Unit-1

Introduction:

ARTIFICIAL INTELIGENCE:

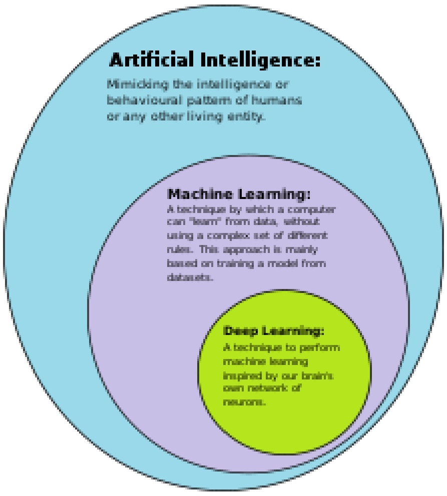
Artificial intelligence (AI) is the ability of a computer or a robot controlled by a computer to do tasks that are usually done by humans because they require human intelligence and discernment. Although there are no AIs that can perform the wide variety of tasks an ordinary human can do, some AIs can match humans in specific tasks.

MACHINE LEARNING:

machine learning is the concept that a computer program can learn and adapt to new data without human intervention. Machine learning is a field of artificial intelligence (AI) that keeps a computer's built-in algorithms current regardless of changes in the worldwide economy.

Machine learning can be applied in a variety of areas, such as in investing, advertising, lending, organizing news, fraud detection, and more.

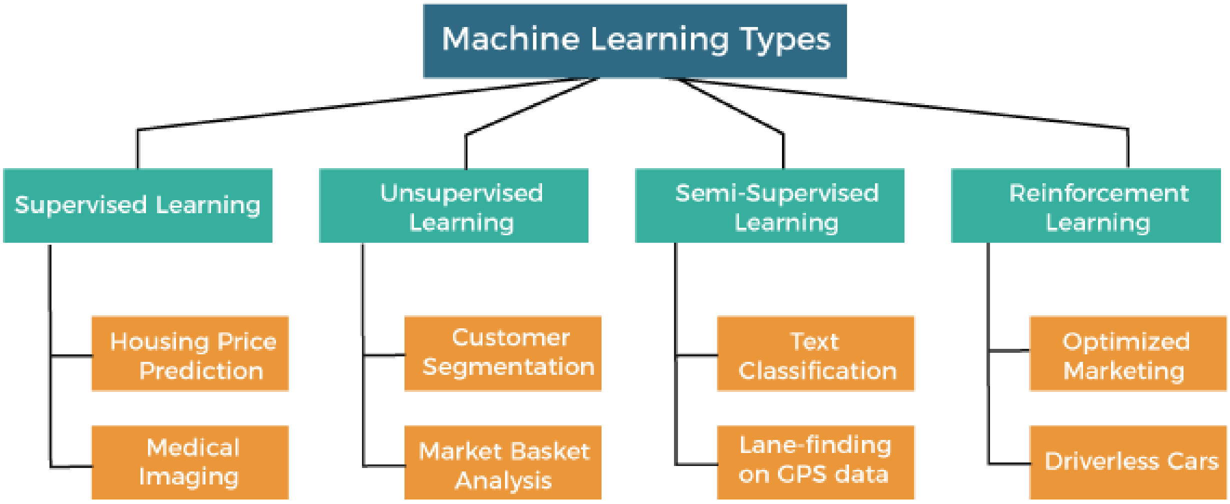
DEEP LEARNING:


Artificial Intelligence:
Mimicking the intelligence or behavioural pattern of humans or any other living entity.
Machine Learning:
A technique by which a computer can "learn" from data, without using a complex set of different rules. This approach is mainly based on training a model from datasets.
Deep Learning:
A technique to perform machine learning inspired by our brain's own network of neurons.

Deep learning is a subset of machine learning (ML), which is itself a subset of artificial intelligence (AI). The concept of AI has been around since the 1950s, with the goal of making computers able to think and reason in a way similar to humans. As part of making machines able to think, ML is focused on how to make them learn without being explicitly programmed. Deep learning goes beyond ML by creating more complex hierarchical models that are meant to mimic how humans learn new information

TYPES OF MECHINE LEARNING:

Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and on the basis of training, they build the model & perform a specific task



1. Supervised Machine Learning

As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

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

- **Classification**
- **Regression**

2. Unsupervised Machine Learning

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

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

- **Clustering**
- **Association**

3. Semi-Supervised Learning

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning**.** It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

Although Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabeled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels.

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced. The main aim of semi-supervised learning is to effectively use all the available data, rather than only labelled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labelled data. It is because labelled data is a comparatively more expensive acquisition than unlabeled data.

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

Advantages and disadvantages of Semi-supervised Learning

Advantages:

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

Disadvantages:

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

4. Reinforcement Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance**.** Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

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

The reinforcement learning process is similar to a human being; for example, a child learns various things by experiences in his day-to-day life. An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score. Agent receives feedback in terms of punishment and rewards.

Due to its way of working, reinforcement learning is employed in different fields such as Game theory, Operation Research, Information theory, multi-agent systems.

A reinforcement learning problem can be formalized using **Markov Decision Process(MDP).** In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.

Categories of Reinforcement Learning

Reinforcement learning is categorized mainly into two types of methods/algorithms:

- **Positive Reinforcement Learning:** Positive reinforcement learning specifies increasing the tendency that the required behaviour would occur again by adding something. It enhances the strength of the behaviour of the agent and positively impacts it.
- **Negative Reinforcement Learning:** Negative reinforcement learning works exactly opposite to the positive RL. It increases the tendency that the specific behaviour would occur again by avoiding the negative condition

. **Advantages:**

- It helps in solving complex real-world problems which are difficult to be solved by general techniques.
- The learning model of RL is similar to the learning of human beings; hence most accurate results can be found.
- Helps in achieving long term results.

Disadvantage

- RL algorithms are not preferred for simple problems.
- RL algorithms require huge data and computations.
- Too much reinforcement learning can lead to an overload of states which can weaken the results.

MAIN CHALLENGES OF MACHINE LEARNING:

- Not enough training data.
- Poor Quality of data.
- Irrelevant features.
- Nonrepresentative training data.
- Overfitting and Underfitting.

1. Not enough training data :

Let’s say for a child, to make him learn what an apple is, all it takes for you to point to an apple and say apple repeatedly. Now the child can recognize all sorts of apples.

Well, machine learning is still not up to that level yet; it takes a lot of data for most of the algorithms to function properly. For a simple task, it needs thousands of examples to make something out of it, and for advanced tasks like image or speech recognition, it may need lakhs(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).”

Image Source

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 and Underfitting :

Let's start with an example, say one day you are walking down a street to buy something, a dog comes out of nowhere you offer him something to eat but instead of eating he starts barking and chasing you but somehow you are safe. After this particular incident, you might think all dogs are not worth treating nicely.

So this **overgeneralization** is what we humans do most of the time, and unfortunately machine learning model also does the same if not paid attention. In machine learning, we call this overfitting i.e model performs well on training data but fails to generalize well.

Overfitting happens when our model is too complex.

Things which we can do to overcome this problem:

1. Simplify the model by selecting one with fewer parameters.
2. By reducing the number of attributes in training data.
3. Constraining the model.
4. Gather more training data.

STATISTICAL LEARNING:

INTRODUCTION:

An Introduction to Statistical Learning provides a broad and less technical treatment of key topics in statistical learning. Each chapter includes an R lab. This book is appropriate for anyone who wishes to use contemporary tools for data analysis.

SUPERVISED LEARNING:

As its name suggests, Supervised machine learning

is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the **shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are**

smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

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

Categories of Supervised Machine Learning

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

- **Classification**
- **Regression**

a) Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as **"Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

Some popular classification algorithms are given below:

- **Random Forest Algorithm**
- **Decision Tree Algorithm**
- **Logistic Regression Algorithm**
- **Support Vector Machine Algorithm**

b) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

- **Simple Linear Regression Algorithm**
- **Multivariate Regression Algorithm**
- **Decision Tree Algorithm**
- **Lasso Regression**

Advantages and Disadvantages of Supervised Learning

Advantages:

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

Disadvantages:

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

Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

- **ImageSegmentation:**
  Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.
- **MedicalDiagnosis:**
  Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labelled data with labels for disease conditions. With such a process, the machine can identify a disease for the new patients.
- **Fraud Detection -** Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.
- **Spam detection -** In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.
- **Speech Recognition -** Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc

UNSUPERVISED LEARNING:

Unsupervised learnin

Is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

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

Let's take an example to understand it more preciously; suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects.

So, now the machine will discover its patterns and differences, such as colour difference, shape difference, and predict the output when it is tested with the test dataset.

Categories of Unsupervised Machine Learning

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

- **Clustering**
- **Association**

1) Clustering

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

Some of the popular clustering algorithms are given below:

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

2) Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc.

Some popular algorithms of Association rule learning are **Apriori Algorithm, Eclat, FP-growth algorithm.**

Advantages and Disadvantages of Unsupervised Learning Algorithm

Advantages:

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

Disadvantages:

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

Applications of Unsupervised Learning

- **Network Analysis:** Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.
- **Recommendation Systems:** Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.
- **Anomaly Detection:** Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent transactions.
- **Singular Value Decomposition:** Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location

TRAINING AND TESTING:

What is Training Dataset?

The ***training data is the biggest (in -size) subset of the original dataset, which is used to train or fit the machine learning model***. Firstly, the training data is fed to the ML algorithms, which lets them learn how to make predictions for the given task.

For example, for training a sentiment analysis model, the training data could be as below:

Input	Output (Labels)
The New UI is Great	Positive
Update is really Slow	Negative

The training data varies depending on whether we are using Supervised Learning or Unsupervised Learning Algorithms.

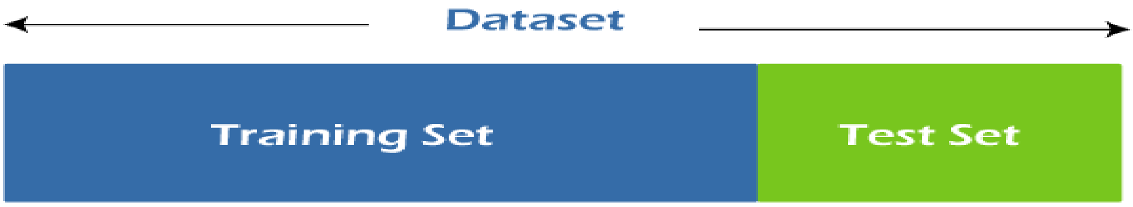
For **Unsupervised learning**, the training data contains unlabeled data points, i.e., inputs are not tagged with the corresponding outputs. Models are required to find the patterns from the given training datasets in order to make predictions.

On the other hand, for supervised learning, the training data contains labels in order to train the model and make predictions.

The type of training data that we provide to the model is highly responsible for the model's accuracy and prediction ability. It means that the better the quality of the training data, the better will be the performance of the model. Training data is approximately more than or equal to 60% of the total data for an ML project.

What is Test Dataset

Once we train the model with the training dataset, it's time to test the model with the test dataset. This dataset evaluates the performance of the model and ensures that the model can generalize well with the new or unseen dataset. ***The test dataset is another subset of original data, which is independent of the training dataset***. However, it has some similar types of features and class probability distribution and uses it as a benchmark for model evaluation once the model training is completed. Test data is a well-organized dataset that contains data for each type of scenario for a given problem that the model would be facing when used in the real world. Usually, the test dataset is approximately 20-25% of the total original data for an ML project.

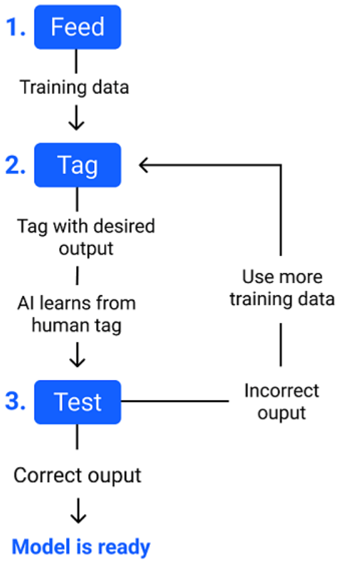


Machine Learning algorithms enable the machines to make predictions and solve problems on the basis of past observations or experiences. These experiences or observations an algorithm can take from the training data, which is fed to it. Further, one of the great things about ML algorithms is that they can learn and improve over time on their own, as they are trained with the relevant training data.

Once the model is trained enough with the relevant training data, it is tested with the test data. We can understand the whole process of training and testing in three steps, which are as follows:

1. **Feed:** Firstly, we need to train the model by feeding it with training input data.
2. **Define:** Now, training data is tagged with the corresponding outputs (in Supervised Learning), and the model transforms the training data into text vectors or a number of data features.
3. **Test:** In the last step, we test the model by feeding it with the test data/unseen dataset. This step ensures that the model is trained efficiently and can generalize well.

The above process is explained using a flowchart given below:



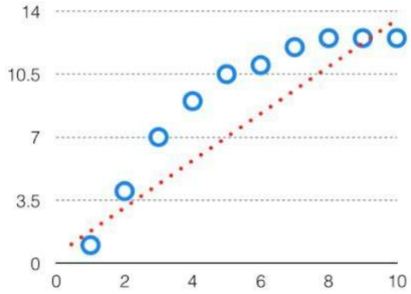
<u>TRADEOFFS IN STATISTICAL LEARNING</u>:

It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine learning algorithm. There is a tradeoff between a model's ability to minimize bias and variance which is referred to as the best solution for selecting a value of **Regularization** constant.

Proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

Bias

The bias is known as the difference between the prediction of the values by the ML model and the correct value. Being high in biasing gives a large error in training as well as testing data. Its recommended that an algorithm should always be low biased to avoid the problem of underfitting. By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as **Underfitting of Data**. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation.



HighBias

In such a problem, a hypothesis looks like follows.

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

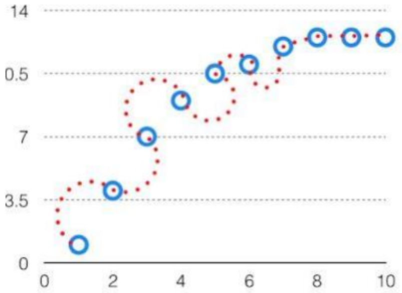
Variance

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

When a model is high on variance, it is then said to as **Overfitting of Data**. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high.

While training a data model variance should be kept low.

The high variance data looks like follows.



High Variance

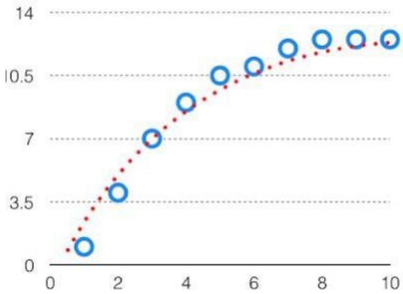
In such a problem, a hypothesis looks like follows.

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \\ + \theta_3 x^3 + \theta_4 x^4$$

Bias Variance Tradeoff

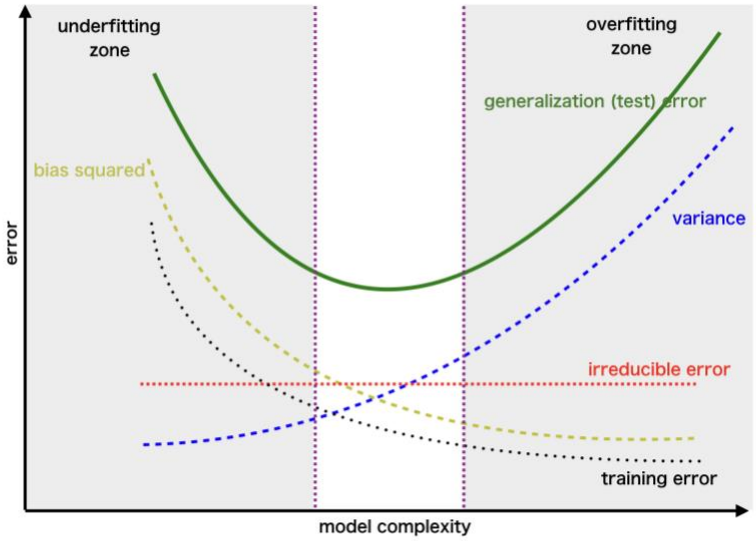
If the algorithm is too simple (hypothesis with linear eq.) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree eq.) then it may be on high variance and low bias. In the latter condition, the new entries will not perform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

This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time. For the graph, the perfect tradeoff will be like.



