vector is a (1xn) matrix where the number of rows is 1 and the number of columns is ‘n’.

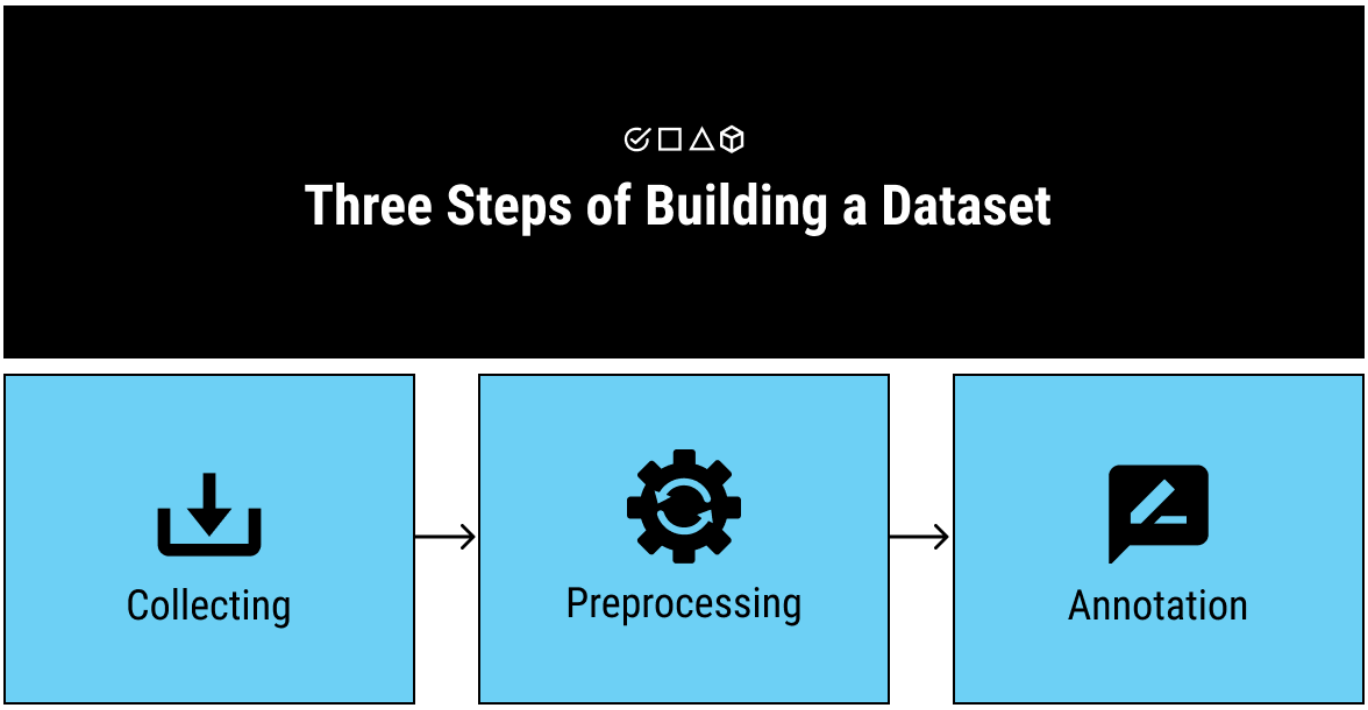
✓ **Column-Vector Representation:**

$$\begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{bmatrix}$$

The column vector is an (nx1) matrix where the number of rows is ‘n’ and the number of columns is 1.

- ✓ **Matrix Representation:**

Say, we have a matrix A as shown below. The vertical numbers i.e. 1 → m correspond to the number of rows, and the horizontal numbers i.e. 1 → n correspond to the number of columns. Therefore, we have a matrix A of size (m x n).



Preprocessing data

The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

In general, learning algorithms benefit from standardization of the data set. If some outliers are present in the set, robust scalers or transformers are more appropriate. The behaviors of the different scalers, transformers, and normalizers on a dataset containing marginal outliers is highlighted .

Normailsation

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

Standardization

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

- Here's the formula for standardization:
- $$X' = \frac{X - \mu}{\sigma}$$
- μ is the mean of the feature values and σ is the standard deviation of the feature values. Note that in this case, the values are not restricted to a particular range.

Covariance Matrix

Covariance Matrix is a measure of how much two random variables gets change together. It is actually used for computing the covariance in between every column of data matrix.

