

Topics

- What is Machine Learning?
- Need for Machine Learning
- Why & When to Make Machines Learn?
- Machine Learning Model Challenges in Machines Learning Applications of Machines Learning

Learning Outcome

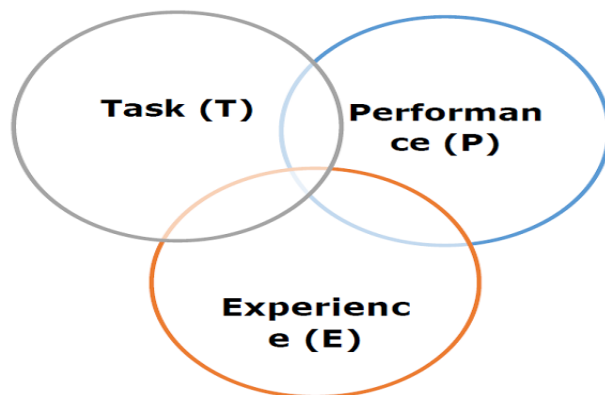
- Why we learn the Machine Learning and
- What is the need and current demand of this technology
- And what challenges he/she has to face in this field.

❖ Machine Learning :- Introduction

Coding has a limit and after exceeding that limit we can't program it anymore. It is the type of AI that provide computer with the ability to learn without being explicitly programmed. When we are using machine learning and you have the computer, you are feeding the data, input as well as examples of output. So, you are putting examples of input, output data and you are getting a program or a model with which you can solve subsequent tasks.

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of Machine Learning.

Definition of machine learning as given by **Tom Mitchell** and this is our definition that is followed very popularly. "A computer program is said to learn from Experience E with respect to some class of tasks T and performance measure P". If its performance on tasks in T, as measured by P improves with Experience E.

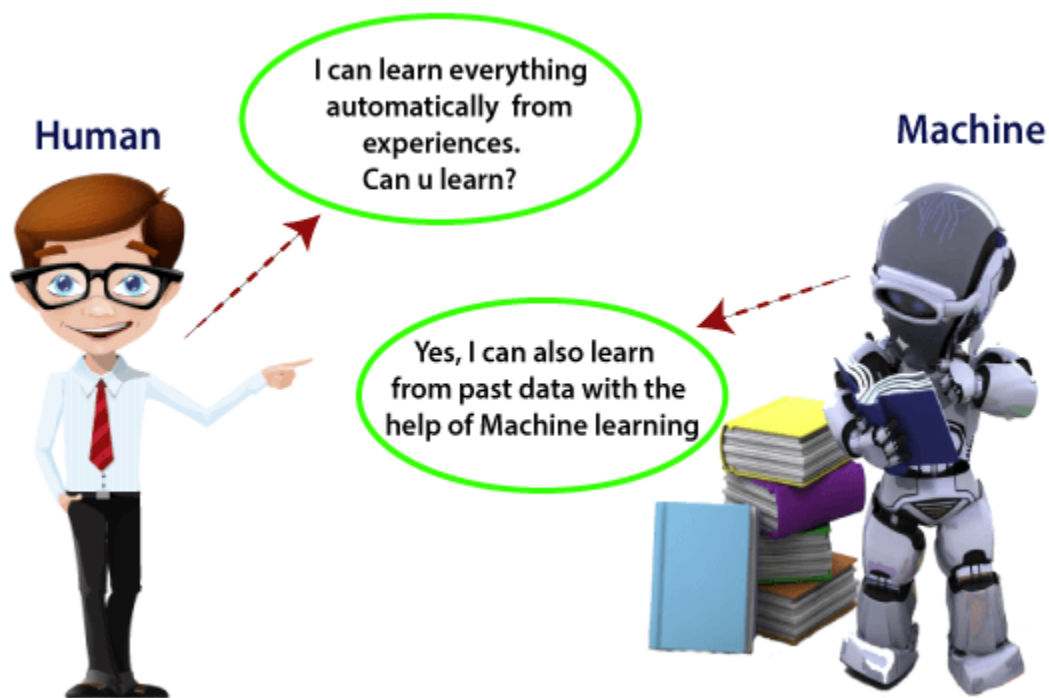


Task- Prediction, Classification

Experience also called Data.

Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel in 1959**. We can define it in a summarized way as:

“Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.”



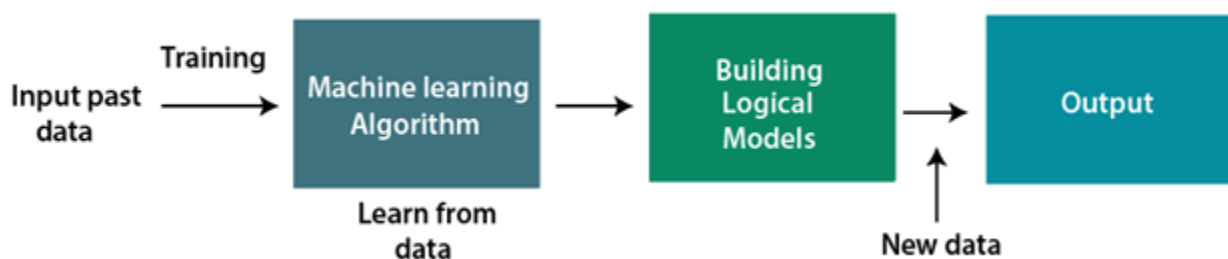
With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.

A machine has the ability to learn if it can improve its performance by gaining more data.

How does Machine Learning work:-

A **Machine Learning system learns from historical data**, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of Writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machines build the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:



Features of Machine Learning:-

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Need for Machine Learning:-

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes machine learning to make things easy for us.

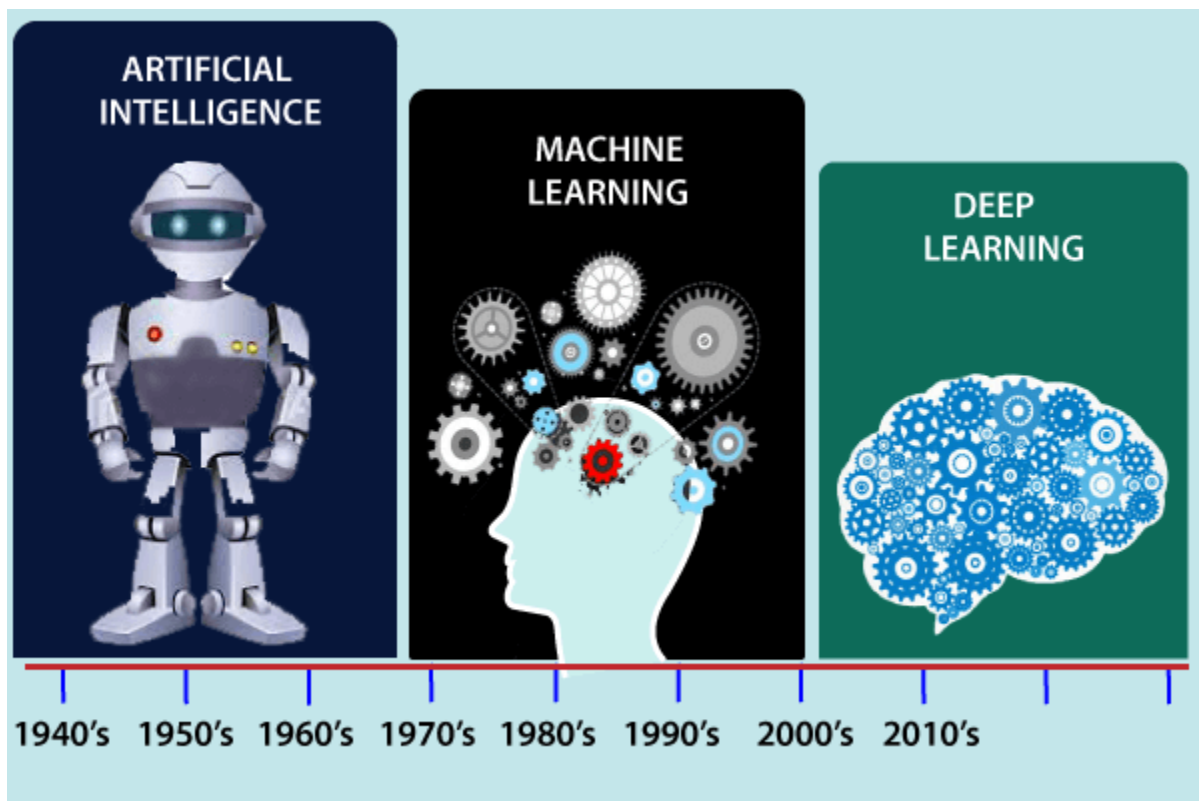
We can train machine learning algorithms by providing them with a huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its use cases. Currently, machine learning is used in self-driving cars, cyber fraud detection, face recognition, and friend suggestions by Facebook, etc. Various top companies such as Netflix and Amazon have built machine learning models that are using a vast amount of data to analyze the user interest and recommend products accordingly.

Some key points which show the importance of Machine Learning:-

- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data.

History of Machine Learning:-



The early history of Machine Learning (Pre-1940):-

- 1834: In 1834, Charles Babbage, the father of the computer, conceived a device that could be programmed with punch cards. However, the machine was never built, but all modern computers rely on its logical structure.
- 1936: In 1936, Alan Turing gave a theory on how a machine can determine and execute a set of instructions.