The best fit will be given by hypothesis on the tradeoff point.

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



This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

ESTIMATING RISK STATISTICS:

Unraveling the genetic background of human diseases serves a number of goals. One aim is to identify genes that modify the susceptibility to disease. In this context, we ask questions like: "Is this genetic variant more frequent in patients with the disease of interest than in unaffected controls?" or "Is the mean phenotype higher in carriers of this genetic variant than in non-carriers?" From the answers, we possibly learn about the pathogenesis of the disease, and we can identify possible targets for therapeutic interventions. Looking back at the past decade, it can be summarized that genome-wide association (GWA) studies have been useful in this endeavor (Hindorff et al. 2012).

When we consider classical measures for strength of association on the one hand, such as the odds ratio (OR), and for classification on the other hand, such as sensitivity (sens) and specificity (spec), there is a simple relationship between them with

$$\mathrm{OR} = \frac{\mathtt{sens}}{1-\mathtt{sens}} \cdot \frac{\mathtt{spec}}{1-\mathtt{spec}}$$

The overall process of rule construction and evaluation is shown in Fig. 1.

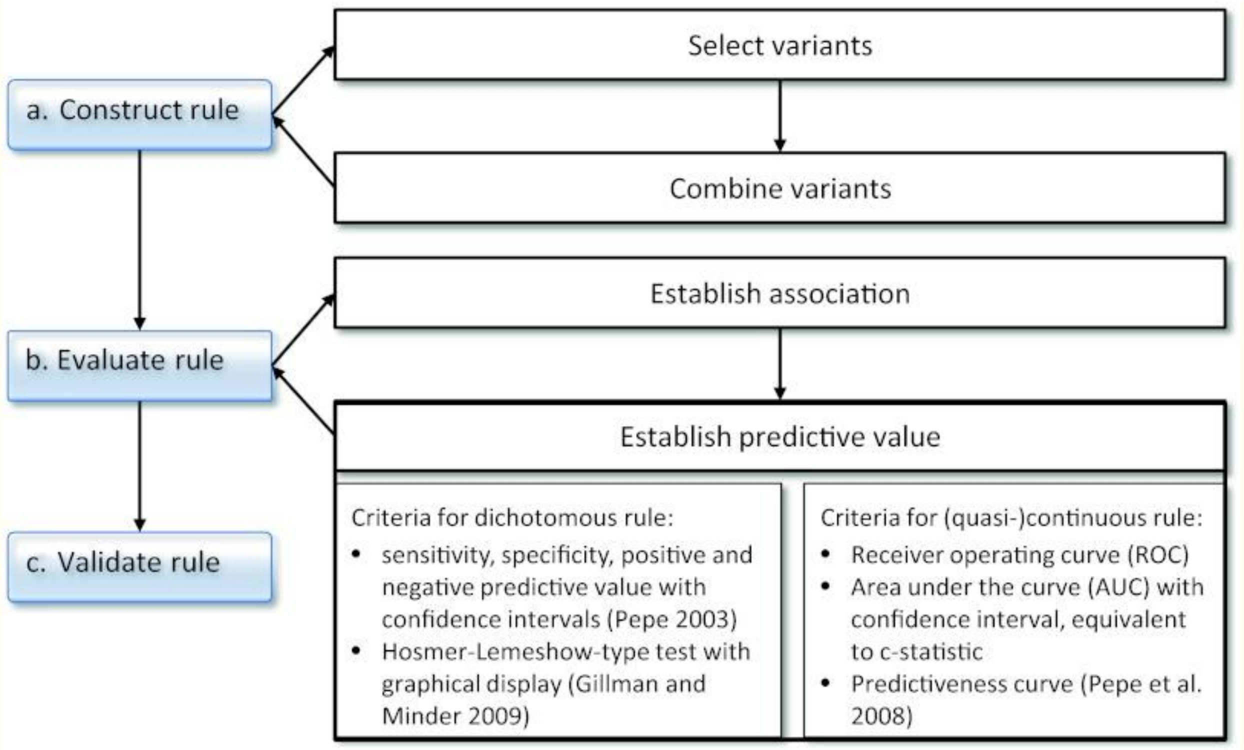

Select variants
a. Construct rule
Combine variants
Establish association
b. Evaluate rule
Establish predictive value
c. Validate rule
Criteria for dichotomous rule:
• sensitivity, specificity, positive and negative predictive value with confidence intervals (Pepe 2003)
• Hosmer-Lemeshow-type test with graphical display (Gillman and Minder 2009)
Criteria for (quasi-)continuous rule:
• Receiver operating curve (ROC)
• Area under the curve (AUC) with confidence interval, equivalent to c-statistic
• Predictiveness curve (Pepe et al. 2008)

Fig. 1
Path to construct, evaluate and validate a rule of classification or probability estimation

SAMPLING DISTRIBUTION OF AN ESTIMATOR:

One of the most important concepts discussed in the context of inferential data analysis is the idea of sampling distributions. Understanding sampling distributions helps us better comprehend and interpret results from our descriptive as well as predictive data analysis investigations. Sampling distributions are also frequently used in decision making under uncertainty and hypothesis testing.

What are sampling distributions?

You may already be familiar with the idea of probability distributions. A probability distribution gives us an understanding of the probability and likelihood associated with values (or range of values) that a random variable may assume. A random variable is a quantity whose value (outcome) is determined randomly. Some examples of a random variable include, the monthly revenue of a retail store, the number of customers arriving at a car wash location on any given day, the number of accidents on a certain highway on any given day, weekly sales volume at a retail store, etc.

Sampling distribution of the sample mean

Assuming that *X* represents the data (population), if X has a distribution with average μ and standard deviation σ, and if X is approximately normally distributed or if the sample size *n* is large,

then the sample mean $\overline{X}$ is normally distributed with average μ and standard error $\frac{\sigma}{\sqrt{n}}$.

The above distribution is only valid if, X is approximately normal **or** sample size *n* is large, and,

- the data (population) standard deviation σ is known.

If X is normal, then $\bar{X}$ is also normally distributed regardless of the sample size *n*. Central Limit Theorem tells us that even if X is not normal, if the sample size is large enough (usually greater than 30), then $\bar{X}$'s distribution is approximately normal (Sharpe, De Veaux, Velleman and Wright, 2020, pp. 318–320). If $\bar{X}$ is normal, we can easily standardize and convert it to the standard normal distribution *Z*.

If the population standard deviation σ is *not* known, we cannot assume that the sample mean $\bar{X}$ is normally distributed. If certain conditions are satisfied (explained below), then we can transform $\bar{X}$ to another random variable *t* such that,

$$t = \frac{\overline{X} - \mu}{s/\sqrt{n}}$$

The random variable *t* is said to follow the *t*-distribution with *n-1* degrees of freedom, where *n* is the sample size. The t-distribution is bell-shaped and symmetric (just like the normal distribution) but has fatter tails compared to the normal distribution. This means values further away from the mean have a higher likelihood of occurring compared to that in the normal distribution.

The conditions to use the t-distribution for the random variable *t* are as follows (Sharpe et al., 2020, pp. 415–420):

If X is normally distributed, even for small sample sizes (n<15), the t-distribution can be used.

If the sample size is between 15 and 40, the t-distribution can be used as long as X is unimodal and reasonably symmetric.

For sample sizes greater than 40, the t-distribution can be used unless X's distribution is heavily skewed

Empirical risk minimization (ERM):

Empirical risk minimization (ERM): It is a principle in statistical learning theory which defines a family of learning algorithms and is used to give theoretical bounds on their performance.

The idea is that we don't know exactly how well an algorithm will work in practice (the true "risk") because we don't know the true distribution of data that the algorithm will work on but as an alternative we can measure its performance on a known set of training data.

We assumed that our samples come from this distribution and use our dataset as an approximation. If we compute the loss using the data points in our dataset, it's called empirical risk.

It is “empirical”and not “true” because we are using a dataset that’s a subset of the whole population.

When our learning model is built, we have to pick a function that minimizes the empirical risk that is the delta between predicted output and actual output for data points in the dataset.

This process of finding this function is called empirical risk minimization (ERM). We want to minimize the true risk.
We don’t have information that allows us to achieve that, so we hope that this empirical risk will almost be the same as the true empirical risk.

In the equation below, we can define the **true error**, which is based on the whole domain **X**:

$$L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \underset{x\sim\mathcal{D}}{\mathbb{P}}[h(x) \neq f(x)] \stackrel{\text{def}}{=} \mathcal{D}(\{x : h(x) \neq f(x)\}).$$

Since we only have access to ***S***, a subset of the input domain, we learn based on that sample of training examples. We don’t have access to the **true error**, but to the **empirical error**:

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}$$

Let’s get a better understanding by Example.

We would want to build a model that can differentiate between a male and a female based on specific features.

If we select 150 random people where women are really short, and men are really tall, then the model might incorrectly assume that height is the differentiating feature.

For building a truly accurate model, we have to gather all the women and men in the world to extract differentiating features.

Unfortunately, that is not possible! So we select a small number of people and hope that this sample is representative of the whole population.

Unit-2

Supervised Learning Algorithm

Supervised learning is a type of Machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labeled dataset. Once the training and processing are done, the model is tested by providing a sample test data to check whether it predicts the correct output.

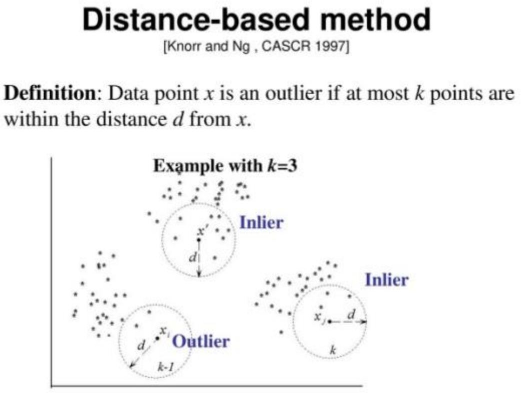
The goal of supervised learning is to map input data with the output data. Supervised learning is based on supervision, and it is the same as when a student learns things in the teacher's supervision. The example of supervised learning is **spam filtering**.

Supervised learning can be divided further into two categories of problem:

- Classification
- Regression

Distance-basedmodels

Like Linear models, distance-based models are based on the geometry of data**. As the name implies, distance-based models work on the concept of distance. In the context of Machine learning, the concept of distance is not based on merely the physical distance between two points.**



Distance-based method

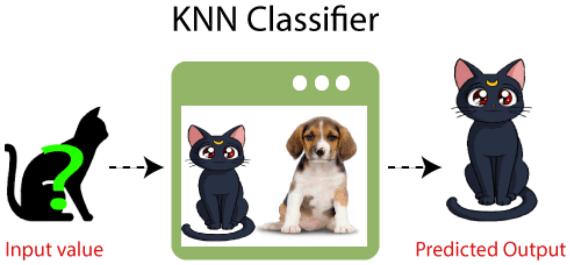
[Knorr and Ng , CASCR 1997]

Definition: Data point *x* is an outlier if at most *k* points are within the distance *d* from *x*.

K-Nearest Neighbor(KNN) Algorithm for Machine Learning

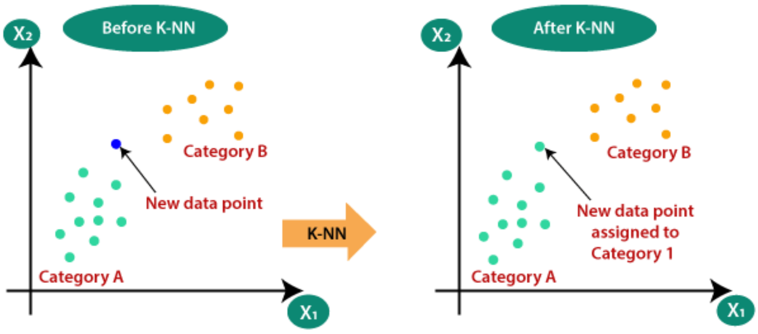
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

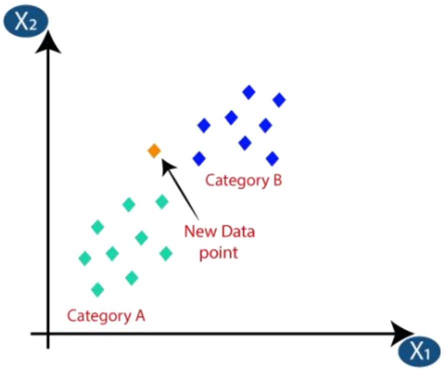


How does K-NN work?

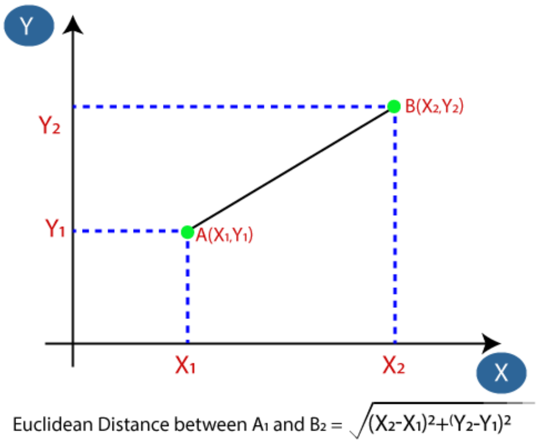
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

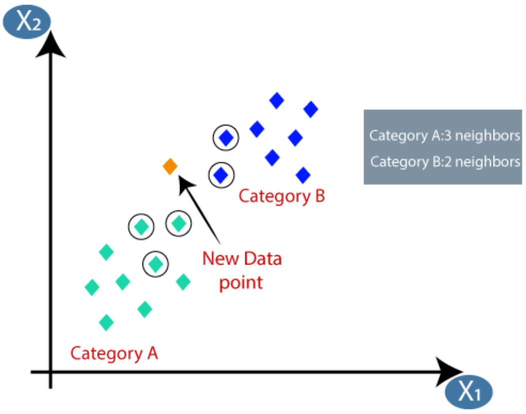
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:


Y
Y_2
$B(X_2,Y_2)$
Y_1
$A(X_1,Y_1)$
X_1
X_2
X
Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

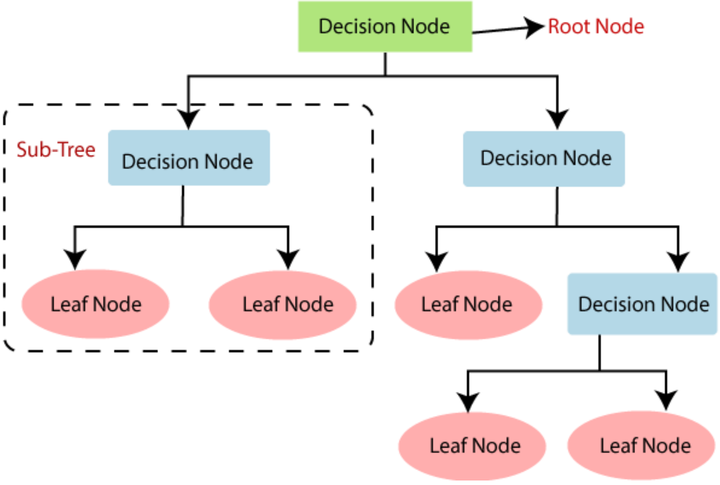
Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

- **Branch/Sub Tree:** A tree formed by splitting the tree.

- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.

- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

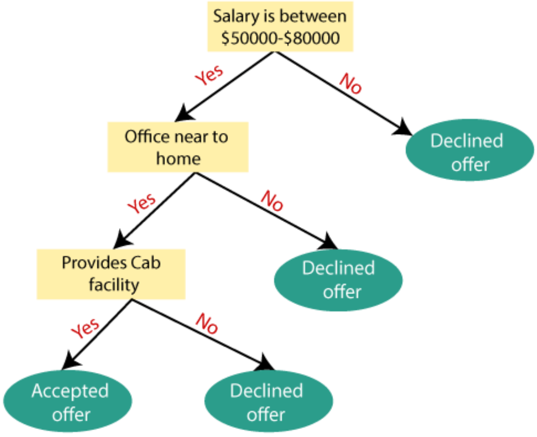
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= 1- $\sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.

- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem

.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35

All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.

- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

Linear Regression

Linear regression is one of the most popular and simple machine learning algorithms that is used for predictive analysis. Here, **predictive analysis** defines prediction of something, and linear regression makes predictions for *continuous numbers* such as **salary, age, etc.**

It shows the linear relationship between the dependent and independent variables, and shows how the dependent variable(y) changes according to the independent variable (x).

It tries to best fit a line between the dependent and independent variables, and this best fit line is knowns as the regression line.

The equation for the regression line is:

$y= a_0+ a*x+ b$

Here, y= dependent variable

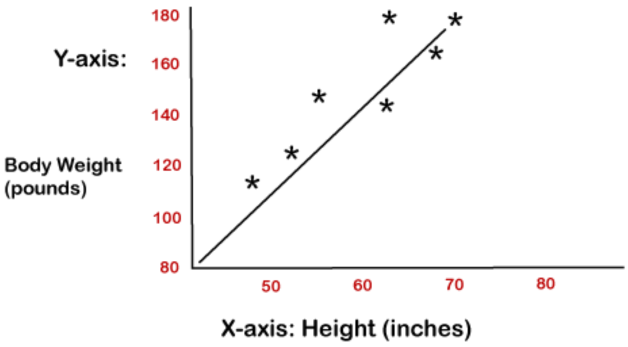
x= independent variable

a_0 = Intercept of line.

Linear regression is further divided into two types:

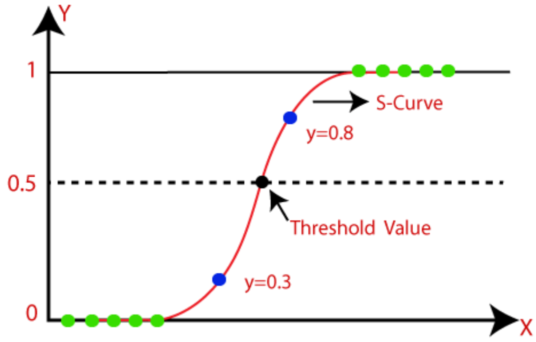
- **Simple Linear Regression:** In simple linear regression, a single independent variable is used to predict the value of the dependent variable.
- **Multiple Linear Regression:** In multiple linear regression, more than one independent variables are used to predict the value of the dependent variable.

The below diagram shows the linear regression for prediction of weight according to height: Read more..



Logistic Regression in Machine Learning

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: *Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.*

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.

- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

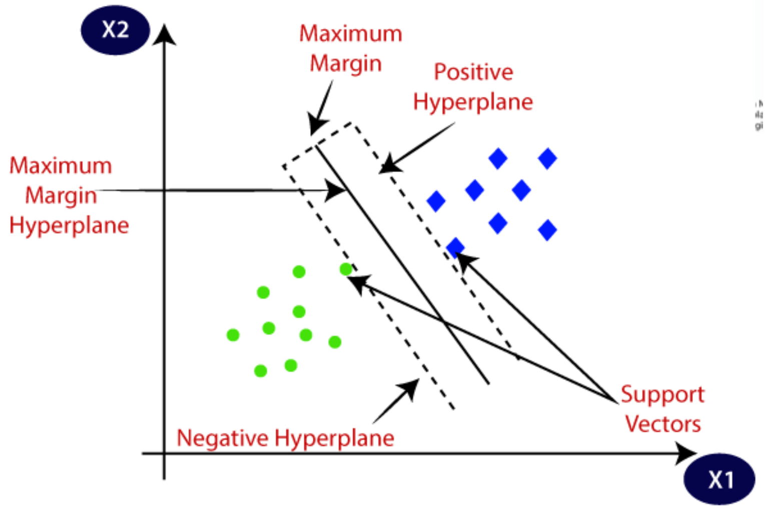
- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Support Vector Machine Algorithm

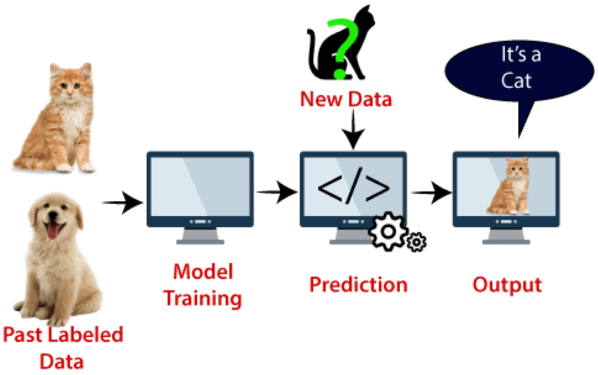
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

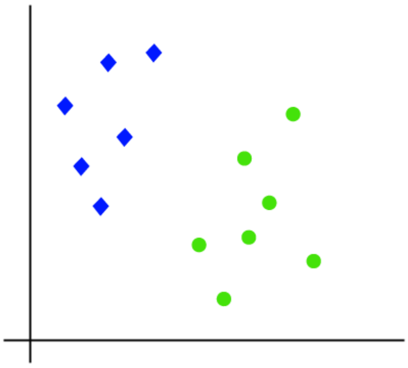
Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

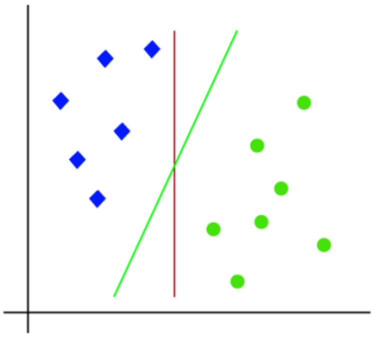
How does SVM works?

Linear SVM:

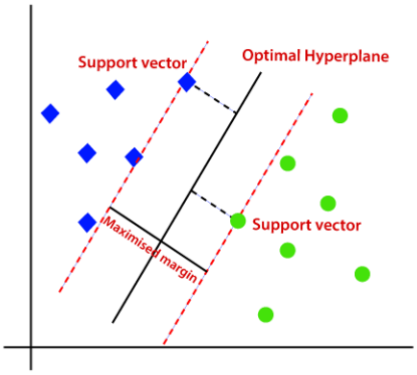
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

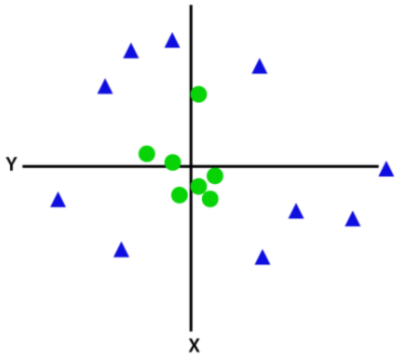


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

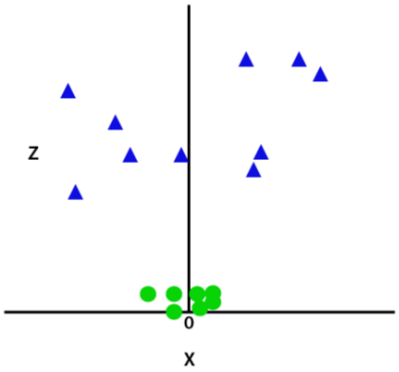
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



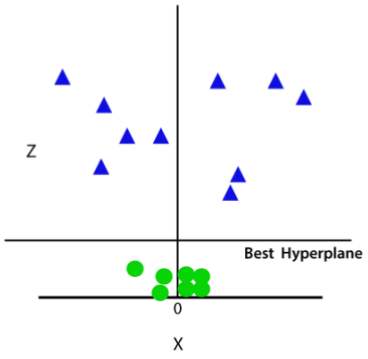
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

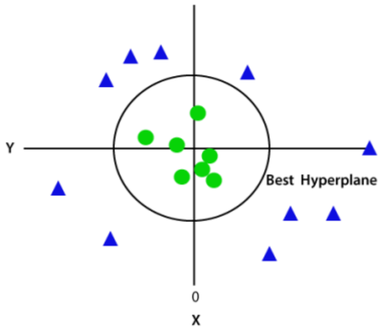
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data.

Generalized Linear Models

Prerequisite:

- Linear Regression
- Logistic Regression

The following article discusses the **Generalized linear models** (GLMs) which explains how Linear regression and Logistic regression are a member of a much broader class of models. GLMs can be used to construct the models for regression and classification problems by using the type of distribution which best describes the data or labels given for training the model. Below given are some types of datasets and the corresponding distributions which would help us in constructing the model for a particular type of data (The term data specified here refers to the **output data** or the **labels** of the dataset).

1. Binary classification data – Bernoulli distribution
2. Real valued data – Gaussian distribution
3. Count-data – Poisson distribution

To understand GLMs we will begin by defining exponential families. Exponential families are a class of distributions whose probability density function(PDF) can be molded into the following form:

Proof – **Bernoulli distribution** is a member of the exponential family.

Therefore, on comparing **Eq1** and **Eq2** :

Note: *As mentioned above the value of* ***phi*** *(which is the same as the activation or* ***sigmoid function*** *for Logistic regression) is not a coincidence. And it will be proved later in the article how Logistic regression model can be derived from the Bernoulli distribution.*

proof – **Gaussian distribution** is a member of the exponential family.

Therefore, on comparing **Eq1** and **Eq3**:

Constructing GLMs:

To construct GLMs for a particular type of data or more generally for linear or logistic classification problems the following three assumptions or design choices are to be considered:

The first assumption is that if **x** is the input data parameterized by theta the resulting output or y will be a member of the exponential family. This means if we are provided with some labeled data our goal is to find the right parameters theta which fits the given model as closely as possible. The third assumption is the least justified and can be considered as a design choice.

Linear Regression Model:

To show that Linear Regression is a special case of the GLMs. It is considered that the output labels are **continuous values** and are therefore a **Gaussian distribution**. So, we have

The first equation above corresponds to the first assumption that the output labels (or **target**

variables) should be the member of an exponential family, Second equation corresponds to the assumption that the **hypothesis** is equal the **expected value** or **mean** of the distribution and lastly, the third equation corresponds to the assumption that natural parameter and the input parameters follow a linear relationship.

Logistic Regression Model:

To show that Logistic Regression is a special case of the GLMs. It is considered that the output labels are **Binary valued** and are therefore a **Bernoulli distribution**. So, we have From the third assumption, it is proven that:

The function that maps the natural parameter to the canonical parameter is known as the **canonical response function** (here, the log-partition function) and the inverse of it is known as the **canonical link function**.

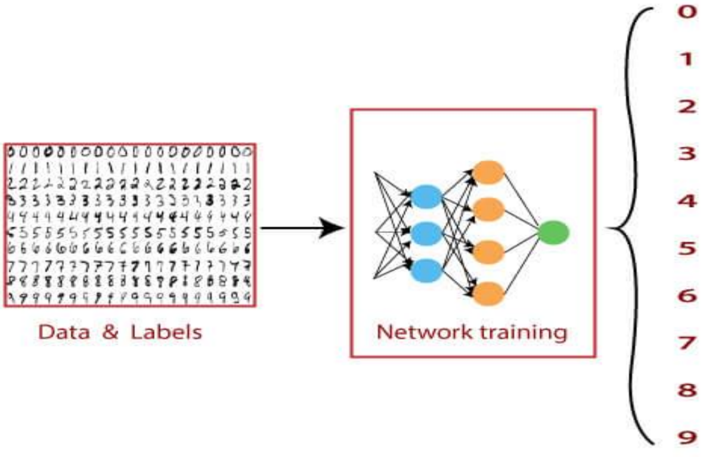
Therefore by using the three assumptions mentioned before it can be proved that the **Logistic** and **Linear Regression** belongs to a much larger family of models known as GLMs.

MNIST Dataset

The MNIST (**Modified National Institute of Standards and Technology**) database is a large database of handwritten numbers or digits that are used for training various image processing systems. The dataset also widely used for training and testing in the field of **machine learning**. The set of images in the MNIST database are a combination of two of NIST's databases: Special Database 1 and Special Database 3.

The MNIST dataset has **60,000** training images and **10,000** testing images.

The **MNIST** dataset can be online, and it is essentially a database of various handwritten digits. The MNIST dataset has a large amount of data and is commonly used to demonstrate the real power of deep neural networks. Our brain and eyes work together to recognize any numbered image. Our mind is a potent tool, and it's capable of categorizing any image quickly. There are so many shapes of a number, and our mind can easily recognize these shapes and determine what number is it, but the same task is not simple for a computer to complete. There is only one way to do this, which is the use of deep neural network which allows us to train a computer to classify the handwritten digits effectively.



So, we have only dealt with data which contains simple data points on a **Cartesian coordinate system**. From starting till now, we have distributed with binary class datasets. And when we use multiclass datasets, we will use the **Softmax activation** function is quite useful for classifying binary datasets. And it was quite effective in arranging values between 0 and 1. The sigmoid function is not effective for multicausal datasets, and for this purpose, we use the softmax activation function, which is capable of dealing with it.

The MNIST dataset is a multilevel dataset consisting of 10 classes in which we can classify numbers from 0 to 9. The major difference between the datasets that we have used before and the MNIST dataset is the method in which MNIST data is inputted in a neural network.

In the perceptual model and linear regression model, each of the data points was defined by a simple x and y coordinate. This means that the input layer needs two nodes to input single data points.

In the MNIST dataset, a single data point comes in the form of an image. These images included in the **MNIST** dataset are typical of **28*28** pixels such as 28 pixels crossing the horizontal axis and 28 pixels crossing the vertical axis. This means that a single image from the MNIST database has a total of 784 pixels that must be analyzed. The input layer of our neural network has 784 nodes to explain one of these images.

MNIST Dataset
Softmax Regression
Implementation
Training
Checking Accuracy
TensorFlow MNIST

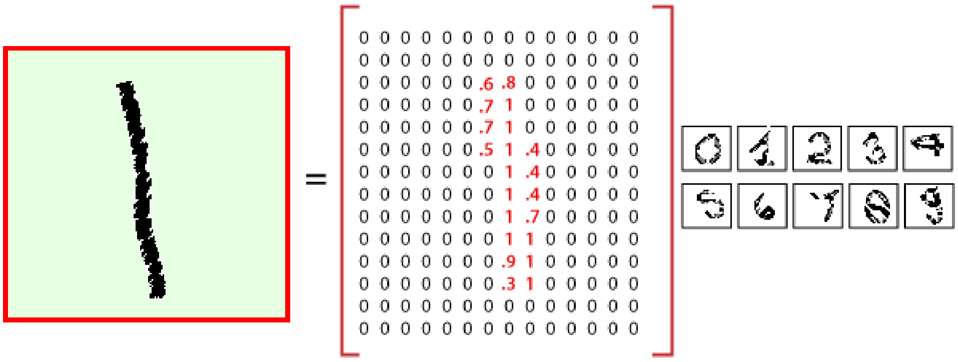
Here, we will see how to create a function that is a model for recognizing handwritten digits by looking at each pixel in the image. Then using TensorFlow to train the model to predict the image by making it look at thousands of examples which are already labeled. We will then check the model's accuracy with a test dataset.

