MACHINE LEARNING

(Employee Attrition)

Summer Internship Report Submitted in partial

fulfillment of the requirement for undergraduate degree

of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By

MARRU RAHUL

221710304035

Under the Guidance of

Assistant Professor

GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM HYDERABAD BENGALURU

Department Of Computer Science and Engineering
GITAM School of Technology

GITAM (Deemed to be
University) Hyderabad-502329

DECLARATION

I submit this industrial training work entitled **“EMPLOYEE ATTRITION Classification**” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science and Engineering**”. I declare that it was carried out independently by me under the guidance of **Mr.**, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: Hyderabad　　Name: MARRU RAHUL

Date: 12-07-2020　　Student RollNo:221710304035

CERTIFICATE

This is to certify that the Industrial Training Report entitled **“EMPLOYEE ATTRITION Classification”** is being submitted by MARRU RAHUL (221710304035) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science And Engineering** at GITAM (Deemed to Be University), Hyderabad during the academic year 2019-20

It is faithful record work carried out by her at the **Computer Science And Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Dr. S.Phani Kumar

Assistant Professor Professor and HOD

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr. N. Siva Prasad,** Pro Vice Chancellor, GITAM Hyderabad and **Dr. N.Seetharamaiah,** Principal, GITAM Hyderabad

I would like to thank respected **S. Phani Kumar,** Head of the Department of Electronics and Communication Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mr.** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

MARRU RAHUL

221710304035

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on data set.

To classify “ Employee Attrition ” is the main motive of this project. Real Dataset is collected from website. Data includes , satisfaction level ,last evaluation, number of projects, average monthly hours, years at company, work accident, left,promotion_last_5years, department ,salary by following above given data we finding the complete job profile of the person. I have adapted the view point of looking at features of the dataset, for deep understanding of the problem. Employee attrition reveals a company's internal power and weaknesses. By this data and by satisfying the above given data we know the sustainable growth. Judges will look for evidence of how employees are engaged in strategic goals employees data and their experience in company and the level of the person in the company and by the graph and the plots we can prove it graphically and we can then control employees working in the company.

Table of Contents

List of Figures

1. MACHINE LEARNING

1.1. Introduction:

Over the past two decades Machine Learning has become one of the mainstays of information technology and with that, a rather central, albeit usually hidden, part of our life. With the ever increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress.

Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might not be completely known at design time. Machine learning methods can be used for on-the-job improvement of existing machine designs. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down. Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign. New knowledge about tasks is constantly being discovered by humans. Vocabulary changes. There is a constant stream of new events in the world. Continuing redesign of AI systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

1.2. Importance of Machine Learning:

Machine learning is a branch of artificial intelligence that aims at enabling machines to perform their jobs skillfully by using intelligent software. The statistical learning methods constitute the backbone of intelligent software that is used to develop machine intelligence. Because machine learning algorithms require data to learn, the discipline must have connection with the discipline of database. Similarly, there are familiar terms such as Knowledge Discovery from Data (KDD), data mining, and pattern recognition. One wonders how to view the big picture in which such connection is illustrated.

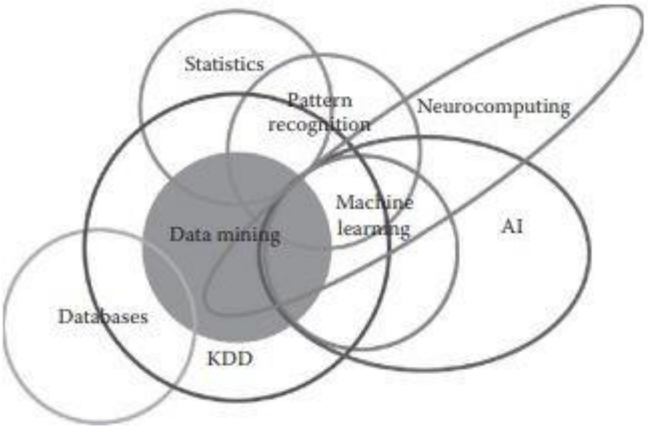


Fig 1.2 usage of Machine learning in different fields

There are some tasks that humans perform effortlessly or with some efforts, but we are unable to explain how we perform them. For example, we can recognize the speech of our friends without much difficulty. If we are asked how we recognize the voices, the answer is very difficult for us to explain. Because of the lack of understanding of such phenomenon (speech recognition in this case), we cannot craft algorithms for such scenarios. Machine learning algorithms are helpful in bridging this gap of understanding.

The idea is very simple. We are not targeting to understand the underlying processes that help us learn. We write computer programs that will make machines learn and enable them to perform tasks, such as prediction. The goal of learning is to construct a model that takes the input and produces the desired result. Sometimes, we can understand the model, whereas, at other times, it can also be like a black box for us, the working of which cannot be intuitively explained. The model can be considered as an approximation of the process we want machines to mimic. In such a situation, it is possible that we obtain errors for some input, but most of the time, the model provides correct answers. Hence, another measure of performance (besides performance of metrics of speed and memory usage) of a machine learning algorithm will be the accuracy of results.

1.3. Uses of Machine Learning:

Artificial Intelligence (AI) is everywhere. Possibility is that you are using it in one way or the other and you don't even know about it. One of the popular applications of AI is Machine Learning (ML), in which computers, software, and devices perform via cognition (very

similar to human brain). Herein, we share few examples of machine learning that we use everyday and perhaps have no idea that they are driven by ML. These are some the uses and applications of ML

i.Virtual Personal Assistants:

Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. All you need to do is activate them and ask "What is my schedule for today?", "What are the flights from Germany to London", or similar questions. For answering, your personal assistant looks out for the information, recalls your related queries, or send a command to other resources (like phone apps) to collect info. You can even instruct assistants for certain tasks like "Set an alarm for 6 AM next morning", "Remind me to visit Visa Office day after tomorrow".

Machine learning is an important part of these personal assistants as they collect and refine the information on the basis of your previous involvement with them. Later, this set of data utilized to render results that are tailored to your preferences.

Virtual Assistants are integrated to a variety of platforms. For example:

- Smart Speakers: Amazon Echo and Google Home
- Smartphones: Samsung Bixby on Samsung S8
- Mobile Apps: Google Allo

ii. Predictions while Commuting:

Traffic Predictions*:* We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. While this helps in preventing the traffic and does congestion analysis, the underlying problem is that there are less number of cars that are equipped with GPS. Machine learning in such scenarios helps to estimate the regions where congestion can be found on the basis of daily experiences.

Online Transportation Networks*:* When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours? The answer is machine learning. Jeff Schneider, the engineering lead at Uber ATC reveals in a an interview

that they use ML to define price surge hours by predicting the rider demand. In the entire cycle of the services, ML is playing a major role.

iii. Social Media Services:

From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits. Here are a few examples that you must be noticing, using, and loving in your social media accounts, without realizing that these wonderful features are nothing but the applications of ML.

- **People You May Know:** Machine learning works on a simple concept: understanding with experiences. Facebook continuously notices the friends that you connect with, the profiles that you visit very often, your interests, workplace, or a group that you share with someone etc. On the basis of continuous learning, a list of Facebook users are suggested that you can become friends with.

- **Face Recognition:** You upload a picture of you with a friend and Facebook instantly recognizes that friend. Facebook checks the poses and projections in the picture, notice the unique features, and then match them with the people in your friend list. The entire process at the backend is complicated and takes care of the precision factor but seems to be a simple application of ML at the front end.

- **Similar Pins:** Machine learning is the core element of Computer Vision, which is a technique to extract useful information from images and videos. Pinterest uses computer vision to identify the objects (or pins) in the images and recommend similar pins accordingly.

iv. Search Engine Result Refining:

Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results. If you open the top results and stay on the web page for long, the search engine assumes that the the results it displayed were in accordance to the query. Similarly, if you reach the second or third page of the search results but do not open any of the results, the search engine estimates that the results served did not match requirement. This way, the algorithms working at the backend improve the search results.

v. Product Recommendations:

You shopped for a product online few days back and then you keep receiving emails for shopping suggestions. If not this, then you might have noticed that the shopping website or the app recommends you some items that somehow matches with your taste. On the basis of your behaviour with the website/app, past purchases, items liked or added to cart, brand preferences etc., the product recommendations are made.

vi. Online Fraud Detection:Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. For example: Paypal is using ML for protection against money laundering. The company uses a set of tools that helps them to compare millions of transactions taking place and distinguish between legitimate or illegitimate transactions taking place between the buyers and sellers.

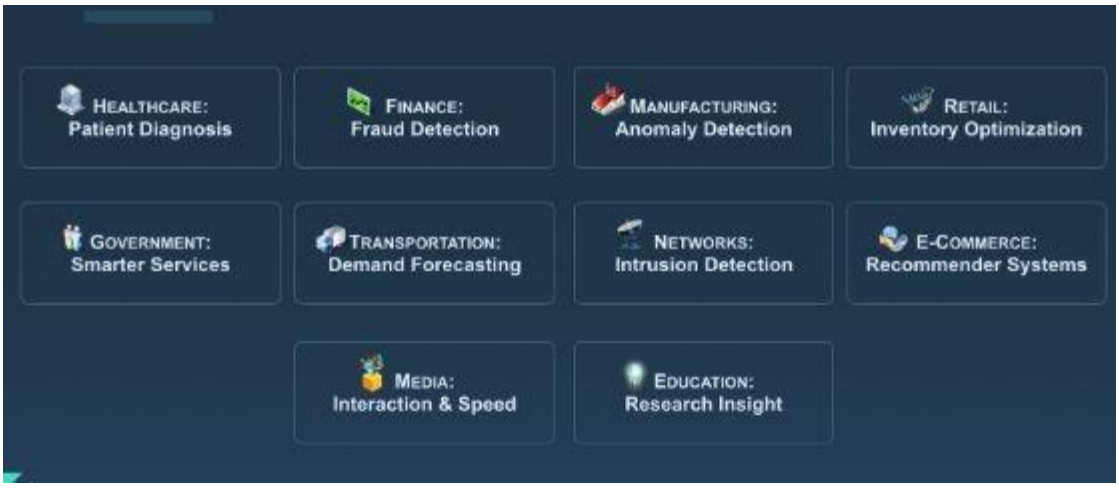


Fig 1.3 Uses of Machine learning

1.4. Types of Machine Learning:

There are 3 types of Machine learning which are widely used in todays world these Are:

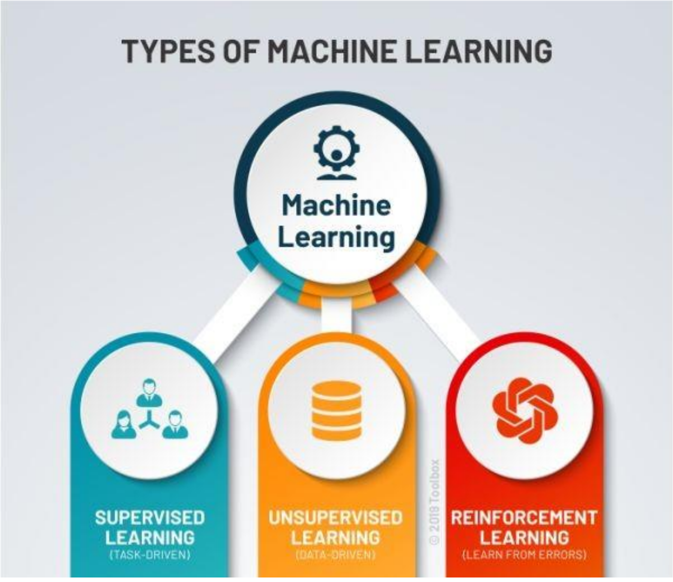


Fig 1.4 types of ML

1.4.1. Supervised Learning:

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem. The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

1.4.2. Unsupervised Learning:

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

1.4.3. Reinforcement Learning:

It directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’.Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result. In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

2. DEEP LEARNING

2.1 Deep Learning and It's Importance:

Deep learning algorithms run data through several "layers" of neural network algorithms, each of which passes a simplified representation of the data to the next layer.Most machine learning algorithms work well on datasets that have up to a few hundred features, or columns.

Basically deep learning is itself a subset of machine learning but in this case the machine learns in a way in which humans are supposed to learn. The structure of deep learning model is highly similar to a human brain with large number of neurons and nodes like neurons in human brain thus resulting in artificial neural network. In applying traditional machine learning algorithms we have to manually select input features from complex data set and then train them which becomes a very tedious job for ML scientist but in neural networks we don't have to manually select useful input features, there are various layers of neural networks for handling complexity of the data set and algorithm as well. In my recent project on human activity recognition , when we applied traditional machine learning algorithm like K-NN then we have to separately detect human and its activity also had to select impactful input parameters manually which became a very tedious task as data set was way too complex but the complexity dramatically reduced on applying artificial neural network, such is the power of deep learning. Yes it's correct that deep learning algorithms take lots of time for training sometimes even weeks as well but its execution on new data is so fast that its not even comparable with traditional ML algorithms. Deep learning has enabled Industrial Experts to overcome challenges which were impossible, a decades ago like Speech and Image recognition and Natural Language Processing. Majority of the Industries are currently depending on it , be it Journalism, Entertainment, Online Retail Store, Automobile, Banking and Finance, Healthcare, Manufacturing or even Digital Sector. Video recommendations,

Mail Services, Self Driving cars, Intelligent Chat bots, Voice Assistants are just trending achievements of Deep Learning.

Furthermore, Deep learning can most profoundly be considered as future of Artificial Intelligence due to constant rapid increase in amount of data as well as the gradual development in hardware field as well, resulting in better computational power.

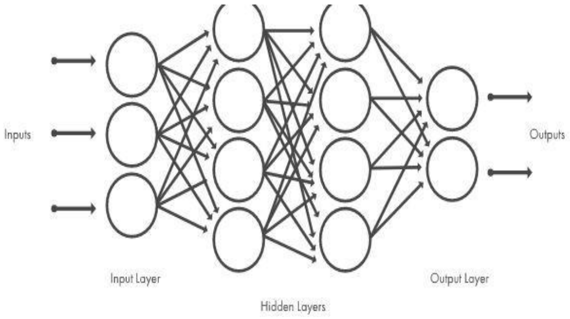


Fig 2.1 Deep Neural network

2.2 Uses of Deep Learning:

i. Translations:

Although automatic machine translation isn’t new, deep learning is helping enhance automatic translation of text by using stacked networks of neural networks and allowing translations from images.

ii. . Adding color to black-and-white images and videos:

It is used to be a very time-consuming process where humans had to add color to black-and-white images and videos by hand can now be automatically done with deep-learning models.

iii. . Language recognition:

Deep learning machines are beginning to differentiate dialects of a language. A machine decides that someone is speaking English and then engages an AI that is learning to tell the differences between dialects. Once the dialect is determined, another AI will step in that specializes in that particular dialect. All of this happens without involvement from a human.

iv. . Autonomous vehicles:

There's not just one AI model at work as an autonomous vehicle drives down the street. Some deep-learning models specialize in streets signs while others are trained to recognize pedestrians. As a car navigates down the road, it can be informed by up to millions of individual AI models that allow the car to act.

v. . Computer vision:

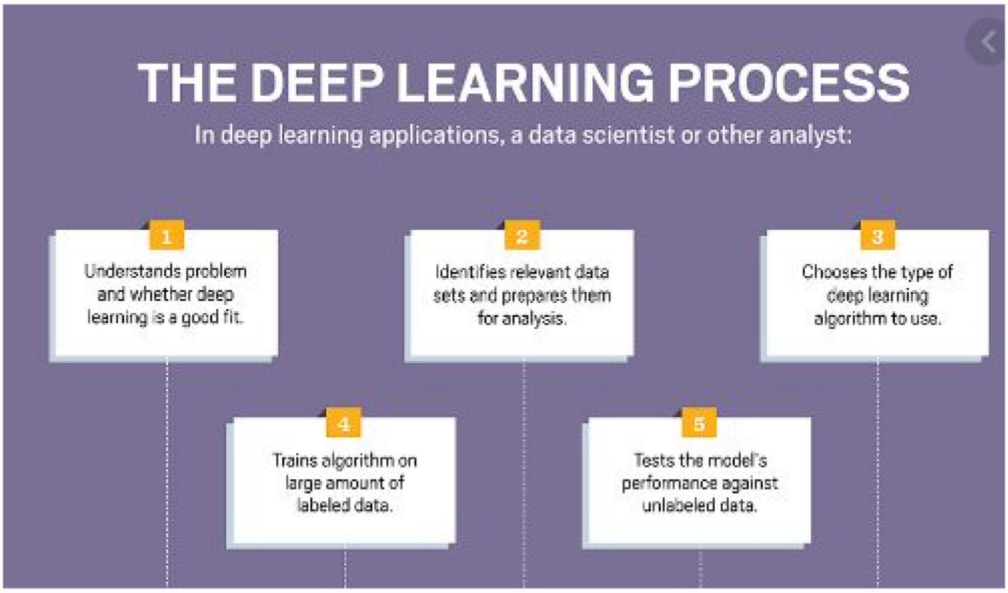
Deep learning has delivered super-human accuracy for image classification, object detection, image restoration and image segmentation—even handwritten digits can be recognized. Deep learning using enormous neural networks is teaching machines to automate the tasks performed by human visual systems.

vi.Text generation:

The machines learn the punctuation, grammar and style of a piece of text and can use the model it developed to automatically create entirely new text with the proper spelling, grammar and style of the example text. Everything from Shakespeare to Wikipedia entries have been created.

vii. Deep-learning robots:

Deep-learning applications for robots are plentiful and powerful from an impressive deep-learning system that can teach a robot just by observing the actions of a human completing a task to a housekeeping robot that’s provided with input from several other AIs in order to take action. Just like how a human brain processes input from past experiences, current input from senses and any additional data that is provided, deep-learning models will help robots execute tasks based on the input of many different AI opinions.



THE DEEP LEARNING PROCESS

In deep learning applications, a data scientist or other analyst:

1. Understands problem and whether deep learning is a good fit.
2. Identifies relevant data sets and prepares them for analysis.
3. Chooses the type of deep learning algorithm to use.
4. Trains algorithm on large amount of labeled data.
5. Tests the model's performance against unlabeled data.

Fig 2.2 The deep learning process

2.3 Relation between Data Mining, Machine Learning and Deep Learning:

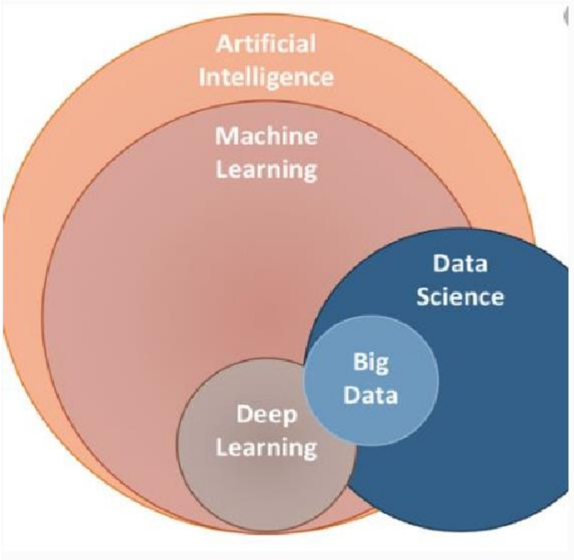


Fig 2.3 Relation between DM,ML,DL

The deep learning, data mining and machine learning share a foundation in data science, and there certainly is overlap between the two. Data mining can use machine learning algorithms to improve the accuracy and depth of analysis, and vice-versa; machine learning can use mined data as its foundation, refining the dataset to achieve better results.