The era of stored program computers:-

- 1940: In 1940, the first manually operated computer, "ENIAC" was invented, which was the first electronic general-purpose computer. After that stored program computer such as EDSAC in 1949 and EDVAC in 1951 were invented.
- 1943: In 1943, a human neural network was modeled with an electrical circuit. In 1950, the scientists started applying their idea to work and analyzed how human neurons might work.

Computer machinery and intelligence:-

- 1950: In 1950, Alan Turing published a seminal paper, "Computer Machinery and Intelligence," on the topic of artificial intelligence. In his paper, he asked, "Can machines think?"
- **1950s:** Samuel's Checker playing program

Machine intelligence in Games:-

- 1952: Arthur Samuel, who was the pioneer of machine learning, created a program that helped an IBM computer to play a checkers game. It performed better the more it played.
- 1959: In 1959, the term "Machine Learning" was first coined by Arthur Samuel.

The first "AI" winter:-

- The duration of 1974 to 1980 was a tough time for AI and ML researchers, and this duration was called AI winter.
- In this duration, failure of machine translation occurred, and people had reduced their interest in AI, which led to reduced funding by the government to the research.

Machine Learning from theory to reality:-

- **1959s:** In 1959, the first neural network was applied to a real-world problem to remove echoes over phone lines using an adaptive filter.
- **1960s:** Neural Network : Rosenblatt's perceptron, Pattern Recognition, Minsky & Paper prove limitations of perceptron.
- **1970s:** Symbolic concept induction
 - Expert systems and knowledge acquisition bottleneck
 - Quinlan's ID3
 - Natural language processing (symbolic)
- **1980s:** Advanced decision tree and rule learning
 - Learning and planning and problem solving
 - Resurgence of neural network Valiant's PAC learning theory
 - Focus on experimental methodology
- **1985:** In 1985, Terry Sejnowski and Charles Rosenberg invented a neural network NETtalk , which was able to teach itself how to correctly pronounce 20,000 words in one week.
- **1997:** IBM's deep blue intelligent computer won the chess game against the chess expert Garry Kasparov, and it became the first computer which had beaten a human chess expert.
- **90's ML and Statistics:** Support Vector Machines
 - Data Mining
 - Adaptive agents and web applications
 - Text learning
 - Reinforcement learning
 - Ensembles

Note:-

- **1994:** Self-driving car road test
 - Bayes Net learning
- **1997:** Deep Blue beats Gary Kasparov
- **2009:** Google builds Self driving car
- **2011:** Watson wins Jeopardy
- **2014:** Human vision surpassed by ML systems

Popularity of this field in recent time and the reasons behind that-

- New software/ algorithms

- Neural networks
 - Deep Learning
- New hardware
- GPU's
 - Cloud Enabled
 - Availability of Big Data

Machine Learning at 21st century:-

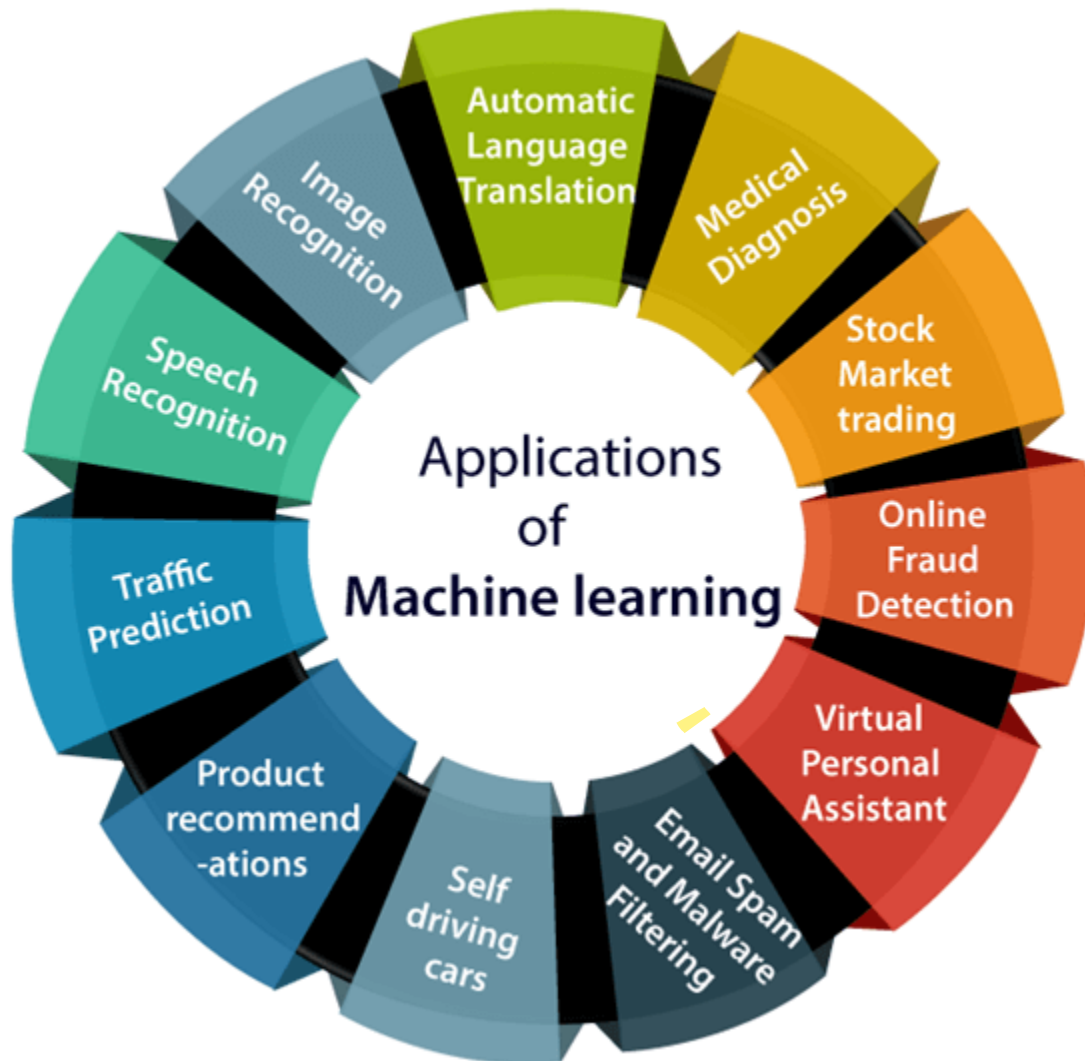
- **2006:** In the year 2006, computer scientist Geoffrey Hinton has given a new name to neural net research as "**deep learning**," and nowadays, it has become one of the most trending technologies.
- **2012:** In 2012, Google created a deep neural network which learned to recognize the image of humans and cats in YouTube videos.
- **2014:** In 2014, the Chabot "**Eugen Goostman**" cleared the Turing Test. It was the first Chabot who convinced the 33% of human judges that it was not a machine.
- **2014: DeepFace** was a deep neural network created by Facebook, and they claimed that it could recognize a person with the same precision as a human can do.
- **2016: AlphaGo** beat the world's number second player **Lee Sedol at Go game**. In 2017 it beat the number one player of this game **Ke Jie**.
-
- **2017:** In 2017, the Alphabet's Jigsaw team built an intelligent system that was able to learn **online trolling**. It used to read millions of comments of different websites to learn to stop online trolling.

Machine Learning at present:-

Now machine learning has made great progress in its research, and it is present everywhere around us, such as self-driving cars, Amazon Alexa, Catboats, recommender systems, and many more.

Modern machine learning models can be used for making various predictions, including weather prediction, disease prediction, stock market analysis, etc.

Applications of Machine Learning: - We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:



1. Image Recognition:-

Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is Automatic friend tagging suggestion: Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with a name, and the technology behind this is machine learning's face detection and recognition algorithm.

It is based on the Facebook project named "Deep Face," which is responsible for face recognition and person identification in the picture.

2. Email Spam and Malware Filtering:-

Whenever we receive a new email, it is filtered automatically as important, normal, and spam. We always receive important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning. Below are some spam filters used by Gmail:

- Content Filter
- Header filter
- General blacklists filter
- Rules-based filters
- Permission filters

Some machine learning algorithms such as Multi-Layer Perceptron, Decision tree, and Naïve Bayes classifier are used for email spam filtering and malware detection.

3. Product recommendations:-

Machine learning is widely used by various e-commerce and entertainment companies such as Amazon, Netflix, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.

Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest.

Similarly, when we use Netflix, we find some recommendations for entertainment series, movies, etc., and this is also done with the help of machine learning.

4. Speech Recognition:-

While using Google, we get an option of "**Search by voice**," it comes under speech recognition, and it's a popular application of machine learning.

Speech recognition is a process of converting voice instructions into text, and it is also known as "**Speech to text**", or "**Computer speech recognition**." At present, machine learning algorithms are widely used by various applications of speech recognition. **Google**

assistant, Siri, Cortana, and Alexa are using speech recognition technology to follow the voice instructions.

5. Traffic prediction:-

If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.

It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:

- Real Time location of the vehicle from Google Map app and sensors
- Average time has taken on past days at the same time.

Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends it back to its database to improve the performance.

6. Self-driving cars:-

One of the most exciting applications of machine learning is self-driving cars. Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company, is working on self-driving cars. It is using an unsupervised learning method to train the car models to detect people and objects while driving.