MNIST dataset in TensorFlow, containing information of handwritten digits spitted into three parts:

- **Training Data (mnist.train) -55000 datapoints**
- **Validation Data (mnist.validate) -5000 datapoint**
- **Test Data (mnist.test) -10000 datapoints**

Now before we start, it is important to note that every data point has two parts: an image (x) and a corresponding label (y) describing the actual image and each image is a 28x28 array, i.e., 784 numbers. The label of the image is a number between 0 and 9 corresponding to the TensorFlow MNIST image. To download and use MNIST dataset, use the following **commands**:

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```



$$
= \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}
$$

Ranking

Ranking is a regression machine learning technique.

- About Ranking
- Ranking Methods
- Ranking Algorithms
  - XGBoost

About Ranking

Ranking is a machine learning technique to rank items.

Ranking is useful for many applications in information retrieval such as e-commerce, social networks, recommendation systems, and so on. For example, a user searches for an article or an item to buy online. To build a recommendation system, it becomes important that similar articles or items of relevance appear to the user such that the user clicks or purchases the item. A simple regression model can predict the probability of a user to click an article or buy an item. However, it is more practical to use ranking technique and be able to order or rank the articles or items to maximize the chances of getting a click or purchase. The prioritization of the articles or the items influence the decision of the users.

The ranking technique directly ranks items by training a model to predict the ranking of one item over another item. In the training model, it is possible to have items, ranking one over the other by having a "score" for each item. Higher ranked items have higher scores and lower ranked items have lower scores. Using these scores, a model is built to predict which item ranks higher than the other.

Ranking Methods

Oracle Machine Learning supports pairwise and listwise ranking methods through XGBoost.

For a training data set, in a number of sets, each set consists of objects and labels representing their ranking. A ranking function is constructed by minimizing a certain loss function on the training data. Using test data, the ranking function is applied to get a ranked list of objects. Ranking is enabled for XGBoost using the regression function. OML4SQL supports pairwise and listwise ranking methods through XGBoost.

Pairwise ranking: This approach regards a pair of objects as the learning instance. The pairs and lists are defined by supplying the same case_id value. Given a pair of objects, this approach gives an optimal ordering for that pair. Pairwise losses are defined by the order of the two objects. In OML4SQL, the algorithm uses LambdaMART to perform pairwise ranking with the goal of minimizing the average number of inversions in ranking.

Listwise ranking: This approach takes multiple lists of ranked objects as learning instance. The items in a list must have the same case_id. The algorithm uses LambdaMART to perform list-wise ranking.

See Also:

- "Ranking Measures and Loss Functions in Learning to Rank" a research paper presentation at https://www.researchgate.net/
- *Oracle Database PL/SQL Packages and Types Reference* for a listing and explanation of the available model settings for XGBoost.

Note:

The term hyperparameter is also interchangeably used for model setting.

Related Topics

- XGBoost
- DBMS_DATA_MINING — Algorithm Settings: XGBoost

Ranking Algorithms

Ranking falls under the Regression function.

OML4SQL supports XGBoost algorithm for ranking.

Related Topics

- XGBoost

Structured outputs

Structured prediction or **structured (output) learning** is an umbrella term for supervised machine learning techniques that involves predicting structured objects, rather than scalar discrete or real values.[1]

Similar to commonly used supervised learning techniques, structured prediction models are typically trained by means of observed data in which the true prediction value is used to adjust model parameters. Due to the complexity of the model and the interrelations of predicted variables the process of prediction using a trained model and of training itself is often computationally infeasible and approximate inference and learning methods are used.

Applications

For example, the problem of translating a natural language sentence into a syntactic representation such as a parse tree can be seen as a structured prediction problem[2] in which the structured output domain is the set of all possible parse trees. Structured prediction is also used in a wide variety of application domains including bioinformatics, natural language processing, speech recognition, and computer vision.

Example: sequence tagging

Sequence tagging is a class of problems prevalent in natural language processing, where input data are often sequences (e.g. sentences of text). The sequence tagging problem appears in several guises, e.g. part-of-speech tagging and named entity recognition. In POS tagging, for example, each word in a sequence must receive a "tag" (class label) that expresses its "type" of word:

This	DT
is	VBZ
a	DT
tagged	JJ
sentence	NN

The main challenge of this problem is to resolve ambiguity: the word "sentence" can also be a verb in English, and so can "tagged".

While this problem can be solved by simply performing classification of individual tokens, that approach does not take into account the empirical fact that tags do not occur independently; instead, each tag displays a strong conditional dependence on the tag of the previous word. This fact can be exploited in a sequence model such as a hidden Markov model or conditional random field[2] that predicts the entire tag sequence for a sentence, rather than just individual tags, by means of the Viterbi algorithm.

Techniques

Probabilistic graphical models form a large class of structured prediction models. In particular, Bayesian networks and random fields are popular. Other algorithms and models for structured prediction include inductive logic programming, case-based reasoning, structured SVMs, Markov logic networks, Probabilistic Soft Logic, and constrained conditional models. Main techniques:

- Conditional random field
- Structured support vector machine
- Structured k-Nearest Neighbours
- Recurrent neural network, in particular Elman network

Structured perceptron

One of the easiest ways to understand algorithms for general structured prediction is the structured perceptron of Collins.[3] This algorithm combines the perceptron algorithm for learning linear classifiers with an inference algorithm (classically the Viterbi algorithm when used on sequence data) and can be described abstractly as follows. First define a "joint feature function" $\Phi(\mathbf{x}, \mathbf{y})$ that maps a training sample **x** and a candidate prediction **y** to a vector of length *n* (**x** and **y** may have any structure; *n* is problem-dependent, but must be fixed for each model). Let GEN be a function that generates candidate predictions.

Unit -3

Ensemble Learning and Random Forests: Introduction, Voting Classifiers, Bagging and Pasting, Random Forests, Boosting, Stacking.

Support Vector Machine: Linear SVM Classification, Nonlinear SVM Classification SVM Regression, Naïve Bayes Classifiers.

VOTING CLASSIFIERS

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.
It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

Voting Classifier supports two types of votings.

1. **Hard Voting:** In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the output class(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction
.
2. **Soft Voting**: In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40). So the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier.

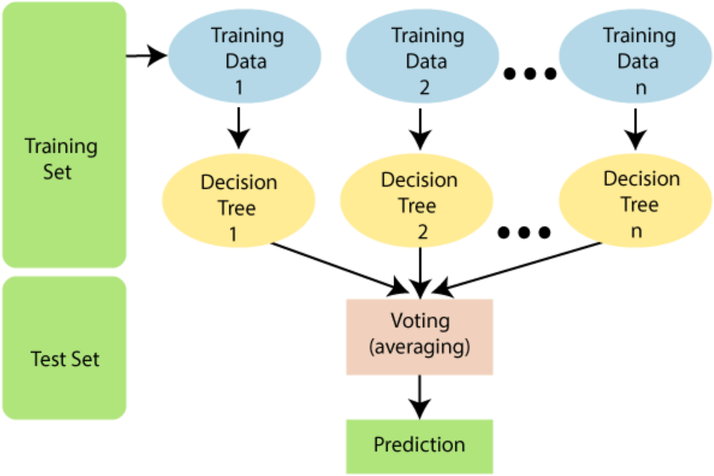
Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the***

predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The below diagram explains the working of the Random Forest algorithm:



USES OF RANDOM FOREST :

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

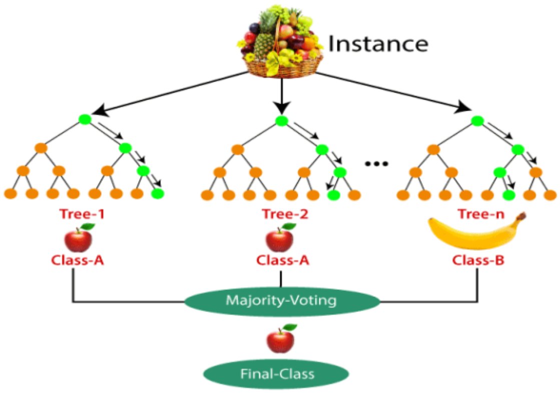
Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision.

Consider the below image:



Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

Stacking in Machine Learning

There are many ways to ensemble models in machine learning, such as Bagging, Boosting, and stacking. ***Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance.*** Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance.

Ensemble learning is one of the most powerful machine learning techniques that use the combined output of two or more models/weak learners and solve a particular computational intelligence problem. E.g., a Random Forest algorithm is an ensemble of various decision trees combined.

"An ensembled model is a machine learning model that combines the predictions from two or more models."

There are 3 most common ensemble learning methods in machine learning. These are as follows:

- Bagging
- Boosting
- Stacking

However, we will mainly discuss Stacking on this topic.

1. Bagging

Bagging is a method of ensemble modeling, which is primarily used to solve supervised machine learning problems. It is generally completed in two steps as follows:

- **Bootstrapping:** It is a random sampling method that is used to derive samples from the data using the replacement procedure. In this method, first, random data samples are fed to the primary model, and then a base learning algorithm is run on the samples to complete the learning process.
- **Aggregation:** This is a step that involves the process of combining the output of all base models and, based on their output, predicting an aggregate result with greater accuracy and reduced variance.

Example: In the Random Forest method, predictions from multiple decision trees are ensembled parallelly. Further, in regression problems, we use an average of these predictions to get the final output, whereas, in classification problems, the model is selected as the predicted class.

2. Boosting

Boosting is an ensemble method that enables each member to learn from the preceding member's mistakes and make better predictions for the future. Unlike the bagging method, in boosting, all base learners (weak) are arranged in a sequential format so that they can learn from the mistakes of their preceding learner. Hence, in this way, all weak learners get turned into strong learners and make a better predictive model with significantly improved performance.

We have a basic understanding of ensemble techniques in machine learning and their two common methods, i.e., bagging and boosting. Now, let's discuss a different paradigm of ensemble learning, i.e., Stacking.

DIFFERENCE

Bagging	Boosting
Various training data subsets are randomly drawn with replacement from the whole training dataset.	Each new subset contains the components that were misclassified by previous models.
Bagging attempts to tackle the over-fitting issue.	Boosting tries to reduce bias.
If the classifier is unstable (high variance), then we need to apply bagging.	If the classifier is steady and straightforward (high bias), then we need to apply boosting.
Every model receives an equal weight.	Models are weighted by their performance.
Objective to decrease variance, not bias.	Objective to decrease bias, not variance.
It is the easiest way of connecting predictions that belong to the same type.	It is a way of connecting predictions that belong to the different types.
Every model is constructed independently.	New models are affected by the performance of the previously developed model.

3. Stacking

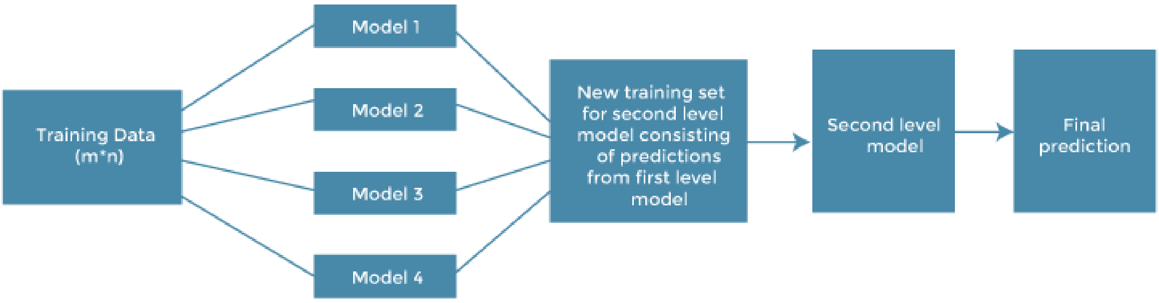
Stacking is one of the popular ensemble modeling techniques in machine learning. Various weak learners are ensembled in a parallel manner in such a way that by combining them with Meta learners, we can predict better predictions for the future.

In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction.

Stacking is also known as **a stacked generalization** and is an extended form of the Model Averaging Ensemble technique in which all sub-models equally participate as per their performance weights and build a new model with better predictions. This new model is stacked up on top of the others; this is the reason why it is named stacking.

Architecture of Stacking

The architecture of the stacking model is designed in such as way that it consists of two or more base/learner's models and a meta-model that combines the predictions of the base models. These base models are called level 0 models, and the meta-model is known as the level 1 model. So, the Stacking ensemble method includes **original (training) data, primary level models, primary level prediction, secondary level model, and final prediction**. The basic architecture of stacking can be represented as shown below the image.


Training Data (m*n)
Model 1
Model 2
Model 3
Model 4
New training set for second level model consisting of predictions from first level model
Second level model
Final prediction

- **Original data:** This data is divided into n-folds and is also considered test data or training data.
- **Base models:** These models are also referred to as level-0 models. These models use training data and provide compiled predictions (level-0) as an output.
- **Level-0 Predictions:** Each base model is triggered on some training data and provides different predictions, which are known as **level-0 predictions.**
- **Meta Model:** The architecture of the stacking model consists of one meta-model, which helps to best combine the predictions of the base models. The meta-model is also known as the **level-1 model**.
- **Level-1 Prediction:** The meta-model learns how to best combine the predictions of the base models and is trained on different predictions made by individual base models, i.e., data not used to train the base models are fed to the meta-model, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

Steps to implement Stacking models:

There are some important steps to implementing stacking models in machine learning.

These are as follows:

- Split training data sets into n-folds using the **RepeatedStratifiedKFold** as this is the most common approach to preparing training datasets for meta-models.
- Now the base model is fitted with the first fold, which is n-1, and it will make predictions for the nth folds.
- The prediction made in the above step is added to the x1_train list.
- Repeat steps 2 & 3 for remaining n-1folds, so it will give x1_train array of size n,
- Now, the model is trained on all the n parts, which will make predictions for the sample data.
- Add this prediction to the y1_test list.
- In the same way, we can find x2_train, y2_test, x3_train, and y3_test by using Model 2 and 3 for training, respectively, to get Level 2 predictions.
- Now train the Meta model on level 1 prediction, where these predictions will be used as features for the model.
- Finally, Meta learners can now be used to make a prediction on test data in the stacking model.

pasting

Bagging is to use the same training for every predictor, but to train them on different random subsets of the training set. When sampling is performed with replacement, this method is called bagging (short for bootstrap aggregating). When sampling is performed without replacement, it is called pasting.

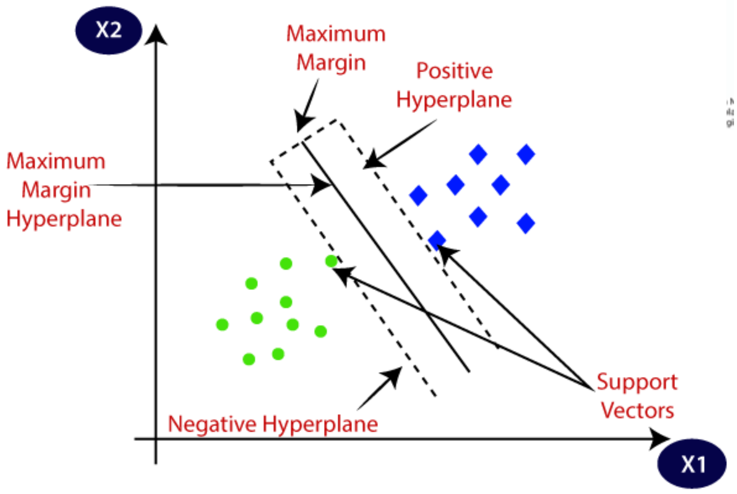
\ **Support Vector Machine:** Linear SVM Classification, Nonlinear SVM Classification SVM Regression, Naïve Bayes Classifiers.

Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Types of SVM

SVM can be of two types:

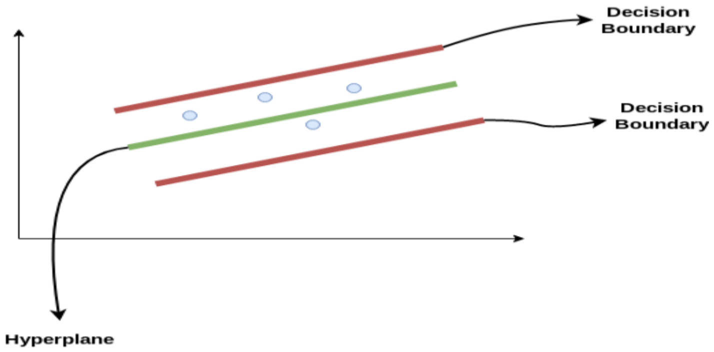
- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Introduction to Support Vector Regression (SVR)

Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems. Let's spend a few minutes understanding the idea behind SVR.

The Idea Behind Support Vector Regression

The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample. So let's now dive deep and understand how SVR works actually.



hyperplane. **Our objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line.** Our best fit line is the hyperplane that has a maximum number of points.

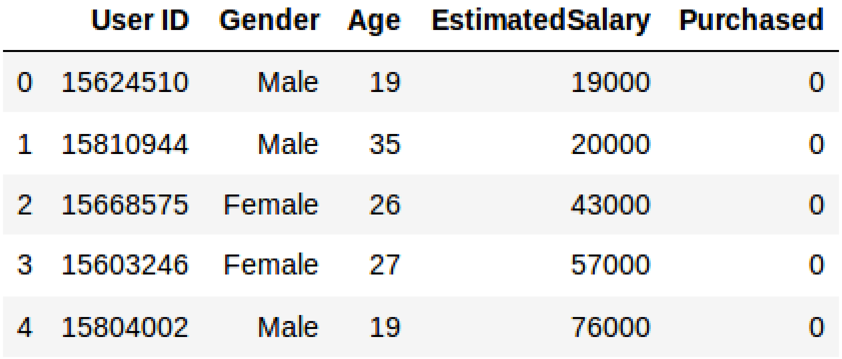
The first thing that we’ll understand is what is the decision boundary (the danger red line above!). Consider these lines as being at any distance, say ‘a’, from the hyperplane. So, these are the lines that we draw at distance ‘+a’ and ‘-a’ from the hyperplane. This ‘a’ in the text is basically referred to as epsilon.

.Our main aim here is to decide a decision boundary at ‘a’ distance from the original hyperplane such that data points closest to the hyperplane or the support vectors are within that boundary line.

Hence, we are going to take only those points that are within the decision boundary and have the least error rate, or are within the Margin of Tolerance. This gives us a better fitting model.

Implementing Support Vector Regression (SVR) in Python

Time to put on our coding hats! In this section, we’ll understand the use of Support Vector Regression with the help of a dataset. Here, we have to predict the salary of an employee given a few independent variables. A classic HR analytics project!



	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Step 1: Importing the libraries

```
import numpy as np

import matplotlib.pyplot as plt
import pandas as pd
```

view rawsvr1.py hosted with by GitHub

Step 2: Reading the dataset

```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

view rawsv2.py hosted with by GitHub

Step 3: Feature Scaling

A real-world dataset contains features that vary in magnitudes, units, and range. I would suggest performing normalization when the scale of a feature is irrelevant or misleading.

Feature Scaling basically helps to normalize the data within a particular range. Normally several common class types contain the feature scaling function so that they make feature scaling automatically. However, the SVR class is not a commonly used class type so we should perform feature scaling using Python.

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

view rawsvr3.py hosted with by GitHub

Step 4: Fitting SVR to the dataset

```
from sklearn.svm import SVR

regressor = SVR(kernel = 'rbf')

regressor.fit(X, y)
```

view rawsvr4.py hosted with by **GitHub**

Kernel is the most important feature. There are many types of kernels – linear, Gaussian, etc. Each is used depending on the dataset. To learn more about this, read this: Support Vector Machine (SVM) in Python and R

Step 5. Predicting a new result

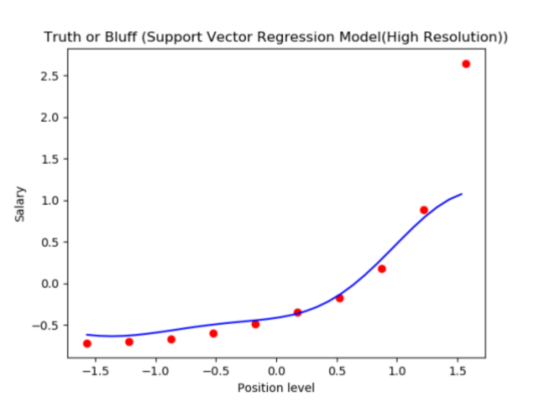
```
y_pred = regressor.predict(6.5)

y_pred = sc_y.inverse_transform(y_pred)
```

view rawsvr5.py hosted with by **GitHub**

So, the prediction for y_pred(6, 5) will be 170,370.

Step 6. Visualizing the SVR results (for higher resolution and smoother curve)

```
X_grid = np.arange(min(X), max(X), 0.01) #this step required because data is feature scaled.

X_grid = X_grid.reshape((len(X_grid), 1))

plt.scatter(X, y, color = 'red')

plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')

plt.title('Truth or Bluff (SVR)')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()
```



This is what we get as output- the best fit line that has a maximum number of points. Quite accurate!

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem

.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

Python Implementation of the Naïve Bayes algorithm:

Now we will implement a Naive Bayes Algorithm using Python. So for this, we will use the "**user_data**" **dataset**, which we have used in our other classification model. Therefore we can easily compare the Naive Bayes model with the other models.

Steps to implement:

- Data Pre-processing step
- Fitting Naive Bayes to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result

1) Data Pre-processing step:

In this step, we will pre-process/prepare the data so that we can use it efficiently in our code. It is similar as we did in data-pre-processing

2) Fitting Naive Bayes to the Training Set:

After the pre-processing step, now we will fit the Naive Bayes model to the Training set. Below is the code for it:

3) Prediction of the test set result:

Now we will predict the test set result. For this, we will create a new predictor variable **y_pred**, and will use the predict function to make the predictions

4) Creating Confusion Matrix:

Now we will check the accuracy of the Naive Bayes classifier using the Confusion matrix. Below is the code for it:

5) Visualizing the training set result:

Next we will visualize the training set result using Naïve Bayes Classifier. Below is the code for it:

Unit-4

Unsupervised learning techniques:

Clustering :

Clustering or cluster analysis is a machine learning technique, which groups the unlabelleddataset. It can be defined as **"A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."**

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method, hence no supervision is provided to the algorithm,and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.

The clustering technique is commonly used for **statistical data analysis.**

Note: Clustering is somewhere similar to the classification algorithm, but the difference is the type of dataset that we are using. In classification, we work with the labeled data set, whereas in clustering, we work with the unlabelled dataset.

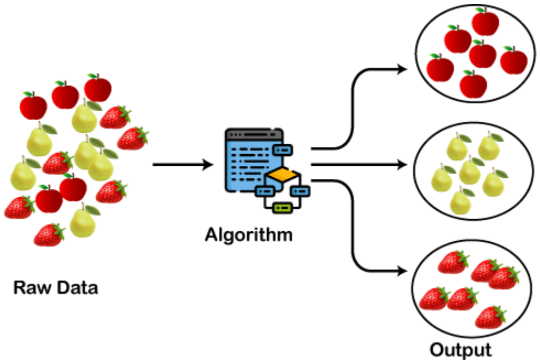
Example: Let's understand the clustering technique with the real-world example of Mall:When we visit any shopping mall, we can observe that the things with similar usage are grouped together. Such as the t-shirts are grouped in one section, and trousers are at other sections, similarly, at vegetable sections, apples, bananas, Mangoes, etc., aregrouped in separate sections, so that we can easily find out the things. The clustering technique also works in the same way. Other examples of clustering are grouping documents according to the topic.

The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

- Market Segmentation
- Statistical data analysis
- Social network analysis
- Image segmentation
- Anomaly detection, etc.

Apart from these general usages, it is used by the **Amazon** in its recommendation systemto provide the recommendations as per the past search of products. **Netflix** also uses this technique to recommend the movies and web-series to its users as per the watch history.

The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.



Types of Clustering Methods

The clustering methods are broadly divided into **Hard clustering** (datapoint belongs to only one group) and **Soft Clustering** (data points can belong to another group also). Butthere are also other various approaches of Clustering exist. Below are the main clusteringmethods used in Machine learning:

1. **Partitioning Clustering**
2. **Density-Based Clustering**
3. **Distribution Model-Based Clustering**
4. **Hierarchical Clustering**
5. **Fuzzy Clustering**

Applications of Clustering

Below are some commonly known applications of clustering technique in MachineLearning:

- **In Identification of Cancer Cells:** The clustering algorithms are widely used for the identification of cancerous cells. It divides the cancerous and non-cancerous data sets into different groups.
- **In Search Engines:** Search engines also work on the clustering technique. The search result appears based on the closest object to the search query. It does it by grouping similar data objects in one group that is far from the other dissimilar objects. The accurateresult of a query depends on the quality of the clustering algorithm used.
- **Customer Segmentation:** It is used in market research to segment the customers based ontheir choice and preferences

- **In Biology:** It is used in the biology stream to classify different species of plants and animals using the image recognition technique.
- **In Land Use:** The clustering technique is used in identifying the area of similar lands use in the GIS database. This can be very useful to find that for what purpose the particular land should be used, that means for which purpose it is more suitable.

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that

need to be created in the process, as if K=2, there will be two clusters, and for K=3, therewill be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover thecategories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The mainaim of this algorithm is to minimize the sum of distances between the data point and theircorresponding clusters.

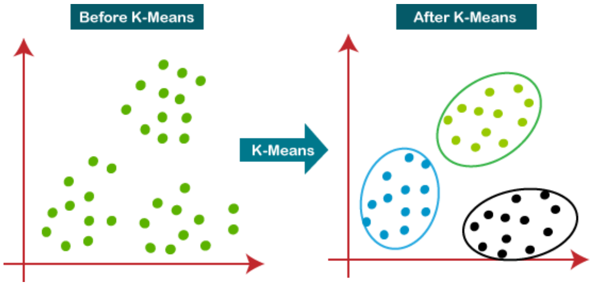
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number ofclusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from otherclusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined Kclusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closestcentroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

limits of k means:

- It executes the K-means clustering on a given dataset for different K values (rangesfrom 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K

Image Segmentation

- A digital image is made up of various components that need to be “analysed”, let’s use that word for simplicity sake and the “analysis”performed on such components can reveal a lot of hidden information from them. This information can help us address a plethora of business problems – which is one of the many end goalsthat are linked with image processing.

- Image Segmentation is the process by which a digital image is partitionedinto various subgroups (of pixels) called Image Objects, which can reducethe complexity of the image, and thus analysing the image becomes simpler.

- We use various image segmentation algorithms to split and group acertain set of pixels together from the image. By doing so, we are actually assigning labels to pixels and the pixels with the same labelfall under a category where they have some or the other thing common in them.

- Using these labels, we can specify boundaries, draw lines, and separate the most required objects in an image from the rest of the not-so-important ones. In the below example, from a main image on the left, we try to get the major components, e.g. chair, table etc. and hence all the chairs are colored uniformly. In the next tab, we have detected instances, which talk about individual objects, and hence the all the chairs have different colors.

- This is how different methods of segmentation of images work in varying degrees of complexity and yield different levels of outputs.

Data Preprocessing in Machine learning

- Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.
- When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Why

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing isrequired tasks for cleaning the data and making it suitable for a machine learning modelwhich also increases the accuracy and efficiency of a machine learning model.

Semi-Supervised Learning:

Semi-Supervised learning is a type of Machine Learning algorithm that represents the intermediate ground between Supervis and Unsupervised learning algorithms. It uses the combination of labeled and unlabeleddatasets during the training period.

Assumptions followed by Semi-Supervised Learning

To work with the unlabeled dataset, there must be a relationship between the objects. Tounderstand this, semi-supervised learning uses any of the following assumptions:

- **Continuity**
- As per the continuity assumption, the objects near each other tend to share the same group or label. This assumption is also used in supervised learning, and the datasets are separated by the decision boundaries. But in semi-supervised, the decision boundaries areadded with the smoothness assumption in low-density boundaries.
- **Cluster assumptions-** In this assumption, data are divided into different discrete clusters. Further, the points in the same cluster share the output label.

- **Manifold assumptions-** This assumption helps to use distances and densities, and this data lie on a manifold of fewer dimensions than input space.
- The dimensional data are created by a process that has less degree of freedom and may be hard to model directly. **(This assumption becomes practical if high).**

Working of Semi-Supervised Learning

Semi-supervised learning uses pseudo labeling to train the model with less labeled training data than supervised learning. The process can combine various neural network models and training ways. The whole working of semi-supervised learning is explained inthe below points:

- Firstly, it trains the model with less amount of training data similar to the supervised learning models. The training continues until the model gives accurate results.
- The algorithms use the unlabeled dataset with pseudo labels in the next step, and now the result may not be accurate.
- Now, the labels from labeled training data and pseudo labels data are linked together.
- The input data in labeled training data and unlabeled training data are also linked.
- In the end, again train the model with the new combined input as did in the first step. It will reduce errors and improve the accuracy of the model.

DBSCAN

Density-Based Clustering refers to one of the most popular unsupervised learning methodologies used in model building and machine learning algorithms. The data points in the region separatedby two clusters of low point density are considered as noise. The surroundings with a radius ε of a given object are known as the ε neighborhood of the object. If the ε neighborhood of the object comprises at least a minimum number, MinPts of objects, then it is called a core object.

There are two different parameters to calculate the density-based clusteringEPS: It

is considered as the maximum radius of the neighborhood.

MinPts: MinPts refers to the minimum number of points in an Eps neighborhood of thatpoint.

NEps (i) : { k belongs to D and dist (i,k) < = Eps}

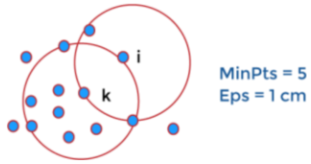
Directly density reachable:

A point i is considered as the directly density reachable from a point k with respect to Eps, MinPts if

i belongs to NEps(k)

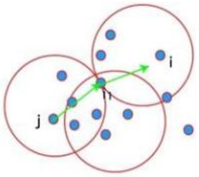
Core point condition:

NEps (k) >= MinPts



Density reachable:

A point denoted by i is a density reachable from a point j with respect to Eps, MinPts if there is a sequence chain of a point i1,…., in, i1 = j, pn = i such that i_i + 1 is directly densityreachable from i_i.



Density connected:

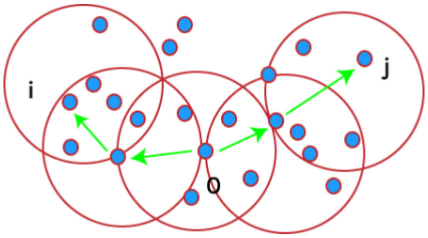
A point i refers to density connected to a point j with respect to Eps, MinPts if there is a point o such that both i and j are considered as density reachable from o with respect toEps and MinPts.

Gaussian Discriminant Analysis

There are two types of Supervised Learning algorithms are used in Machine Learning for classification.

1. Discriminative Learning Algorithms
2. Generative Learning Algorithms

Logistic Regression, Perceptron, and other Discriminative Learning Algorithms are examples of discriminative learning algorithms. These algorithms attempt to determine aboundary between classes in the learning process. A Discriminative Learning Algorithm might be used to solve a classification problem that will determine if a patient has malaria.The boundary is then checked to see if the new example falls on the boundary, **P(y|X)**, i.e., Given a feature set X, what is its probability of belonging to the class "y".



Generative Learning Algorithms, on the other hand, take a different approach. They try tocapture each class distribution separately rather than finding a boundary between classes.A Generative Learning Algorithm, as mentioned, will examine the distribution of infectedand healthy patients separately. It will then attempt to learn each distribution's features individually. When a new example is presented, it will be compared to both distributions,and the class that it most closely resembles will be assigned, **P(X|y)** for a given **P(y)** here,P(y) is known as a class prior.

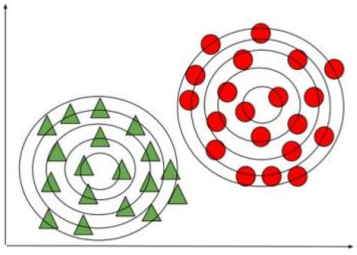
These Bayes Theory predictions are used to predict generative learning algorithms

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

$$where, P(X) = P(X|y=1).P(y=1) + P(X|y=0).P(y=0)$$

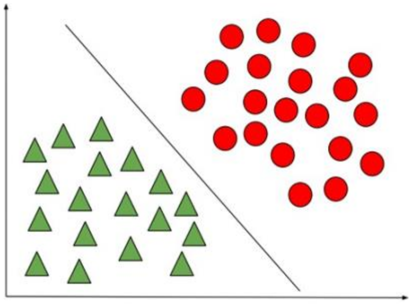
By analysing only, the numbers of **P(X|y)** as well as **P(y)** in the specific class, we can determine P(y), i.e., considering the characteristics of a sample, how likely is it that it belongs to class "y".

Gaussian Discriminant Analysis is a Generative Learning Algorithm that aims to determinethe distribution of every class. It attempts to create the Gaussian distribution to each category of data in a separate way. The likelihood of an outcome in the case using an algorithm known as the Generative learning algorithm is very high if it is close to the centre of the contour, which corresponds to its class. It diminishes when we move away from the middle of the contour. Below are images that illustrate the differences betweenDiscriminative as well as Generative Learning Algorithms.



(a) Generative Learning Algorithm (GDA)

Curse of dimensionality : Handling the high-dimensional data is very difficult in practice, commonly known as the *curse of dimensionality.* If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex. As the number of features increases, the number of samples also gets increased



(a) Discriminative Learning Algorithm

proportionally, and the chance of overfitting also increases. If the machine learning modelis trained on high-dimensional data, it becomes overfitted and results in poor performance.

Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction.

Dimensionality reduction & Methods:

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.

Methods of Dimensionality Reduction

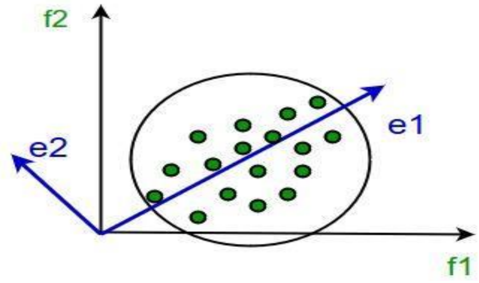
The various methods used for dimensionality reduction include:

- 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 linearmethod, 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 dimensionalspace 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 eigen values are used to reconstruct a large fraction of variance ofthe 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.

Principal Component Analysis

- Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.
- PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.
- PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.*** It is a feature extraction technique, so it contains the important variables and drops the least important variable.
- The PCA algorithm is based on some mathematical concepts such as:

  - Variance and Covariance
  - Eigenvalues and Eigen

factors Some common terms used in

PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

What is Scikit-Learn (Sklearn)

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**.

Origin of Scikit-Learn

It was originally called ***scikits.learn*** and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010.

Let’s have a look at its version history −

- May 2019: scikit-learn 0.21.0
- March 2019: scikit-learn 0.20.3
- December 2018: scikit-learn 0.20.2
- November 2018: scikit-learn 0.20.1
- September 2018: scikit-learn 0.20.0
- July 2018: scikit-learn 0.19.2

Features

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

Supervised Learning algorithms − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

Unsupervised Learning algorithms − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

Clustering − This model is used for grouping unlabeled data.

Cross Validation − It is used to check the accuracy of supervised models on unseen data.

Dimensionality Reduction − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

Ensemble methods − As name suggest, it is used for combining the predictions of multiple supervised models. **Feature extraction** − It is used to extract the features from

data to define the attributes in image and text data. **Feature selection** − It is used to

identify useful attributes to create supervised mode

data to define the attributes in image and text data. **Feature selection** − It is used to

identify useful attributes to create supervised mode

KERNEL PCA:

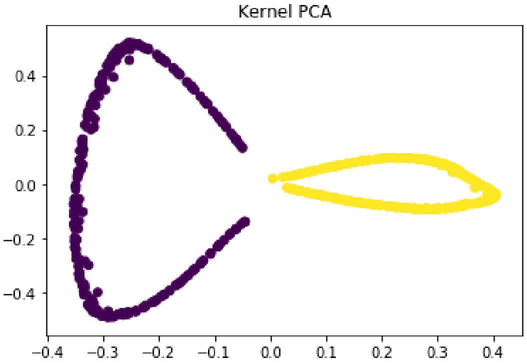
PCA is a linear method. That is it can only be applied to datasets which are linearly separable.

It does an excellentjob for datasets, which are linearly separable. But, if we use it to non-linear datasets, we might get a result which may not be the optimal dimensionality reduction. Kernel PCA uses a kernel function to project dataset into a higher dimensional feature space, where it is linearly separable. It is similar to the idea of Support Vector Machines.

There are various kernel methods like linear, polynomial, and gaussian.

Example: Applying kernel PCA on this dataset with RBF kernel with a gamma value of 15.from sklearn.decomposition import KernelPCA