You could also argue that data mining and machine learning are similar in that they both seek to address the question of how we can learn from data. However, the way in which they achieve this end, and their applications, form the basis of some significant differences.

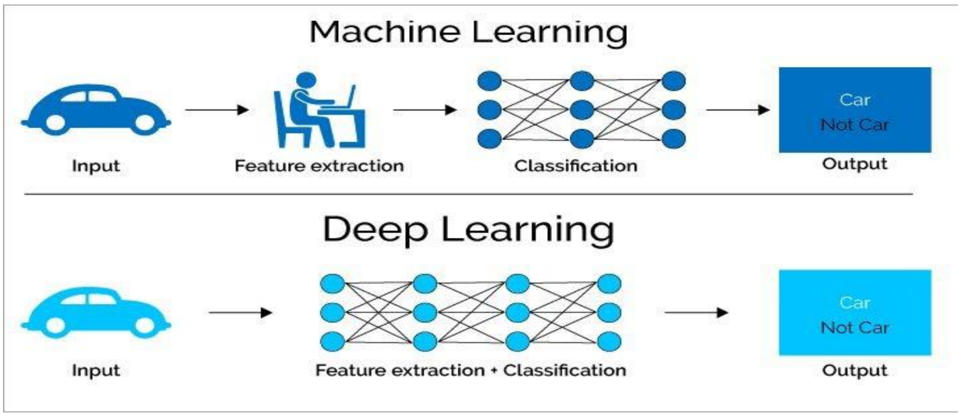


Fig 2.3 process in machine learning and deep learning

Machine Learning comprises of the ability of the machine to learn from trained data set and predict the outcome automatically. It is a subset of artificial intelligence.

Deep Learning is a subset of machine learning. It works in the same way on the machine just like how the human brain processes information. Like a brain can identify the patterns by comparing it with previously memorized patterns, deep learning also uses this concept.

Deep learning can automatically find out the attributes from raw data while machine learning selects these features manually which further needs processing. It also employs artificial neural networks with many hidden layers, big data, and high computer resources.

Data Mining is a process of discovering hidden patterns and rules from the existing data. It uses relatively simple rules such as association, correlation rules for the decision-making process, etc. Deep Learning is used for complex problem processing such as voice recognition etc. It uses Artificial Neural Networks with many hidden layers for processing. At times data mining also uses deep learning algorithms for processing the data.

3. PYTHON

3.1 Introduction:

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is dynamically typed and garbage-collected.It supports multiple programming paradigms,including structured , object-oriented,and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python is an interpreted, object-oriented, high-level programming language with

dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity

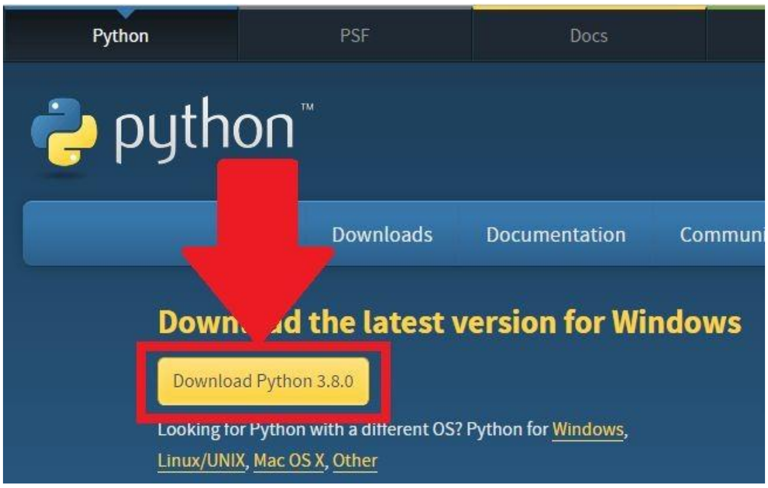
and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

3.2 Setup of Python:

- Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

3.2.1 Installation(using python IDLE):

- To start, go to python.org/downloads and then click on the button to download the latest version of Python
- We can download python IDLE in windows,mac and linux operating systems also.

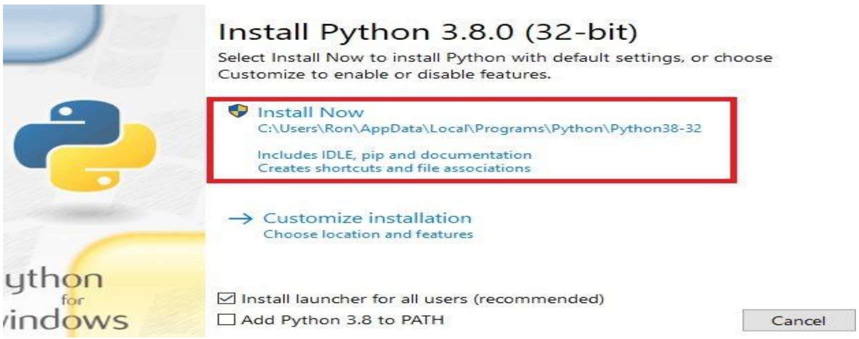


Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Figure 3.2.1 : Python download

- Run the .exe file that you just downloaded and start the installation of Python by clicking on Install Now
- We can give environmental variable i.e path after completion of downloading



Install Python 3.8.0 (32-bit)

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

Install Now
C:\Users\Ron\AppData\Local\Programs\Python\Python38-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.8 to PATH

Fig 3.2.1.1 python installation

- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

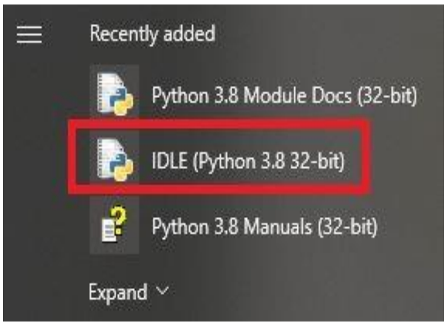


Fig 3.2.1.2 IDLE

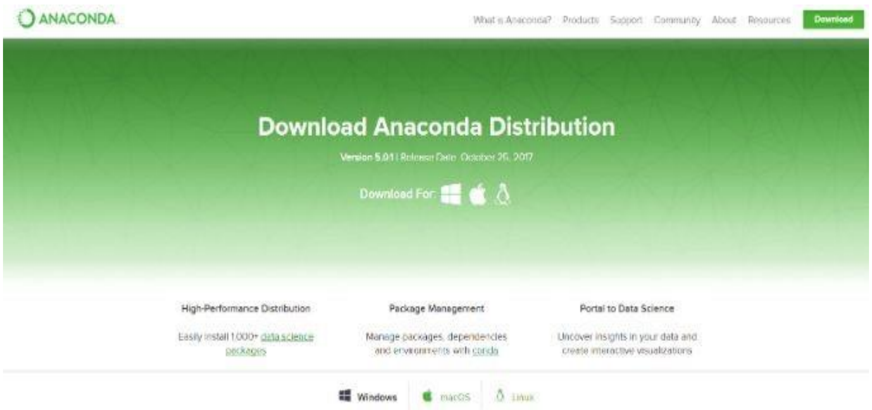
3.2.2 Python Installation using Anaconda:

- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.

- Conda is a package manager quickly installs and manages packages.

  Anaconda for Windows installation:

  i. Go to the following link: Anaconda.com/downloads



Download Anaconda Distribution

Version 5.0.1 | Release Date: October 26, 2017

Download For:

High-Performance Distribution

Easily install 1,000+ data science packages

Package Management

Manage packages, dependencies and environments with conda

Portal to Data Science

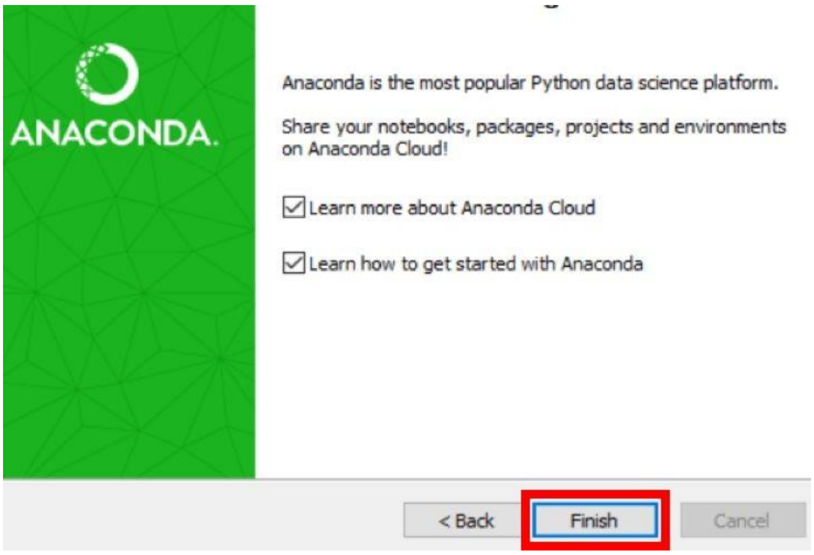
Uncover insights in your data and create interactive visualizations

ii. Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)

iii. Select path(i.e. add anaconda to path & register anaconda as default python 3.4)

iv. Click finish

v. Open jupyter notebook



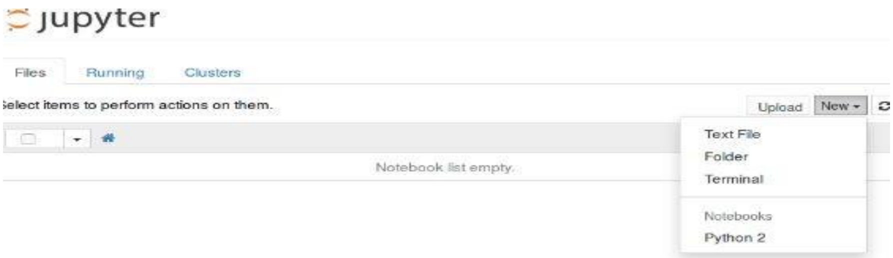
Anaconda is the most popular Python data science platform.

Share your notebooks, packages, projects and environments on Anaconda Cloud!

☑ Learn more about Anaconda Cloud

☑ Learn how to get started with Anaconda

Fig 3.2.2.1 After installation



Select items to perform actions on them.

Notebook list empty.

Fig 3.2.2.2 jupyter notebook

3.3 Features:

i. **Readable:** Python is a very readable language.

ii. **Easy to Learn:** Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn

iii. **Cross platform:** Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.

iv. **Open Source:** Python is a open source programming language.

v. **Large standard library:** Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.

vi. **Free:** Python is free to download and use. This means you can download it for free and use it in your application. Python is an example of a FLOSS (Free/Libre Open Source Software), which means you can freely distribute copies of this software, read its source code and modify it.

vii. **Supports exception handling:** If you are new, you may wonder what is an exception? An exception is an event that can occur during program exception and can disrupt the normal flow of program. Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.

viii. **Advanced features:** Supports generators and list comprehensions. We will cover these features later.

ix. **Automatic memory management:** Python supports automatic memory management which means the memory is cleared and freed automatically. You do not have to bother clearing the memory.

3.4 Variable Types:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Python has five standard data types –

- Numbers
- Strings
- Lists
- Tuples
- Dictionary

3.4.1 Python Numbers:

Number data types store numeric values. They are immutable data types, means that changing the value of a number data type results in a newly allocated object.

Python supports four different numerical types −

- **int (signed integers)** − They are often called just integers or ints, are positive or negative whole numbers with no decimal point.
- **long (long integers)** − Also called longs, they are integers of unlimited size, written like integers and followed by an uppercase or lowercase L.

- **float (floating point real values)** − Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = 2.5 x 10^2 = 250).

3.4.2 Python Strings:

In Python, Strings can be created by simply enclosing characters in quotes. Python does not support character types. These are treated as length-one strings, and are also considered as substrings. Substrings are immutable and can't be changed once created.Strings are the ordered blocks of text that are enclosed in single or double quotations. Thus, whatever is written in quotes, is considered as string. Though it can be written in single or double quotations, double quotation marks allow the user to extend strings over multiple lines without backslashes, which is usually the signal of continuation of an expression, e.g., 'abc', "ABC".

3.4.3 Python lists:

- List is a collection data type in python. It is ordered and allows duplicate entries as well. Lists in python need not be homogeneous, which means it can contain different data types like integers, strings and other collection data types. It is mutable in nature and allows indexing to access the members in a list.

- To declare a list, we use the square brackets.
- List is like any other array that we declare in other programming languages. Lists in python are often used to implement stacks and queues. The lists are mutable in nature. Therefore, the values can be changed even after a list is declared.

3.4.4 python tuples:

- A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square

  brackets.Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also

3.4.5 python Dictionary:

- It is a collection data type just like a list or a set, but there are certain features that make python dictionary unique. A dictionary in python is not ordered and is changeable as well. We can make changes in a dictionary unlike sets or strings which are immutable in nature. Dictionary contains key-value pairs like a map that we have in other programming languages. A dictionary has indexes. Since the value of the keys we declare in a dictionary are always unique, we can use them as indexes to access the elements in a dictionary.

3.5 Functions:

3.5.1 Defining a Function:

- Function blocks begin with the keyword def followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or docstring.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

3.5.2 Calling a Function:

- Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.
- Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt

3.6 OOPs Concepts:

3.6.1 Class:

- Python is an object oriented programming language. Unlike procedure oriented programming, where the main emphasis is on functions, object oriented programming stresses on objects..

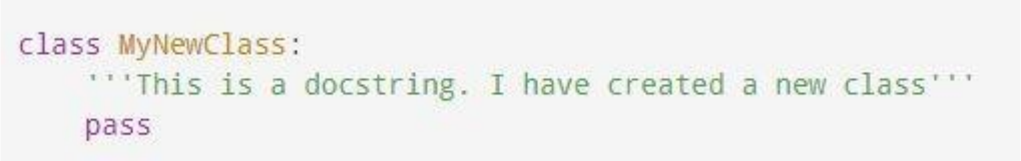
- An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.

- We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

- As many houses can be made from a house's blueprint, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is called **instantiation**.

- Like function definitions begin with the def keyword in Python, class definitions begin with a class keyword.

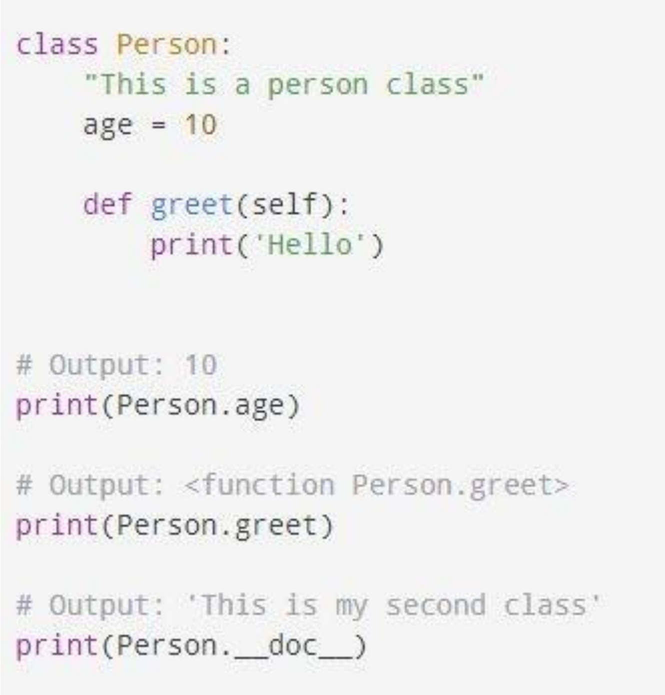
- The first string inside the class is called docstring and has a brief description about the class



```
class MyNewClass:
    '''This is a docstring. I have created a new class'''
    pass
```

Fig 3.6.1 Class defining

- As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.



```
class Person:
    "This is a person class"
    age = 10

    def greet(self):
        print('Hello')


# Output: 10
print(Person.age)

# Output: <function Person.greet>
print(Person.greet)

# Output: 'This is my second class'
print(Person.__doc__)
```

Fig 3.6.1.1 Example of class

4. EMPLOYEE ATTRITION Classification

4.1 Project Requirements:

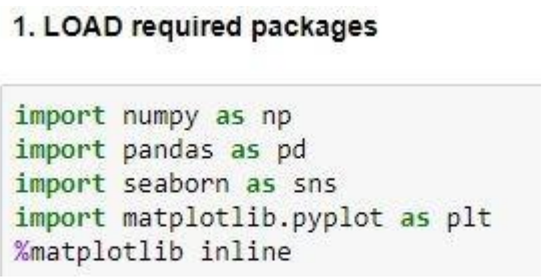
4.1.1. Packages used:

- **Numpy:** In Python we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster that traditional Python lists.The array object in NumPy is called ndata, it provides a lot of supporting functions that make working with ndarray very easy.Arrays are very frequently used in data science, where speed and resources are very important.

- **Pandas:** Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

- **Seaborn:** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Mathplotlib : Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.



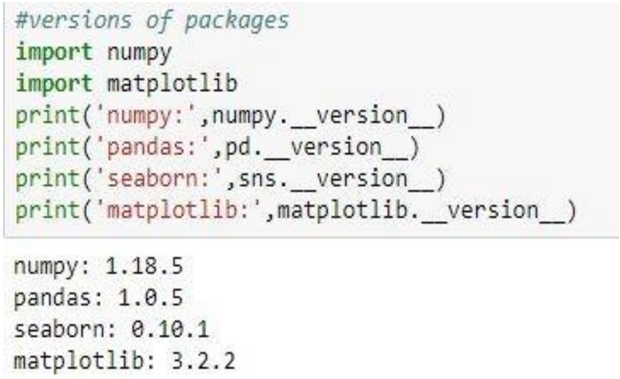
1. LOAD required packages

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Fig 4.1.1 packages

4.1.2. Versions of the packages:

The versions of the packages are found by following command



```
#versions of packages
import numpy
import matplotlib
print('numpy:',numpy.__version__)
print('pandas:',pd.__version__)
print('seaborn:',sns.__version__)
print('matplotlib:',matplotlib.__version__)
```

```
numpy: 1.18.5
pandas: 1.0.5
seaborn: 0.10.1
matplotlib: 3.2.2
```

Fig 4.1.2 versions

4.1.3. Algorithms used:

Here , 3 algorithms are used they are:

- Logistic regression
- Random Forest
- K-nearest neighbors(KNN)

4.2. Problem Statement:

This project aims through machine learning techniques at creating a model Employee Attrition Classification.