7. Online Fraud Detection:

Machine learning is making our online transactions safe and secure by detecting fraud transactions. Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as fake accounts, fake ids, and steal money in the middle of a transaction. So to detect this, Feed Forward Neural network helps us by checking whether it is a genuine transaction or a fraud transaction.

For each genuine transaction, the output is converted into some hash values, and these values become the input for the next round. For each genuine transaction, there is a specific pattern which gets changed for the fraud transaction hence, it detects it and makes our online transactions more secure.

8. Virtual Personal Assistant:-

We have various virtual personal assistants such as Google assistant, Alexa, Cortana, Siri. As the name suggests, they help us in finding the information using our voice instruction.

These assistants can help us in various ways just by our voice instructions such as Play music, call someone, Open an email, Scheduling an appointment, etc.

These virtual assistants use machine learning algorithms as an important part.

These assistants record our voice instructions, send it over the server on a cloud, and decode it using ML algorithms and act accordingly.

9. Medical Diagnosis:-

In medical science, machine learning is used for disease diagnoses. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain.

It helps in finding brain tumors and other brain-related diseases easily.

10. Automatic Language Translation:-

Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages. Google's GNMT (Google Neural Machine Translation) provides this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it is called as automatic translation.

The technology behind the automatic translation is a sequence to sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

11. Stock Market trading:-

Machine learning is widely used in stock market trading. In the stock market, there is always a risk of ups and downs in shares, so for this machine learning's **long short term memory neural network** is used for the prediction of stock market trends.

Python Ecosystem

An Introduction to Python

Python is a popular object-oriented programming language having the capabilities of a high-level programming language. It's easy to learn the syntax and its portability capability makes it popular these days. The followings facts give us an introduction to Python –

- Python was developed by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands.
- It was written as the successor of a programming language named 'ABC'.
- Its first version was released in 1991.
- The name Python was picked by Guido van Rossum from a TV show named Monty Python's Flying Circus.
- It is an open-source programming language which means that we can freely download it and use it to develop programs. It can be downloaded from www.python.org.
- Python programming language is having the features of Java and C both. It is having elegant 'C' code and on the other hand, it is having classes and objects like Java for object-oriented programming.
- It is an interpreted language, which means the source code of the Python program would be first converted into bytecode and then executed by Python virtual machine.

Strengths and Weaknesses of Python

Every programming language has some strengths as well as weaknesses, so does Python too.

Strengths

According to studies and surveys, Python is the fifth most important language as well as the most popular language for machine learning and data science. It is because of the following strengths that Python has –

Easy to learn and understand – The syntax of Python is simpler; hence it is relatively easy, even for beginners also, to learn and understand the language.

Multi-purpose language – Python is a multi-purpose programming language because it supports structured programming, object-oriented programming as well as functional programming.

A huge number of modules – Python has a huge number of modules covering every aspect of programming. These modules are easily available for use hence making Python an extensible language.

Support of open-source community – As open source programming language, Python is supported by a very large developer community. Due to this, the bugs are easily fixed by the Python community. This characteristic makes Python very robust and adaptive.

Scalability – Python is a scalable programming language because it provides an improved structure for supporting large programs than shell scripts.

Weakness

Although Python is a popular and powerful programming language, it has its own weakness of slow execution speed.

The execution speed of Python is slow as compared to compiled languages because Python is an interpreted language. This can be a major area of improvement for the Python community.

Installing Python

We can perform the installation of Python in any of the following two ways –

- Installing Python individually
- Using Pre-packaged Python distribution – Anaconda

● Installing Python Individually

If you want to install Python on your computer, then then you need to download only the binary code applicable to your platform. Python distribution is available for Windows, Linux, and Mac platforms.

The following is a quick overview of installing Python on the above-mentioned platforms –

On Unix and Linux platforms

With the help of the following steps, we can install Python on Unix and Linux platforms –

- First, go to www.python.org/downloads/.
- Next, click on the link to download the zipped source code available for Unix/Linux.
- Now, Download and extract files.
- Next, we can edit the Modules/Setup file if we want to customize some options.
 - Next, write the command **run ./configure script**
 - make
 - make install

On Windows platform

With the help of the following steps, we can install Python on the Windows platform –

- First, go to www.python.org/downloads/.
- Next, click on the link for the Windows installer python-XYZ.msi file. Here XYZ is the version we wish to install.
- Now, we must run the file that is downloaded. It will take us to the Python install wizard, which is easy to use. Now, accept the default settings and wait until the installation is finished.

On Macintosh platform

For Mac OS X, Homebrew, a great and easy-to-use package installer is recommended to install Python 3. In case if you don't have Homebrew, you can install it with the help of the following command –

```
$ ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

It can be updated with the command below –

```
$ brew update
```

Now, to install Python3 on your system, we need to run the following command –

```
$ brew install python3
```

Using Pre-packaged Python Distribution: Anaconda

Anaconda is a packaged compilation of Python which have all the libraries widely used in Data science. We can follow the following steps to set up a Python environment using Anaconda –

- **Step 1** – First, we need to download the required installation package from Anaconda distribution. The link for the same is www.anaconda.com/distribution/. You can choose from Windows, Mac, and Linux OS as per your requirement.
- **Step 2** – Next, select the Python version you want to install on your machine. The latest Python version is 3.7. There you will get the options for 64-bit and 32-bit Graphical installers.
- **Step 3** – After selecting the OS and Python version, it will download the Anaconda installer on your computer. Now, double-click the file and the installer will install the Anaconda package.
- **Step 4** – For checking whether it is installed or not, open a command prompt and type Python as follows –

A screenshot of a Windows Command Prompt window titled "Command Prompt - Python". The window has a black background with white text. The text shows the user navigating to the Anaconda3 directory and running the Python command, which opens the Python 3.7.1 shell. The prompt shows the user is in the directory C:\Users\Leekha\Anaconda3 and has run the command 'Python'. The output shows 'Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32' followed by a prompt for help or license information. The prompt is currently '>>>'.

```
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Leekha>cd Anaconda3
C:\Users\Leekha\Anaconda3>Python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Why Python for Data Science?

Python is the fifth most important language as well as the most popular language for Machine learning and data science. The following are the features of Python that make it the preferred choice of language for data science –

An extensive set of packages

Python has an extensive and powerful set of packages that are ready to be used in various domains. It also has packages like **numpy**, **scipy**, **pandas**, **sci-kit-learn** etc. which are required for machine learning and data science.

Easy prototyping

Another important feature of Python that makes it the choice of language for data science is the easy and fast prototyping. This feature is useful for developing new algorithms.

Collaboration feature

The field of data science basically needs good collaboration and Python provides many useful tools that make this extremely.

One language for many domains

A typical data science project includes various domains like data extraction, data manipulation, data analysis, feature extraction, modeling, evaluation, deployment, and updating the solution. As Python is a multi-purpose language, it allows the data scientist to address all these domains from a common platform.

Components of Python ML Ecosystem

In this section, let us discuss some core Data Science libraries that form the components of the Python Machine learning ecosystem. These useful components make Python an important language for Data Science. Though there are many such components, let us discuss some of the important components of the Python ecosystem here –

Jupyter Notebook

Jupyter notebooks basically provide an interactive computational environment for developing Python-based Data Science applications. They are formerly known as python notebooks. The following are some of the features of Jupyter notebooks that make it one of the best components of the Python ML ecosystem –

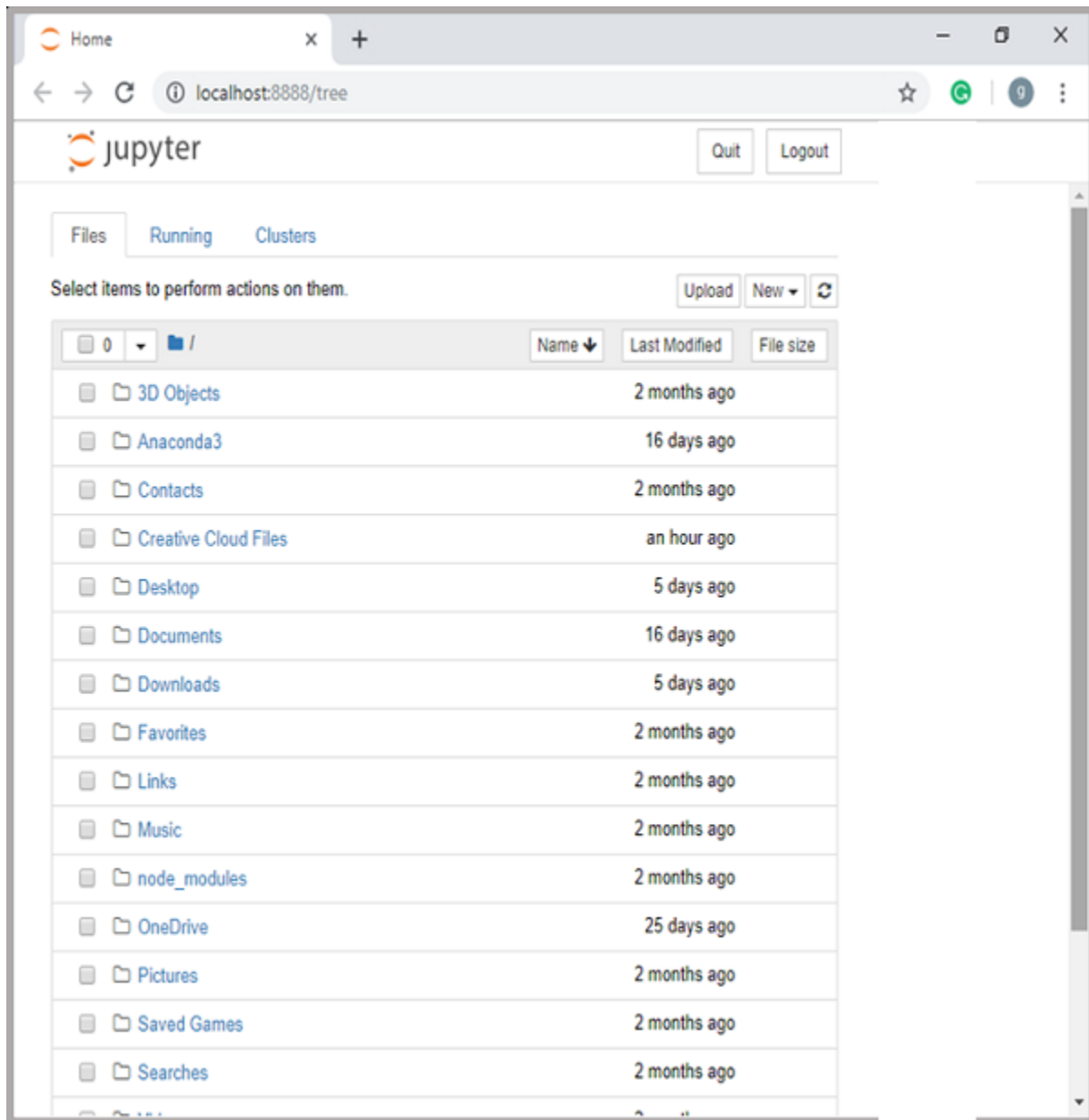
- Jupyter notebooks can illustrate the analysis process step by step by arranging the stuff like code, images, text, output, etc. in a step-by-step manner.
- It helps a data scientist to document the thought process while developing the analysis process.
- One can also capture the result as part of the notebook.
- With the help of Jupiter notebooks, we can share our work with a peer also.

