MACHINE LEARNING-(EMPLOYEE ATTRITITION)

Summer Internship Report Submitted in partial of the requirement for undergraduate degree of
Bachelor of Technology

In

Computer Science Engineering

By

Ravi Nishita

221710304046

Under the Guidance of

Department Of ComputerScience and Engineering

GITAM School of Technology

GITAM (Deemed to be University), Hyderabad-502329

DECLARATION

I submit this industrial training work entitled “EMPLOYEE ATTRITION” 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 Engineering”. I declare that it was carried out independently by me under the guidance 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 **RAVI NISHITA**

Date:13-07-2020 **221710304046**

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"EMPLOYEE ATTRITION**" is being submitted by **Ravi Nishita(221710304046)** in partial fulfillment of the requirement for the award of **Bachelor of Technology in ComputerScience Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2020.

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

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 ProVice Chancellor,GITAM Hyderabad Principal, GITAM Hyderabad.

I would like to thank respected, Head of the Department of Computer Science 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 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.

RAVI NISHITA

221710304046

ABSTRACT

Employee Attrition is a process in which the workforce dwindles at a company, following a period in which a number of people retire or resign, and are not replaced. A reduction in staff due to attrition is often called a hiring freeze and is seen as a less disruptive way to trim the workforce and reduce payroll than layoffs.The attrition rate has always been a sensitive issue for all organizations. Calculating employee turnover rate is not that simple as it seems to be. No common formula can be used by all the organizations. A formula had to be devised keeping in view the nature of the business and different job functions.

This paper investigates the performance of naïve bayes, random forest and logistic regression on highly skewed employee attrition data . Dataset of employee attrition contains 1470 rows and 34 columns. A hybrid technique of under-sampling and oversampling is carried out on the skewed data. The three techniques are applied on the raw and preprocessed data. The work is implemented in Python. The performance of the techniques is evaluated based on accuracy and is visulazised using heatmap of their respective classification report.