The Covariance Matrix is also known as dispersion matrix and variance-covariance matrix. The covariance between two jointly distributed real-valued random variables X and Y with finite second moments is defined as.

$$Cov(X,Y) = \sum \frac{(X_i - \overline{X})(Y_i - \overline{Y})}{N} = \sum \frac{x_i y_i}{N}$$

Where,

N = Number of scores in each set of data

X = Mean of the N scores in the first data set

X_i = i^{th} raw score in the first set of scores

x_i = i^{th} deviation score in the first set of scores

Y = Mean of the N scores in the second data set

Y_i = i^{th} raw score in the second set of scores

y_i = i^{th} deviation score in the second set of scores

Cov(X, Y) = Covariance of corresponding scores in the two sets of data

Unit – 3

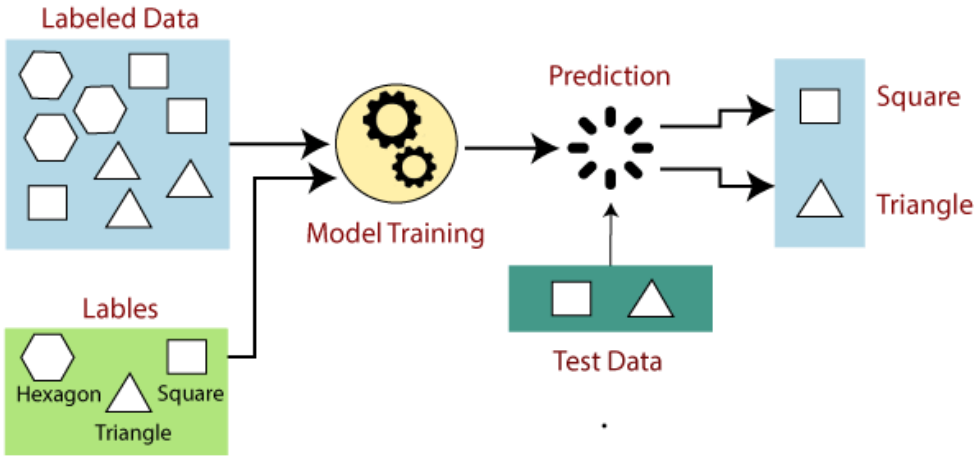
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

In the real-world, supervised learning can be used for **Risk Assessment, Image classification, Fraud Detection, spam filtering**, etc.

Supervised Learning

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
- If the given shape has three sides, then it will be labelled as a **triangle**.
- If the given shape has six equal sides then it will be labelled as **hexagon**.

Now, after training we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Supervised learning uses a training set to teach models to yield the desired output. This training dataset includes inputs and correct outputs, which allow the model to learn over

time. The algorithm measures its accuracy through the loss function, adjusting until the error has been sufficiently minimized.

Supervised learning can be separated into two types of problems when data mining—classification and regression:

- **Classification** uses an algorithm to accurately assign test data into specific categories. It recognizes specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined. Common classification algorithms are linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbor, and random forest, which are described in more detail below.
- **Regression** is used to understand the relationship between dependent and independent variables. It is commonly used to make projections, such as for sales revenue for a given business. Linear regression, logistical regression, and polynomial regression are popular regression algorithms.

Supervised learning algorithms

Primarily leveraged for deep learning algorithms, neural networks process training data by mimicking the interconnectivity of the human brain through layers of nodes. Each node is made up of inputs, weights, a bias (or threshold), and an output. If that output value exceeds a given threshold, it "fires" or activates the node, passing data to the next layer in the network. Neural networks learn this mapping function through supervised learning, adjusting based on the loss function through the process of gradient descent. When the cost function is at or near zero, we can be confident in the model's accuracy to yield the correct answer.

1. Naive Bayes

Naive Bayes is classification approach that adopts the principle of class conditional independence from the Bayes Theorem. This means that the presence of one feature does not impact the presence of another in the probability of a given outcome, and each predictor has an equal effect on that result. There are three types of Naïve Bayes classifiers: Multinomial Naïve Bayes, Bernoulli Naïve Bayes, and Gaussian Naïve Bayes. This technique is primarily used in text classification, spam identification, and recommendation systems.

2. Linear regression

Linear regression is used to identify the relationship between a dependent variable and one or more independent variables and is typically leveraged to make predictions about future outcomes. When there is only one independent variable and one dependent variable, it is known as simple linear regression. As the number of independent variables increases, it is referred to as multiple linear regression. For each type of linear regression, it seeks to plot a line of best fit, which is calculated through the method of least squares. However, unlike other regression models, this line is straight when plotted on a graph.

3. **Logistic regression**

While linear regression is leveraged when dependent variables are continuous, logistical regression is selected when the dependent variable is categorical, meaning they have binary outputs, such as "true" and "false" or "yes" and "no." While both regression models seek to understand relationships between data inputs, logistic regression is mainly used to solve binary classification problems, such as spam identification.