Installation and Execution

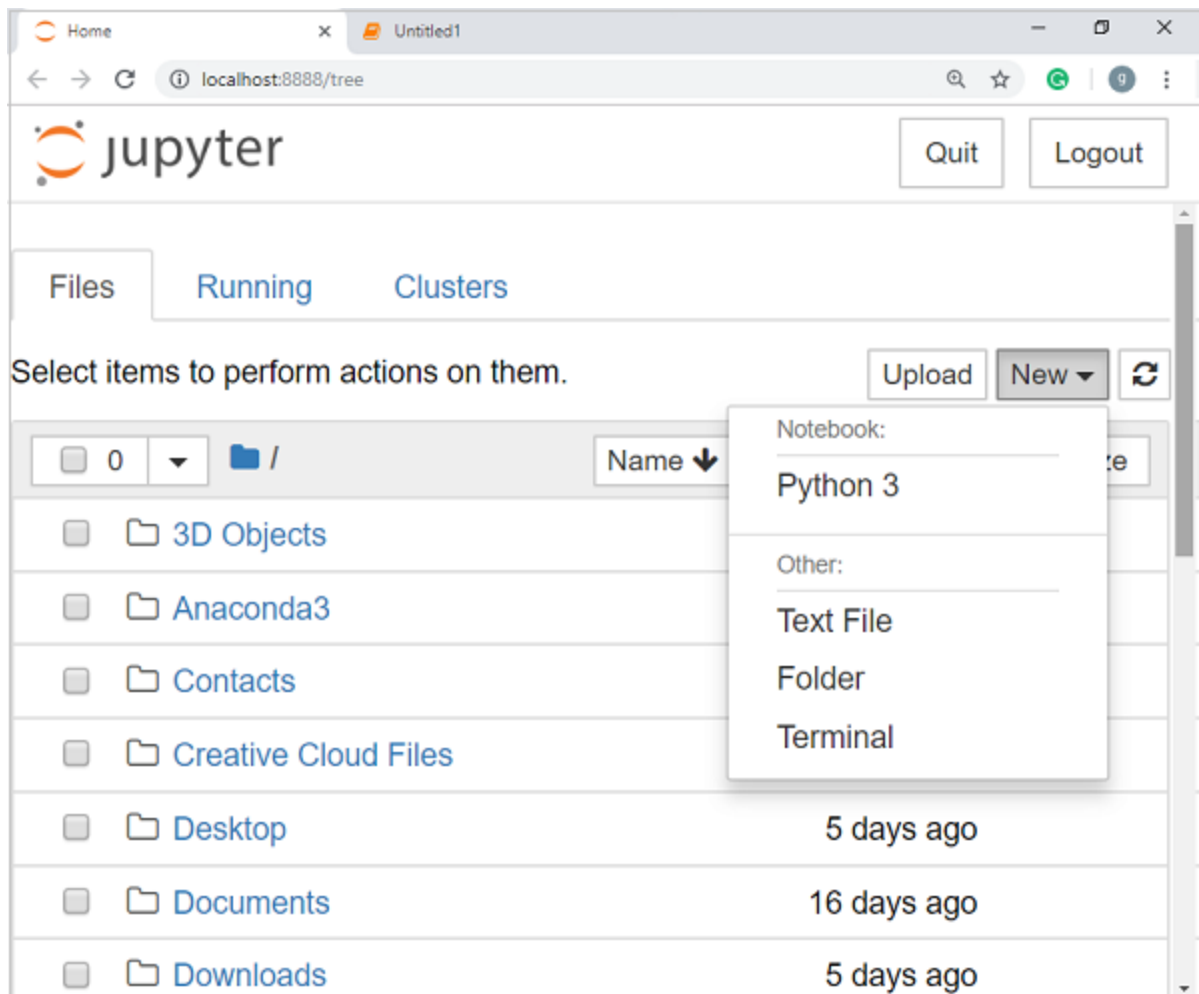
If you are using Anaconda distribution, then you need not install jupyter notebook separately as it is already installed with it. You just need to go to Anaconda Prompt and type the following command –

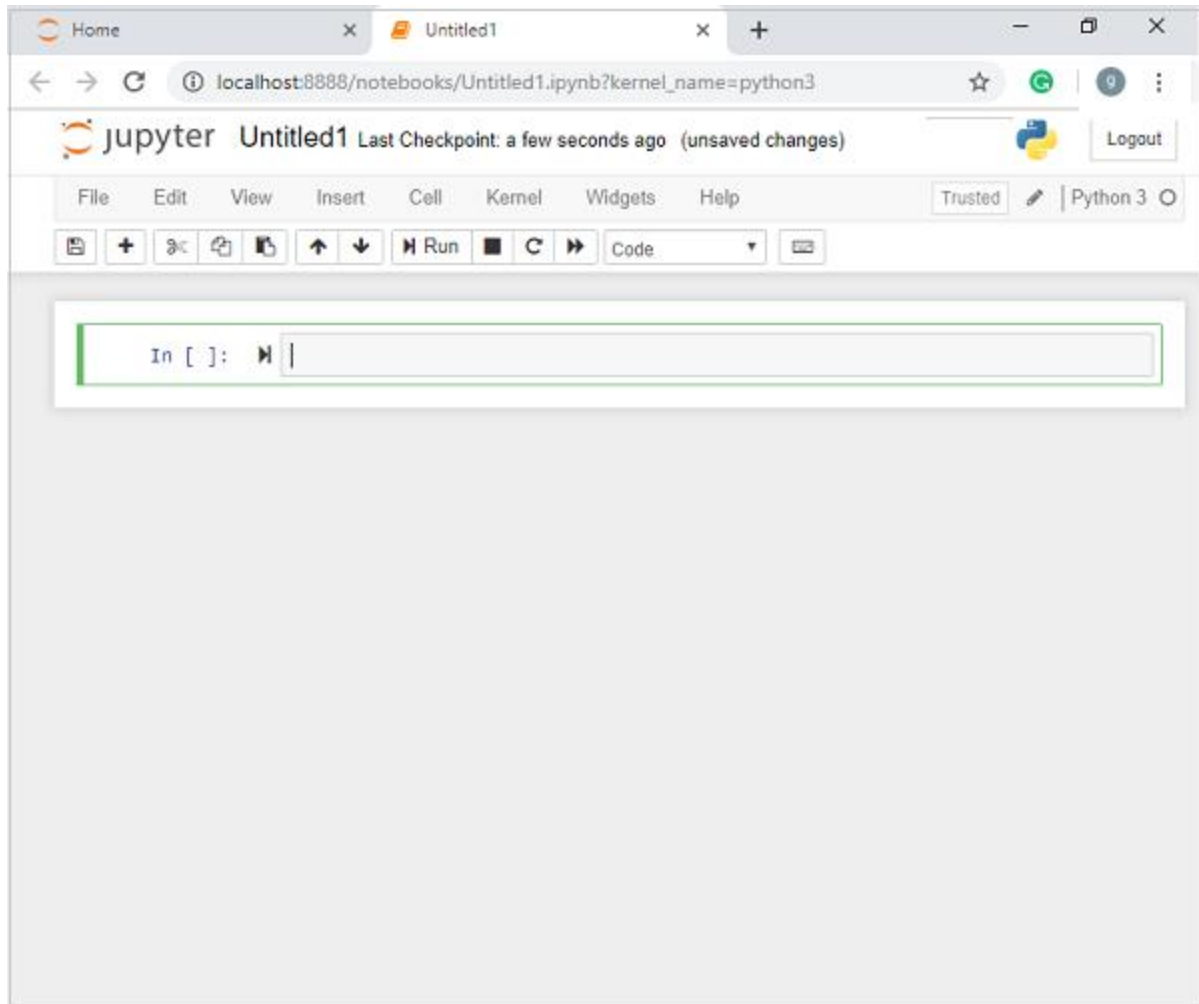
```
C:\>jupyter notebook
```

After pressing enter, it will start a notebook server at localhost: 8888 of your computer. It is shown in the following screenshot –



Now, after clicking the new tab, you will get a list of options. Select Python 3 and it will take you to the new notebook to start working in it. You will get a glimpse of it in the following screenshots –





On the other hand, if you are using standard Python distribution then jupyter notebook can be installed using popular python package installer, pip.

```
pip install jupyter
```

Types of Cells in Jupyter Notebook

The following are the three types of cells in a jupyter notebook –

Code cells – As the name suggests, we can use these cells to write code. After writing the code/content, it will send it to the kernel that is associated with the notebook.