RAVI NISHITA

221710304046

TABLE OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

Figure 1 : The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

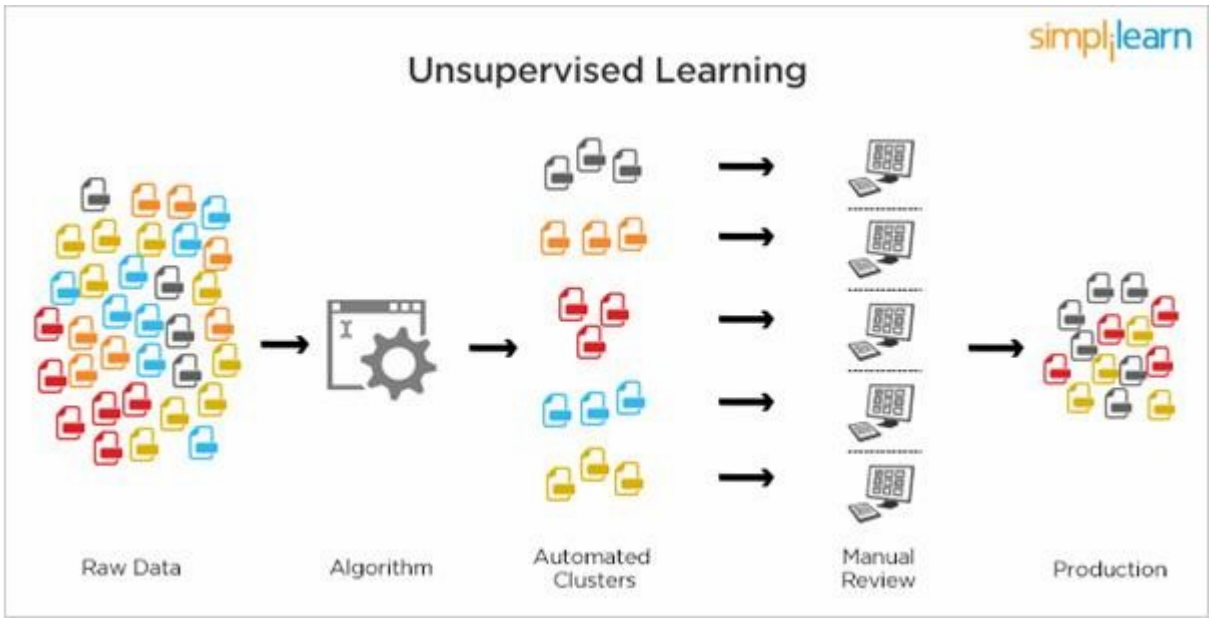


Figure 2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

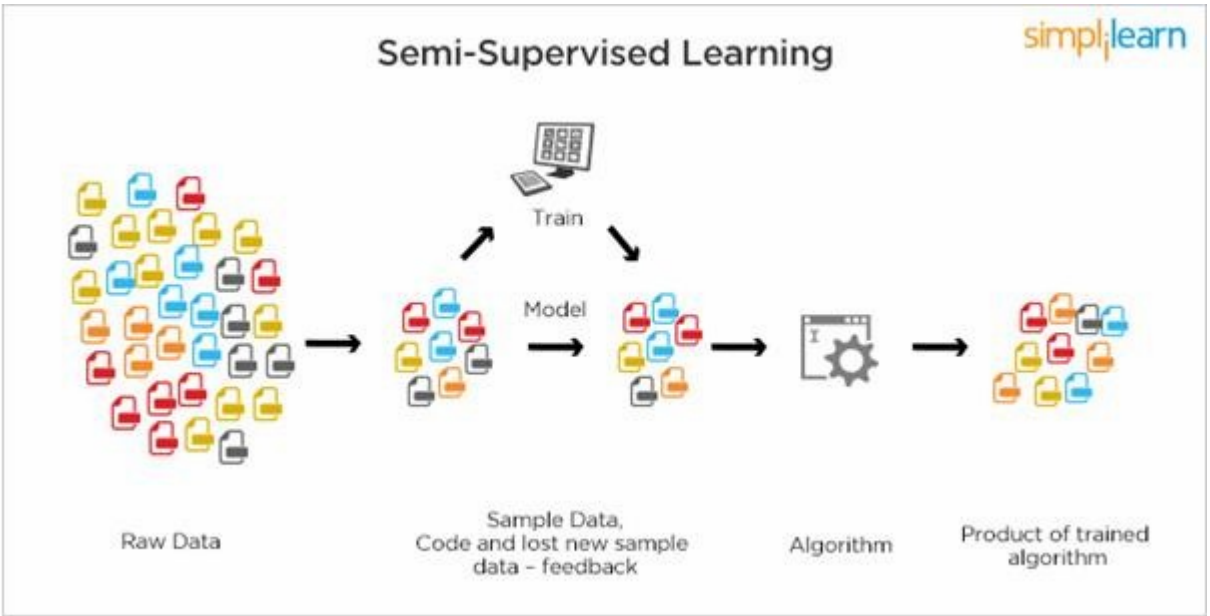


Figure 3 : Semi Supervised Learning

1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special

types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 2

PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 INTRODUCTION TO PYHTON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7 , it is generally called as python3

2.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.

- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

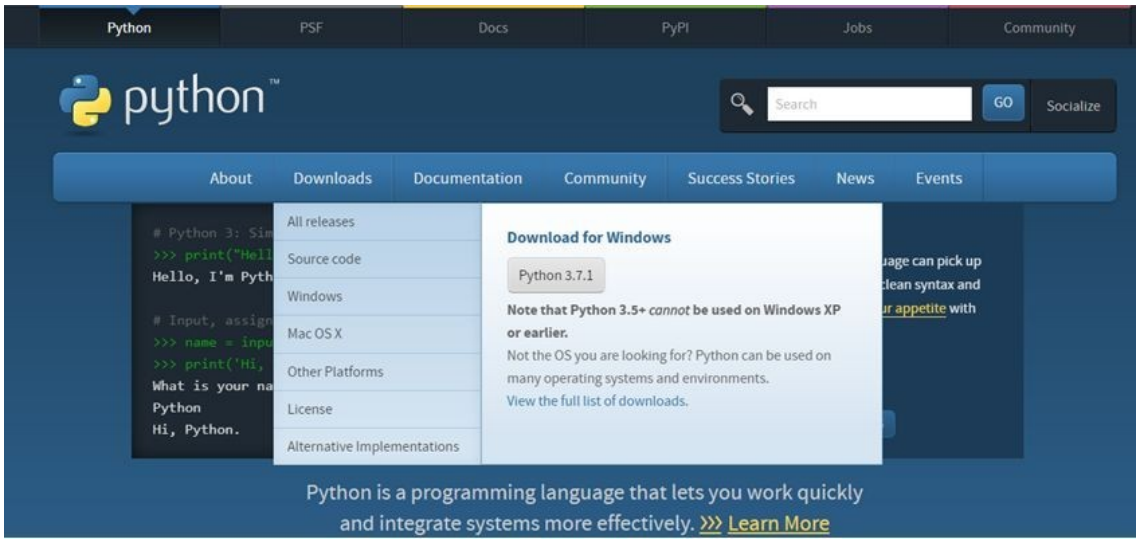


Figure 4 : Python download

2.4.2 Installation(using Anaconda):

- Python programs are also executed 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.

- In WINDOWS:
- In windows
    - Step 1: Open Anaconda.com/downloads in web browser.
    - Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
    - Step 3: select installation type(all users)
    - Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
    - Step 5: Open jupyter notebook (it opens in default browser)