Data includes Education,joblevel,jobinvolvement,Dailyrate,MonthlyIncome, Experience,No.ofCompaniesworked,Noofcompanieschanged ,Totalworkexperience,Yearsatcompany,Yearsatcurrentrole, Yearssincelastpromotion,Yearswithcurrentmanager,Training timeslastyear,Performancerate,Attrition,By following above given data we finding the complete job profile of the person.

.

By this data and by satisfying the above given data we know the sustainable growth.Judges will look for evidence of how employees are engaged in strategic goals employees data and their experience in company and the level of the person in the company.

By the graph and the plots we can prove it graphically. And we can the no of employees working in the company.

4.3. Dataset Description:

In this data the columns contains:

- Education
- joblevel
- jobinvolvement
- Dailyrate
- MonthlyIncome
- Experience
- No.ofCompaniesworked
- Noofcompanieschanged
- Totalworkexperience
- Yearsatcompany
- Yearsatcurrentrole
- Yearssincelastpromotion
- Yearswithcurrentmanager
- Training timeslastyear
- Performancerate
- Attrition

4.4. Objective of the Case Study:

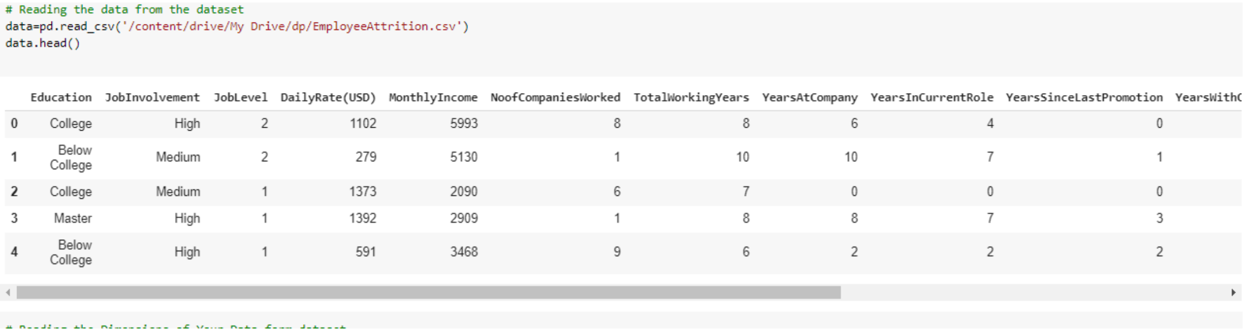
By this data and by satisfying the above given data we know the sustainable growth.

Judges will look for evidence of how employees are engaged in strategic goals employees Data and their experience in company and the level of the person in the company .And by the graph and the plots we can prove it graphically.And we can the no of employees working in the company. Celebrating organizations seeing the best returnonan investment in people,this Award recognizes the employee strategy that best attracts,retains and develops talent–and how being an employer of choice contributes to and company values, and how human and supporting resources have been optimized to achieve and sustain commercial and competitivesuccess. They will also look at investment in people,initiatives to create a more collaborative culture,and the impactthishashad on commercialperformance.

5. DATA PREPROCESSING/FEATURE ENGINEERING AND EDA

5.1 Statistical Analysis:

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a DataFrame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the dataframe. Any missing value or NaN value have to be cleaned.



```
# Reading the data from the dataset
data=pd.read_csv('/content/drive/My Drive/dp/EmployeeAttrition.csv')
data.head()
```

	Education	JobInvolvement	JobLevel	DailyRate(USD)	MonthlyIncome	NoofCompaniesWorked	TotalWorkingYears	YearsAtCompany	YearsInCurrentRole	YearsSinceLastPromotion	YearsWithC
0	College	High	2	1102	5993	8	8	6	4	0	
1	Below College	Medium	2	279	5130	1	10	10	7	1	
2	College	Medium	1	1373	2090	6	7	0	0	0	
3	Master	High	1	1392	2909	1	8	8	7	3	
4	Below College	High	1	591	3468	9	6	2	2	2	

```
# Reading the Dimensions of Your Data form dataset
```

Fig 5.1 loading data set

Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset’s distribution, excluding nan values.Analyzes both numeric and object series, as well as DataFrame column sets of mixed data types. The output will vary depending on what is provided.

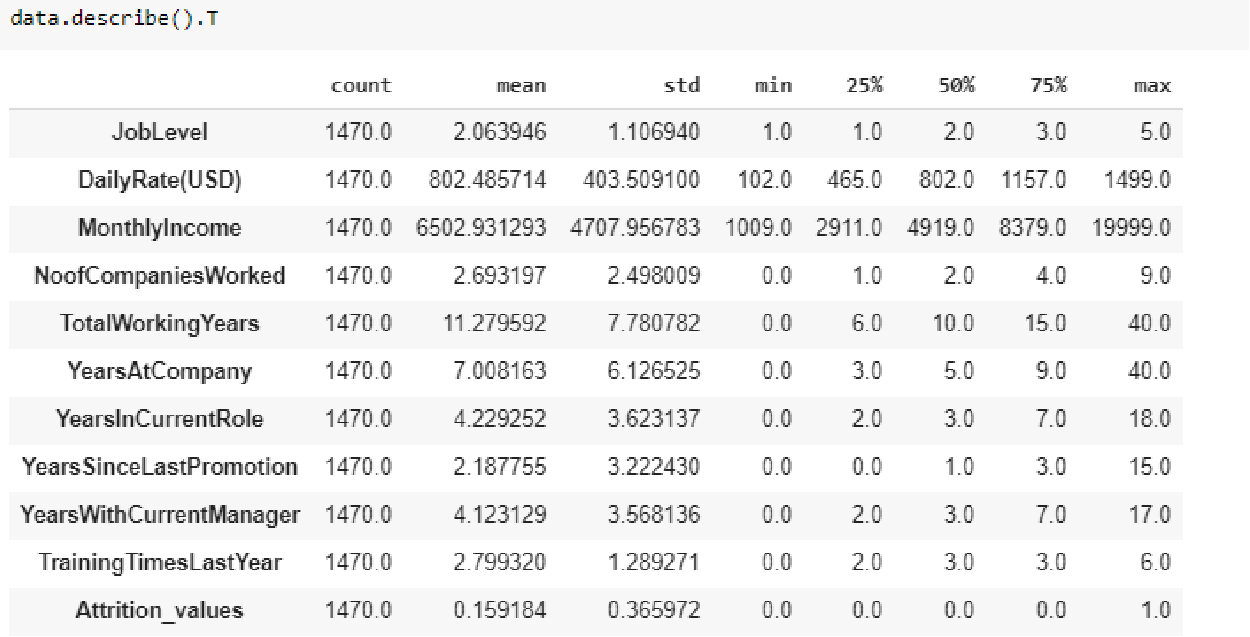
For numeric data, the result’s index will include count, mean, std, min, max as well as lower, 50 and upper percentiles. By default the lower percentile is 25 and the upper percentile is 75. The 50 percentile is the same as the median.

For object data (e.g. strings or timestamps), the result’s index will include count, unique, top,and freq. The top is the most common value. The freq is the most common value’s frequency. Timestamps also include the first and last items.

If multiple object values have the highest count, then the count and top results will be arbitrarily chosen from among those with the highest count.

For mixed data types provided via a DataFrame, the default is to return only an analysis of numeric columns. If the dataframe consists only of object and categorical data without any numeric columns, the default is to return an analysis of both the object and categorical columns. If include='all' is provided as an option, the result will include a union of attributes of each type.

Fig 5.1.1 Statistical data



```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
JobLevel	1470.0	2.063946	1.106940	1.0	1.0	2.0	3.0	5.0
DailyRate(USD)	1470.0	802.485714	403.509100	102.0	465.0	802.0	1157.0	1499.0
MonthlyIncome	1470.0	6502.931293	4707.956783	1009.0	2911.0	4919.0	8379.0	19999.0
NoofCompaniesWorked	1470.0	2.693197	2.498009	0.0	1.0	2.0	4.0	9.0
TotalWorkingYears	1470.0	11.279592	7.780782	0.0	6.0	10.0	15.0	40.0
YearsAtCompany	1470.0	7.008163	6.126525	0.0	3.0	5.0	9.0	40.0
YearsInCurrentRole	1470.0	4.229252	3.623137	0.0	2.0	3.0	7.0	18.0
YearsSinceLastPromotion	1470.0	2.187755	3.222430	0.0	0.0	1.0	3.0	15.0
YearsWithCurrentManager	1470.0	4.123129	3.568136	0.0	2.0	3.0	7.0	17.0
TrainingTimesLastYear	1470.0	2.799320	1.289271	0.0	2.0	3.0	3.0	6.0
Attrition_values	1470.0	0.159184	0.365972	0.0	0.0	0.0	0.0	1.0

Observations:-

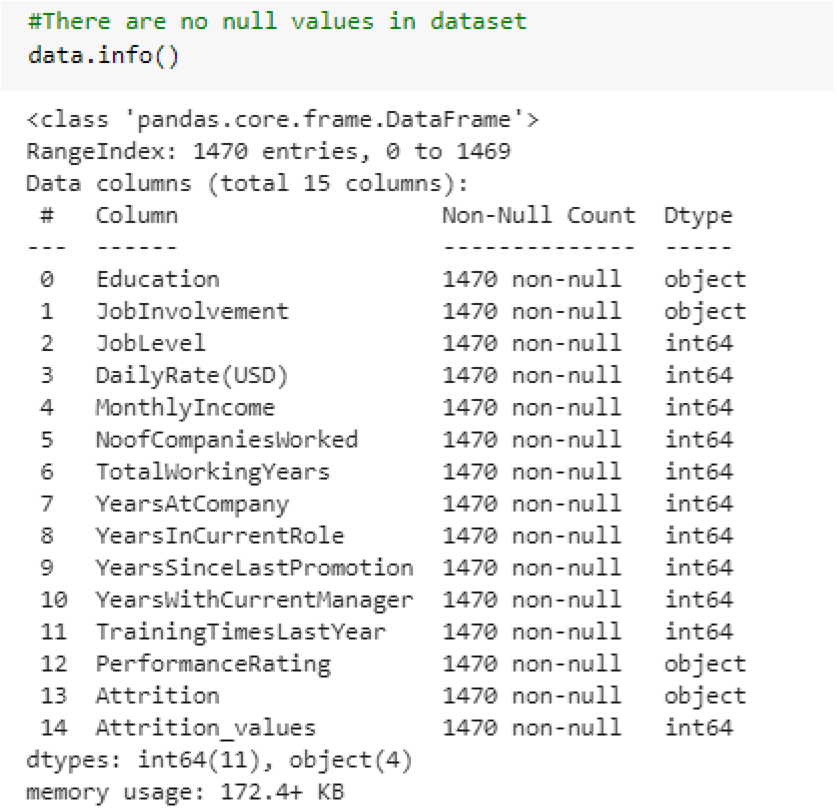
1.Max MontlyIncome is 19999 and Min MontlyIncome is 1009.

2.Count of employess in company is 1470.

3.Mean of DailyRate(USD) is 6502.931 and std is 4707.9567

5.2 Data Type Conversions:

When doing data analysis, it is important to make sure you are using the correct data types; otherwise you may get unexpected results or errors. In the case of pandas, it will correctly infer data types in many cases and you can move on with your analysis without any further thought on the topic.Despite how well pandas works, at some point in your data analysis processes, you will likely need to explicitly convert data from one type to another. This article will discuss the basic pandas data types (aka dtypes), how they map to python and numpy data types and the options for converting from one pandas type to another.



```
#There are no null values in dataset
data.info()
```

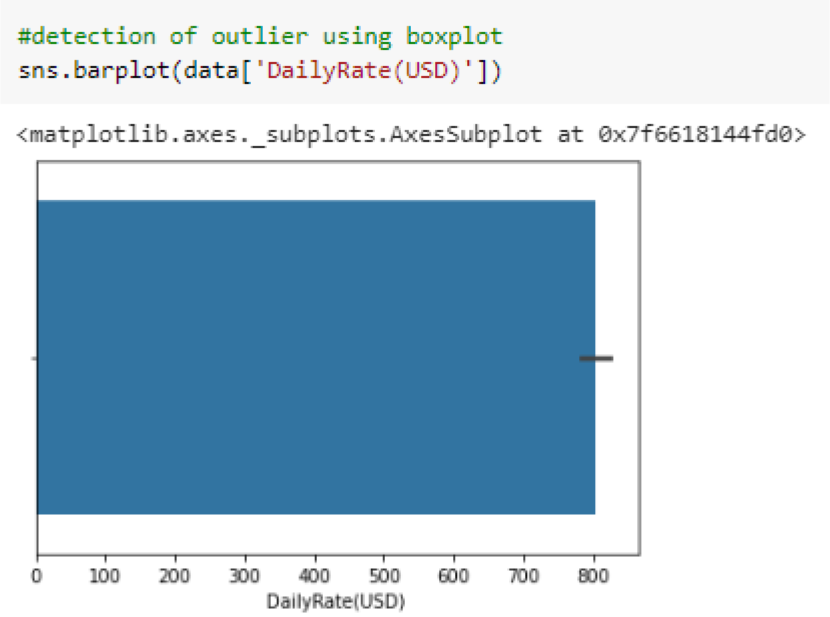
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Education                1470 non-null   object
 1   JobInvolvement           1470 non-null   object
 2   JobLevel                 1470 non-null   int64 
 3   DailyRate(USD)           1470 non-null   int64 
 4   MonthlyIncome            1470 non-null   int64 
 5   NoofCompaniesWorked      1470 non-null   int64 
 6   TotalWorkingYears        1470 non-null   int64 
 7   YearsAtCompany           1470 non-null   int64 
 8   YearsInCurrentRole       1470 non-null   int64 
 9   YearsSinceLastPromotion  1470 non-null   int64 
 10  YearsWithCurrentManager  1470 non-null   int64 
 11  TrainingTimesLastYear    1470 non-null   int64 
 12  PerformanceRating        1470 non-null   object
 13  Attrition                1470 non-null   object
 14  Attrition_values         1470 non-null   int64 
dtypes: int64(11), object(4)
memory usage: 172.4+ KB
```

Fig 5.2 datatypes

5.3 Detection of Outliers:

Perhaps the most common or familiar type of outlier is the observations that are far from the rest of the observations or the center of mass of observations.This is easy to understand when we have one or two variables and we can visualize the data as a histogram or scatter plot, although it becomes very challenging when we have many input variables defining a high-dimensional input feature space.In this case, simple statistical methods for identifying outliers can break down, such as methods that use standard deviations or the interquartile range.It can be important to identify and remove outliers from data when training machine learning algorithms for predictive modeling.Outliers can skew statistical measures and data distributions, providing a misleading representation of the underlying data and relationships. Removing outliers from training data prior to modeling can result in a better fit of the data and, in turn, more skillful predictions.Thankfully, there are a variety of automatic model-based methods for identifying outliers in input data. Importantly, each method approaches the definition of an outlier is slightly different ways, providing alternate approaches to preparing a training dataset that can be evaluated and compared, just like any other data preparation step in a modeling pipeline.

There are no outliers in a data set.



```
#detection of outlier using boxplot
sns.barplot(data['DailyRate(USD)'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6618144fd0>
```

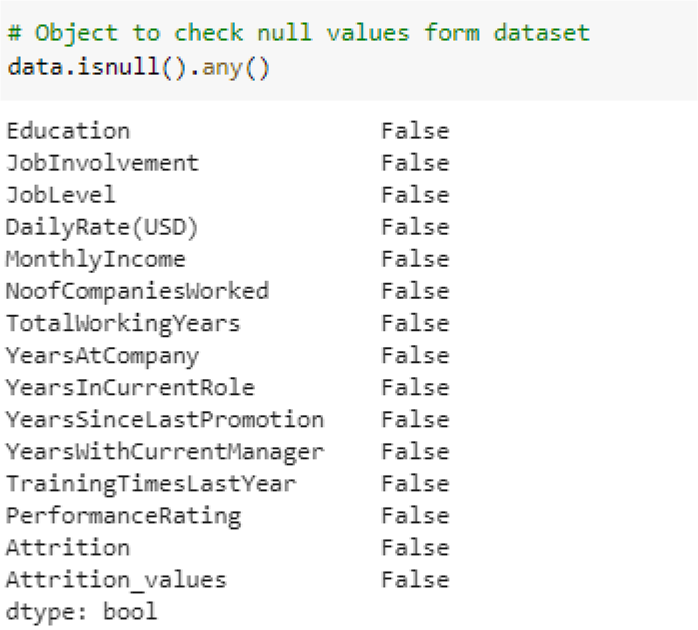
Fig 5.3 detection of outlier using boxplot

5.4 Handling Missing Values:

There are a number of schemes that have been developed to indicate the presence of missing data in a table or DataFrame. Generally, they revolve around one of two strategies:

using a mask that globally indicates missing values, or choosing a sentinel value that indicates a missing entry.In the masking approach, the mask might be an entirely separate Boolean array, or it may involve appropriation of one bit in the data representation to locally indicate the null status of a value.In the sentinel approach, the sentinel value could be some data-specific convention, such as indicating a missing integer value with -9999 or some rare bit pattern, or it could be a more global convention, such as indicating a missing floating-point value with NaN (Not a Number), a special value which is part of the IEEE floating-point specification.

Fig 5.4 There are no missing values in a dataset



```
# Object to check null values form dataset
data.isnull().any()
```

```
Education                  False
JobInvolvement             False
JobLevel                   False
DailyRate(USD)             False
MonthlyIncome              False
NoofCompaniesWorked        False
TotalWorkingYears          False
YearsAtCompany             False
YearsInCurrentRole         False
YearsSinceLastPromotion    False
YearsWithCurrentManager    False
TrainingTimesLastYear      False
PerformanceRating          False
Attrition                  False
Attrition_values           False
dtype: bool
```

5.5 Encoding Categorical Data:

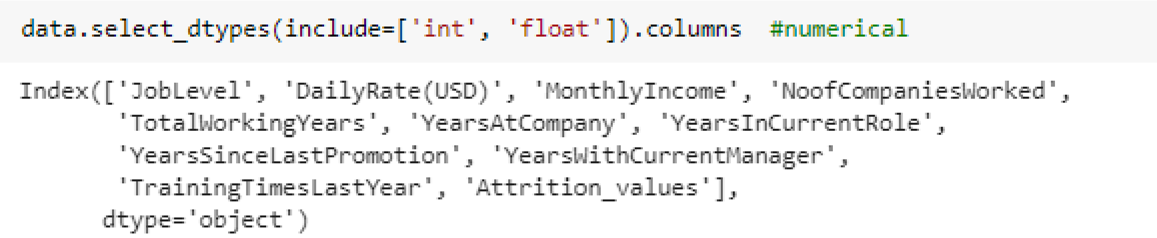
Categorical Variables are of two types: Nominal and Ordinal

● Nominal: The categories do not have any numeric ordering in between them. They don't have any ordered relationship between each of them. Examples: Male or Female, any colour

- Ordinal: The categories have a numerical ordering in between them. Example: Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer satisfaction survey, high low medium

- Categorical data can be handled by using dummy variables, which are also called as indicator variables.