Markdown cells – We can use these cells for notating the computation process. They can contain stuff like text, images, Latex equations, HTML tags, etc.

Raw cells – The text written in them is displayed as it is. These cells are basically used to add the text that we do not wish to be converted by the automatic conversion mechanism of jupyter notebook.

Methods for Machine Learning

There are various ML algorithms, techniques and methods that can be used to build models for solving real-life problems by using data. In this chapter, we are going to discuss such different kinds of methods.

Different Types of Methods

The following are various ML methods based on some broad categories –

Based on human supervision

In the learning process, some of the methods that are based on human supervision are as follows –

Supervised Learning

Supervised learning algorithms or methods are the most commonly used ML algorithms. This method or learning algorithm take the data sample i.e. the training data and its associated output i.e. labels or responses with each data samples during the training process.

The main objective of supervised learning algorithms is to learn an association between input data samples and corresponding outputs after performing multiple training data instances.

For example, we have

x: Input variables and

Y: Output variable

Now, apply an algorithm to learn the mapping function from the input to output as follows –

$$Y=f(x)$$

Now, the main objective would be to approximate the mapping function so well that even when we have new input data (x), we can easily predict the output variable (Y) for that new input data.

It is called supervised because the whole process of learning can be thought as it is being supervised by a teacher or supervisor. Examples of supervised machine learning algorithms includes **Decision tree, Random Forest, KNN, Logistic Regression** etc.

Based on the ML tasks, supervised learning algorithms can be divided into following two broad classes –

- Classification
- Regression

Classification

The key objective of classification-based tasks is to predict categorical output labels or responses for the given input data. The output will be based on what the model has learned in training phase. As we know that the categorical output responses means unordered and discrete values, hence each output response will belong to a specific class or category. We will discuss Classification and associated algorithms in detail in the upcoming chapters also.

Regression

The key objective of regression-based tasks is to predict output labels or responses which are continues numeric values, for the given input data. The output will be based on what the model has learned in its training phase. Basically, regression models use the input data features (independent variables) and their corresponding continuous numeric output values (dependent or outcome variables) to learn specific association between inputs and corresponding outputs. We will discuss regression and associated algorithms in detail in further chapters also.

Unsupervised Learning

As the name suggests, it is opposite to supervised ML methods or algorithms which means in unsupervised machine learning algorithms we do not have any supervisor to provide any sort of guidance. Unsupervised learning algorithms are handy in the scenario in which we do not have the liberty, like in supervised learning algorithms, of having pre-labeled training data and we want to extract useful pattern from input data.

For example, it can be understood as follows –

Suppose we have –

x: Input variables, then there would be no corresponding output variable and the algorithms need to discover the interesting pattern in data for learning.

Examples of unsupervised machine learning algorithms includes K-means clustering, **K-nearest neighbors** etc.

Based on the ML tasks, unsupervised learning algorithms can be divided into following broad classes –

- Clustering
- Association
- Dimensionality Reduction

Clustering

Clustering methods are one of the most useful unsupervised ML methods. These algorithms used to find similarity as well as relationship patterns among data samples and then cluster those samples into groups having similarity based on features. The real-world example of clustering is to group the customers by their purchasing behavior.

Association

Another useful unsupervised ML method is **Association** which is used to analyze large dataset to find patterns which further represents the interesting relationships between various items. It is also termed as **Association Rule Mining** or **Market basket analysis** which is mainly used to analyze customer shopping patterns.

Dimensionality Reduction

This unsupervised ML method is used to reduce the number of feature variables for each data sample by selecting set of principal or representative features. A question arises here is that why we need to reduce the dimensionality? The reason behind is the problem of feature space complexity which arises when we start analyzing and extracting millions of features from data samples. This problem generally refers to “curse of dimensionality”. PCA (Principal Component Analysis), K-nearest neighbors and discriminant analysis are some of the popular algorithms for this purpose.

Anomaly Detection

This unsupervised ML method is used to find out the occurrences of rare events or observations that generally do not occur. By using the learned knowledge, anomaly detection methods would be able to differentiate between anomalous or a normal data point. Some of the unsupervised algorithms like clustering, KNN can detect anomalies based on the data and its features.

Semi-supervised Learning

Such kind of algorithms or methods are neither fully supervised nor fully unsupervised. They basically fall between the two i.e. supervised and unsupervised learning methods. These kinds of algorithms generally use small supervised learning component i.e. small amount of pre-labeled annotated data and large unsupervised learning component i.e. lots of unlabeled data for training. We can follow any of the following approaches for implementing semi-supervised learning methods –

- The first and simple approach is to build the supervised model based on small amount of labeled and annotated data and then build the unsupervised model by applying the same to the large amounts of unlabeled data to get more labeled samples. Now, train the model on them and repeat the process.
- The second approach needs some extra efforts. In this approach, we can first use the unsupervised methods to cluster similar data samples, annotate these groups and then use a combination of this information to train the model.

Reinforcement Learning

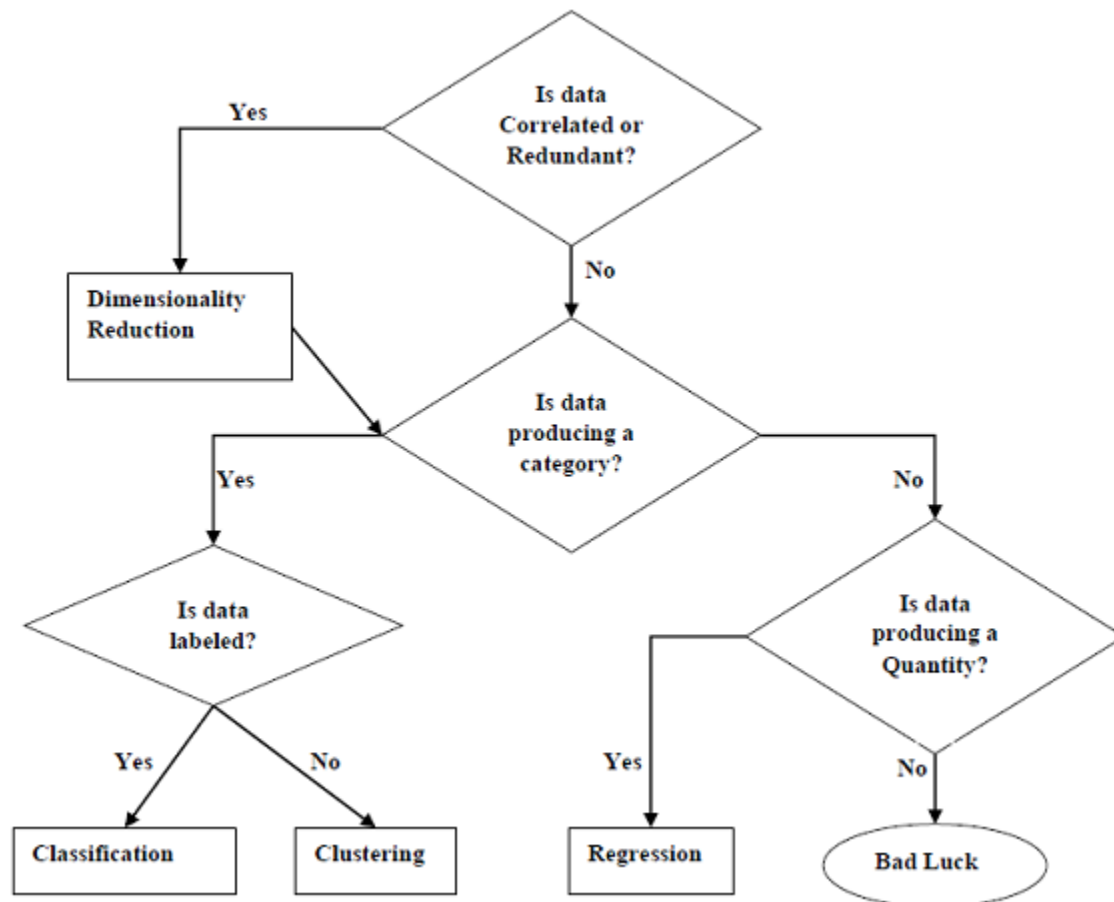
These methods are different from previously studied methods and very rarely used also. In this kind of learning algorithms, there would be an agent that we want to train over a period of time so that it can interact with a specific environment. The agent will follow a set of strategies for interacting with the environment and then after observing the environment it will take actions regards the current state of the environment. The following are the main steps of reinforcement learning methods –

- **Step 1** – First, we need to prepare an agent with some initial set of strategies.
- **Step 2** – Then observe the environment and its current state.