4. **Support vector machine (SVM)**

A support vector machine is a popular supervised learning model developed by Vladimir Vapnik, used for both data classification and regression. That said, it is typically leveraged for classification problems, constructing a hyperplane where the distance between two classes of data points is at its maximum. This hyperplane is known as the decision boundary, separating the classes of data points (e.g., oranges vs. apples) on either side of the plane.

5. **K-nearest neighbor**

K-nearest neighbor, also known as the KNN algorithm, is a non-parametric algorithm that classifies data points based on their proximity and association to other available data. This algorithm assumes that similar data points can be found near each other. As a result, it seeks to calculate the distance between data points, usually through Euclidean distance, and then it assigns a category based on the most frequent category or average.

Its ease of use and low calculation time make it a preferred algorithm by data scientists, but as the test dataset grows, the processing time lengthens, making it less appealing for classification tasks. KNN is typically used for recommendation engines and image recognition.

Unit – 4

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

Clustering

Clustering is a data mining technique which groups unlabeled data based on their similarities or differences. Clustering algorithms are used to process raw, unclassified data objects into groups represented by structures or patterns in the information. Clustering algorithms can be categorized into a few types, specifically exclusive, overlapping, hierarchical, and probabilistic.

Exclusive and Overlapping Clustering

Exclusive clustering is a form of grouping that stipulates a data point can exist only in one cluster. This can also be referred to as “hard” clustering. The K-means clustering algorithm is an example of exclusive clustering.

- **K-means clustering**

K means it is an iterative clustering algorithm which helps you to find the highest value for every iteration. Initially, the desired number of clusters are selected. In this clustering method, you need to cluster the data points into k groups. A larger k means smaller groups with more granularity in the same way. A lower k means larger groups with less granularity.

The output of the algorithm is a group of “labels.” It assigns data point to one of the k groups. In k-means clustering, each group is defined by creating a centroid for each group. The centroids are like the heart of the cluster, which captures the points closest to them and adds them to the cluster.

K-mean clustering further defines two subgroups:

- Agglomerative clustering
- Dendrogram

1. **Agglomerative clustering**

This type of K-means clustering starts with a fixed number of clusters. It allocates all data into the exact number of clusters. This clustering method does not require the number of clusters K as an input. Agglomeration process starts by forming each data as a single cluster.

This method uses some distance measure, reduces the number of clusters (one in each iteration) by merging process. Lastly, we have one big cluster that contains all the objects.

2. **Dendrogram**

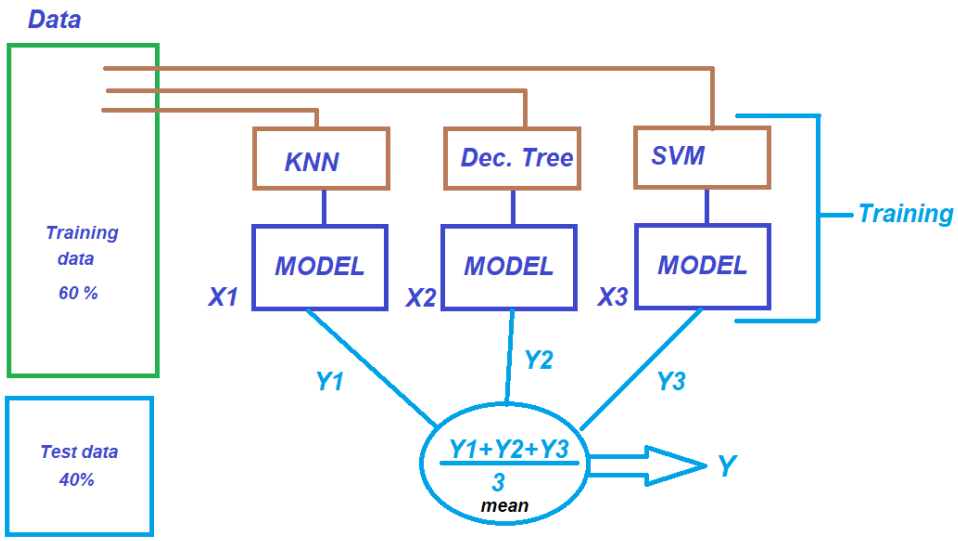
In the Dendrogram clustering method, each level will represent a possible cluster. The height of dendrogram shows the level of similarity between two join clusters. The closer to the bottom of the process they are more similar cluster which is finding of the group from dendrogram which is not natural and mostly subjective.

Ensemble methods

Ensembles are methods that combine multiple machine learning models (KNN, Decision Tree, SVM etc..) to create more powerful models (Image 1). The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner. There are many models in the machine learning literature that belong to this category: **Random Forests** , **Gradient Boosted Decision Tree** etc.

Why ensembles?