In the given dataset I have not used any encoding because dataset is numerical



```
data.select_dtypes(include=['int', 'float']).columns  #numerical
```

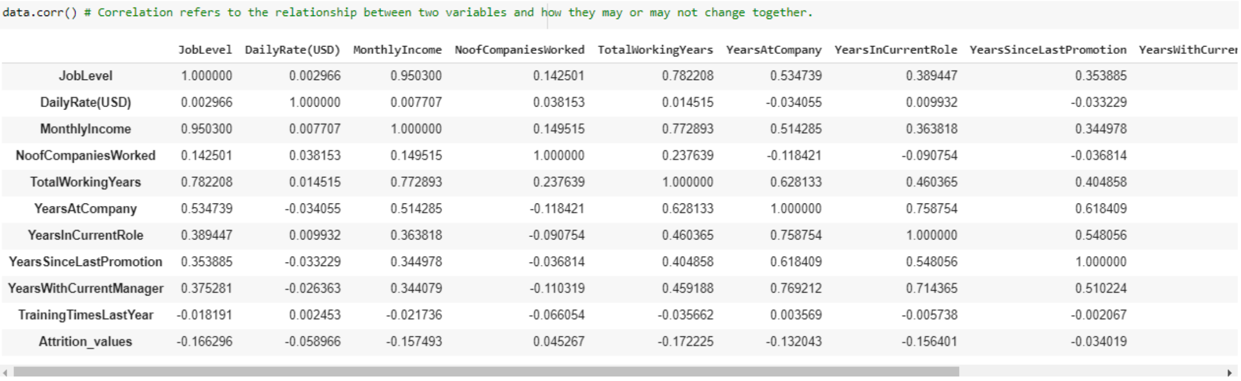
```
Index(['JobLevel', 'DailyRate(USD)', 'MonthlyIncome', 'NoofCompaniesWorked',
       'TotalWorkingYears', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrentManager',
       'TrainingTimesLastYear', 'Attrition_values'],
      dtype='object')
```

Fig 5.5 numerical in the dataset

5.6 Generating Plots:

5.6.1. Visualize the data between Target and the Features:

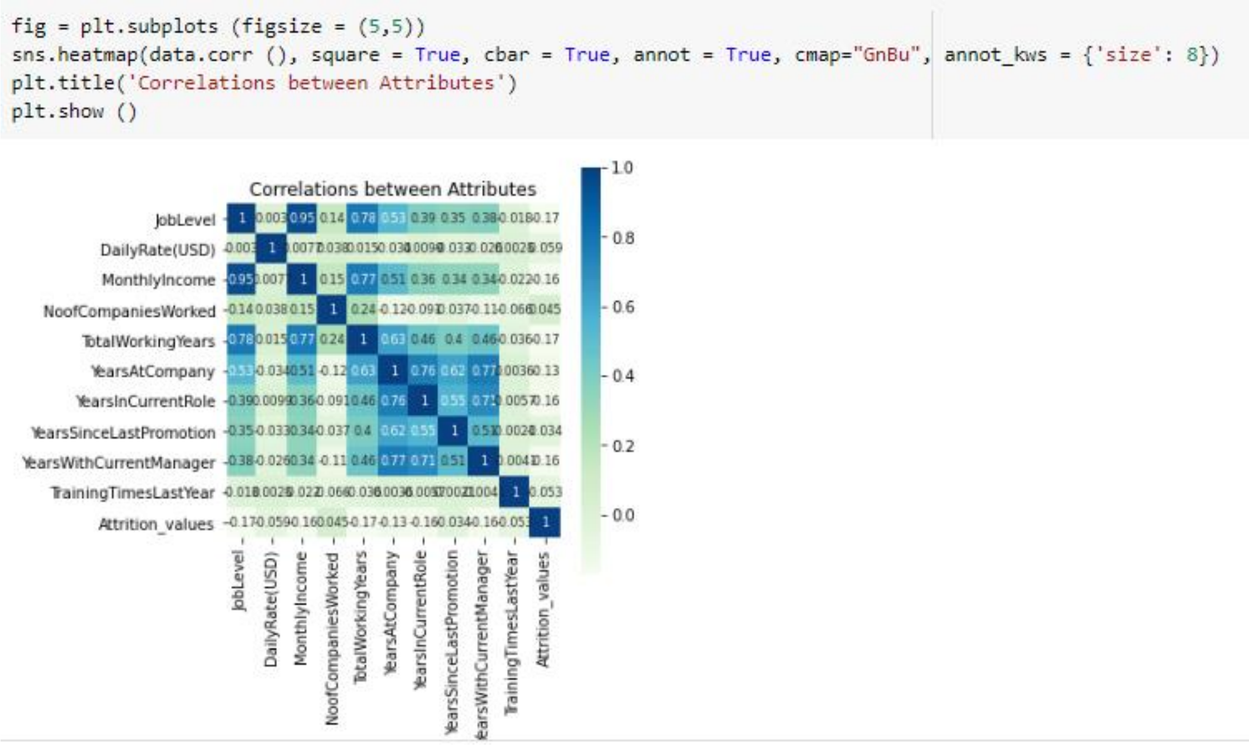
i. Correlation:



```
data.corr() # Correlation refers to the relationship between two variables and how they may or may not change together.
```

	JobLevel	DailyRate(USD)	MonthlyIncome	NoofCompaniesWorked	TotalWorkingYears	YearsAtCompany	YearsInCurrentRole	YearsSinceLastPromotion	YearsWithCurre
JobLevel	1.000000	0.002966	0.950300	0.142501	0.782208	0.534739	0.389447	0.353885	
DailyRate(USD)	0.002966	1.000000	0.007707	0.038153	0.014515	-0.034055	0.009932	-0.033229	
MonthlyIncome	0.950300	0.007707	1.000000	0.149515	0.772893	0.514285	0.363818	0.344978	
NoofCompaniesWorked	0.142501	0.038153	0.149515	1.000000	0.237639	-0.118421	-0.090754	-0.036814	
TotalWorkingYears	0.782208	0.014515	0.772893	0.237639	1.000000	0.628133	0.460365	0.404858	
YearsAtCompany	0.534739	-0.034055	0.514285	-0.118421	0.628133	1.000000	0.758754	0.618409	
YearsInCurrentRole	0.389447	0.009932	0.363818	-0.090754	0.460365	0.758754	1.000000	0.548056	
YearsSinceLastPromotion	0.353885	-0.033229	0.344978	-0.036814	0.404858	0.618409	0.548056	1.000000	
YearsWithCurrentManager	0.375281	-0.026363	0.344079	-0.110319	0.459188	0.769212	0.714365	0.510224	
TrainingTimesLastYear	-0.018191	0.002453	-0.021736	-0.066054	-0.035662	0.003569	-0.005738	-0.002067	
Attrition_values	-0.166296	-0.058966	-0.157493	0.045267	-0.172225	-0.132043	-0.156401	-0.034019	

Fig 5.6.1 correlation



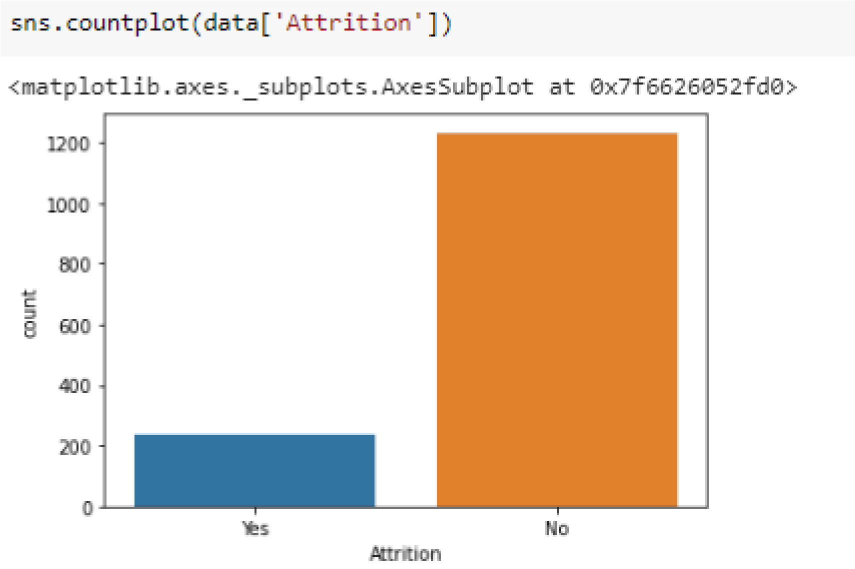
```
fig = plt.subplots (figsize = (5,5))
sns.heatmap(data.corr (), square = True, cbar = True, annot = True, cmap="GnBu", annot_kws = {'size': 8})
plt.title('Correlations between Attributes')
plt.show ()
```

Fig 5.6.1.1 co-relation graph

We can see these attributes having relationship with each other:

- Attrition vs. MonthlyIncome: high positive correlation
- JobLevel vs. DailyRate(USD): positive correlation
- NoofCompaniesWorked vs. TotalWorkingYears : positive correlation
- YearsAtCompany vs. YearsSinceLastPromotion : positive correlation

ii.Visualize the number of employess that stayed and left company



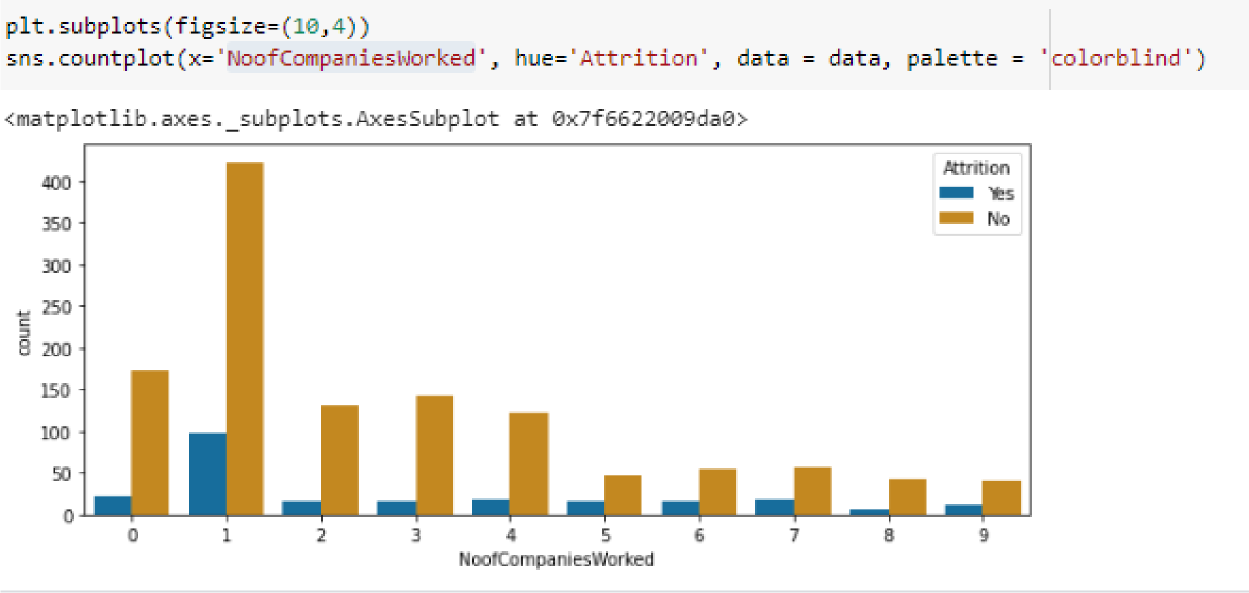
```
sns.countplot(data['Attrition'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6626052fd0>
```

Fig 5.6.1.2 Attrition(yes/no)

The above bar chart shows the number of employess that stayed 1300(aprox) and left company 250(aprox)

iii. Shows the number of employees that left and corresponding companiesworked.



```
plt.subplots(figsize=(10,4))
sns.countplot(x='NoofCompaniesWorked', hue='Attrition', data = data, palette = 'colorblind')
```

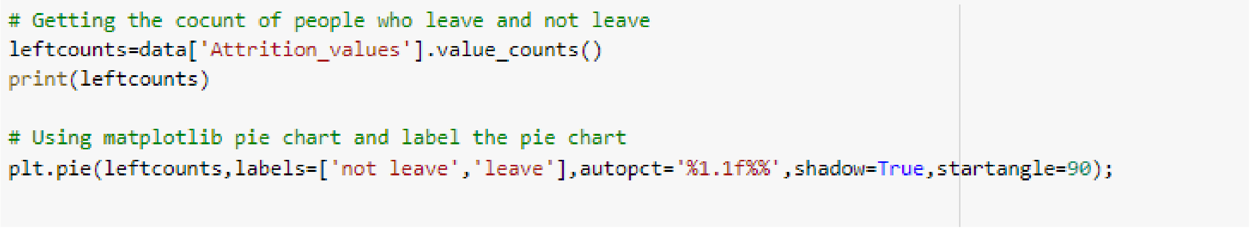
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6622009da0>
```

Fig 5.6.1.3 CompaniesWorked vs Attrition

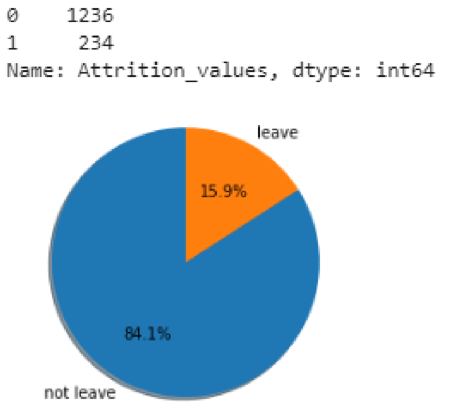
The above bar chart shows the number of employess that worked on different compaines(NoofCompanies Worked) with employees leaving(highest 1 company and 8 compaines lowest) and not leaving(highest 1 company and 8 compaines lowest)

5.6.2 Visualize the data between all the Features:

i. shows the profile leaving employees:



```
# Getting the cocunt of people who leave and not leave
leftcounts=data['Attrition_values'].value_counts()
print(leftcounts)

# Using matplotlib pie chart and label the pie chart
plt.pie(leftcounts,labels=['not leave','leave'],autopct='%1.1f%%',shadow=True,startangle=90);
```

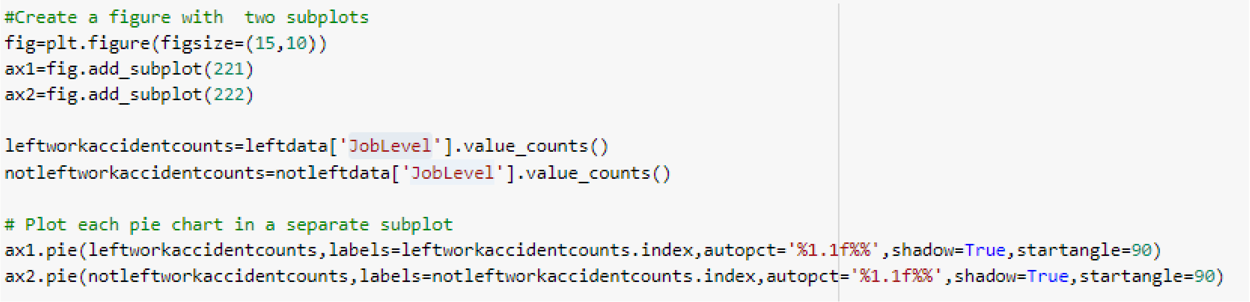


```
0    1236
1     234
Name: Attrition_values, dtype: int64
```

Fig 5.6.2.1 Attrition(leave/not leave)

The above pie chart shows the % of employees leaving(15.9%) and not leaving(84.1%)

ii. JobLevel% of employees leaving not leaving:



```
#Create a figure with  two subplots
fig=plt.figure(figsize=(15,10))
ax1=fig.add_subplot(221)
ax2=fig.add_subplot(222)

leftworkaccidentcounts=leftdata['JobLevel'].value_counts()
notleftworkaccidentcounts=notleftdata['JobLevel'].value_counts()

# Plot each pie chart in a separate subplot
ax1.pie(leftworkaccidentcounts,labels=leftworkaccidentcounts.index,autopct='%1.1f%%',shadow=True,startangle=90)
ax2.pie(notleftworkaccidentcounts,labels=notleftworkaccidentcounts.index,autopct='%1.1f%%',shadow=True,startangle=90)
```

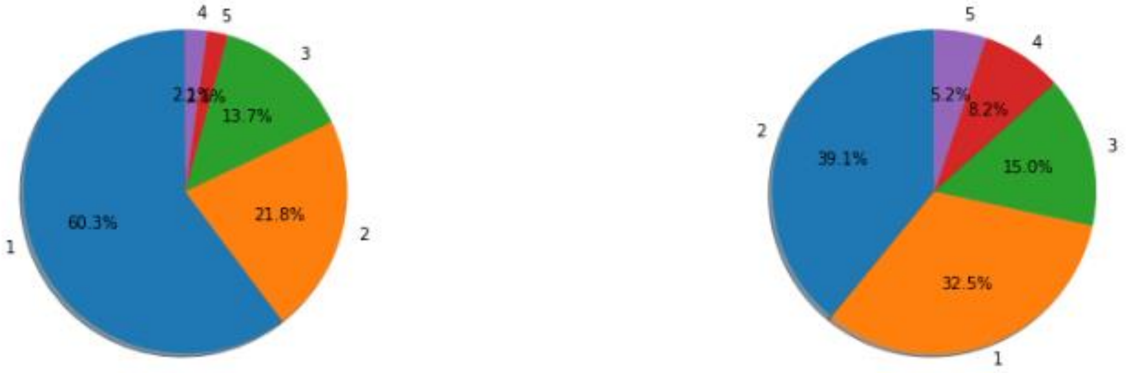
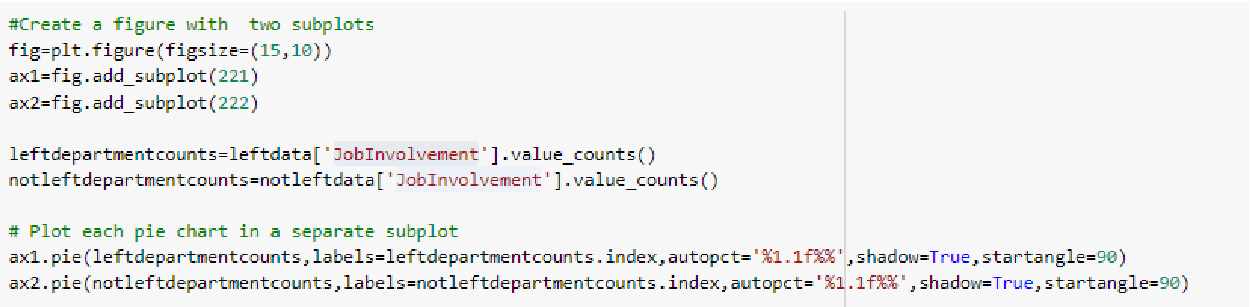


Fig 5.6.2.2 Attrition(leave/not leave) vs JobLevel

The above pie chart shows the JobLevel% of employees leaving(max 60.3% and min 2.2%) and not leaving(max 39.1% and min 5.2%)

iii. JobInvolvement% of employees leaving not leaving:



```
#Create a figure with  two subplots
fig=plt.figure(figsize=(15,10))
ax1=fig.add_subplot(221)
ax2=fig.add_subplot(222)

leftdepartmentcounts=leftdata['JobInvolvement'].value_counts()
notleftdepartmentcounts=notleftdata['JobInvolvement'].value_counts()

# Plot each pie chart in a separate subplot
ax1.pie(leftdepartmentcounts,labels=leftdepartmentcounts.index,autopct='%1.1f%%',shadow=True,startangle=90)
ax2.pie(notleftdepartmentcounts,labels=notleftdepartmentcounts.index,autopct='%1.1f%%',shadow=True,startangle=90)
```

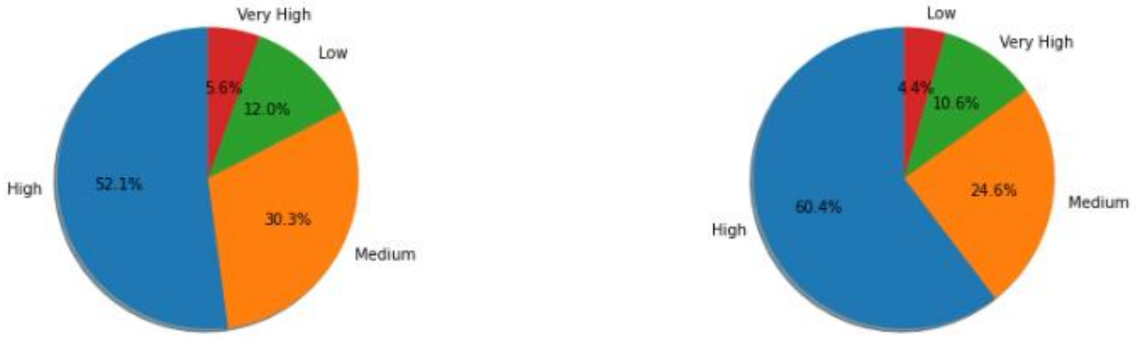
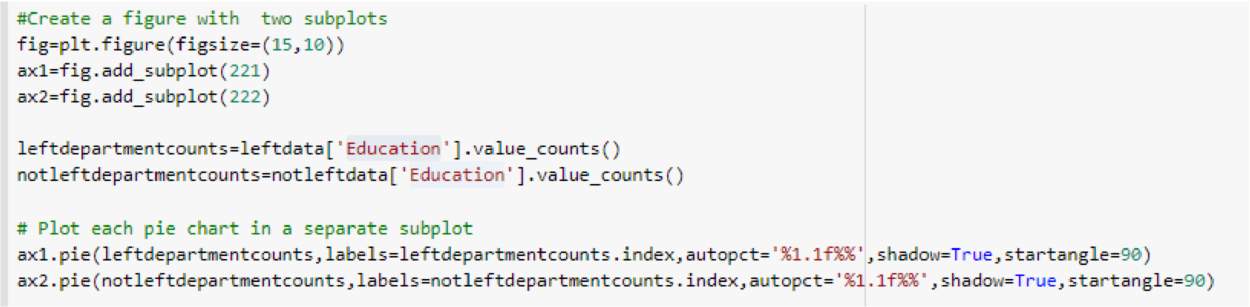


Fig 5.6.2.3 Attrition(leave/not leave) vs JobInvolvement

The above pie chart shows the JobInvolvement% of employees leaving(max 52.1% and min 5.6%) and not leaving(max 60.4% and min 4.4%)

iv. Education% of employees leaving not leaving:



```
#Create a figure with  two subplots
fig=plt.figure(figsize=(15,10))
ax1=fig.add_subplot(221)
ax2=fig.add_subplot(222)

leftdepartmentcounts=leftdata['Education'].value_counts()
notleftdepartmentcounts=notleftdata['Education'].value_counts()

# Plot each pie chart in a separate subplot
ax1.pie(leftdepartmentcounts,labels=leftdepartmentcounts.index,autopct='%1.1f%%',shadow=True,startangle=90)
ax2.pie(notleftdepartmentcounts,labels=notleftdepartmentcounts.index,autopct='%1.1f%%',shadow=True,startangle=90)
```

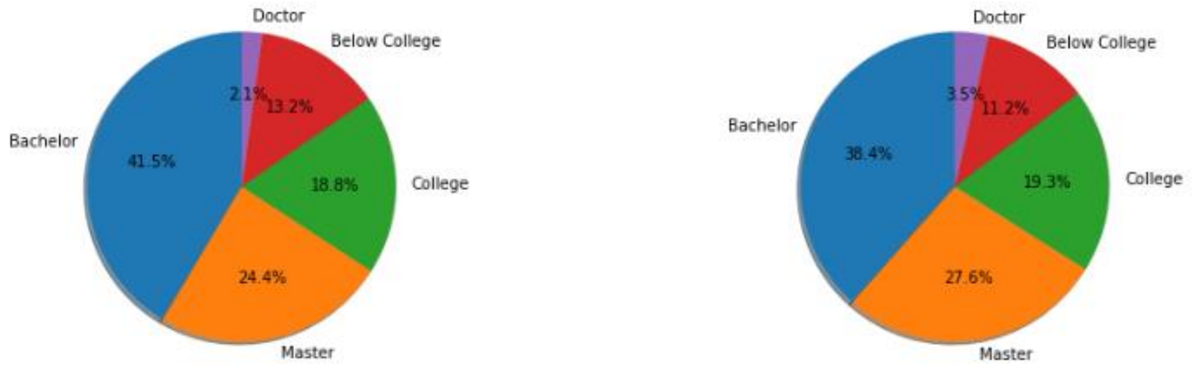
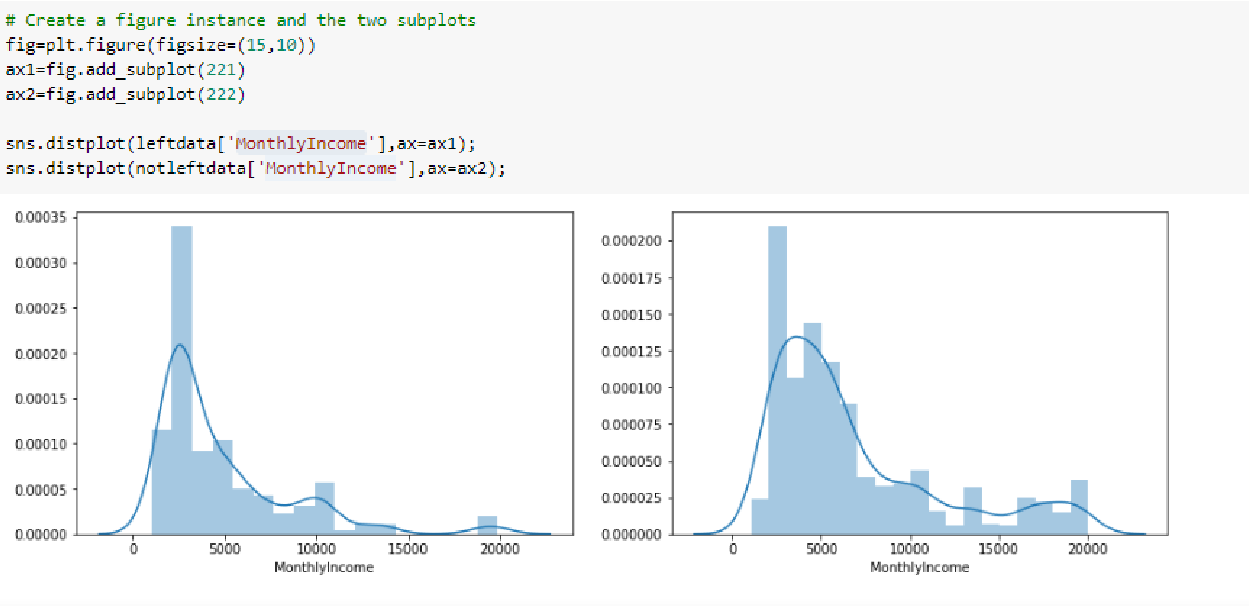


Fig 5.6.2.4 Attrition(leave/not leave) vs Education

The above pie chart shows the Education% of employees leaving(max 41.5% and min 2.1%) and not leaving(max 38.4% and min 3.5%)

v. MonthlyIncome of leave and not leave employees:

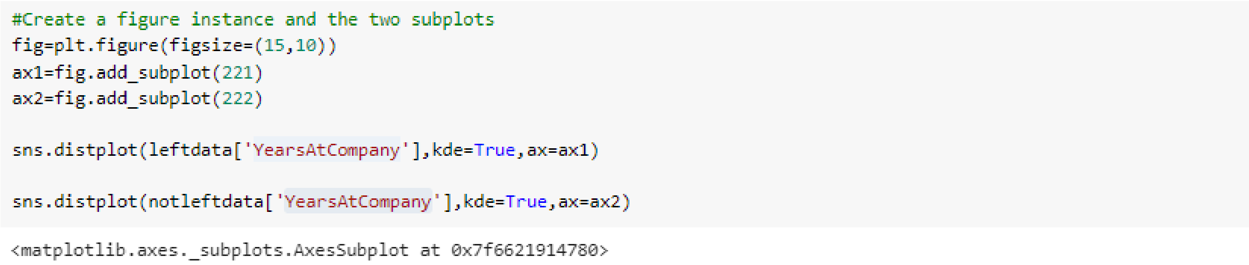


```
# Create a figure instance and the two subplots
fig=plt.figure(figsize=(15,10))
ax1=fig.add_subplot(221)
ax2=fig.add_subplot(222)

sns.distplot(leftdata['MonthlyIncome'],ax=ax1);
sns.distplot(notleftdata['MonthlyIncome'],ax=ax2);
```

Fig 5.6.2.5 Attrition(leave/not leave) vs MonthlyIncome

The above plot tells about the MonthlyIncome of leave and not leave employees.There is a somewhat aver age veriation .

vi. YearsAtCompany of leave and not leave employees:



```
#Create a figure instance and the two subplots
fig=plt.figure(figsize=(15,10))
ax1=fig.add_subplot(221)
ax2=fig.add_subplot(222)

sns.distplot(leftdata['YearsAtCompany'],kde=True,ax=ax1)

sns.distplot(notleftdata['YearsAtCompany'],kde=True,ax=ax2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6621914780>
```

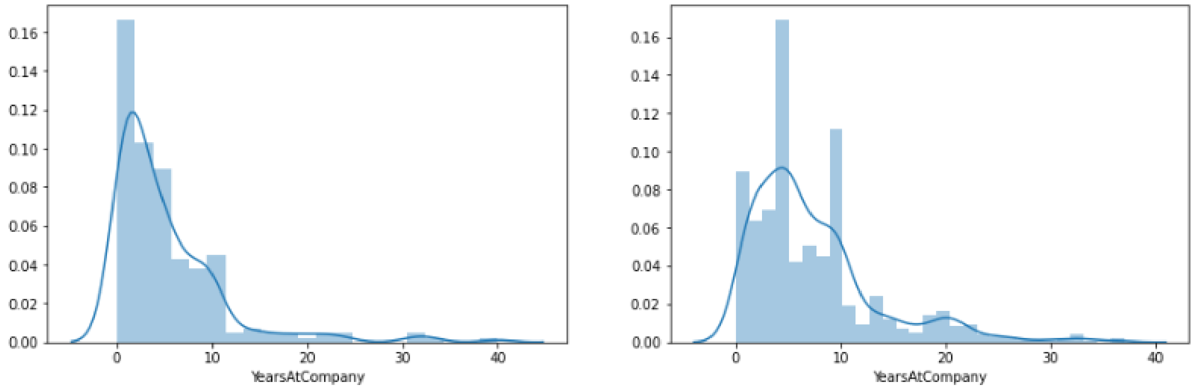
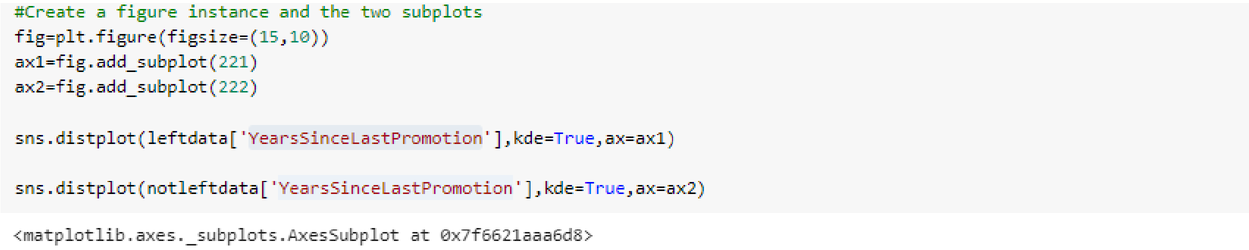


Fig 5.6.2.6 Attrition(leave/not leave) vs YearsAtCompany

The above plot tells about the YearsAtCompany of leave and not leave employees.There is a average/little veriation .

vii. YearsSinceLastPromotion of leave and not leave employees:



```
#Create a figure instance and the two subplots
fig=plt.figure(figsize=(15,10))
ax1=fig.add_subplot(221)
ax2=fig.add_subplot(222)

sns.distplot(leftdata['YearsSinceLastPromotion'],kde=True,ax=ax1)

sns.distplot(notleftdata['YearsSinceLastPromotion'],kde=True,ax=ax2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6621aaa6d8>
```

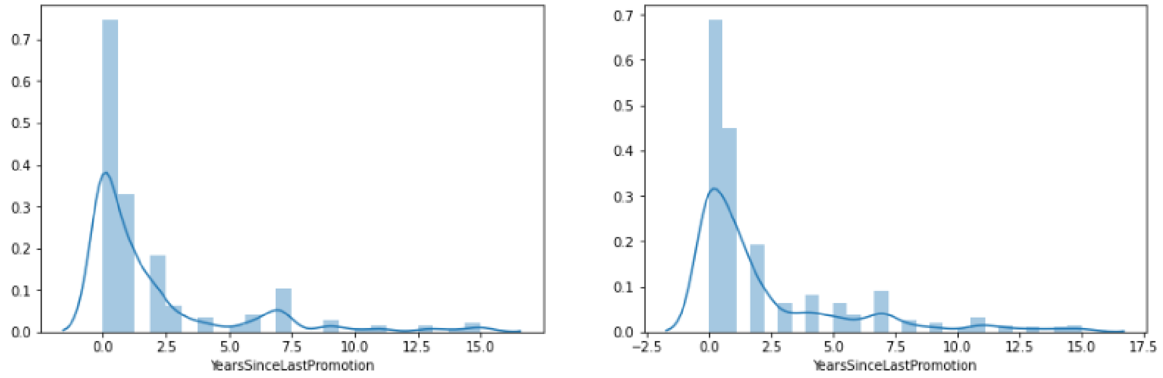
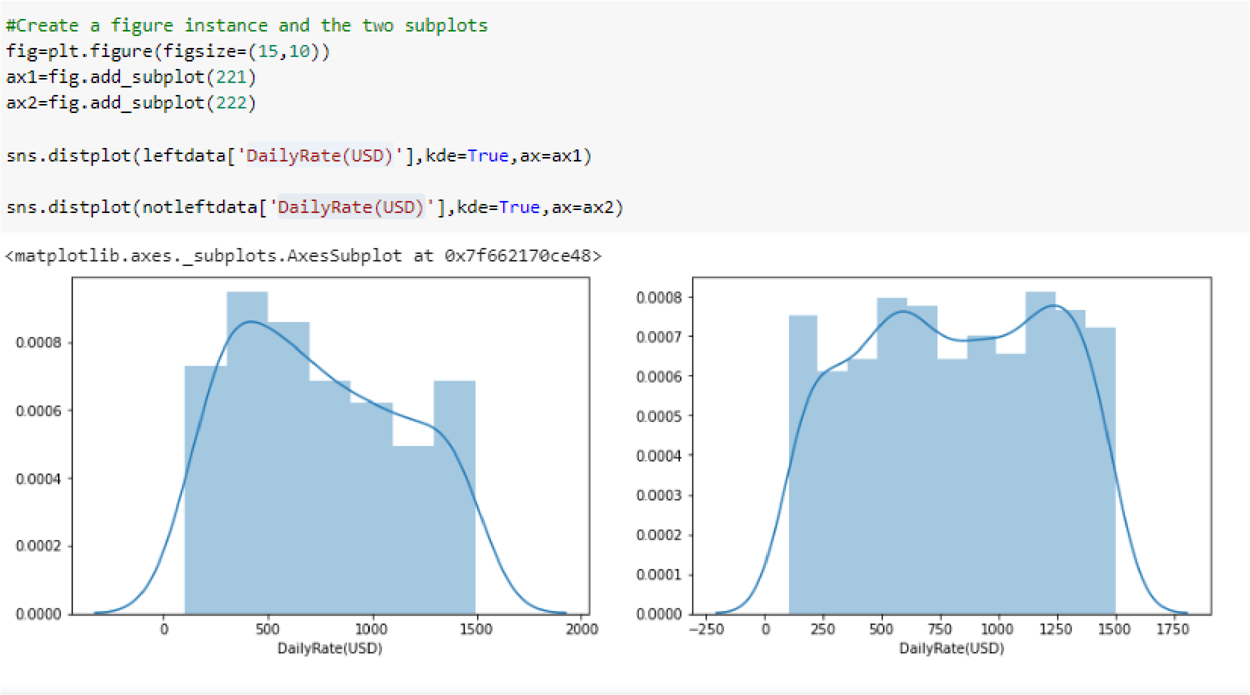


Fig 5.6.2.7 Attrition(leave/not leave) vs YearsSinceLastPromotion

The above plot tells about the YearsSinceLastPromotion of leave and not leave employees.There is a little veriation .

viii. DailyRate(USD) of leave and not leave employees:



```
#Create a figure instance and the two subplots
fig=plt.figure(figsize=(15,10))
ax1=fig.add_subplot(221)
ax2=fig.add_subplot(222)

sns.distplot(leftdata['DailyRate(USD)'],kde=True,ax=ax1)

sns.distplot(notleftdata['DailyRate(USD)'],kde=True,ax=ax2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f662170ce48>
```

Fig 5.6.2.8 Attrition(leave/not leave) vs DailyRate(USD)

6.FEATURE SELECTION

6.1 Select relevant features for the analysis:

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

· **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.

· **Improves Accuracy**: Less misleading data means modeling accuracy improves.

· **Reduces Training Time**: fewer data points reduce algorithm complexity and algorithms train faster.

Feature Selection Methods:

I will share 3 Feature selection techniques that are easy to use and also gives good results.

1. Univariate Selection

2. Feature Importance

3. Univariate Selection

4. Feature Importance

5.Correlation Matrix with Heatmap

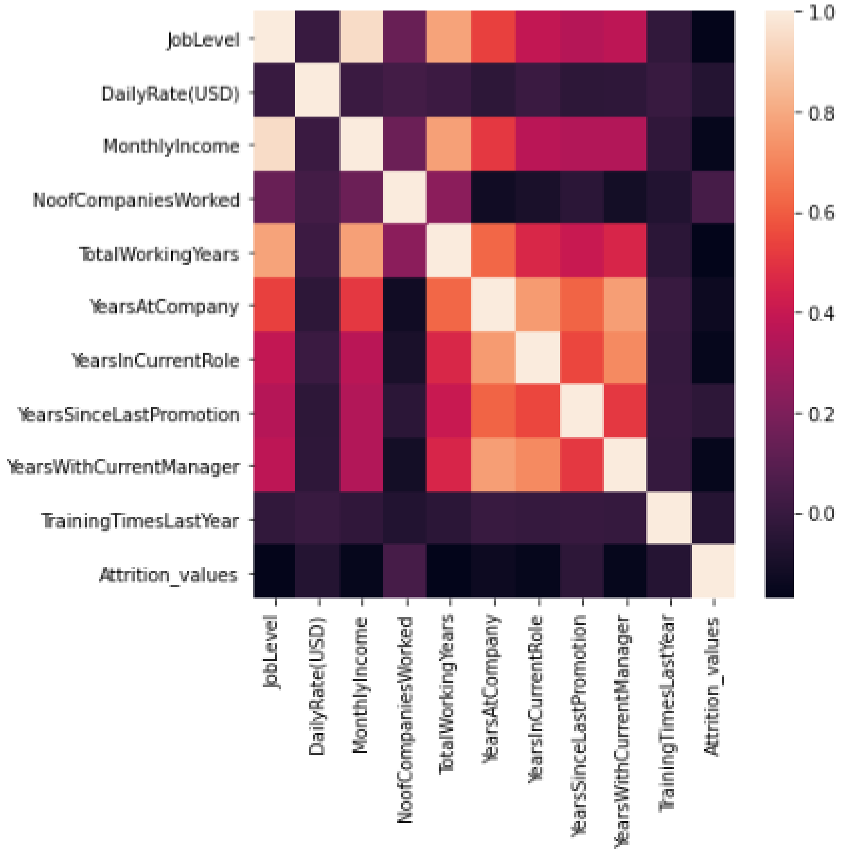


Fig 6.1 top features in dataset

6.2 Drop irrelevant features:

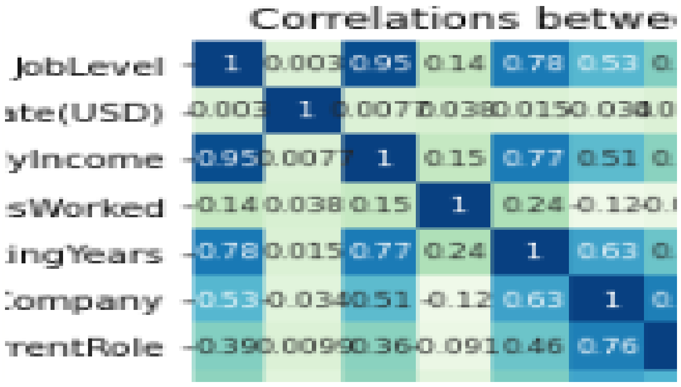
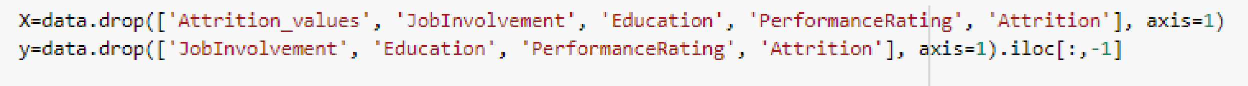


Fig 6.2 correlation between attributes

6.3 Train-Test-Split:

One of the first decisions to make when starting a modeling project is how to utilize the existing data. One common technique is to split the data into two groups typically referred to as the training and testing sets. The training set is used to develop models and feature sets; they are the substrate for estimating parameters, comparing models, and all of the other activities required to reach a final model. The test set is used only at the conclusion of these activities for estimating a final, unbiased assessment of the model's performance. It is critical that the test set not be used prior to this point. Looking at the test sets results would bias the outcomes since the testing data will have become part of the model development process.

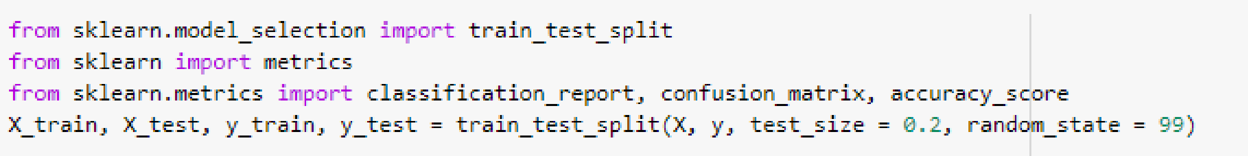
There are a number of ways to split the data into training and testing sets. The most common approach is to use some version of random sampling. Completely random sampling is a straightforward strategy to implement and usually protects the process from being biased towards any characteristic of the data. However this approach can be problematic when the response is not evenly distributed across the outcome. A less risky splitting strategy would be to use a stratified random sample based on the outcome. For classification models, this is accomplished by selecting samples at random within each class. This approach ensures that the frequency distribution of the outcome is approximately equal within the training and test sets. When the outcome is numeric, artificial strata can be constructed based on the quartiles of the data. For example, in the Ames housing price data, the quartiles of the outcome distribution would break the data into four artificial groups containing roughly 230 houses. The training/test split would then be conducted within these four groups and the four different training set portions are pooled together (and the same for the test set).



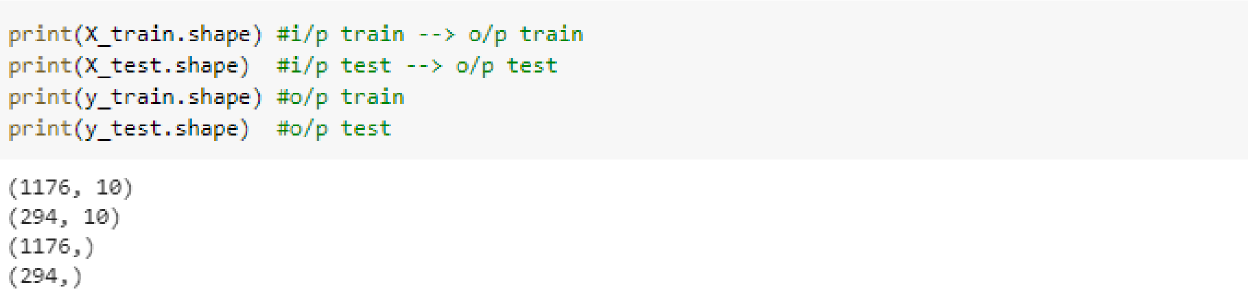
```
X=data.drop(['Attrition_values', 'JobInvolvement', 'Education', 'PerformanceRating', 'Attrition'], axis=1)
y=data.drop(['JobInvolvement', 'Education', 'PerformanceRating', 'Attrition'], axis=1).iloc[:,-1]
```

Fig 6.3.1 dividing input and output and target values for output

Importing packages:



```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 99)
```



```
print(X_train.shape) #i/p train --> o/p train
print(X_test.shape)  #i/p test --> o/p test
print(y_train.shape) #o/p train
print(y_test.shape)  #o/p test

(1176, 10)
(294, 10)
(1176,)
(294,)
```

6.4 Feature Scaling:

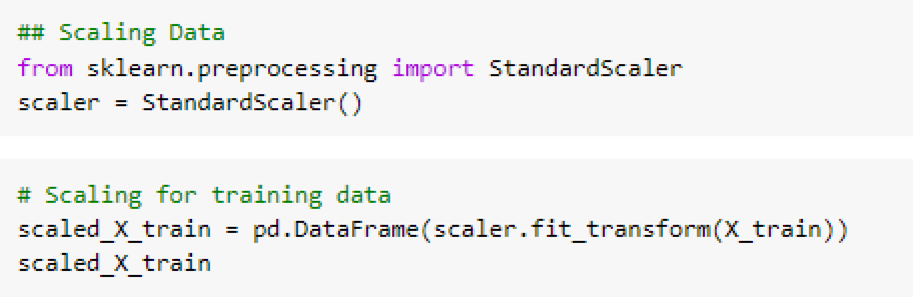
It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm. Real world dataset contains features that highly vary in magnitudes, units, and range. Normalisation should be performed when the scale of a feature is irrelevant or misleading and not should Normalise when the scale is meaningful.

The algorithms which use Euclidean Distance measure are sensitive to Magnitudes. Here feature scaling helps to weigh all the features equally.

Formally, If a feature in the dataset is big in scale compared to others then in algorithms where Euclidean distance is measured this big scaled feature becomes dominating and needs to be normalized.

$$z = \frac{x - \mu}{\sigma}$$

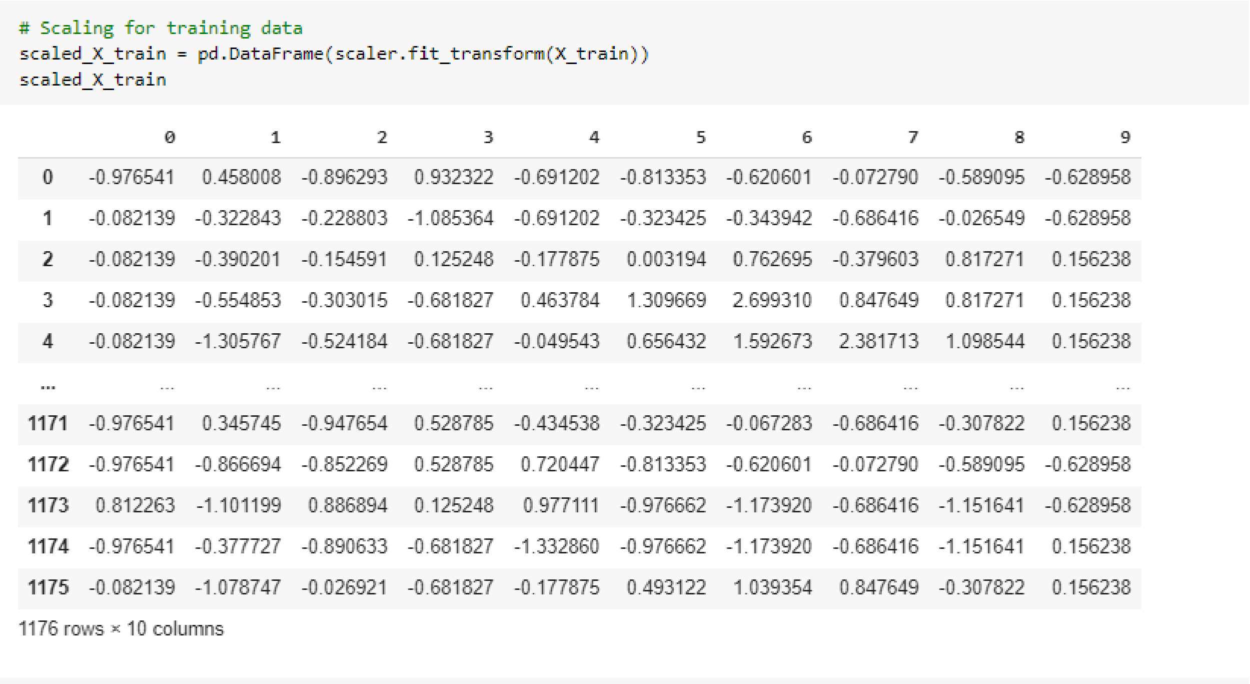
Fig 6.4.1 formula for scaling



```
## Scaling Data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
# Scaling for training data
scaled_X_train = pd.DataFrame(scaler.fit_transform(X_train))
scaled_X_train
```

Fig 6.4.2 scaled X train



```
# Scaling for training data
scaled_X_train = pd.DataFrame(scaler.fit_transform(X_train))
scaled_X_train
```

	0	1	2	3	4	5	6	7	8	9
0	-0.976541	0.458008	-0.896293	0.932322	-0.691202	-0.813353	-0.620601	-0.072790	-0.589095	-0.628958
1	-0.082139	-0.322843	-0.228803	-1.085364	-0.691202	-0.323425	-0.343942	-0.686416	-0.026549	-0.628958
2	-0.082139	-0.390201	-0.154591	0.125248	-0.177875	0.003194	0.762695	-0.379603	0.817271	0.156238
3	-0.082139	-0.554853	-0.303015	-0.681827	0.463784	1.309669	2.699310	0.847649	0.817271	0.156238
4	-0.082139	-1.305767	-0.524184	-0.681827	-0.049543	0.656432	1.592673	2.381713	1.098544	0.156238
...	...	...	...	...	...	...	...	...	...	...
1171	-0.976541	0.345745	-0.947654	0.528785	-0.434538	-0.323425	-0.067283	-0.686416	-0.307822	0.156238
1172	-0.976541	-0.866694	-0.852269	0.528785	0.720447	-0.813353	-0.620601	-0.072790	-0.589095	-0.628958
1173	0.812263	-1.101199	0.886894	0.125248	0.977111	-0.976662	-1.173920	-0.686416	-1.151641	-0.628958
1174	-0.976541	-0.377727	-0.890633	-0.681827	-1.332860	-0.976662	-1.173920	-0.686416	-1.151641	0.156238
1175	-0.082139	-1.078747	-0.026921	-0.681827	-0.177875	0.493122	1.039354	0.847649	-0.307822	0.156238

1176 rows × 10 columns

1. **K-Means** uses the Euclidean distance measure here feature scaling matters.

2. **K-Nearest-Neighbours** also require feature scaling.

3. **Principal Component Analysis (PCA)**: Tries to get the feature with maximum variance, here too feature scaling is required.

4. **Gradient Descent**: Calculation speed increase as Theta calculation becomes faster after feature scaling

7. MODEL BUILDING AND EVALUATION

7.1 Brief about the algorithms used:

i.Logistic Regression:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

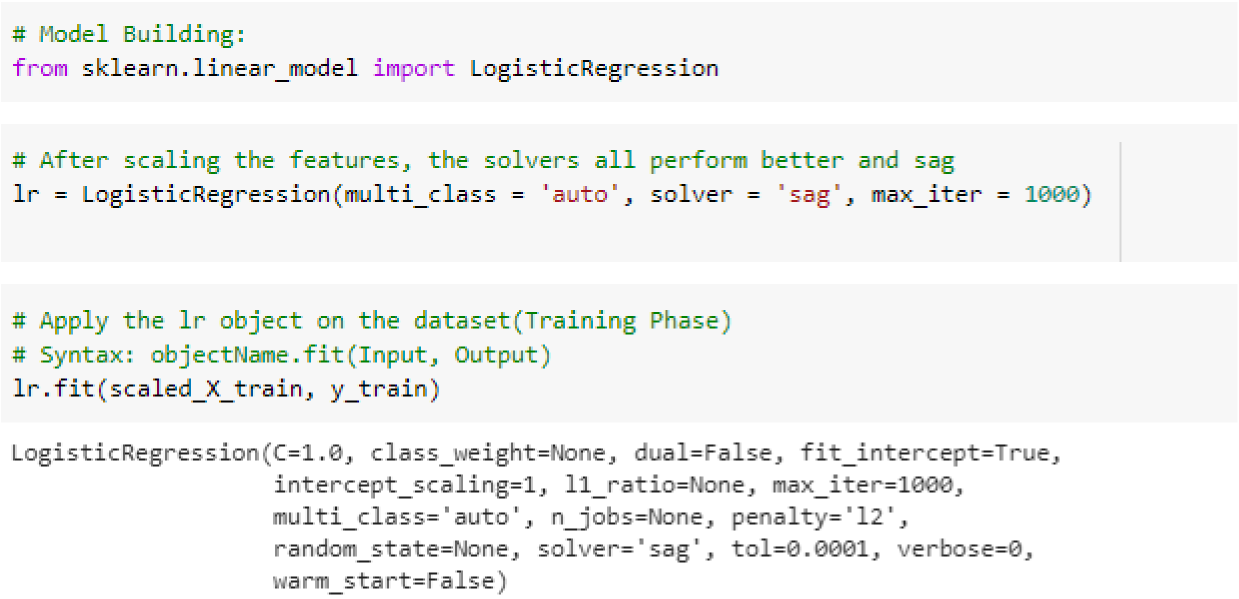
Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same −

- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-collinearity in the model, which means the

independent variables must be independent of each other.

- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression

.