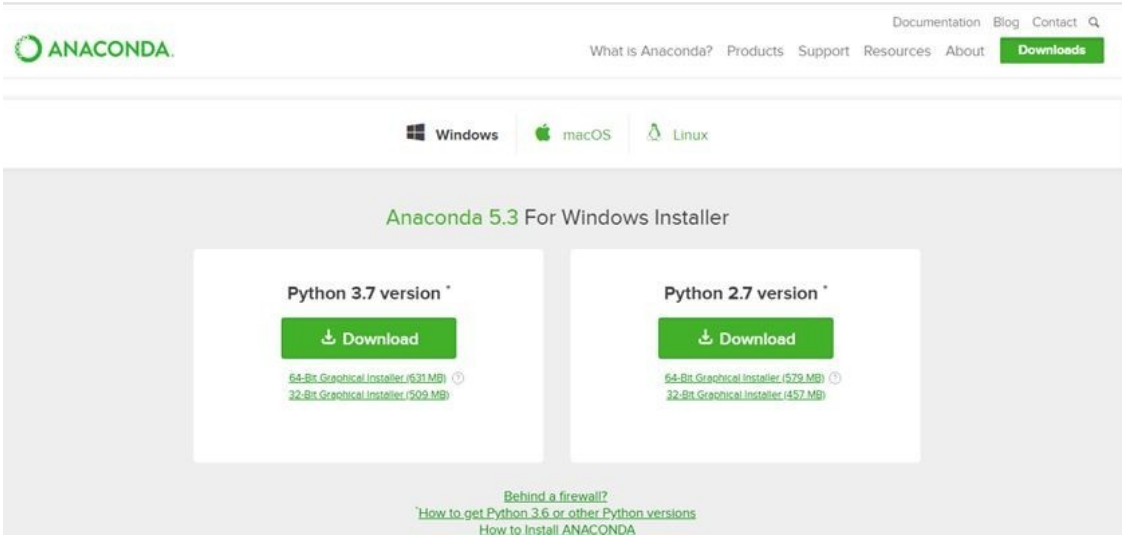


Figure 5 : Anaconda download

Figure 6 : Jupyter notebook

2.5 PYTHON 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.
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types –
    - Numbers
    - Strings
    - Lists

- Tuples
- Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets

([]).

- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays

or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

2.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

2.6.2 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.

2.7 PYTHON USING OOP's CONCEPTS:

2.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

- Defining a Class:
    - We define a class in a very similar way how we define a function.
    - Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is

indented like a functions body is.

```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Figure 7 : Defining a Class

2.7.2 _init____method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The init method has a special name that starts and ends with two underscores: init ().

CHAPTER 3

CASE STUDY

3.1 PROBLEM STATEMENT:

To uncover the factors that lead to Employee Attrition and to predict the number of employees who left and who stayed in the company based on the feature column Attrition using the following Machine Learning Algorithms

- RandomForest
- LogisticRegression
- Naive bayes

3.2 DATA SET

The data set consists of different features as

Age

Attrition

Education

Income

Education

Marital status

Experience

Work life balance

Job role

Job level

Overtime

Worklife balance

Years in current role

CHAPTER 4

4.1 PREPROCESSING OF THE DATA:

Preprocessing of the data actually involves the following steps:

4.1.1 GETTING THE DATASET:

We can get the data set from the database or

we can get the data from client.

4.1.2 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the algorithm.

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

Figure 8: importing packages

4.1.3IMPORTING THE DATA-SET:

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.

```
df = pd.read_csv('Employee_Attrition.csv')
df
```

```
## checking the shape
df.shape
```

```
(1470, 35)
```

Figure 9: reading and checking the shape

DATA CLEANING

Checking for null values duplicates and the description of the columns respectively

```
In [4]: #Getting the column data types
        df.dtypes
```

```
Out[4]: Age                        int64
        Attrition                 object
        BusinessTravel            object
        DailyRate                  int64
        Department                object
        DistanceFromHome           int64
        Education                  int64
        EducationField            object
        EmployeeCount              int64
        EmployeeNumber             int64
        EnvironmentSatisfaction    int64
        Gender                    object
        HourlyRate                 int64
        JobInvolvement             int64
        JobLevel                   int64
        JobRole                   object
        JobSatisfaction            int64
        MaritalStatus             object
        MonthlyIncome              int64
        MonthlyRate                int64
        NumCompaniesWorked         int64
        Over18                    object
        OverTime                  object
        PercentSalaryHike          int64
        PerformanceRating          int64
        RelationshipSatisfaction   int64
```

Figure 10: checking type

```
## getting the description
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	1470.0	36.923810	9.135373	18.0	30.00	36.0	43.00	60.0
DailyRate	1470.0	802.485714	403.509100	102.0	465.00	802.0	1157.00	1499.0
DistanceFromHome	1470.0	9.192517	8.106864	1.0	2.00	7.0	14.00	29.0
Education	1470.0	2.912925	1.024165	1.0	2.00	3.0	4.00	5.0
EmployeeCount	1470.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0
EmployeeNumber	1470.0	1024.865306	602.024335	1.0	491.25	1020.5	1555.75	2068.0
EnvironmentSatisfaction	1470.0	2.721769	1.093082	1.0	2.00	3.0	4.00	4.0
HourlyRate	1470.0	65.891156	20.329428	30.0	48.00	66.0	83.75	100.0
JobInvolvement	1470.0	2.729932	0.711561	1.0	2.00	3.0	3.00	4.0
JobLevel	1470.0	2.063946	1.106940	1.0	1.00	2.0	3.00	5.0
JobSatisfaction	1470.0	2.728571	1.102846	1.0	2.00	3.0	4.00	4.0
MonthlyIncome	1470.0	6502.931293	4707.956783	1009.0	2911.00	4919.0	8379.00	19999.0
MonthlyRate	1470.0	14313.103401	7117.786044	2094.0	8047.00	14235.5	20461.50	26999.0
NumCompaniesWorked	1470.0	2.693197	2.498009	0.0	1.00	2.0	4.00	9.0
PercentSalaryHike	1470.0	15.209524	3.659938	11.0	12.00	14.0	18.00	25.0

Figure 11: description

4.1.3 HANDLING MISSING VALUES:

Missing values can be handled in many ways using some inbuilt methods:

(a)dropna()

- dropna() function has a parameter called how which works as follows
- if how = ’all’ is passed then it drops the rows where all the columns of the particular row are missing
- if how = ’any’ is passed then it drops the rows where all the columns of the particular row are missing

(b)fillna()

(c)interpolate():

(d)mean and median imputation if any null values are present

```
#checking for null values
df.isna().sum()
```

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
```

Figure 12: checking the null values

```
# checking for null values
df.isnull().values.any()
```

```
False
```

```
## visual representation of null values using heatmap
sns.heatmap(df.isna())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1790aae8708>
```

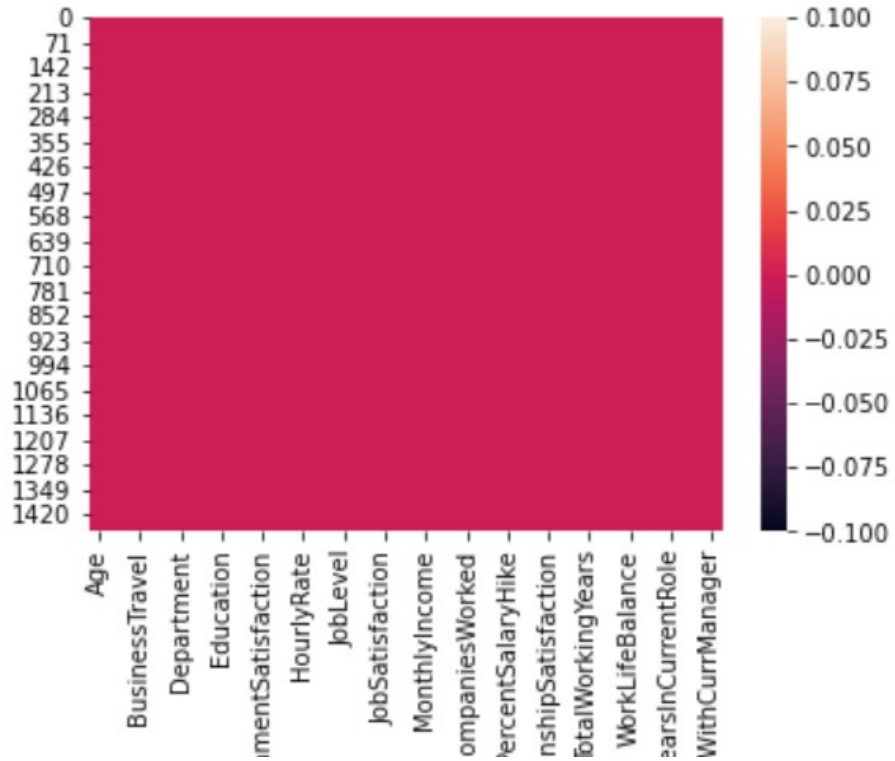


Figure 13: visualizing null values

```
: ## checking for duplicate values
  df.duplicated().sum()
```

```
: 0
```

```
: ## checking for unique values
  df.nunique()
```

```
JobSatisfaction              4
MaritalStatus                3
MonthlyIncome             1349
MonthlyRate               1427
NumCompaniesWorked          10
Over18                       1
OverTime                     2
PercentSalaryHike           15
PerformanceRating            2
RelationshipSatisfaction     4
StandardHours                1
StockOptionLevel             4
TotalWorkingYears           40
TrainingTimesLastYear        7
WorkLifeBalance              4
YearsAtCompany              37
YearsInCurrentRole          19
YearsSinceLastPromotion     16
YearsWithCurrManager        18
dtype: int64
```

Checking the counts of different features present in the dataset

Figure 14: unique values

```
df['JobInvolvement'].value_counts()
```

```
3    868
2    375
4    144
1     83
Name: JobInvolvement, dtype: int64
```

```
df['Education'].value_counts()
```

```
3    572
4    398
2    282
1    170
5     48
Name: Education, dtype: int64
```

```
#Get a count of the number of employee attrition employees who stayed (no) and who left (yes)
df['Attrition'].value_counts()
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

Figure 15: checking counts of features

4.1.4 CATEGORICAL DATA:

- Machine Learning models are based on equations, we need to replace the text by numbers. So that we can include the numbers in the equations.
- 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.
- Handling categorical data using dummies:

  In pandas library we have a method called get_dummies() which creates dummy variables for those categorical data in the form of 0's and 1's.

Once these dummies got created we have to concat this dummy set to our dataframe or we can add that dummy set to dataframe

In our dataset we have non numerical columns such as

- Attrition
- Business travel
- Education field
- Department

```
#Transform non-numeric columns into numerical columns
from sklearn.preprocessing import LabelEncoder## label encoder
for column in df.columns:
        if df[column].dtype == np.number:
            continue
        df[column] = LabelEncoder().fit_transform(df[column])
df
```

Figure 16 categorical transformation

Removing unwanted columns from the dataset such as

- StandardHours
- EmployeeCount
- EmployeeNumber
- Over18

```
#Removing unwanted columns columns
df = df.drop('StandardHours', axis = 1)
df = df.drop('EmployeeCount', axis = 1)
df = df.drop('EmployeeNumber', axis = 1)
df = df.drop('Over18', axis = 1)
```