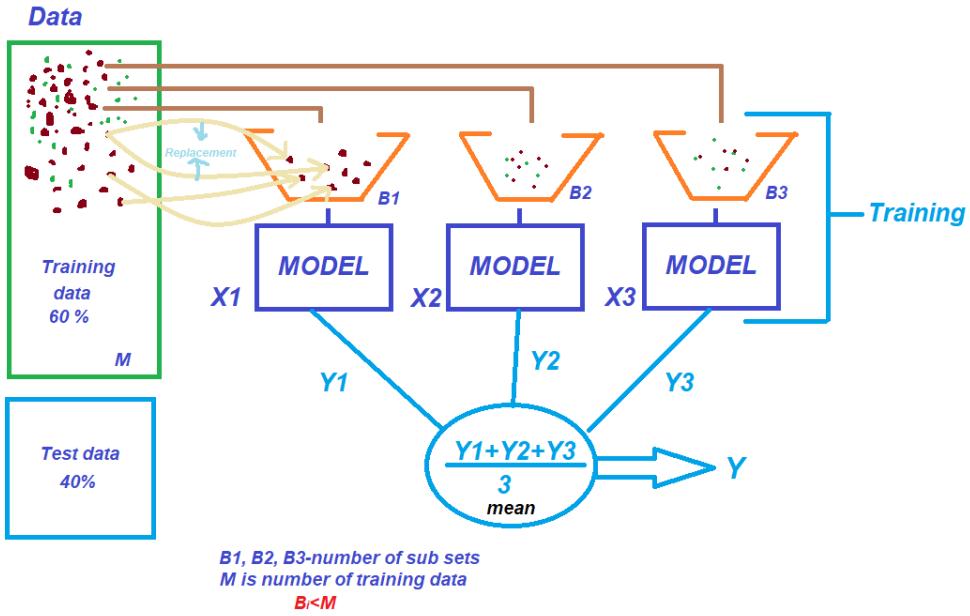
1. Lower error
2. Less overfitting
3. Reduces variance



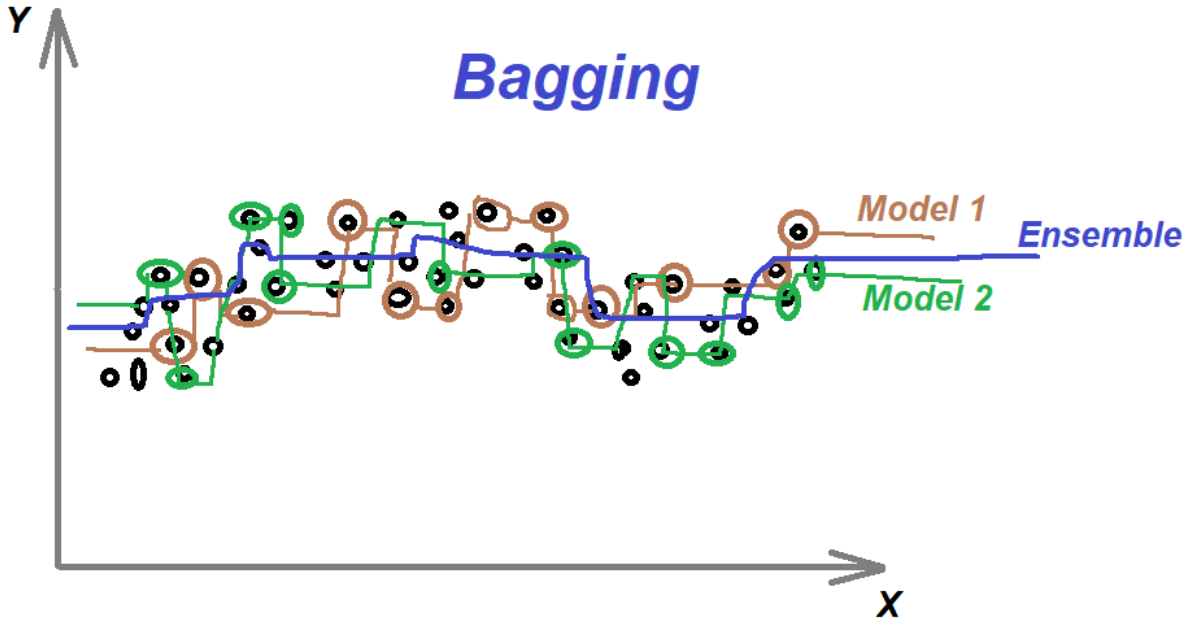
Bagging (bootstrap aggregating)

Bagging is a machine learning ensemble algorithm designed to improve the accuracy of machine learning algorithms used in classification and regression. It also reduces variance

and helps to avoid overfitting. We can choose randomly sub sets from training data with replacement.As a result get average of predictions for each bag.Take a look (Image 2)



Below we have chosen two sub set (bag) and trained.