- **Step 3** – Next, select the optimal policy regards the current state of the environment and perform important action.
- **Step 4** – Now, the agent can get corresponding reward or penalty as per accordance with the action taken by it in previous step.
- **Step 5** – Now, we can update the strategies if it is required so.
- **Step 6** – At last, repeat steps 2-5 until the agent got to learn and adopt the optimal policies.

Tasks Suited for Machine Learning

The following diagram shows what type of task is appropriate for various ML problems –



Based on learning ability

In the learning process, the following are some methods that are based on learning ability –

Batch Learning

In many cases, we have end-to-end Machine Learning systems in which we need to train the model in one go by using whole available training data. Such kind of learning method or algorithm is called **Batch or Offline learning**. It is called Batch or Offline learning

because it is a one-time procedure and the model will be trained with data in one single batch. The following are the main steps of Batch learning methods –

- **Step 1** – First, we need to collect all the training data for start training the model.
- **Step 2** – Now, start the training of model by providing whole training data in one go.
- **Step 3** – Next, stop learning/training process once you got satisfactory results/performance.
- **Step 4** – Finally, deploy this trained model into production. Here, it will predict the output for new data sample.

Online Learning

It is completely opposite to the batch or offline learning methods. In these learning methods, the training data is supplied in multiple incremental batches, called mini-batches, to the algorithm. Followings are the main steps of Online learning methods –

- **Step 1** – First, we need to collect all the training data for starting training of the model.
- **Step 2** – Now, start the training of model by providing a mini-batch of training data to the algorithm.
- **Step 3** – Next, we need to provide the mini-batches of training data in multiple increments to the algorithm.
- **Step 4** – As it will not stop like batch learning hence after providing whole training data in mini-batches, provide new data samples also to it.
- **Step 5** – Finally, it will keep learning over a period of time based on the new data samples.

Based on Generalization Approach

In the learning process, followings are some methods that are based on generalization approaches –

Instance based Learning

Instance based learning method is one of the useful methods that build the ML models by doing generalization based on the input data. It is opposite to the previously studied learning methods in the way that this kind of learning involves ML systems as well as methods that uses the raw data points themselves to draw the outcomes for newer data samples without building an explicit model on training data.

In simple words, instance-based learning basically starts working by looking at the input data points and then using a similarity metric, it will generalize and predict the new data points.

Model based Learning

In Model based learning methods, an iterative process takes place on the ML models that are built based on various model parameters, called hyperparameters and in which input data is used to extract the features. In this learning, hyperparameters are optimized based

on various model validation techniques. That is why we can say that Model based learning methods uses more traditional ML approach towards generalization.

Data Loading for ML Projects

Suppose if you want to start a ML project then what is the first and most important thing you would require? It is the data that we need to load for starting any of the ML project. With respect to data, the most common format of data for ML projects is CSV (comma-separated values).

Basically, CSV is a simple file format which is used to store tabular data (number and text) such as a spreadsheet in plain text. In Python, we can load CSV data into with different ways but before loading CSV data we must have to take care about some considerations.

Consideration While Loading CSV data

CSV data format is the most common format for ML data, but we need to take care about following major considerations while loading the same into our ML projects –

File Header

In CSV data files, the header contains the information for each field. We must use the same delimiter for the header file and for data file because it is the header file that specifies how should data fields be interpreted.

The following are the two cases related to CSV file header which must be considered –

- **Case-I: When Data file is having a file header** – It will automatically assign the names to each column of data if data file is having a file header.
- **Case-II: When Data file is not having a file header** – We need to assign the names to each column of data manually if data file is not having a file header.

In both the cases, we must need to specify explicitly whether our CSV file contains header or not.

Comments

Comments in any data file are having their significance. In CSV data file, comments are indicated by a hash (#) at the start of the line. We need to consider comments while loading CSV data into ML projects because if we are having comments in the file then we may need to indicate, depends upon the method we choose for loading, whether to expect those comments or not.

Delimiter

In CSV data files, comma (,) character is the standard delimiter. The role of delimiter is to separate the values in the fields. It is important to consider the role of delimiter while uploading the CSV file into ML projects because we can also use a different delimiter such as a tab or white space. But in the case of using a different delimiter than standard one, we must have to specify it explicitly.

Quotes

In CSV data files, double quotation (“ ”) mark is the default quote character. It is important to consider the role of quotes while uploading the CSV file into ML projects because we can also use other quote character than double quotation mark. But in case of using a different quote character than standard one, we must have to specify it explicitly.

Methods to Load CSV Data File

While working with ML projects, the most crucial task is to load the data properly into it. The most common data format for ML projects is CSV and it comes in various flavors and varying difficulties to parse. In this section, we are going to discuss about three common approaches in Python to load CSV data file –

Load CSV with Python Standard Library

The first and most used approach to load CSV data file is the use of Python standard library which provides us a variety of built-in modules namely **csv module** and the `reader()` function. The following is an example of loading CSV data file with the help of it –

Example

In this example, we are using the iris flower data set which can be downloaded into our local directory. After loading the data file, we can convert it into **NumPy** array and use it for ML projects. Following is the Python script for loading CSV data file –

First, we need to import the csv module provided by Python standard library as follows –

```
import csv
```

Next, we need to import Numpy module for converting the loaded data into NumPy array.

```
import numpy as np
```

Now, provide the full path of the file, stored on our local directory, having the CSV data file –

```
path = r"c:\iris.csv"
```

Next, use the `csv.reader()` function to read data from CSV file –

```
with open(path, 'r') as f:
    reader = csv.reader(f, delimiter = ',')
    headers = next(reader)
    data = list(reader)
    data = np.array(data).astype(float)
```

We can print the names of the headers with the following line of script –

```
print(headers)
```

The following line of script will print the shape of the data i.e. number of rows & columns in the file –

```
print(data.shape)
```

Next script line will give the first three line of data file –

```
print(data[:3])
```

Output

```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
(150, 4)
[ [5.1  3.5  1.4  0.2]
  [4.9  3.   1.4  0.2]
  [4.7  3.2  1.3  0.2]
]
```

Load CSV with NumPy

Another approach to load CSV data file is NumPy and `numpy.loadtxt()` function. The following is an example of loading CSV data file with the help of it –

Example

In this example, we are using the Pima Indians Dataset having the data of diabetic patients. This dataset is a numeric dataset with no header. It can also be downloaded into our local directory. After loading the data file, we can convert it into NumPy array and use it for ML projects. The following is the Python script for loading CSV data file –

```
from numpy import loadtxt
path = r"C:\pima-indians-diabetes.csv"
datapath= open(path, 'r')
data = loadtxt(datapath, delimiter=",")
print(data.shape)
print(data[:3])
```

Output

```
(768, 9)
[ [ 6. 148.  72.  35.  0. 33.6  0.627  50.  1.]
  [ 1.  85.  66.  29.  0. 26.6  0.351  31.  0.]
  [ 8. 183.  64.  0.  0. 23.3  0.672  32.  1.]
]
```

Load CSV with Pandas

Another approach to load CSV data file is by **Pandas** and **`pandas.read_csv()`** function. This is the very flexible function that returns a **`pandas.DataFrame`** which can be used immediately for plotting. The following is an example of loading CSV data file with the help of it –

Example

Here, we will be implementing two Python scripts, first is with Iris data set having headers and another is by using the Pima Indians Dataset which is a numeric dataset with no header. Both the datasets can be downloaded into local directory.

Script-1

The following is the Python script for loading CSV data file using Pandas on Iris Data set –

```
from pandas import read_csv
path = r"C:\iris.csv"
data = read_csv(path)
print(data.shape)
print(data[:3])
```

Output:

```
(150, 4)
   sepal_length  sepal_width  petal_length  petal_width
0         5.1         3.5         1.4         0.2
1         4.9         3.0         1.4         0.2
2         4.7         3.2         1.3         0.2
```

Script-2

The following is the Python script for loading CSV data file, along with providing the headers names too, using Pandas on Pima Indians Diabetes dataset –

```
from pandas import read_csv
path = r"C:\pima-indians-diabetes.csv"
headernames = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=headernames)
print(data.shape)
print(data[:3])
```

Output

```
(768, 9)
   preg  plas  pres  skin  test  mass  pedi  age  class
0     6   148    72    35     0   33.6  0.627   50     1
1     1     85    66    29     0   26.6  0.351   31     0
2     8   183    64     0     0   23.3  0.672   32     1
```

The difference between above used three approaches for loading CSV data file can easily be understood with the help of given examples.

Understanding Data with Statistics

Introduction

While working with machine learning projects, usually we ignore two most important parts called **mathematics** and **data**. It is because, we know that ML is a data driven approach and our ML model will produce only as good or as bad results as the data we provided to it.

In the previous chapter, we discussed how we can upload CSV data into our ML project, but it would be good to understand the data before uploading it. We can understand the data by two ways, with statistics and with visualization.

In this chapter, with the help of following Python recipes, we are going to understand ML data with statistics.

Looking at Raw Data

The very first recipe is for looking at your raw data. It is important to look at raw data because the insight we will get after looking at raw data will boost our chances to better pre-processing as well as handling of data for ML projects.