```
# Model Building:
from sklearn.linear_model import LogisticRegression
```

```
# After scaling the features, the solvers all perform better and sag
lr = LogisticRegression(multi_class = 'auto', solver = 'sag', max_iter = 1000)
```

```
# Apply the lr object on the dataset(Training Phase)
# Syntax: objectName.fit(Input, Output)
lr.fit(scaled_X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='sag', tol=0.0001, verbose=0,
                   warm_start=False)
```

Fig 7.1.1 logistic regression for train data

ii. K-nearest neighbors(KNN) :

- K-nearest neighbors (KNN) will be focus on primarily how does the algorithm work and how does the input parameter affects the output/prediction.
- KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

  a. Ease to interpret output

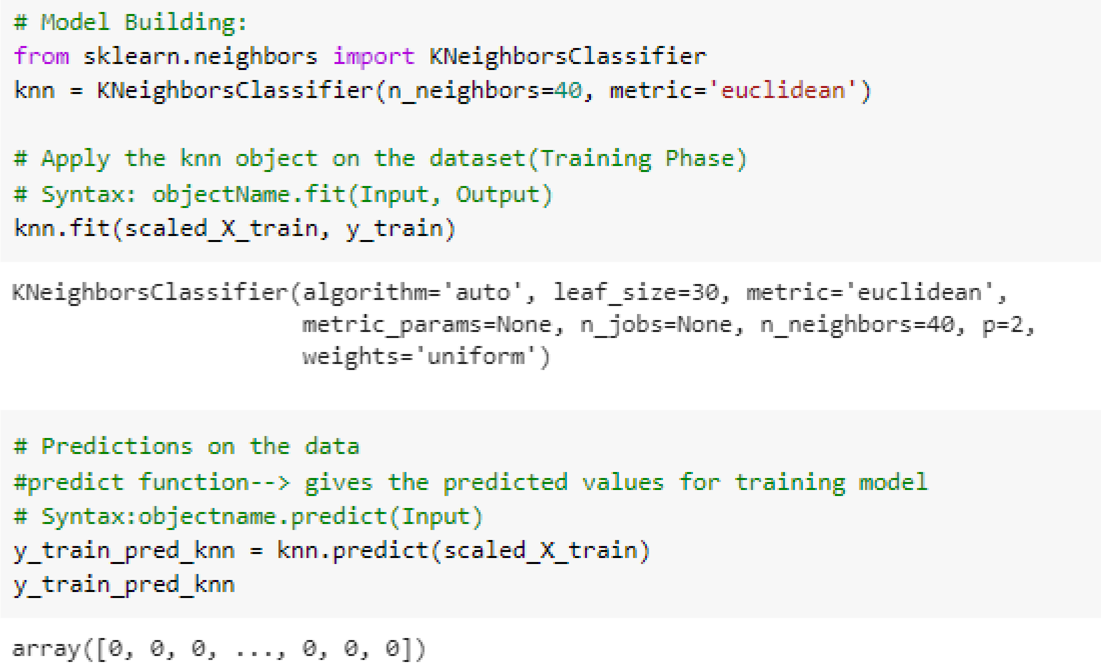
  b. Calculation time

c. Predictive Power

- KNN algorithm fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

We can implement a KNN model by following the below steps:

1. Load the data
2. Initialise the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
    1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it’s the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
    2. Sort the calculated distances in ascending order based on distance values
    3. Get top k rows from the sorted array
    4. Get the most frequent class of these rows
    5. Return the predicted class



```
# Model Building:
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=40, metric='euclidean')

# Apply the knn object on the dataset(Training Phase)
# Syntax: objectName.fit(Input, Output)
knn.fit(scaled_X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=40, p=2,
                     weights='uniform')
```

```
# Predictions on the data
#predict function--> gives the predicted values for training model
# Syntax:objectname.predict(Input)
y_train_pred_knn = knn.predict(scaled_X_train)
y_train_pred_knn
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

Fig 7.1.2 K-nearest neighbors(KNN) for train data

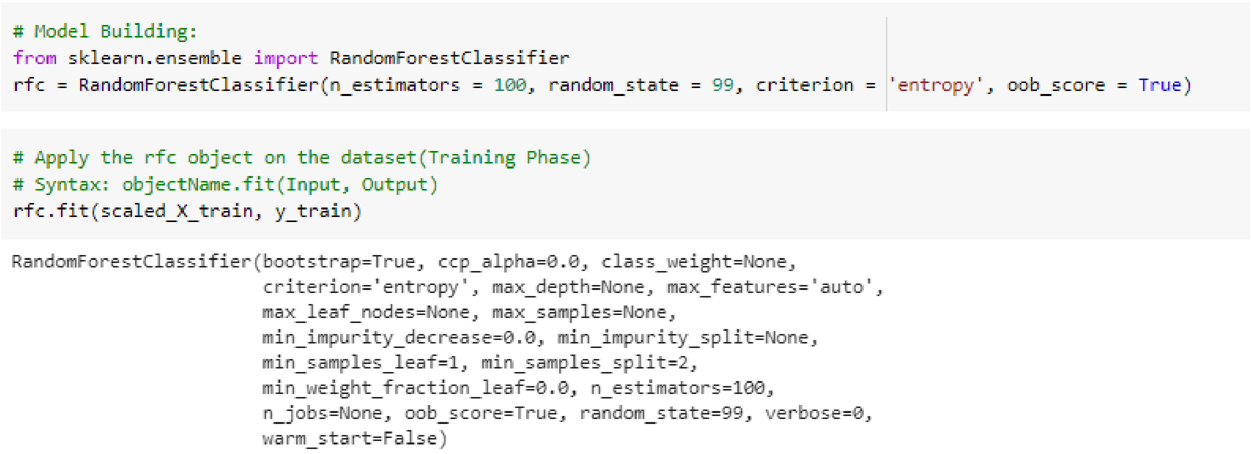
iii. Random forest:

Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression tasks). In this post we'll learn how the random forest algorithm works, how it differs from other algorithms and how to use it.

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.



```
# Model Building:
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 100, random_state = 99, criterion = 'entropy', oob_score = True)
```

```
# Apply the rfc object on the dataset(Training Phase)
# Syntax: objectName.fit(Input, Output)
rfc.fit(scaled_X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=True, random_state=99, verbose=0,
                       warm_start=False)
```

Fig 7.1.3 Random Forests for train data

7.2 Train the Models:

Splitting the data : after the preprocessing is done then the data is split into train and test sets

● In Machine Learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'test set'. An important point to note is that during training the classifier only uses the training set . The test set must not be used during training the classifier. The test set will only be available during testing the classifier.

● training set - a subset to train a model.(Model learns patterns between Input and Output)

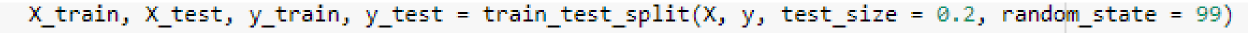
● test set - a subset to test the trained model.(To test whether the model has correctly learnt)

● The amount or percentage of Splitting can be taken as specified

● First we need to identify the input and output variables and we need to separate the input set and output set

● In scikit learn library we have a package called model_selection in which train_test_split method is available .we need to import this method

● This method splits the input and output data to train and test based on the percentage specified by the user and assigns them to four different variables.



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 99)
```

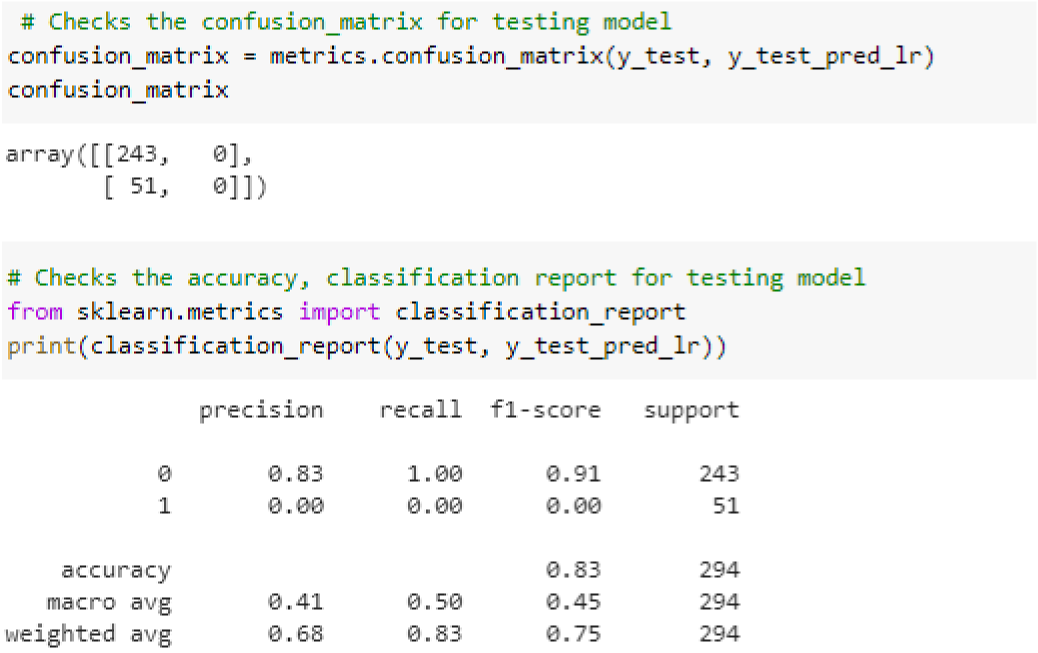
Fig 7.2.1 Training data

7.3 Make Predictions:

- Then we have to test the model for the test set ,that is done as follows

- We have a method called predict , using this method we need to predict the output for input test set and we need to compare the out but with the output test data

- If the predicted values and the original values are close then we can say that model is trained with good accuracy



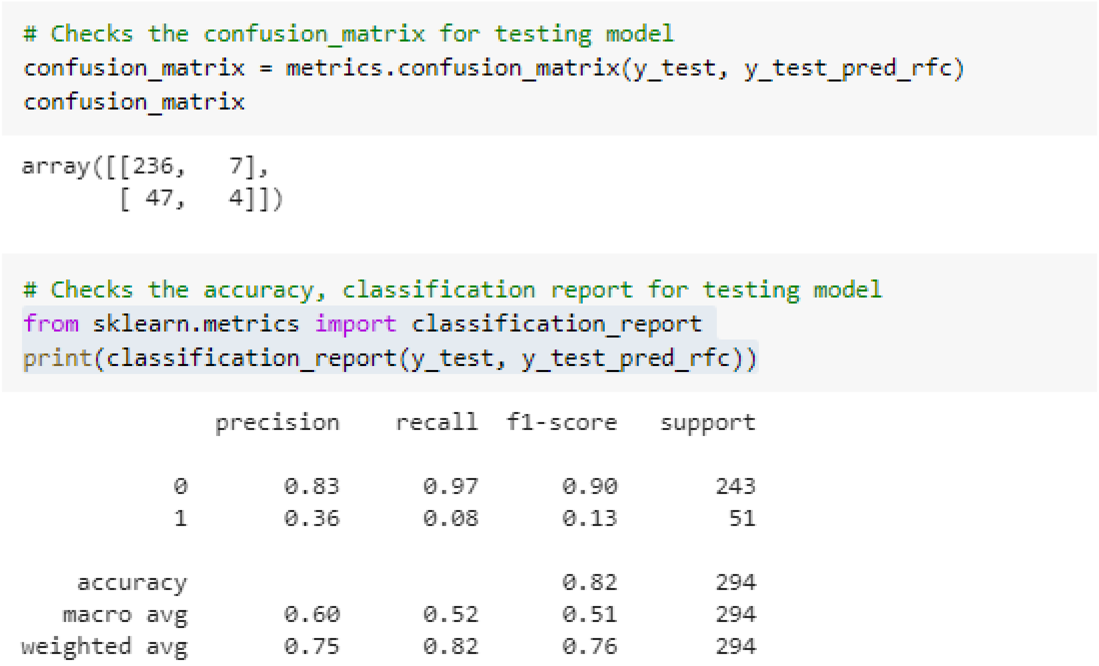
```
# Checks the confusion_matrix for testing model
confusion_matrix = metrics.confusion_matrix(y_test, y_test_pred_lr)
confusion_matrix
```

```
array([[243,   0],
       [ 51,   0]])
```

```
# Checks the accuracy, classification report for testing model
from sklearn.metrics import classification_report
print(classification_report(y_test, y_test_pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91       243
           1       0.00      0.00      0.00        51

    accuracy                           0.83       294
   macro avg       0.41      0.50      0.45       294
weighted avg       0.68      0.83      0.75       294
```

Fig 7.3.1 Predicting test in logistic regression



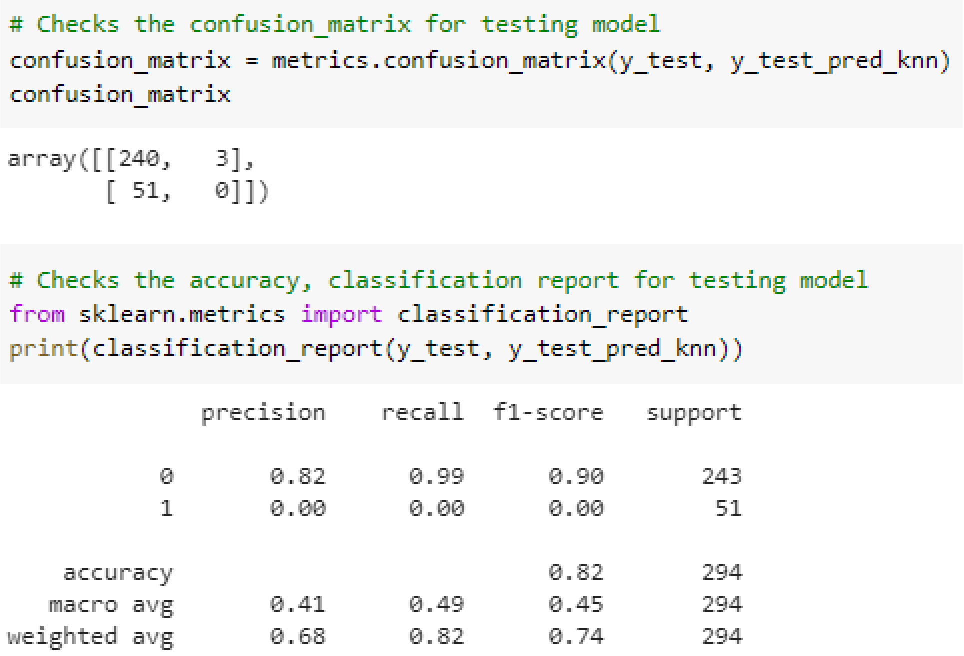
```
# Checks the confusion_matrix for testing model
confusion_matrix = metrics.confusion_matrix(y_test, y_test_pred_rfc)
confusion_matrix
```

```
array([[236,   7],
       [ 47,   4]])
```

```
# Checks the accuracy, classification report for testing model
from sklearn.metrics import classification_report
print(classification_report(y_test, y_test_pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.83      0.97      0.90       243
           1       0.36      0.08      0.13        51

    accuracy                           0.82       294
   macro avg       0.60      0.52      0.51       294
weighted avg       0.75      0.82      0.76       294
```

Fig 7.3.2 Predicting test in RandomForestClassifition



```
# Checks the confusion_matrix for testing model
confusion_matrix = metrics.confusion_matrix(y_test, y_test_pred_knn)
confusion_matrix
```

```
array([[240,   3],
       [ 51,   0]])
```

```
# Checks the accuracy, classification report for testing model
from sklearn.metrics import classification_report
print(classification_report(y_test, y_test_pred_knn))
```

```
              precision    recall  f1-score   support

           0       0.82      0.99      0.90       243
           1       0.00      0.00      0.00        51

    accuracy                           0.82       294
   macro avg       0.41      0.49      0.45       294
weighted avg       0.68      0.82      0.74       294
```

Fig 7.3.3 Predicting test in K-nearest neighbors(KNN)

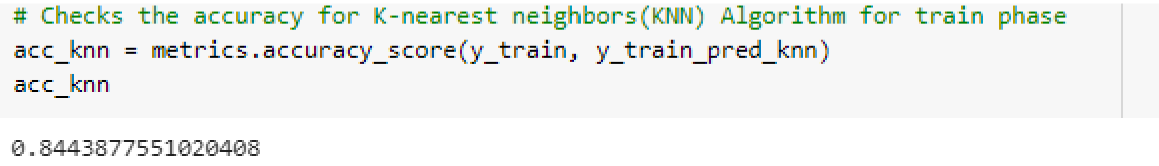
7.4 Validate the Models:

Model validation is the process of evaluating a trained model on test data set. This provides the generalization ability of a trained model. Here I provide a step by step approach to complete first iteration of model validation in minutes.

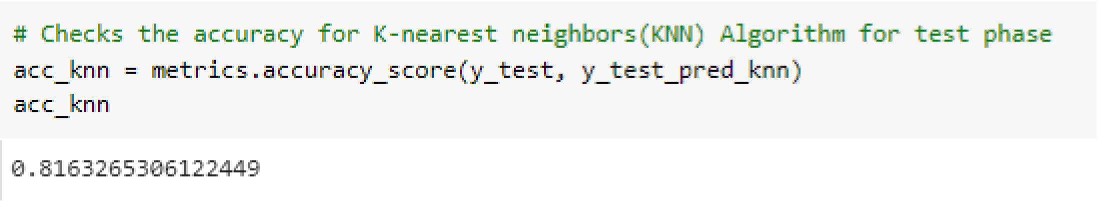
- The model are validate after completion of training and testing the model.

- Checking the accuracy scores as metrics to validate the models

- We have to check the accuracy among the models and validate best model among those.

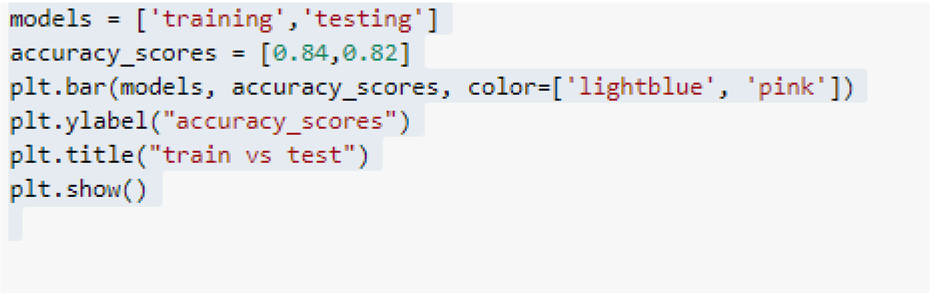


```
# Checks the accuracy for K-nearest neighbors(KNN) Algorithm for train phase
acc_knn = metrics.accuracy_score(y_train, y_train_pred_knn)
acc_knn

0.8443877551020408
```

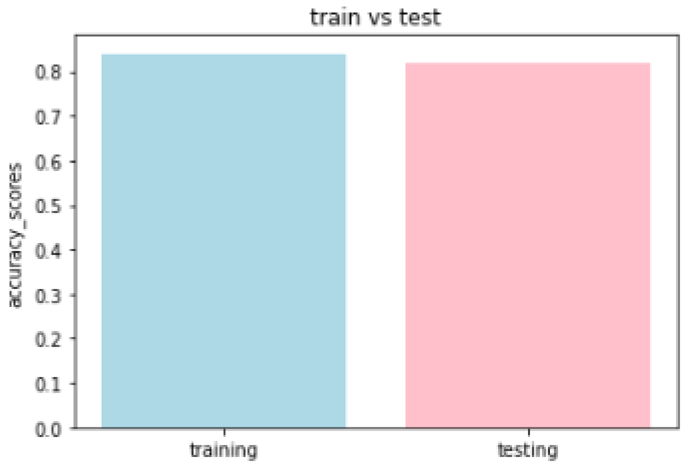


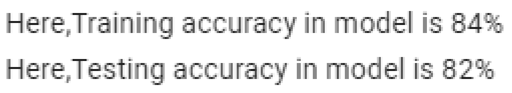
```
# Checks the accuracy for K-nearest neighbors(KNN) Algorithm for test phase
acc_knn = metrics.accuracy_score(y_test, y_test_pred_knn)
acc_knn

0.8163265306122449
```

Fig 7.4.1 Test and train accuracy in K-nearest neighbors(KNN)