- **Boosting (Ada boost)**

For boosting-we build our first bag of data with select randomly from training data and train model in a usual way. Next is take all our training data and use it to test the model. We will discover that some of the points are not well predicted (significant error). For second bag we choose randomly data again but each instance is weighted according to this error.Now we

test our system altogether and combine their outputs and again we measure error across all this data.Thus we build next bag and so on..

Building Random Forests

Random forests are collection of decision trees, where each tree is slightly different from the others. To get more information about Decision Tree you can read my another article (int this article I explained what is Decision Tree and how it works).

The idea of random forests is to build many trees, so we can reduce the amount of overfitting by averaging their results.

In order to implement this , we need to build many decision trees that should be different from the other trees.

In order to build random forests we need to decide how many trees should we choose. There is n_estimator (number of trees) parameter of RandomForestRegressor or RandomForestClassifier in Scikit-Learn.

To start to build Random Forest model first of all we need to understand how ***bootstrap*** works. Bootstrap means we can choose data randomly with replacement by repeating count of data. Let's say we have data like this **data=[a, b, c, d, e , f]**, so count of data is **n=len(data)=6.** A possible bootstrap sample would be [b,d,d,c,e,f], another possible sample would be [c, d, b, b, f, f] and so on..after repeat 6 times.

Performance measurement of models in terms of accuracy

In the field of machine learning, other than building models, it's equally important to measure the performance of the model. Basically, we check how good are the predictions made by our model.

In this series of articles, we will try to understand what are the various performance measures of a model.

Accuracy

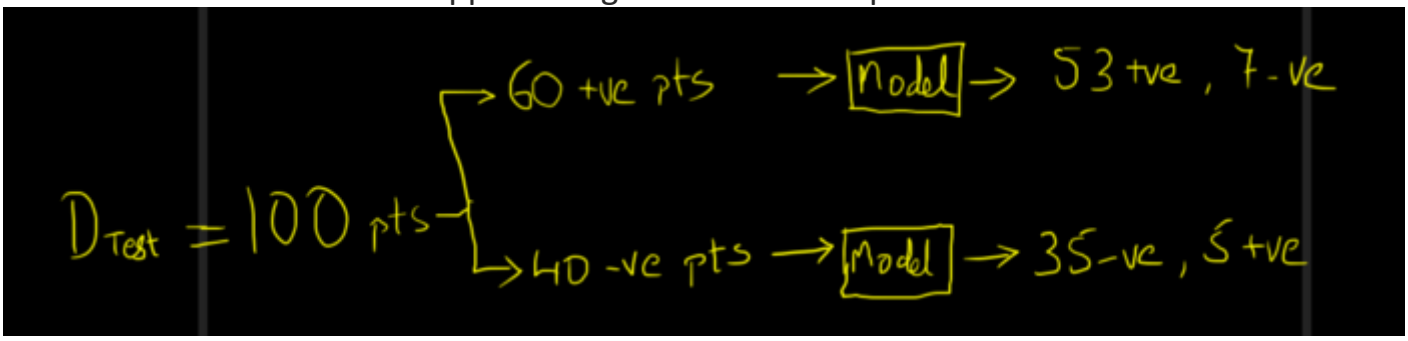
This is probably the simplest performance metrics. It is defined as:

$$Accuracy = \frac{Number\ of\ correctly\ classified\ points}{Total\ number\ of\ point}$$

Accuracy value lies between 0 and 1. If the value is closer to 0 it's considered as bad performance, whereas if the value is closer to 1 then its considered good performance. It is one of the simplest and easy to understand metric.