Figure 17: dropping unwanted columns

4.1.5 Visualizations:

This plot gives the relation with the attrition which is yes or no

```
#Visualizing this count
sns.countplot(df['Attrition'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x179002a5d48>
```

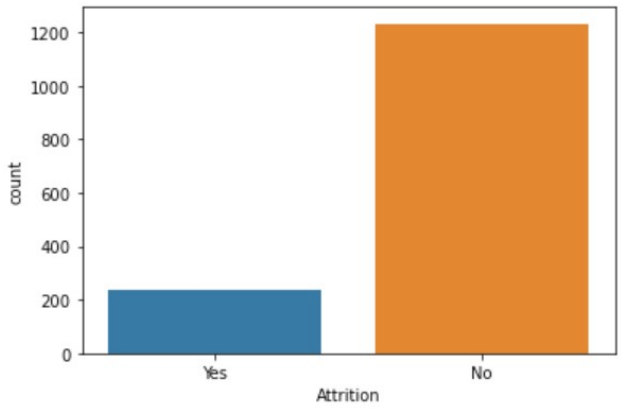


Figure 18: countplot

```
#Showing  the number of employees that left and stayed by age
fig_dims = (12, 4)
fig= plt.subplots(figsize=fig_dims)
sns.countplot(x='Age', hue='Attrition', data = df)
plt.title("age vs attrition")
```

```
Text(0.5, 1.0, 'age vs attrition')
```

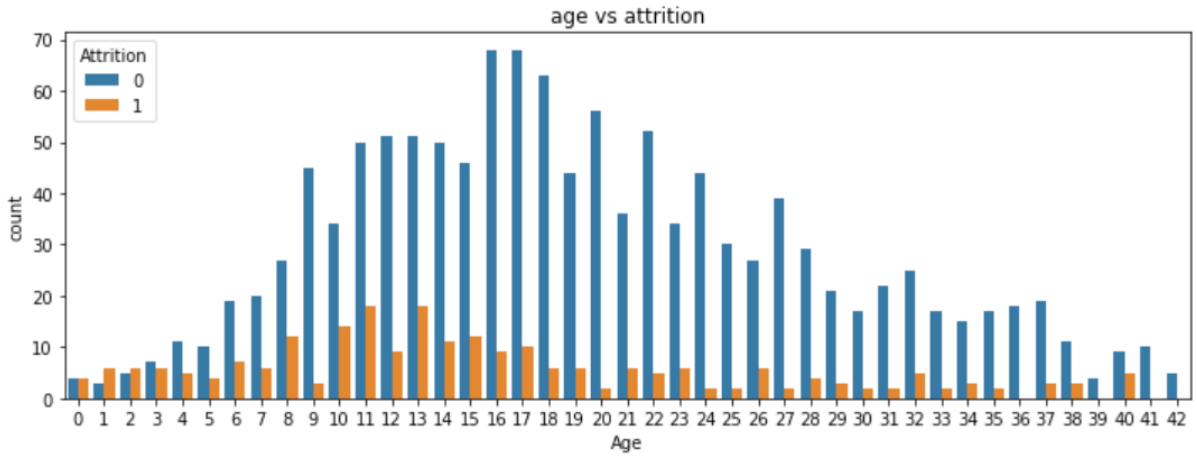


Figure 19: countplot of age vs attrition

Histogram for visualization

```
df.hist(edgecolor='black', linewidth=1.2, figsize=(20, 20));
```