```
models = ['training','testing']
accuracy_scores = [0.84,0.82]
plt.bar(models, accuracy_scores, color=['lightblue', 'pink'])
plt.ylabel("accuracy_scores")
plt.title("train vs test")
plt.show()
```

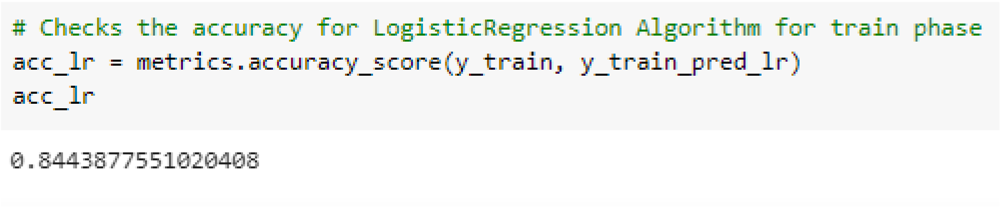




Here,Training accuracy in model is 84%

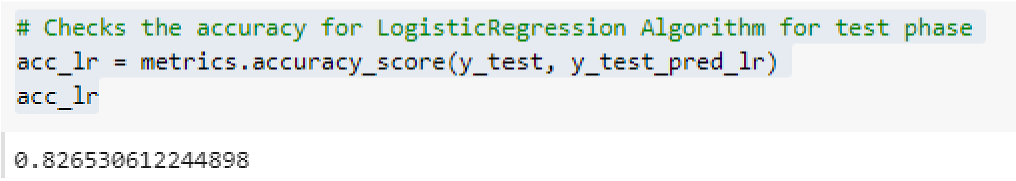
Here,Testing accuracy in model is 82%

Fig 7.4.2 test vs train in K-nearest neighbors(KNN)



```
# Checks the accuracy for LogisticRegression Algorithm for train phase
acc_lr = metrics.accuracy_score(y_train, y_train_pred_lr)
acc_lr
```

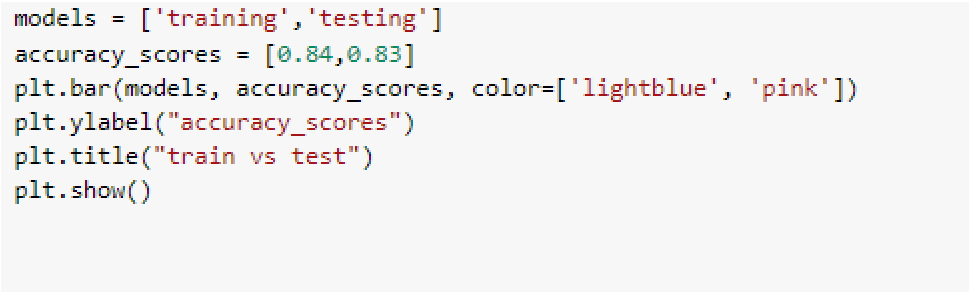
```
0.8443877551020408
```



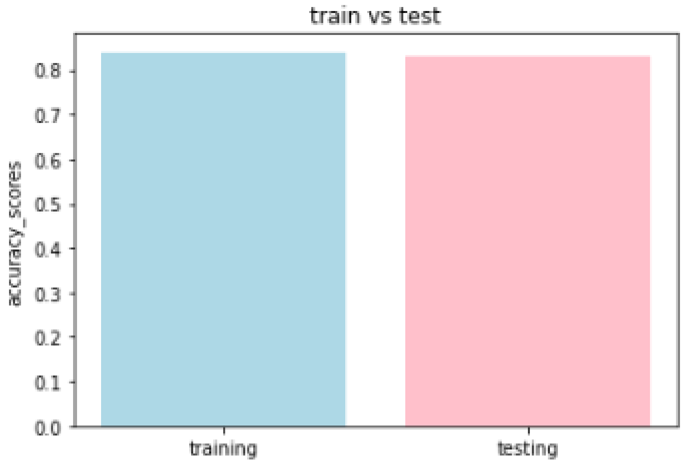
```
# Checks the accuracy for LogisticRegression Algorithm for test phase
acc_lr = metrics.accuracy_score(y_test, y_test_pred_lr)
acc_lr
```

```
0.826530612244898
```

Fig 7.4.3 Test and train accuracy in LogisticRegression



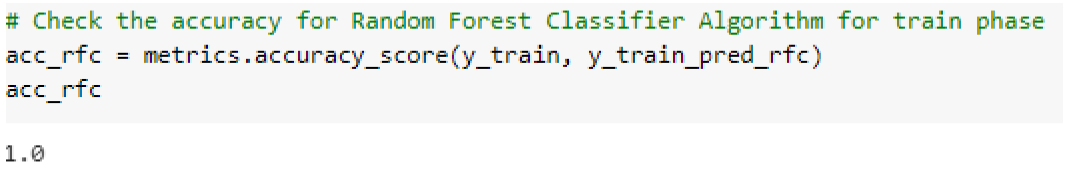
```
models = ['training','testing']
accuracy_scores = [0.84,0.83]
plt.bar(models, accuracy_scores, color=['lightblue', 'pink'])
plt.ylabel("accuracy_scores")
plt.title("train vs test")
plt.show()
```



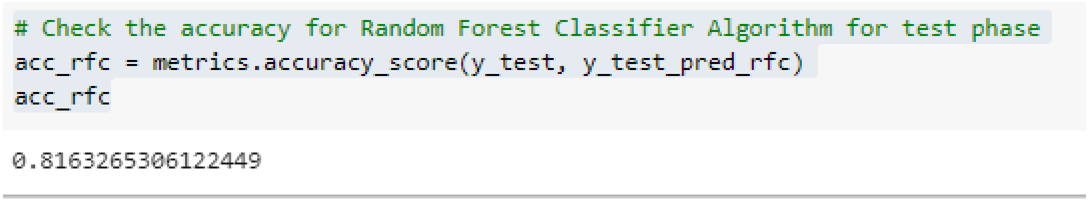
Here,Training accuracy in model is 84%

Here,Testing accuracy in model is 83%

Fig 7.4.4 test vs train in LogisticRegression

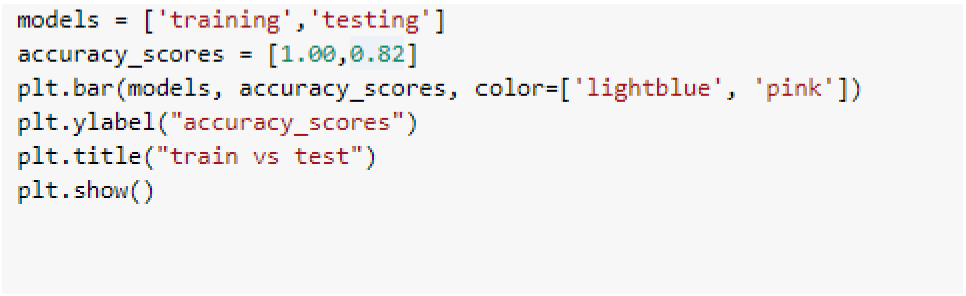


```
# Check the accuracy for Random Forest Classifier Algorithm for train phase
acc_rfc = metrics.accuracy_score(y_train, y_train_pred_rfc)
acc_rfc

1.0
```

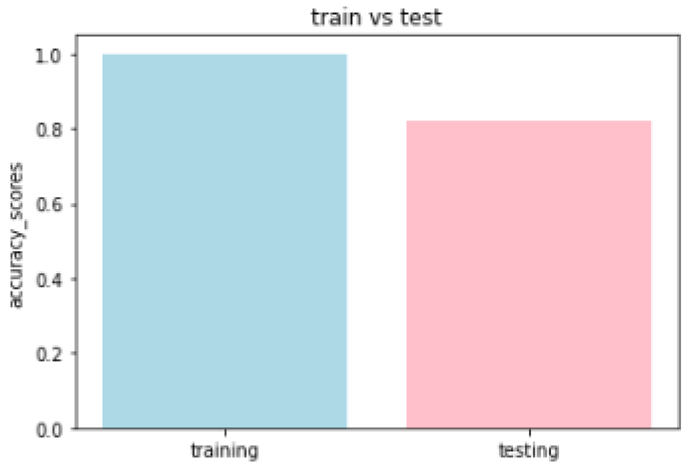


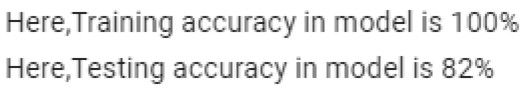
```
# Check the accuracy for Random Forest Classifier Algorithm for test phase
acc_rfc = metrics.accuracy_score(y_test, y_test_pred_rfc)
acc_rfc

0.8163265306122449
```

Fig 7.4.5 Test and train accuracy in Random Forest Classifier



```
models = ['training','testing']
accuracy_scores = [1.00,0.82]
plt.bar(models, accuracy_scores, color=['lightblue', 'pink'])
plt.ylabel("accuracy_scores")
plt.title("train vs test")
plt.show()
```





Here,Training accuracy in model is 100%

Here,Testing accuracy in model is 82%

Fig 7.4.6 test vs train in Random Forest Classifier

7.5 Parameter Tuning:

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters, and have to be tuned so that the model can optimally solve the machine learning problem. Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss. Cross-validation is often used to estimate this generalization performance.

i. Grid search:

The traditional way of performing hyperparameter optimization has been grid search, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.

since the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters, manually set bounds and discretization may be necessary before applying grid search.

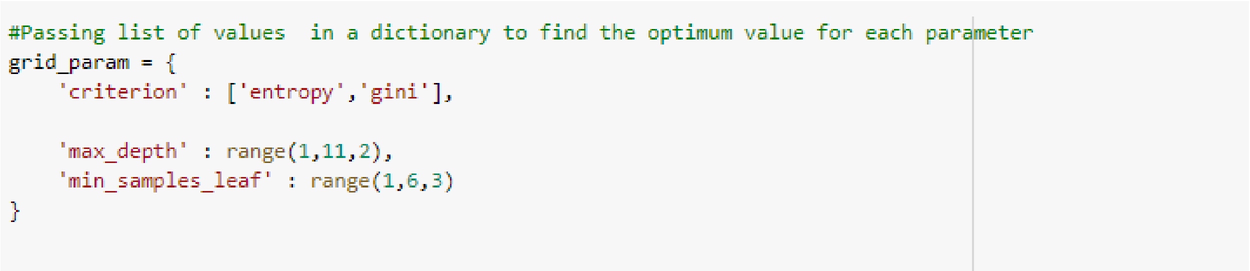
For example, a typical soft-margin SVM classifier equipped with an RBF kernel has at least two hyperparameters that need to be tuned for good performance on unseen data: a regularization constant *C* and a kernel hyperparameter γ. Both parameters are continuous, so to perform grid search, one selects a finite set of "reasonable" values for each, say

ii.Random search:

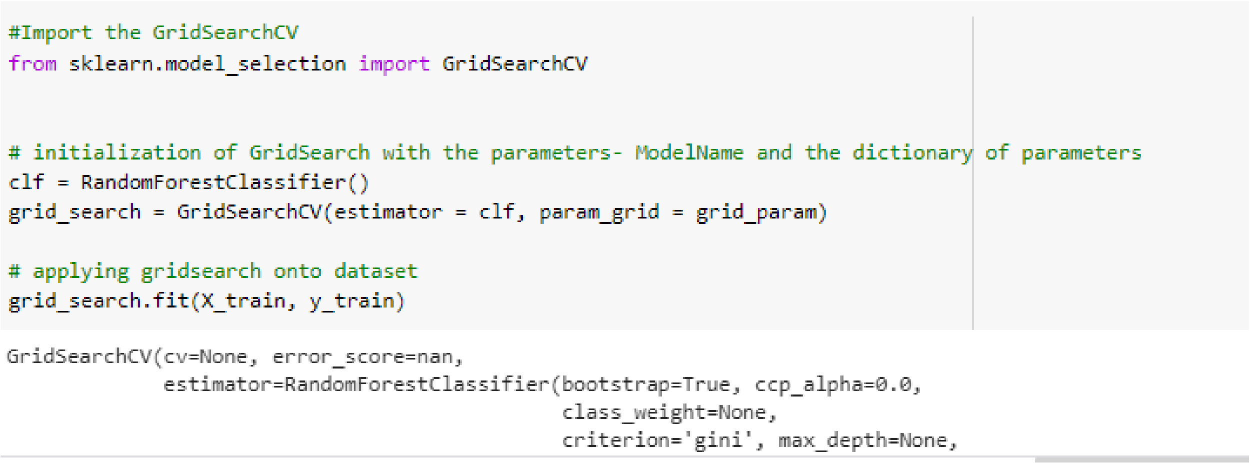
Random Search replaces the exhaustive enumeration of all combinations by selecting them randomly. This can be simply applied to the discrete setting described above, but also generalizes to continuous and mixed spaces. It can outperform Grid search, especially when only a small number of hyperparameters affects the final performance of the machine learning algorithm In this case, the optimization problem is said to have a low intrinsic dimensionality. Random Search is also embarrassingly parallel, and additionally allows the inclusion of prior knowledge by specifying the distribution from which to sample.

iii.Bayesian optimization:

Bayesian optimization is a global optimization method for noisy black-box functions. Applied to hyperparameter optimization, Bayesian optimization builds a probabilistic model of the function mapping from hyperparameter values to the objective evaluated on a validation set. By iteratively evaluating a promising hyperparameter configuration based on the current model, and then updating it, Bayesian optimization, aims to gather observations revealing as much information as possible about this function and, in particular, the location of the optimum. It tries to balance exploration (hyperparameters for which the outcome is most uncertain) and exploitation (hyperparameters expected close to the optimum). In practice, Bayesian optimization has been shown to obtain better results in fewer evaluations compared to grid search and random search, due to the ability to reason about the quality of experiments before they are run.



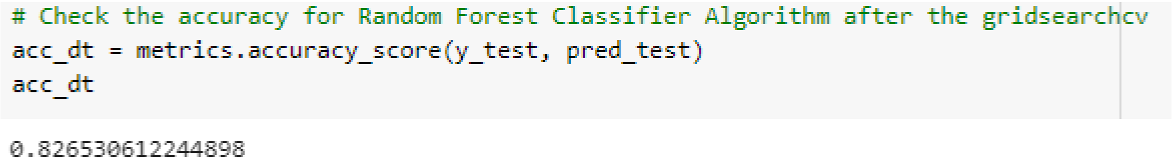
```
#Passing list of values  in a dictionary to find the optimum value for each parameter
grid_param = {
    'criterion' : ['entropy','gini'],

    'max_depth' : range(1,11,2),
    'min_samples_leaf' : range(1,6,3)
}
```



```
#Import the GridSearchCV
from sklearn.model_selection import GridSearchCV


# initialization of GridSearch with the parameters- ModelName and the dictionary of parameters
clf = RandomForestClassifier()
grid_search = GridSearchCV(estimator = clf, param_grid = grid_param)

# applying gridsearch onto dataset
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=None, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
```

Fig 7.5.1 grid search cv



```
# Check the accuracy for Random Forest Classifier Algorithm after the gridsearchcv
acc_dt = metrics.accuracy_score(y_test, pred_test)
acc_dt
```

```
0.826530612244898
```

Fig 7.5.2 accuracy of test using grid search cv i.e accuracy is increased in Random Forest Classifier

7.6 Predictions from raw data:

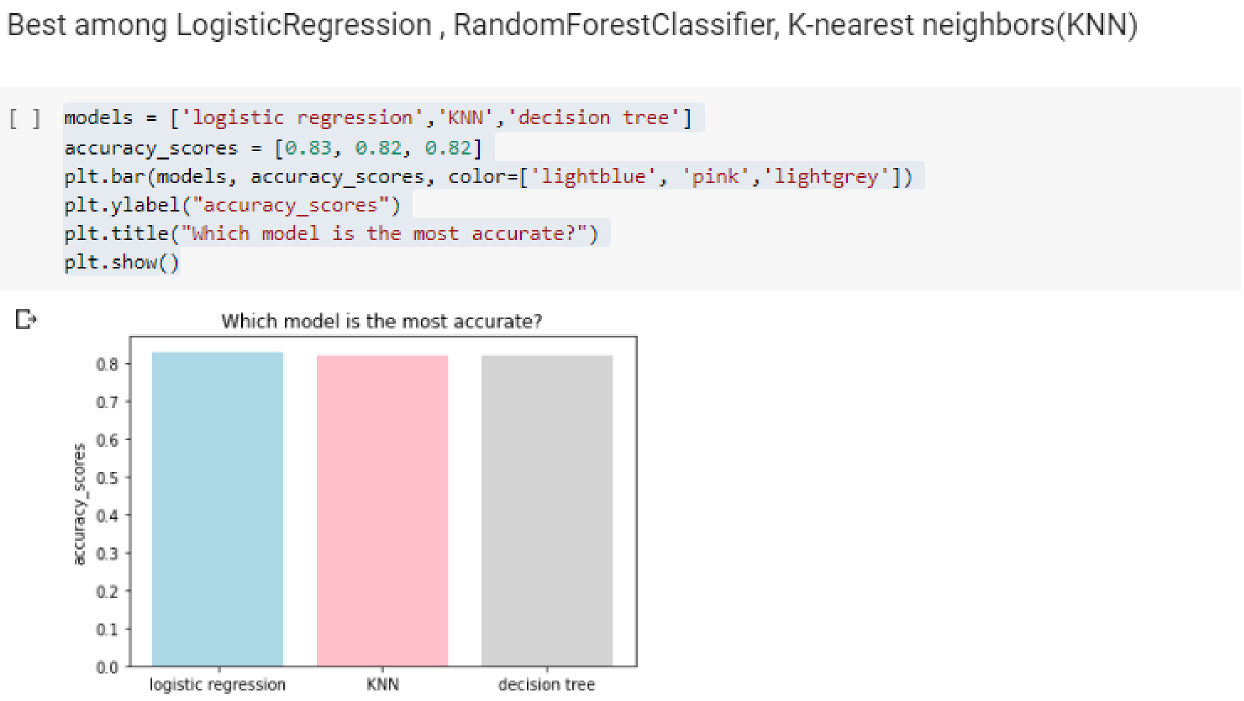
- After completion of accuracy we have to predict best model among them onto raw data
- Load the test dataset to predict

- Check the rows and columns and also shape

- After completion predict using best algorithm onto train dataset

- Add new column of predicted data i.e output

- Check the relation between test and train dataset to verify output



Best among LogisticRegression , RandomForestClassifier, K-nearest neighbors(KNN)

```
[ ] models = ['logistic regression','KNN','decision tree']
    accuracy_scores = [0.83, 0.82, 0.82]
    plt.bar(models, accuracy_scores, color=['lightblue', 'pink','lightgrey'])
    plt.ylabel("accuracy_scores")
    plt.title("Which model is the most accurate?")
    plt.show()
```

Fig 7.6.1 best accuracy model among algorithms

- Almost these LogisticRegression , RandomForestClassifier, K-nearest neighbors Algorithm.
- shows best Accuracy as LogisticRegression is somewhat very little high with

in 83% accuracy_score..

8. Conclusion

It is concluded after performing thorough Exploratory Data analysis which include Statistics models which are computed to get accuracy and also Heat maps which are computed to get a clear under standing of the data set and its come to point of getting the solution for the problem statement being , that the Employee turn over(attrition) is a major cost to an organization, and predicting turn over is at the forefront ofneeds of Human Resources(HR)in many organizations.

7.REFERENCES

- https://en.wikipedia.org/wiki/Machine_learning
- https://www.kaggle.com/iabhishekofficial/employee-attrition-classification/kernels
- https://mode.com/blog/python-data-visualization-libraries/
- https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/
- https://builtin.com/data-science/random-forest-algorithm
- https://towardsdatascience.com/supervised-machine-learning-model-validation-a-step-by-step-approach-771109ae0253