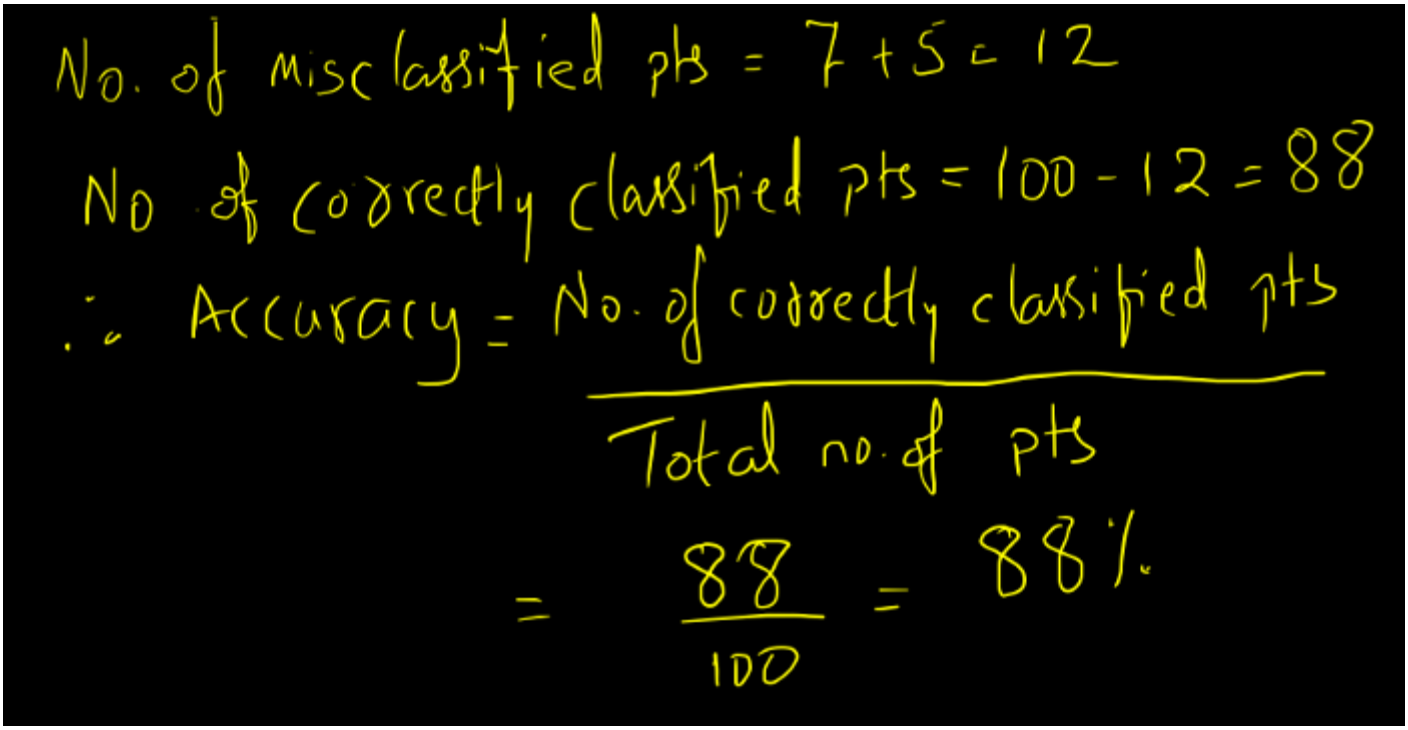
Let’s understand this metric using an example:

Assume we have already trained our model using training data. Now, we want to use the test data and check how accurate the predictions are. Let's say we have a classification problem with 100 data points in our test data set. The objective is to classify whether the point is positive or negative. Assume out of the 100 points, we have 60 positive points and 40 negative points (Note that this is the original/actual class label). Now, when we feed this test data to our model and suppose we get the below output:



So basis the above example, our model has misclassified 7 points as negative and 5 points as positive. So overall misclassified points = 7+5 = 12.

So the accuracy of the model can be calculated as:



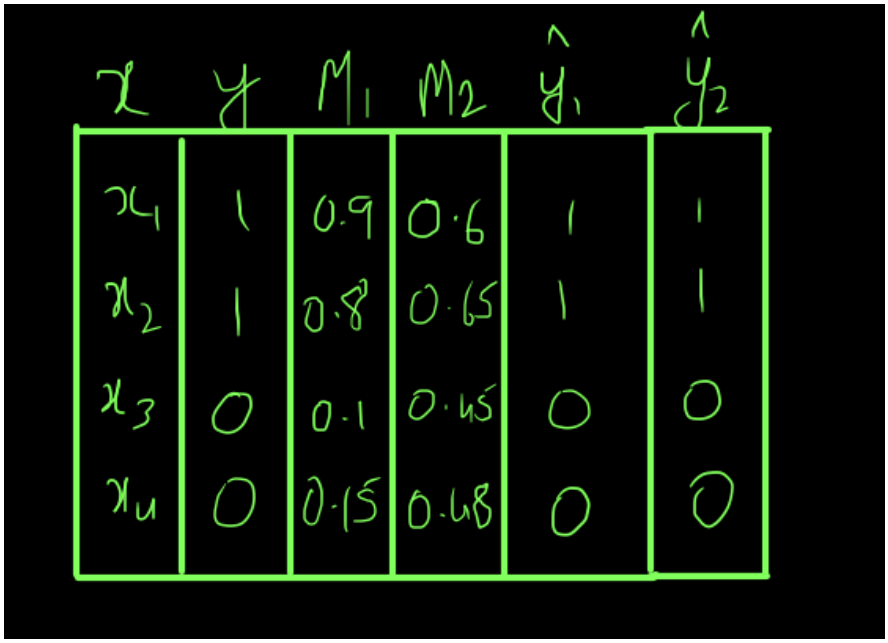
Now that we have understood how to calculate accuracy, let's understand some of the problems associated with it.