Following is a Python script implemented by using head() function of Pandas DataFrame on Pima Indians diabetes dataset to look at the first 50 rows to get better understanding of it –

Example

```
from pandas import read_csv
path = r"C:\pima-indians-diabetes.csv"
headernames = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=headernames)
print(data.head(50))
```

Output

	preg	plas	pres	skin	test	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1

12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1
17	7	107	74	0	0	29.6	0.254	31	1
18	1	103	30	38	83	43.3	0.183	33	0
19	1	115	70	30	96	34.6	0.529	32	1
20	3	126	88	41	235	39.3	0.704	27	0
21	8	99	84	0	0	35.4	0.388	50	0
22	7	196	90	0	0	39.8	0.451	41	1
23	9	119	80	35	0	29.0	0.263	29	1
24	11	143	94	33	146	36.6	0.254	51	1
25	10	125	70	26	115	31.1	0.205	41	1
26	7	147	76	0	0	39.4	0.257	43	1
27	1	97	66	15	140	23.2	0.487	22	0
28	13	145	82	19	110	22.2	0.245	57	0
29	5	117	92	0	0	34.1	0.337	38	0
30	5	109	75	26	0	36.0	0.546	60	0
31	3	158	76	36	245	31.6	0.851	28	1
32	3	88	58	11	54	24.8	0.267	22	0
33	6	92	92	0	0	19.9	0.188	28	0
34	10	122	78	31	0	27.6	0.512	45	0
35	4	103	60	33	192	24.0	0.966	33	0
36	11	138	76	0	0	33.2	0.420	35	0
37	9	102	76	37	0	32.9	0.665	46	1
38	2	90	68	42	0	38.2	0.503	27	1
39	4	111	72	47	207	37.1	1.390	56	1
40	3	180	64	25	70	34.0	0.271	26	0
41	7	133	84	0	0	40.2	0.696	37	0
42	7	106	92	18	0	22.7	0.235	48	0
43	9	171	110	24	240	45.4	0.721	54	1
44	7	159	64	0	0	27.4	0.294	40	0
45	0	180	66	39	0	42.0	1.893	25	1
46	1	146	56	0	0	29.7	0.564	29	0
47	2	71	70	27	0	28.0	0.586	22	0
48	7	103	66	32	0	39.1	0.344	31	1
49	7	105	0	0	0	0.0	0.305	24	0

We can observe from the above output that first column gives the row number which can be very useful for referencing a specific observation.

Checking Dimensions of Data

It is always a good practice to know how much data, in terms of rows and columns, we are having for our ML project. The reasons behind are –

- Suppose if we have too many rows and columns then it would take long time to run the algorithm and train the model.

- Suppose if we have too less rows and columns then it we would not have enough data to well train the model.

Following is a Python script implemented by printing the shape property on Pandas Data Frame. We are going to implement it on iris data set for getting the total number of rows and columns in it.

Example

```
from pandas import read_csv
path = r"C:\iris.csv"
data = read_csv(path)
print(data.shape)
```

Output

(150, 4)

We can easily observe from the output that iris data set, we are going to use, is having 150 rows and 4 columns.

Getting Each Attribute's Data Type

It is another good practice to know data type of each attribute. The reason behind is that, as per to the requirement, sometimes we may need to convert one data type to another. For example, we may need to convert string into floating point or int for representing categorial or ordinal values. We can have an idea about the attribute's data type by looking at the raw data, but another way is to use dtypes property of Pandas DataFrame. With the help of dtypes property we can categorize each attributes data type. It can be understood with the help of following Python script –

Example

```
from pandas import read_csv
path = r"C:\iris.csv"
data = read_csv(path)
print(data.dtypes)
```

Output

```
sepal_length float64
sepal_width  float64
petal_length float64
petal_width  float64
dtype: object
```

From the above output, we can easily get the datatypes of each attribute.

Statistical Summary of Data

We have discussed Python recipe to get the shape i.e. number of rows and columns, of data but many times we need to review the summaries out of that shape of data. It can

be done with the help of describe() function of Pandas DataFrame that further provide the following 8 statistical properties of each & every data attribute –

- Count
- Mean
- Standard Deviation
- Minimum Value
- Maximum value
- 25%
- Median i.e. 50%
- 75%

Example

```
from pandas import read_csv
from pandas import set_option
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=names)
set_option('display.width', 100)
set_option('precision', 2)
print(data.shape)
print(data.describe())
```

Output

(768, 9)

	preg	plas	pres	skin	test	mass	pedi	age	class
count	768.00	768.00	768.00	768.00	768.00	768.00	768.00	768.00	768.00
mean	3.85	120.89	69.11	20.54	79.80	31.99	0.47	33.24	0.35
std	3.37	31.97	19.36	15.95	115.24	7.88	0.33	11.76	0.48
min	0.00	0.00	0.00	0.00	0.00	0.00	0.08	21.00	0.00
25%	1.00	99.00	62.00	0.00	0.00	27.30	0.24	24.00	0.00
50%	3.00	117.00	72.00	23.00	30.50	32.00	0.37	29.00	0.00
75%	6.00	140.25	80.00	32.00	127.25	36.60	0.63	41.00	1.00
max	17.00	199.00	122.00	99.00	846.00	67.10	2.42	81.00	1.00

From the above output, we can observe the statistical summary of the data of Pima Indian Diabetes dataset along with shape of data.

Reviewing Class Distribution

Class distribution statistics is useful in classification problems where we need to know the balance of class values. It is important to know class value distribution because if we have highly imbalanced class distribution i.e. one class is having lots more observations than other class, then it may need special handling at data preparation stage of our ML project. We can easily get class distribution in Python with the help of Pandas DataFrame.

Example

```
from pandas import read_csv
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=names)
count_class = data.groupby('class').size()
print(count_class)
```

Output

```
Class
0 500
1 268
dtype: int64
```

From the above output, it can be clearly seen that the number of observations with class 0 are almost double than number of observations with class 1.

Reviewing Correlation between Attributes

The relationship between two variables is called correlation. In statistics, the most common method for calculating correlation is Pearson's Correlation Coefficient. It can have three values as follows –

- **Coefficient value = 1** – It represents full **positive** correlation between variables.
- **Coefficient value = -1** – It represents full **negative** correlation between variables.
- **Coefficient value = 0** – It represents **no** correlation at all between variables.

It is always good for us to review the pairwise correlations of the attributes in our dataset before using it into ML project because some machine learning algorithms such as linear regression and logistic regression will perform poorly if we have highly correlated attributes. In Python, we can easily calculate a correlation matrix of dataset attributes with the help of `corr()` function on Pandas DataFrame.

Example

```
from pandas import read_csv
from pandas import set_option
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=names)
set_option('display.width', 100)
set_option('precision', 2)
correlations = data.corr(method='pearson')
print(correlations)
```

Output

	preg	plas	pres	skin	test	mass	pedi	age	class
preg	1.00	0.13	0.14	-0.08	-0.07	0.02	-0.03	0.54	0.22
plas	0.13	1.00	0.15	0.06	0.33	0.22	0.14	0.26	0.47
pres	0.14	0.15	1.00	0.21	0.09	0.28	0.04	0.24	0.07

skin	-0.08	0.06	0.21	1.00	0.44	0.39	0.18	-0.11	0.07
test	-0.07	0.33	0.09	0.44	1.00	0.20	0.19	-0.04	0.13
mass	0.02	0.22	0.28	0.39	0.20	1.00	0.14	0.04	0.29
pedi	-0.03	0.14	0.04	0.18	0.19	0.14	1.00	0.03	0.17
age	0.54	0.26	0.24	-0.11	-0.04	0.04	0.03	1.00	0.24
class	0.22	0.47	0.07	0.07	0.13	0.29	0.17	0.24	1.00

The matrix in above output gives the correlation between all the pairs of the attribute in dataset.

Reviewing Skew of Attribute Distribution

Skewness may be defined as the distribution that is assumed to be Gaussian but appears distorted or shifted in one direction or another, or either to the left or right. Reviewing the skewness of attributes is one of the important tasks due to following reasons –

- Presence of skewness in data requires the correction at data preparation stage so that we can get more accuracy from our model.
- Most of the ML algorithms assumes that data has a Gaussian distribution i.e. either normal or bell curved data.

In Python, we can easily calculate the skew of each attribute by using `skew()` function on Pandas DataFrame.

Example

```
from pandas import read_csv
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=names)
print(data.skew())
```

Output

```
preg 0.90
plas 0.17
pres -1.84
skin 0.11
test 2.27
mass -0.43
pedi 1.92
age 1.13
class 0.64
dtype: float64
```

From the above output, positive or negative skew can be observed. If the value is closer to zero, then it shows less skew.

Preparing Data

Introduction

Machine Learning algorithms are completely dependent on data because it is the most crucial aspect that makes model training possible. On the other hand, if we won't be able to make sense out of that data, before feeding it to ML algorithms, a machine will be useless. In simple words, we always need to feed right data i.e. the data in correct scale, format and containing meaningful features, for the problem we want machine to solve.

This makes data preparation the most important step in ML process. Data preparation may be defined as the procedure that makes our dataset more appropriate for ML process.

Why Data Pre-processing?

After selecting the raw data for ML training, the most important task is data pre-processing. In broad sense, data preprocessing will convert the selected data into a form we can work with or can feed to ML algorithms. We always need to preprocess our data so that it can be as per the expectation of machine learning algorithm.