```
kpca = KernelPCA(kernel ='rbf', gamma =
15)X_kpca = kpca.fit_transform(X)

plt.title("Kernel PCA")
plt.scatter(X_kpca[:, 0], X_kpca[:, 1], c = y)
plt.show()
```



In the kernel space the two classes are linearly separable. Kernel PCA uses a kernel function to project the dataset into a higher-dimensional space, where it is linearly separable.

Randomized PCA algorithm

Both SVD and NIPALS are not very efficient when number of rows in dataset is very large (e.g. hundreds of thousandsvalues or even more). Such datasets can be easily obtained in case of for example hyperspectral images. Direct use of the traditional algorithms with such datasets often leads to a lack of memory and long computational time.

One of the solution here is to use probabilistic algorithms, which allow to reduce the number of values needed fo restimation of principal components. Starting from *0.9.0* one of the probabilistic approach is also implemented
in *mdatools*. The original idea can be found in this paper and some examples on using the approach for PCA analysis ofhyperspectral

A Randomized Algorithm for PCA 2
• Form Y = AΩ. •
QR decompose Y and discard R.
The main theoretical result is:

$$E\|A - QQ{*}A\| \leq 1 + 4 * \sqrt{\ } k + p\ p - 1\ p \min(m, n)\ \sigma k+1(A).$$

Proof Sketch.

Apply the triangle inequality many times in order to split the error into a part that involves optimizing over a space of dimension k and a separate high dimensional part.

Let $\Omega \in R\ n\times(k+12)$, $W \in R\ (k+12)\times n$ and $Z \in R\ k\times(k+12)$

$$||A - QQ*A|| \leq 2||A - A\Omega W|| + 2||A\Omega - QZ||||W||.$$

we want to choose W and Z to show

$$||A - QQ*A|| \leq C\sigma k+1(A).$$

The algorithm forms Q's columns from singular vectors corresponding to the k + p greatest singular values of $A\Omega$. This lets us choose Z such that

$$||A - QQ*A|| \leq \sigma k+1(A\Omega) \leq ||\Omega||\sigma k+1(A)$$

where we understand the second inequality by recalling that we are working with the spectral norm in this note.

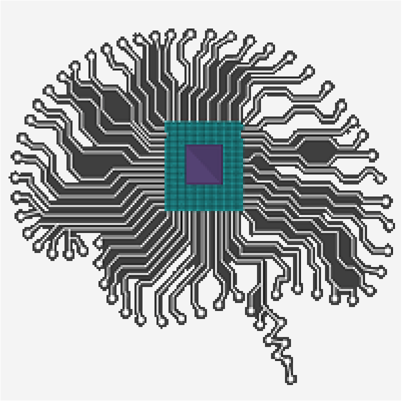
The existence of a $(k +p)\times n$ matrix W such that $||A-A\Omega W|| \leq C\sigma k+1(A)$ is tedious and shown in the appendix of using results from [1]. A few notes about the result:

- A few iterations of the power method in our computation of Y can improve the accuracy of our method.
- We expect the bound in to involve a factor of $\sigma k+1(A)$ as $\sigma k+1(A)$ is the theoretical best bound we can find.
- Notice that increasing p greatly improves accuracy.14

Unit V:

Neural Networks and Deep Learning: Introduction to Artificial Neural Networks with Keras, Implementing MLPs with Keras, Installing TensorFlow 2, Loading and Preprocessing Data with TensorFlow

Artificial Neural Network Tutorial



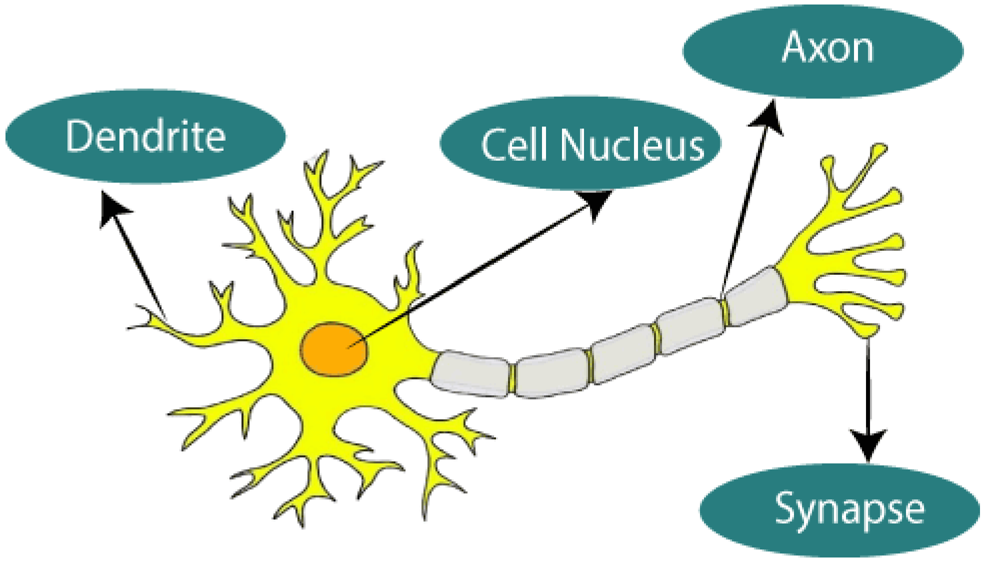
Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions.

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

Artificial neural network tutorial covers all the aspects related to the artificial neural network. In this tutorial, we will discuss ANNs, Adaptive resonance theory, Kohonen self-organizing map, Building blocks, unsupervised learning, Genetic algorithm, etc.

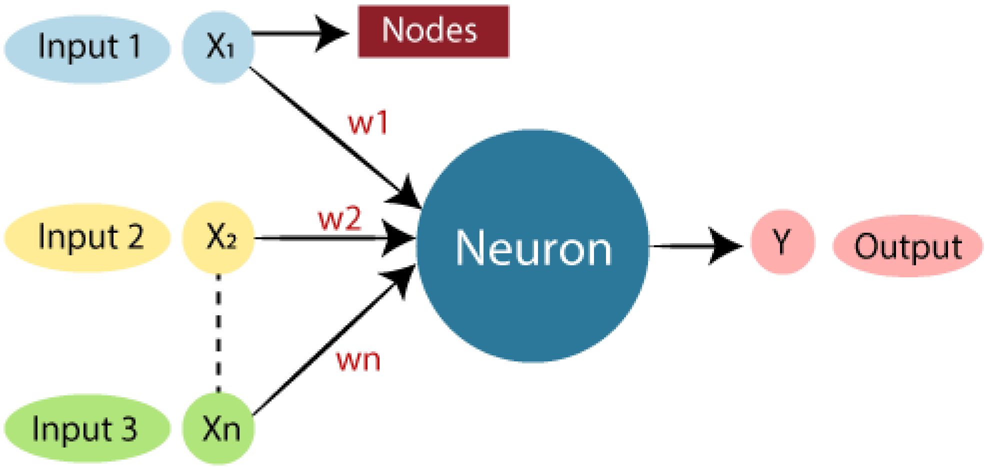
What is Artificial Neural Network?

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



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

The typical Artificial Neural Network looks something like the given figure.



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

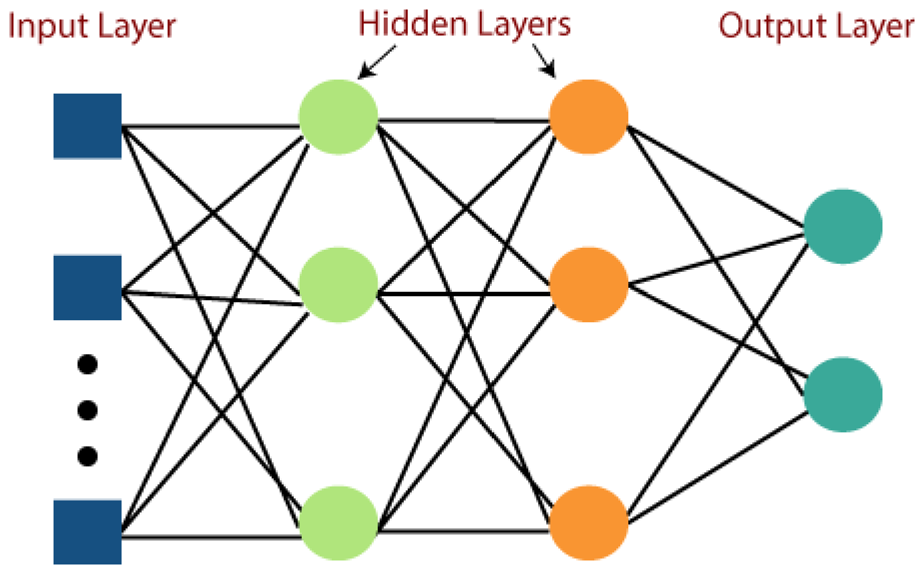
An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

Multi-Layer perceptron defines the most complex architecture of artificial neural networks. It is substantially formed from multiple layers of the perceptron. TensorFlow is a very popular deep learning framework released by, and this notebook will guide to build a neural network with this library. If we want to understand what is a Multi-layer perceptron, we have to develop a multi-layer perceptron from scratch using Numpy.

The pictorial representation of multi-layer perceptron learning is as shown below-



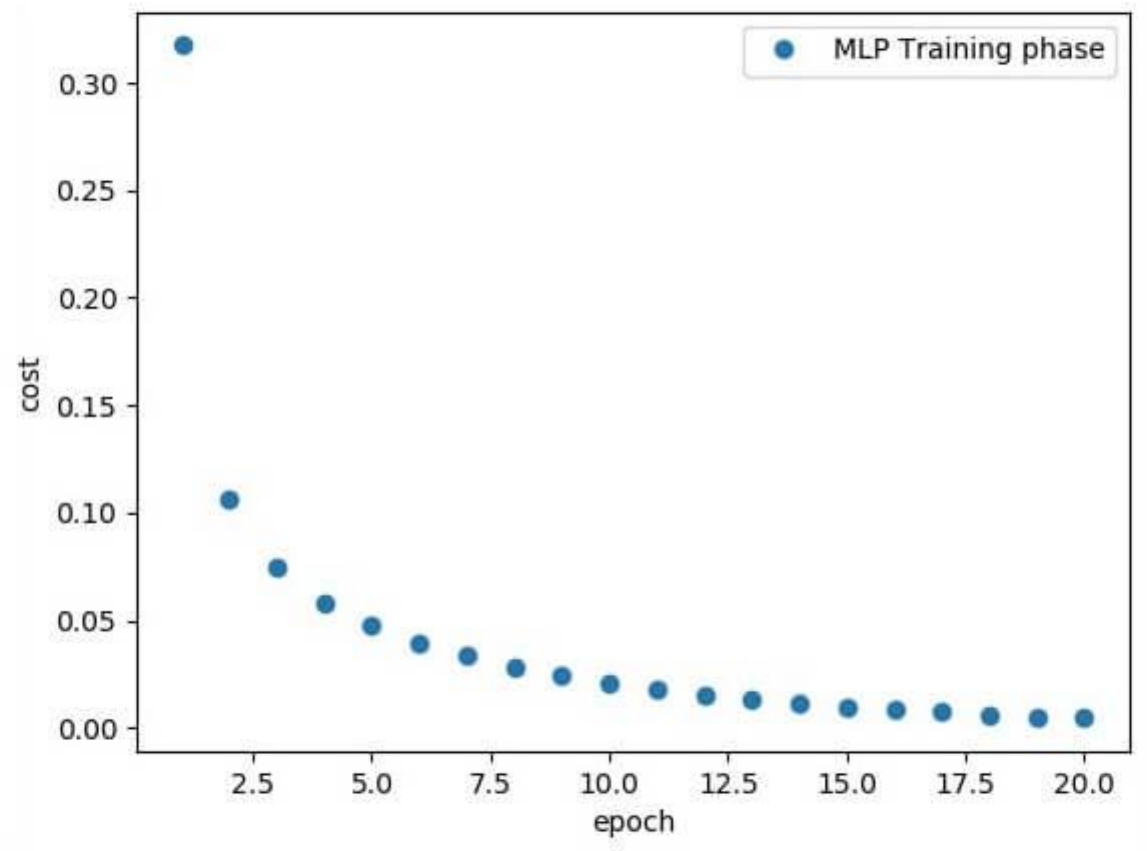
MLP networks are used for supervised learning format. A typical learning algorithm for MLP networks is also called **back propagation's algorithm**.

A multilayer perceptron (MLP) is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers. MLP uses backpropagation for training the network. MLP is a deep learning method.

Now, we are focusing on the implementation with MLP for an image classification problem.