Imbalanced data set

Let's say we have a model that returns negative class as my output. Now, suppose we have an imbalanced data set which is also my test data set, and say 90% of the total test data set is negative. Now when we input this test data to our model, we will get 90% of the classification correct as my model returns negative class label and 90% of my test data set is negative. In such a scenario, even the dumb model gives me an accuracy of 90%. So stay away from accuracy when you have an imbalanced data set.

Accuracy doesn’t consider probability scores

Consider the below example to understand this:



x	y	M1	M2	$\hat{y}_1$	$\hat{y}_2$
x_1	1	0.9	0.6	1	1
x_2	1	0.8	0.65	1	1
x_3	0	0.1	0.45	0	0
x_4	0	0.15	0.48	0	0

x ➜ Datapoints

y ➜ Actual class label

M1➜ Probability score of model M1

M2 ➜ Probability score of model M2

Y1 ➜ Predicted class label of model M1

Y2 ➜ Predicted class label of model M2

Let's assume we ran our data through 2 models M1 and M2 and these models returned the probability scores. So given a data point, we get a probability of P(y=1).

M1 can be read as the probability score of x1 having y1=1 is 0.9 when ran through model M1. Similarly, the probability score of x3 having y3=1 is 0.1, which means the probability of y3=1 is very less when ran through model M1 (means P(y3=0) = 0.9).

Consider x1. For x1, the actual class label is 1. Our model M1 gives a probability score of P(y=1) = 0.9 and is predicted to belong to class label 1. On the other hand model, M2 gives a

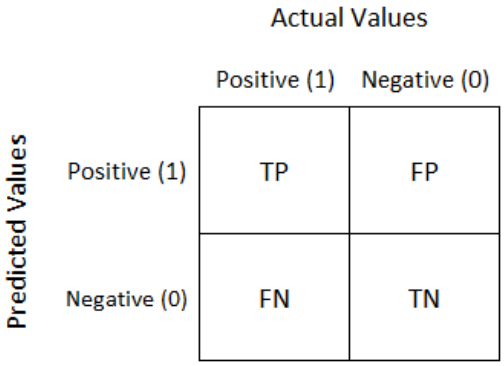
probability score of P(y=1) = 0.6, and hence it is also classified as class label 1. But if we consider the probability scores, its clear that my model M1 is performing better than my model M2. Similarly, for x2, x3, and x4 the probability scores of model M1 are much better compared to model M2. However, their predicted class labels remain the same in both models. Accuracy as a measure doesn't distinguish which model is better since it doesn't use probability scores. It can only use predicted class labels and since it uses predicted class labels to calculate accuracy it will say that model M1 and M2 have the same accuracy but from probability scores, it is clear that M1 is better than M2.

Confusion Matrix

Confusion Matrix

A much better way to evaluate the performance of a classifier is to look at the confusion matrix. The general idea is to count the number of times instances of class A are classified as class B.

To understand the confusion matrix, let's take a binary classification task where the objective is to classify the class label as either 0 (negative) or 1 (positive). Let's construct the confusion matrix for the same.



TN ➜ True Negative

FN ➜ False Negative

FP ➜ False Positive

TP ➜ True Positive

N ➜ Total no. of negative points

P ➜ Total no. of positive points

Let's understand each of the above terms:

- **True Negative** ➔ when the actual value is 0 and the model predicted value is also 0
- **False Negative** ➔ when the actual value is 1 and the model predicted value is 0
- **True Positive** ➔ when the actual value is 1 and the model predicted value is also 1
- **False Positive** ➔ when the actual value is 0 and the model predicted value is 1

Now that we have understood how to construct a confusion matrix and also its basic terminologies, let's understand some of the key metrics associated with it.

True Positive Rate (TPR) = # of TP / Total # of P

True Negative Rate (TNR) = # of TN / Total # of N

False Positive Rate (FPR) = # of FP / Total # of N

False Negative Rate (FNR) = # of FN / Total # of P

Precision = TP / (TP + FP)
It means, of all the points the model predicted to be positive, what % of them are actually positive. In precision, we are not bothered about the negative class. Our only focus is on the positive class label.

Recall = TPR = TP / Total # of P
It means, of all the points that “actually” belong to class label 1, how many the model predicted to be class label 1.

For a good model, we would always want the values of precision and recall to be high.