Data Pre-processing Techniques

We have the following data preprocessing techniques that can be applied on data set to produce data for ML algorithms –

Scaling

Most probably our dataset comprises of the attributes with varying scale, but we cannot provide such data to ML algorithm hence it requires rescaling. Data rescaling makes sure that attributes are at same scale. Generally, attributes are rescaled into the range of 0 and 1. ML algorithms like gradient descent and k-Nearest Neighbors requires scaled data. We can rescale the data with the help of MinMaxScaler class of scikit-learn Python library.

Example

In this example we will rescale the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded (as done in the previous chapters) and then with the help of MinMaxScaler class, it will be rescaled in the range of 0 and 1.

The first few lines of the following script are same as we have written in previous chapters while loading CSV data.

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn import preprocessing
path = r'C:\pima-indians-diabetes.csv'
```

```
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use MinMaxScaler class to rescale the data in the range of 0 and 1.

```
data_scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
data_rescaled = data_scaler.fit_transform(array)
```

We can also summarize the data for output as per our choice. Here, we are setting the precision to 1 and showing the first 10 rows in the output.

```
set_printoptions(precision=1)
print ("\nScaled data:\n", data_rescaled[0:10])
```

Output

Scaled data:

```
[
  [0.4 0.7 0.6 0.4 0. 0.5 0.2 0.5 1. ]
  [0.1 0.4 0.5 0.3 0. 0.4 0.1 0.2 0. ]
  [0.5 0.9 0.5 0. 0. 0.3 0.3 0.2 1. ]
  [0.1 0.4 0.5 0.2 0.1 0.4 0. 0. 0. ]
  [0. 0.7 0.3 0.4 0.2 0.6 0.9 0.2 1. ]
  [0.3 0.6 0.6 0. 0. 0.4 0.1 0.2 0. ]
  [0.2 0.4 0.4 0.3 0.1 0.5 0.1 0.1 1. ]
  [0.6 0.6 0. 0. 0. 0.5 0. 0.1 0. ]
  [0.1 1. 0.6 0.5 0.6 0.5 0. 0.5 1. ]
  [0.5 0.6 0.8 0. 0. 0. 0.1 0.6 1. ]
]
```

From the above output, all the data got rescaled into the range of 0 and 1.

Normalization

Another useful data preprocessing technique is Normalization. This is used to rescale each row of data to have a length of 1. It is mainly useful in Sparse dataset where we have lots of zeros. We can rescale the data with the help of Normalizer class of scikit-learn Python library.

Types of Normalization

In machine learning, there are two types of normalization preprocessing techniques as follows –

L1 Normalization

It may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the absolute values will always be up to 1. It is also called Least Absolute Deviations.

Example

In this example, we use L1 Normalize technique to normalize the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded and then with the help of Normalizer class it will be normalized.

The first few lines of following script are same as we have written in previous chapters while loading CSV data.

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn.preprocessing import Normalizer
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use Normalizer class with L1 to normalize the data.

```
Data_normalizer = Normalizer(norm='l1').fit(array)
Data_normalized = Data_normalizer.transform(array)
```

We can also summarize the data for output as per our choice. Here, we are setting the precision to 2 and showing the first 3 rows in the output.

```
set_printoptions(precision=2)
print ("\nNormalized data:\n", Data_normalized [0:3])
```

Output

Normalized data:

```
[
  [0.02 0.43 0.21 0.1  0.  0.1  0.  0.14 0. ]
  [0.  0.36 0.28 0.12 0.  0.11 0.  0.13 0. ]
  [0.03 0.59 0.21 0.  0.  0.07 0.  0.1  0. ]
]
```

L2 Normalization

It may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the squares will always be up to 1. It is also called least squares.

Example

In this example, we use L2 Normalization technique to normalize the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded (as done in previous chapters) and then with the help of Normalizer class it will be normalized.

The first few lines of following script are same as we have written in previous chapters while loading CSV data.

```
from pandas import read_csv
from numpy import set_printoptions
```

```
from sklearn.preprocessing import Normalizer
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use Normalizer class with L1 to normalize the data.

```
Data_normalizer = Normalizer(norm='l2').fit(array)
Data_normalized = Data_normalizer.transform(array)
```

We can also summarize the data for output as per our choice. Here, we are setting the precision to 2 and showing the first 3 rows in the output.

```
set_printoptions(precision=2)
print ("\nNormalized data:\n", Data_normalized [0:3])
```

Output

Normalized data:

```
[
  [0.03 0.83 0.4  0.2  0. 0.19 0. 0.28 0.01]
  [0.01 0.72 0.56 0.24 0. 0.22 0. 0.26 0. ]
  [0.04 0.92 0.32 0.  0. 0.12 0. 0.16 0.01]
]
```

Binarization

As the name suggests, this is the technique with the help of which we can make our data binary. We can use a binary threshold for making our data binary. The values above that threshold value will be converted to 1 and below that threshold will be converted to 0. For example, if we choose threshold value = 0.5, then the dataset value above it will become 1 and below this will become 0. That is why we can call it **binarizing** the data or **thresholding** the data. This technique is useful when we have probabilities in our dataset and want to convert them into crisp values.

We can binarize the data with the help of Binarizer class of scikit-learn Python library.

Example

In this example, we will rescale the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded and then with the help of Binarizer class it will be converted into binary values i.e. 0 and 1 depending upon the threshold value. We are taking 0.5 as threshold value.

The first few lines of following script are same as we have written in previous chapters while loading CSV data.

```
from pandas import read_csv
from sklearn.preprocessing import Binarizer
path = r'C:\pima-indians-diabetes.csv'
```



```
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use Binarize class to convert the data into binary values.

```
binarizer = Binarizer(threshold=0.5).fit(array)
Data_binarized = binarizer.transform(array)
```

Here, we are showing the first 5 rows in the output.

```
print ("\nBinary data:\n", Data_binarized [0:5])
```

Output

Binary data:

```
[
  [1. 1. 1. 1. 0. 1. 1. 1. 1.]
  [1. 1. 1. 1. 0. 1. 0. 1. 0.]
  [1. 1. 1. 0. 0. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1. 1. 0. 1. 0.]
  [0. 1. 1. 1. 1. 1. 1. 1. 1.]
]
```

Standardization

Another useful data preprocessing technique which is basically used to transform the data attributes with a Gaussian distribution. It differs the mean and SD (Standard Deviation) to a standard Gaussian distribution with a mean of 0 and a SD of 1. This technique is useful in ML algorithms like linear regression, logistic regression that assumes a Gaussian distribution in input dataset and produce better results with rescaled data. We can standardize the data (mean = 0 and SD = 1) with the help of StandardScaler class of scikit-learn Python library.

Example

In this example, we will rescale the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded and then with the help of StandardScaler class it will be converted into Gaussian Distribution with mean = 0 and SD = 1.

The first few lines of following script are same as we have written in previous chapters while loading CSV data.

```
from sklearn.preprocessing import StandardScaler
from pandas import read_csv
from numpy import set_printoptions
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use StandardScaler class to rescale the data.

```
data_scaler = StandardScaler().fit(array)
data_rescaled = data_scaler.transform(array)
```

We can also summarize the data for output as per our choice. Here, we are setting the precision to 2 and showing the first 5 rows in the output.

```
set_printoptions(precision=2)
print ("\nRescaled data:\n", data_rescaled [0:5])
```

Output

Rescaled data:

```
[
  [ 0.64  0.85  0.15  0.91 -0.69  0.2   0.47  1.43  1.37]
  [-0.84 -1.12 -0.16  0.53 -0.69 -0.68 -0.37 -0.19 -0.73]
  [ 1.23  1.94 -0.26 -1.29 -0.69 -1.1   0.6  -0.11  1.37]
  [-0.84 -1.   -0.16  0.15  0.12 -0.49 -0.92 -1.04 -0.73]
  [-1.14  0.5  -1.5   0.91  0.77  1.41  5.48 -0.02  1.37]
]
```

Data Labeling

We discussed the importance of good data for ML algorithms as well as some techniques to pre-process the data before sending it to ML algorithms. One more aspect in this regard is data labeling. It is also very important to send the data to ML algorithms having proper labeling. For example, in case of classification problems, lot of labels in the form of words, numbers etc. are there on the data.

What is Label Encoding?

Most of the sklearn functions expect that the data with number labels rather than word labels. Hence, we need to convert such labels into number labels. This process is called label encoding. We can perform label encoding of data with the help of LabelEncoder() function of scikit-learn Python library.

Example

In the following example, Python script will perform the label encoding.

First, import the required Python libraries as follows –

```
import numpy as np
from sklearn import preprocessing
```

Now, we need to provide the input labels as follows –

```
input_labels = ['red','black','red','green','black','yellow','white']
```

The next line of code will create the label encoder and train it.

```
encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)
```

The next lines of script will check the performance by encoding the random ordered list –

```
test_labels = ['green','red','black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
print("Encoded values =", list(encoded_values))
encoded_values = [3,0,4,1]
decoded_list = encoder.inverse_transform(encoded_values)
```

We can get the list of encoded values with the help of following python script –

```
print("\nEncoded values =", encoded_values)
print("\nDecoded labels =", list(decoded_list))
```

Output

```
Labels = ['green', 'red', 'black']
Encoded values = [1, 2, 0]
Encoded values = [3, 0, 4, 1]
Decoded labels = ['white', 'black', 'yellow', 'green']
```