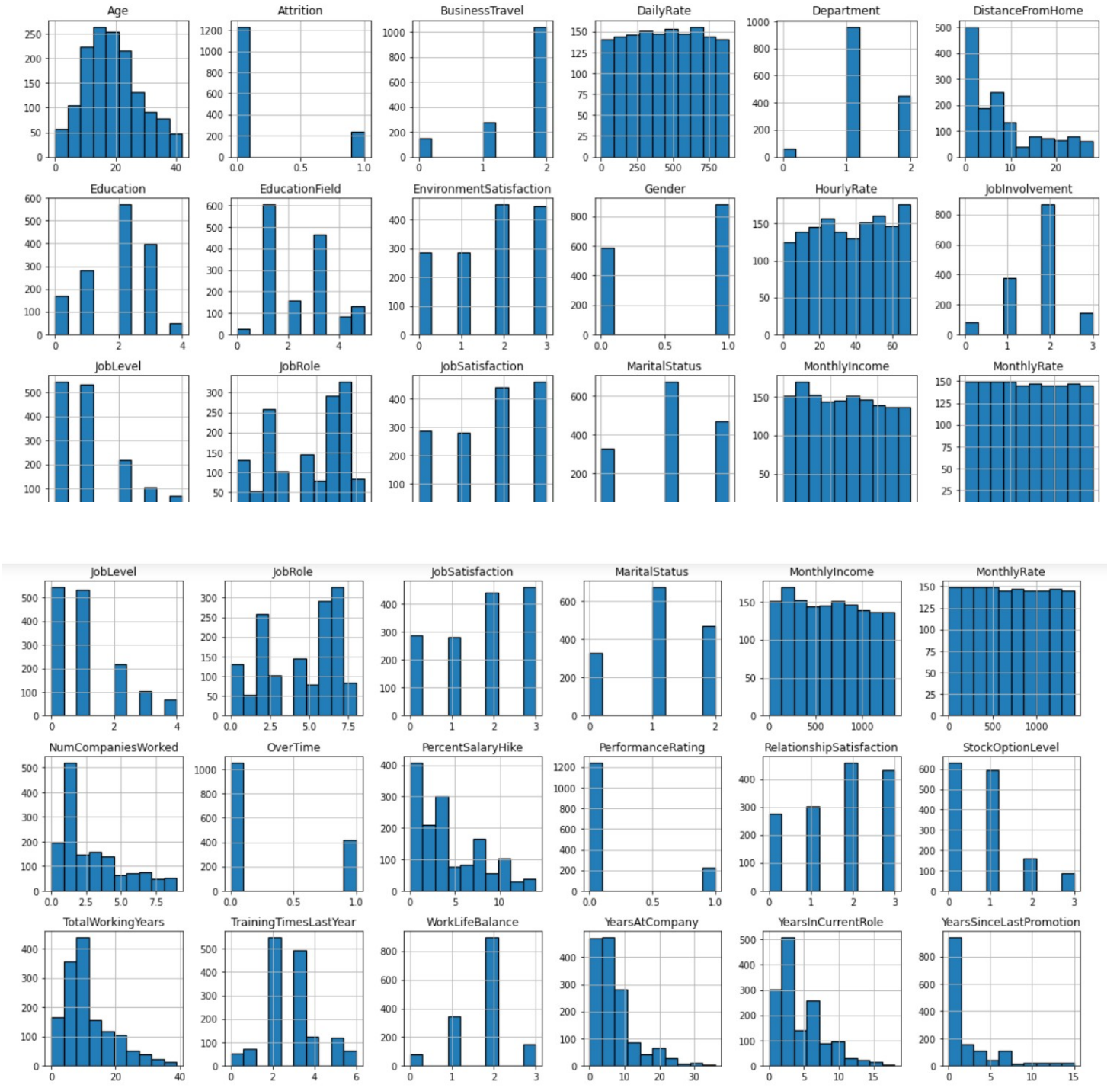


Figure 20: histogram

Distribution plot

```
fig,ax = plt.subplots(3,3, figsize=(10,10))
sns.distplot(df['TotalWorkingYears'], ax = ax[0,0])
sns.distplot(df['YearsAtCompany'], ax = ax[0,1])
sns.distplot(df['DistanceFromHome'], ax = ax[0,2])
sns.distplot(df['YearsInCurrentRole'], ax = ax[1,0])
sns.distplot(df['YearsWithCurrManager'], ax = ax[1,1])
sns.distplot(df['YearsSinceLastPromotion'], ax = ax[1,2])
sns.distplot(df['PercentSalaryHike'], ax = ax[2,0])
sns.distplot(df['YearsSinceLastPromotion'], ax = ax[2,1])
sns.distplot(df['TrainingTimesLastYear'], ax = ax[2,2])
plt.show()
```

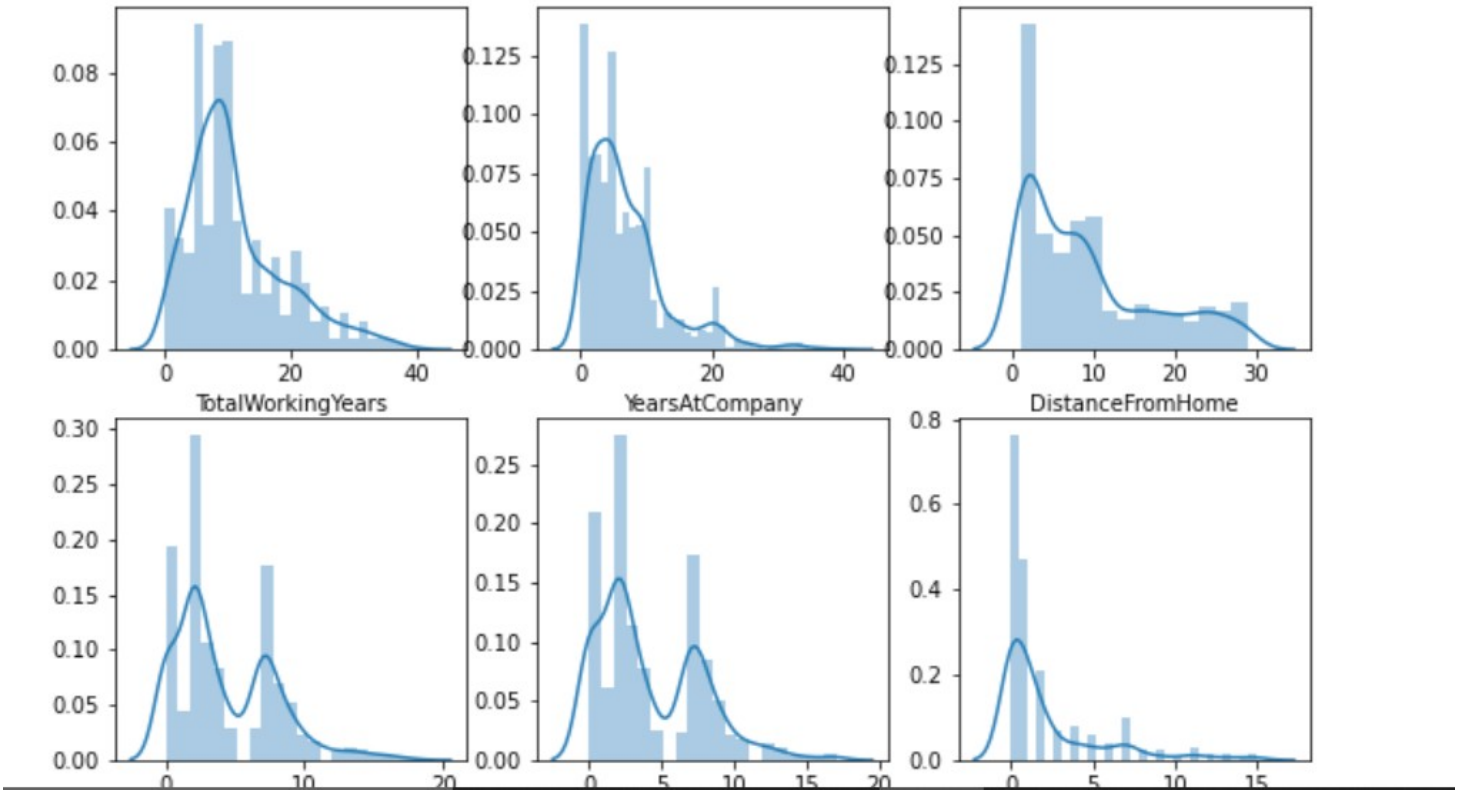
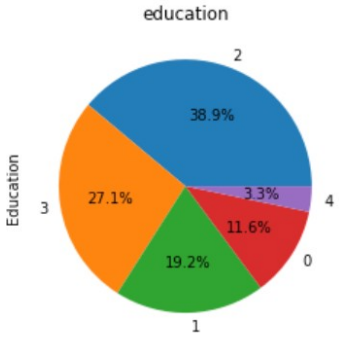