F1-score:

It's the combination of both metrics precision and recall and is given as follows:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Receiver Operating Characteristic Curve (ROC) curve

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

True Positive Rate (**TPR**) is a synonym for recall and is therefore defined as follows:

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

False Positive Rate (**FPR**) is defined as follows:

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

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

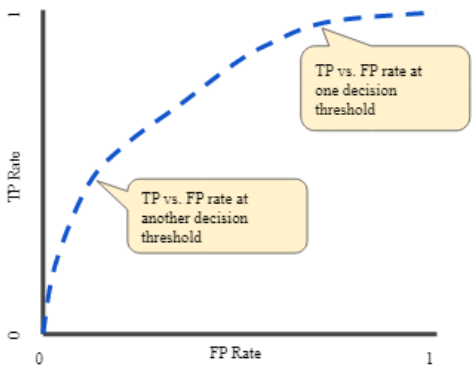


Figure 4. TP vs. FP rate at different classification thresholds.

To compute the points in an ROC curve, we could evaluate a logistic regression model many times with different classification thresholds, but this would be inefficient. Fortunately,

there's an efficient, sorting-based algorithm that can provide this information for us, called AUC.

AUC: Area Under the ROC Curve

AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

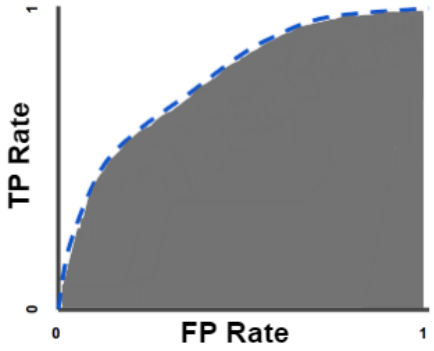


Figure 5. AUC (Area under the ROC Curve).

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:

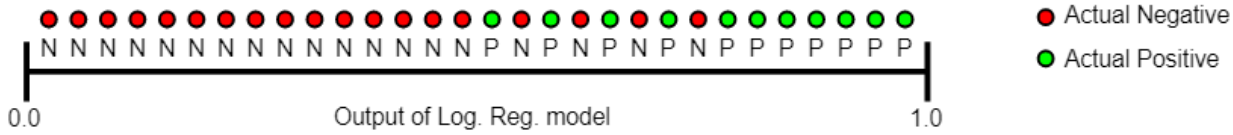


Figure 6. Predictions ranked in ascending order of logistic regression score.

AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example.

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

AUC is desirable for the following two reasons:

- AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.
- AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

However, both these reasons come with caveats, which may limit the usefulness of AUC in certain use cases:

- **Scale invariance is not always desirable.** For example, sometimes we really do need well calibrated probability outputs, and AUC won't tell us about that.
- **Classification-threshold invariance is not always desirable.** In cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimize one type of classification error. For example, when doing email spam detection, you likely want to prioritize minimizing false positives (even if that results in a significant increase of false negatives). AUC isn't a useful metric for this type of optimization.

Median Absolute Deviation (MAD)

The Median Absolute Deviation (MAD) is a simple way to quantify variation. Half the values are closer to the median than the MAD, and half are further away. GraphPad Prism does not (yet) compute the MAD.

The best way to understand the MAD is to understand how it is calculated. First, compute the median of all the values. The median is the 50th percentile. Then calculate how far each value is from the median of all values. Regardless of whether the value is greater or less than the median, express the distance between it and the median as a positive value (in math terminology, take the absolute value of the difference between the value and the median). Now find the median of that set of differences. The result is the Median Absolute Deviation, abbreviated MAD.

Note one point of confusion. There are two distinct computations of the median. First one computes the median of the actual data. Then one computes the median of the distances of the values from that median.

There are three error metrics that are 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)

F1 Score

F1 Score is used to measure a test's accuracy

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).

High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as :

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

- F1 Score tries to find the balance between precision and recall.
- **Precision :** It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

- **Recall :** It is the number of correct positive results divided by the number of ***all*** relevant samples (all samples that should have been identified as positive).

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Mean Absolute Error

Mean Absolute Error is the average of the difference between the Original Values and the Predicted Values. It gives us the measure of how far the predictions were from the actual output. However, they don't gives us any idea of the direction of the error i.e. whether we are under predicting the data or over predicting the data. Mathematically, it is represented as :

$$MeanAbsoluteError = \frac{1}{N}\sum_{j=1}^{N} |y_j - \hat{y}_j|$$

Mean Squared Error

Mean Squared Error(MSE) is quite similar to Mean Absolute Error, the only difference being that MSE takes the average of the **square** of the difference between the original values and

the predicted values. The advantage of MSE being that it is easier to compute the gradient, whereas Mean Absolute Error requires complicated linear programming tools to compute the gradient. As, we take square of the error, the effect of larger errors become more pronounced then smaller error, hence the model can now focus more on the larger errors.

$$MeanSquaredError = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2$$

- **RMSE: Root Mean Square Error**

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.