```
# Import MINST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot = True)

import tensorflow as tf
import matplotlib.pyplot as plt

# Parameters
learning_rate = 0.001
training_epochs = 20
batch_size = 100
display_step = 1

# Network Parameters
n_hidden_1 = 256

# 1st layer num features
n_hidden_2 = 256 # 2nd layer num features
n_input = 784 # MNIST data input (img shape: 28*28) n_classes = 10
# MNIST total classes (0-9 digits)
  # tf Graph input   x = tf.placeholder("float", [None, n_input])
y = tf.placeholder("float", [None, n_classes])
  # weights layer 1
h = tf.Variable(tf.random_normal([n_input, n_hidden_1])) # bias layer 1   bias_layer_1 = tf.V
ariable(tf.random_normal([n_hidden_1]))
# layer 1 layer_1 = tf.nn.sigmoid(tf.add(tf.matmul(x, h), bias_layer_1))

# weights layer 2
w = tf.Variable(tf.random_normal([n_hidden_1, n_hidden_2]))

# bias layer 2   bias_layer_2 = tf.Variable(tf.random_normal([n_hidden_2]))
# layer 2
layer_2 = tf.nn.sigmoid(tf.add(tf.matmul(layer_1, w), bias_layer_2))

# weights output layer
output = tf.Variable(tf.random_normal([n_hidden_2, n_classes]))

# biar output layer
bias_output = tf.Variable(tf.random_normal([n_classes])) # output layer
output_layer = tf.matmul(layer_2, output) + bias_output

# cost function
cost = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(
  logits = output_layer, labels = y))

#cost = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(output_layer, y))
# optimizer
optimizer = tf.train.AdamOptimizer(learning_rate = learning_rate).minimize(cost)

# optimizer = tf.train.GradientDescentOptimizer(
  learning_rate = learning_rate).minimize(cost)

# Plot settings
avg_set = []
epoch_set = []

# Initializing the variables
init = tf.global_variables_initializer()

# Launch the graph
with tf.Session() as sess:
  sess.run(init)
     # Training cycle
  for epoch in range(training_epochs):
    avg_cost = 0.
    total_batch = int(mnist.train.num_examples / batch_size)

    # Loop over all batches
    for i in range(total_batch):
      batch_xs, batch_ys = mnist.train.next_batch(batch_size)
      # Fit training using batch data sess.run(optimizer, feed_dict = {
        x: batch_xs, y: batch_ys})          # Compute average loss
      avg_cost += sess.run(cost, feed_dict = {x: batch_xs, y: batch_ys}) / total_batch
    # Display logs per epoch step
    if epoch % display_step == 0:
      print           Epoch:", '%04d' % (epoch + 1), "cost=", "{:.9f}".format(avg_cost)
    avg_set.append(avg_cost)
    epoch_set.append(epoch + 1)
  print
  "Training phase finished"
     plt.plot(epoch_set, avg_set, 'o', label = 'MLP Training phase')
  plt.ylabel('cost')
  plt.xlabel('epoch')
  plt.legend()
  plt.show()

  # Test model
  correct_prediction = tf.equal(tf.argmax(output_layer, 1), tf.argmax(y, 1))
    # Calculate accuracy
      accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
      print
  "Model Accuracy:", accuracy.eval({x: mnist.test.images, y: mnist.test.labels})
```

The above line of codes generating the following output-



Creating an interactive section

We have two basic options when using TensorFlow to run our code:

- Build graphs and run sessions [Do all the set-up and then execute a session to implement a session to evaluate tensors and run operations].
- Create our coding and run on the fly.

For this first part, we will use the interactive session that is more suitable for an environment like Jupiter notebook.

sess = tf.InteractiveSession()

Installation of TensorFlow Through pip

In this tutorial, we will describe that how to install TensorFlow in Windows 10.

We can download TensorFlow in our system in 2 ways:

1. Through pip (Python package library)

2. Through Anaconda Navigator (conda)

1. Through pip

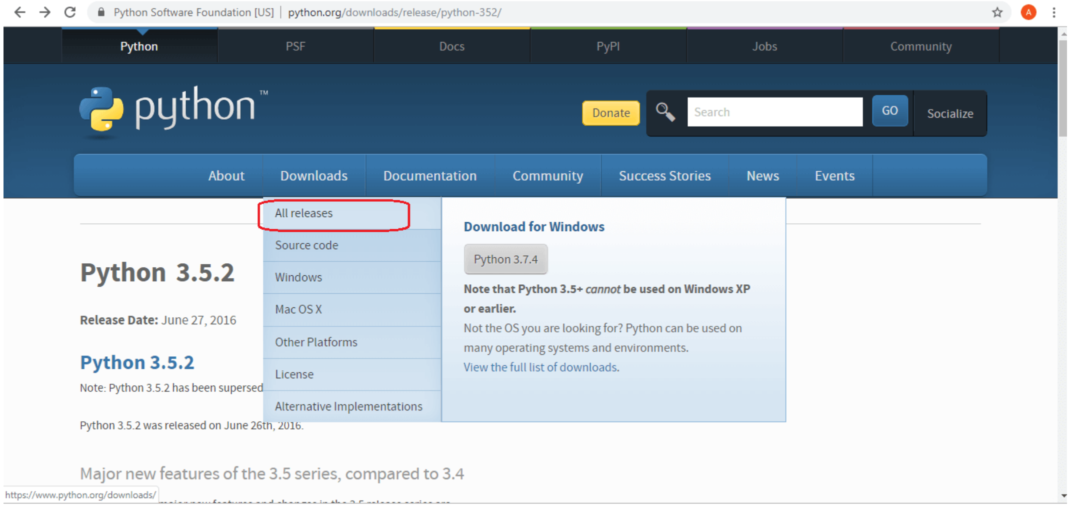
So, firstly we have to install and set-up anaconda in our system through pip.

The following are the requirement for TensorFlow to work on our computer.

Stay

- TensorFlow has only supported 64-bit Python **3.5.x** or Python **3.6.x** on Windows
- When we download the Python **3.5.x** version, it comes with the **pip3** package manager. (Which is the program that we are going to need for our users to install TensorFlow on Windows).

Step 1: Download Python from the below link:



Python 3.5.2

Release Date: June 27, 2016

Python 3.5.2

Note: Python 3.5.2 has been supersed

Python 3.5.2 was released on June 26th, 2016.

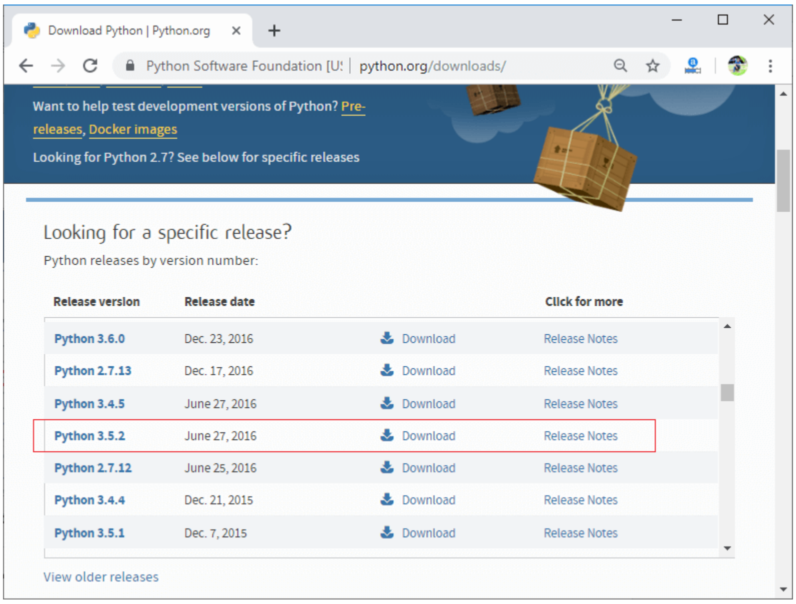
Major new features of the 3.5 series, compared to 3.4

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

https://www.python.org/downloads/release/python-352/



Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Looking for a specific release?

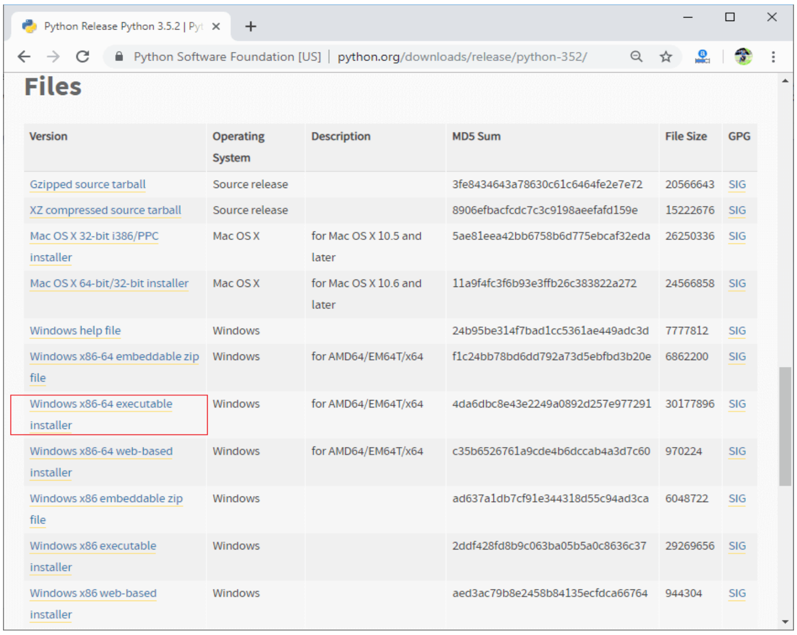
Python releases by version number:

Release version	Release date		Click for more
Python 3.6.0	Dec. 23, 2016	Download	Release Notes
Python 2.7.13	Dec. 17, 2016	Download	Release Notes
Python 3.4.5	June 27, 2016	Download	Release Notes
Python 3.5.2	June 27, 2016	Download	Release Notes
Python 2.7.12	June 25, 2016	Download	Release Notes
Python 3.4.4	Dec. 21, 2015	Download	Release Notes
Python 3.5.1	Dec. 7, 2015	Download	Release Notes

View older releases

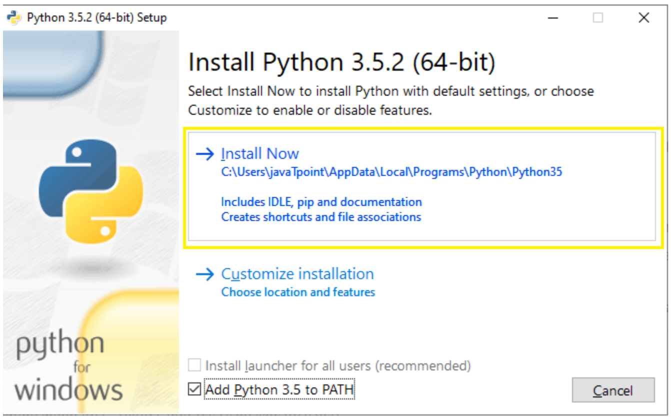
After that,

Step 3: We will be brought to another page, where we will need to select either the **x86-64** or **amd64 installer** to install Python.


Python Release Python 3.5.2 | Pyt
Python Software Foundation [US]
python.org/downloads/release/python-352/

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		3fe8434643a78630c61c6464fe2e7e72	20566643	SIG
XZ compressed source tarball	Source release		8906efbacfcdc7c3c9198aeefafd159e	15222676	SIG
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later	5ae81eea42bb6758b6d775ebcaf32eda	26250336	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	11a9f4fc3f6b93e3ffb26c383822a272	24566858	SIG
Windows help file	Windows		24b95be314f7bad1cc5361ae449adc3d	7777812	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	f1c24bb78bd6dd792a73d5ebfbd3b20e	6862200	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	4da6dbc8e43e2249a0892d257e977291	30177896	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	c35b6526761a9cde4b6dccab4a3d7c60	970224	SIG
Windows x86 embeddable zip file	Windows		ad637a1db7cf91e344318d55c94ad3ca	6048722	SIG
Windows x86 executable installer	Windows		2ddf428fd8b9c063ba05b5a0c8636c37	29269656	SIG
Windows x86 web-based installer	Windows		aed3ac79b8e2458b84135ecfdca66764	944304	SIG



Install Python 3.5.2 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

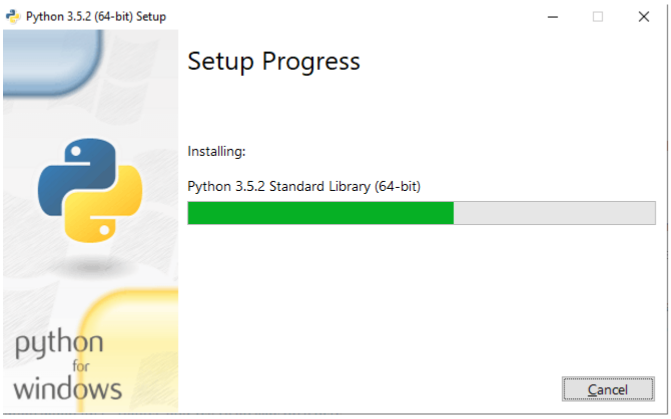
→ Install Now
C:\Users\javaTpoint\AppData\Local\Programs\Python\Python35

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☐ Install launcher for all users (recommended)
☑ Add Python 3.5 to PATH

Now, Python is installing successfully.

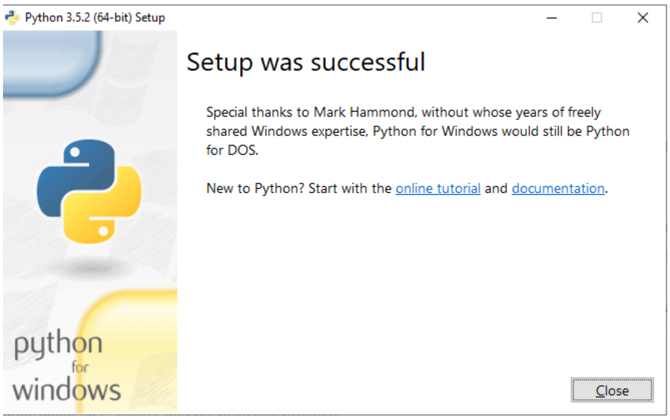


Setup Progress

Installing:

Python 3.5.2 Standard Library (64-bit)

Step 4: For this tutorial, I'll be choosing to Add **Python 3.5 to PATH**.



Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

Step 5: Now, we will be able to see the message "**Set-up was successful**." A way to confirm that it hs installed successfully is to open your **Command Prompt** and check the version.

What is pip?

pip is known as a **package management system** which is used to install and manage the software package, which is written in Python or any other languages. pip is used to download, search, install, uninstall, and manage the 3rd party python package. (pip3 is the latest version of it which comes with new Python 3.5.x version that we had just downloaded)

Installing our TensorFlow

Once we have downloaded the latest version of Python, we can now put our finishing touches by installing our **TensorFlow**.

Step 1: To install TensorFlow, start the terminal. Make sure that we run the cmd as an administrator.

If we do not know how to run your cmd as an administrator

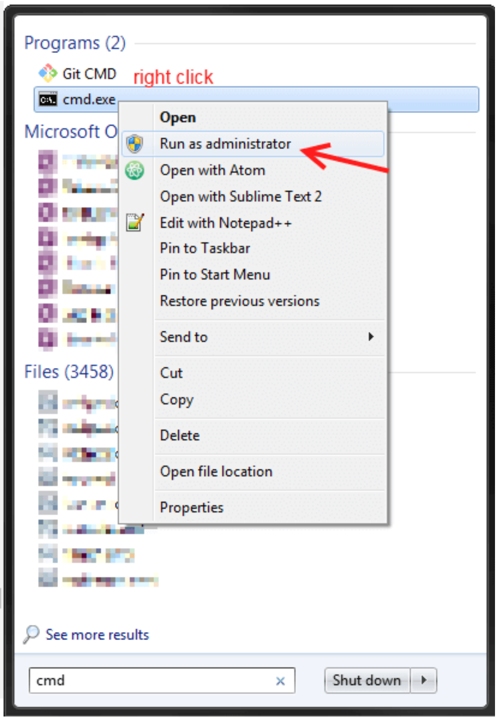
Here's how we can run in our cmd as an administrator.

Open the Start menu, search for **cmd**, and then right-click on it and **Run as an administrator**.

Open the Start menu, search for **cmd**, and then right-click on it and **Run as an administrator**.

Step 2: Once we are done with that, then we have to write the command in **command prompt** for finish installing Tensorflow in our Windows.

Enter this command:



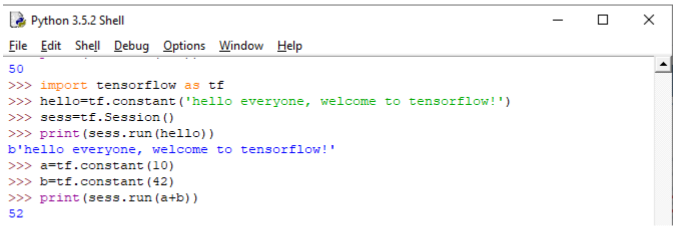
C:\pip3 install -upgrade tensorflow

Now, TensorFlow is successfully installed in our system.

Testing our TensorFlow

Here, we try and prove whether our new TensorFlow works smoothly without any problems.

Below is an example that you can write to the test.

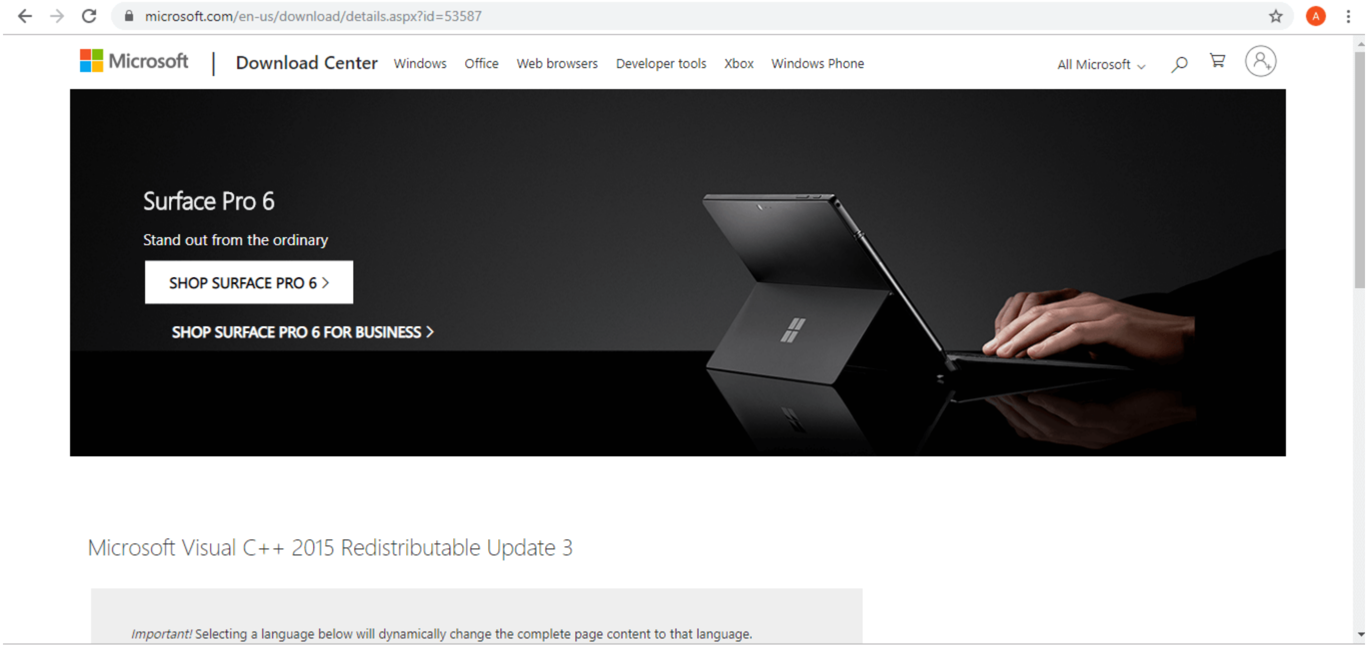