Figure 21: distribution plot

Relation between Employee's qualification and Attrition

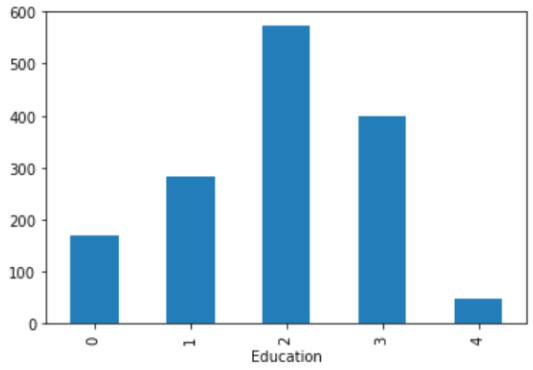
```
In [78]:
df.Education.value_counts().plot(kind = 'pie', autopct = '%1.1f%%')
plt.title('education')
```

```
Out[78]: Text(0.5, 1.0, 'education')
```




```
In [67]: df.groupby(['Education'])['Attrition'].count().plot(kind = 'bar')
```

```
Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x179048ad848>
```



Employees with Bachelors degree have highest attrition rate followed by employees with Masters degree

Figure 22: plot of education vs attrition

```
fig,ax = plt.subplots(2,2, figsize=(10,10))
sns.boxplot(df['Attrition'], df['MonthlyIncome'], ax = ax[0,0])  # Plot on 1st axes
sns.boxplot(df['Gender'], df['MonthlyIncome'], ax = ax[0,1])  # Plot on 2nd axes
plt.xticks( rotation=90)
sns.boxplot(df['Department'], df['MonthlyIncome'], ax = ax[1,0])       # Plot on 3rd axes
plt.xticks( rotation=90)
sns.boxplot(df['JobRole'], df['MonthlyIncome'], ax = ax[1,1])     # Plot on 4 the axes
plt.show()
```