```
50
>>> import tensorflow as tf
>>> hello=tf.constant('hello everyone, welcome to tensorflow!')
>>> sess=tf.Session()
>>> print(sess.run(hello))
b'hello everyone, welcome to tensorflow!'
>>> a=tf.constant(10)
>>> b=tf.constant(42)
>>> print(sess.run(a+b))
52
```

TensorFlow is successfully working now.

Otherwise, If we are getting any problem to run the program, then we have to install **Microsoft Visual C++ 2015**. And set up in our system then the TensorFlow will be run in the system.

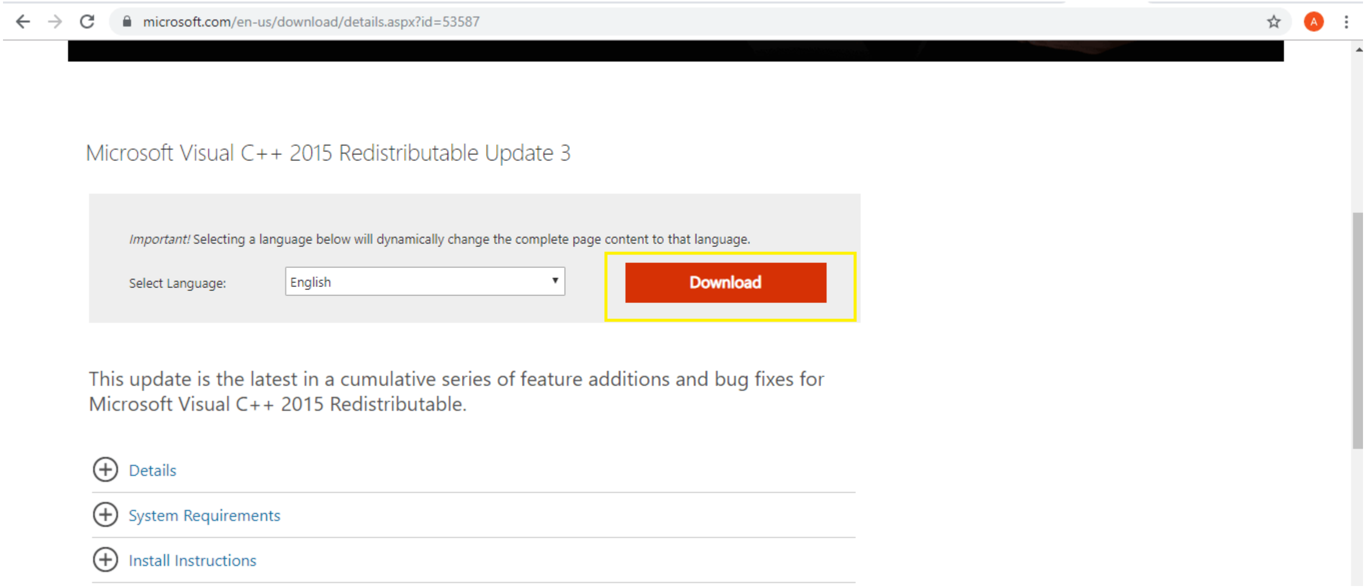
We can download **Microsoft Visual C++ 2015** from the below link: https://www.microsoft.com/en-us/download/confirmation.aspx?id=53587



Microsoft Visual C++ 2015 Redistributable Update 3

Important! Selecting a language below will dynamically change the complete page content to that language.

We can download from here.



microsoft.com/en-us/download/details.aspx?id=53587

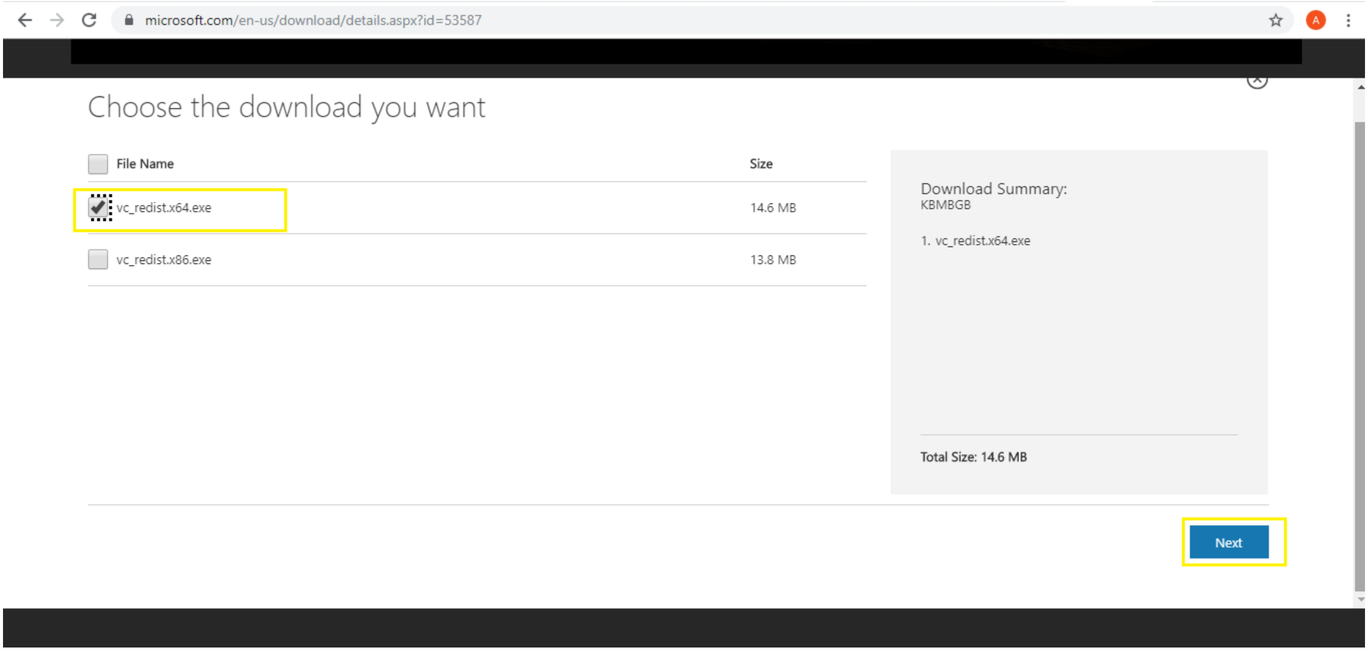
Microsoft Visual C++ 2015 Redistributable Update 3

Important! Selecting a language below will dynamically change the complete page content to that language.

Select Language: English

This update is the latest in a cumulative series of feature additions and bug fixes for Microsoft Visual C++ 2015 Redistributable.

Choose the **vc_redist.x64.exe** on the page and click on "**Next"** after that it will be downloaded.



microsoft.com/en-us/download/details.aspx?id=53587

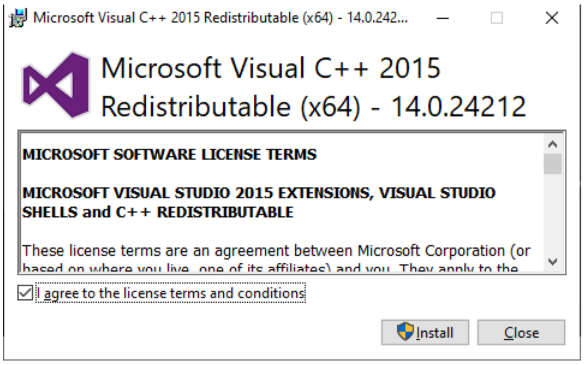
Choose the download you want

File Name	Size
☑ vc_redist.x64.exe	14.6 MB
□ vc_redist.x86.exe	13.8 MB

Download Summary:
KBMBGB

1. vc_redist.x64.exe

Total Size: 14.6 MB



Microsoft Visual C++ 2015 Redistributable (x64) - 14.0.242...

Microsoft Visual C++ 2015 Redistributable (x64) - 14.0.24212

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT VISUAL STUDIO 2015 EXTENSIONS, VISUAL STUDIO SHELLS and C++ REDISTRIBUTABLE

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. They apply to the

☑ I agree to the license terms and conditions

At last, it will successfully installed in our system.

We will read **conda install TensorFlow** in our next tutorial.

How is data loaded with TensorFlow?

In memory data

For any small CSV dataset the simplest way to train a TensorFlow model on it is to **load it into memory as a pandas Dataframe or a NumPy array**. A relatively simple example is the abalone dataset. The dataset is small. All the input features are all limited-range floating point values.

Data preprocessing for ML using TensorFlow Transform

1. Run Notebook 1.
2. Overview of the pipeline.
3. Read raw training data from BigQuery.
4. Transform raw training data.
5. Write transformed training data.
6. Read, transform, and write evaluation data.
7. Save the graph.