Figure 23

The below plots represent the relation between monthly income and

- attrition

- gender
- Department
- job role

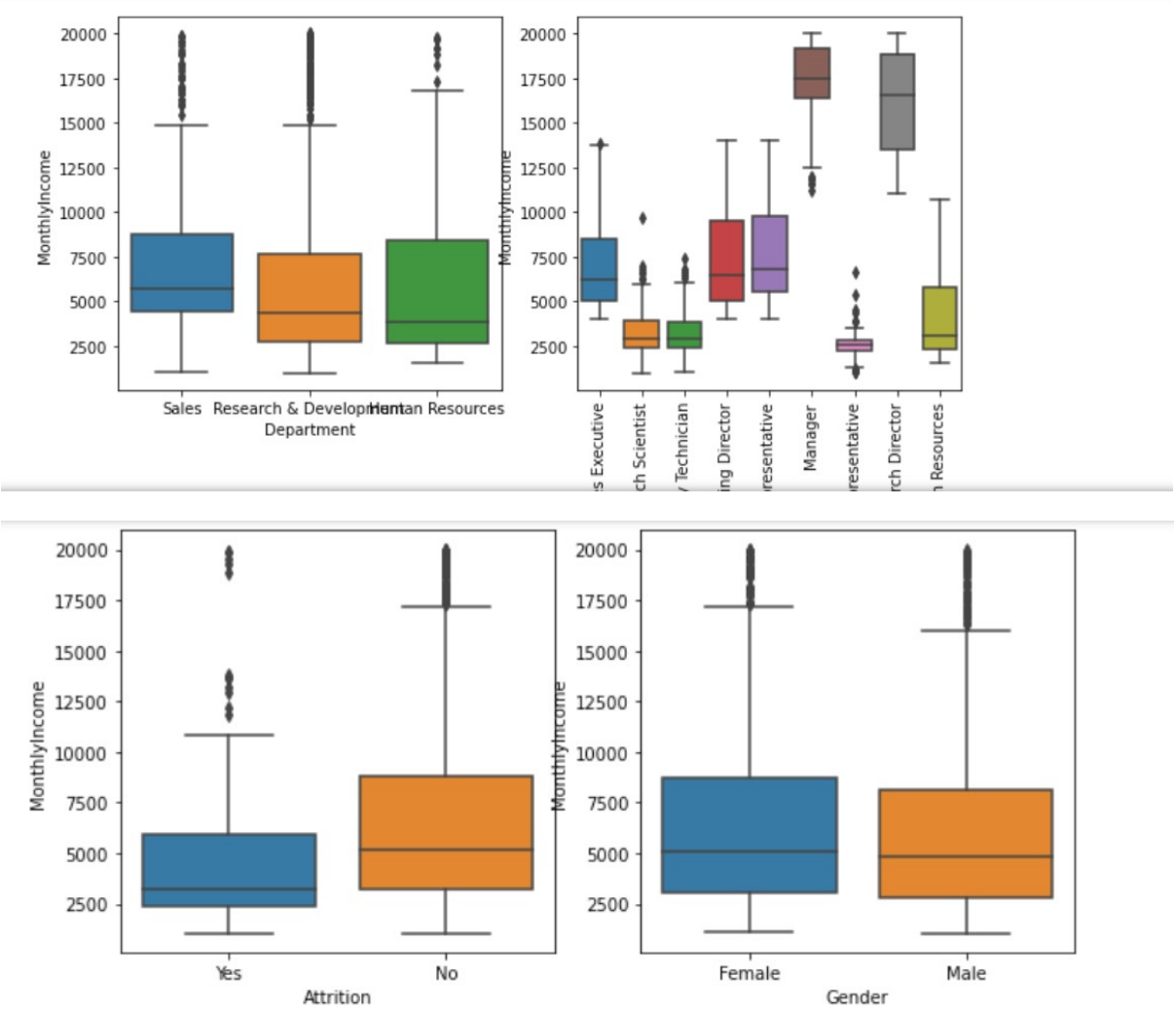


Figure 24: boxplot

4.1.6 Correlation:

Correlation coefficients quantify the association between variables or features of a dataset. These statistics are of high importance for science and technology, and Python has great tools that you can use to calculate them. SciPy, NumPy, and Pandas correlation methods are fast, comprehensive, and well-documented.

- We can also find the column which is effecting the R-Squared value so that we can perform operations on that specific column or we can remove that column , this can be done using pair plot.
- In pair plot we need to find the correlation between two variables .

Figure 25

```
|: #Getting  the correlation of the columns
   df.corr()
```

|:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction
Age	1.000000	-0.159205	0.024751	0.010557	-0.031882	-0.001686	0.208034	-0.040873	0.010146
Attrition	-0.159205	1.000000	0.000074	-0.056605	0.063991	0.077924	-0.031373	0.026846	-0.103369
BusinessTravel	0.024751	0.000074	1.000000	-0.004103	-0.009044	-0.024469	0.000757	0.023724	0.004174
DailyRate	0.010557	-0.056605	-0.004103	1.000000	0.007796	-0.004738	-0.016550	0.038129	0.018462
Department	-0.031882	0.063991	-0.009044	0.007796	1.000000	0.017225	0.007996	0.013720	-0.019395
DistanceFromHome	-0.001686	0.077924	-0.024469	-0.004738	0.017225	1.000000	0.021042	0.002013	-0.016075
Education	0.208034	-0.031373	0.000757	-0.016550	0.007996	0.021042	1.000000	-0.039592	-0.027128
EducationField	-0.040873	0.026846	0.023724	0.038129	0.013720	0.002013	-0.039592	1.000000	0.043163
EnvironmentSatisfaction	0.010146	-0.103369	0.004174	0.018462	-0.019395	-0.016075	-0.027128	0.043163	1.000000
Gender	-0.036311	0.029453	-0.032981	-0.011194	-0.041583	-0.001851	-0.016547	-0.002504	0.000508
HourlyRate	0.024287	-0.006846	0.026528	0.022916	-0.004144	0.031131	0.016775	-0.021941	-0.049857
JobInvolvement	0.029820	-0.130016	0.039062	0.046072	-0.024586	0.008783	0.042438	-0.002655	-0.008278
JobLevel	0.509604	-0.169105	0.019311	0.003355	0.101963	0.005303	0.101589	-0.044933	0.001212
JobRole	-0.122427	0.067151	0.002724	-0.009127	0.662431	-0.001015	0.004236	0.015599	-0.017321
JobSatisfaction	-0.004892	-0.103481	-0.033962	0.030712	0.021001	-0.003669	-0.011296	-0.034401	-0.006784

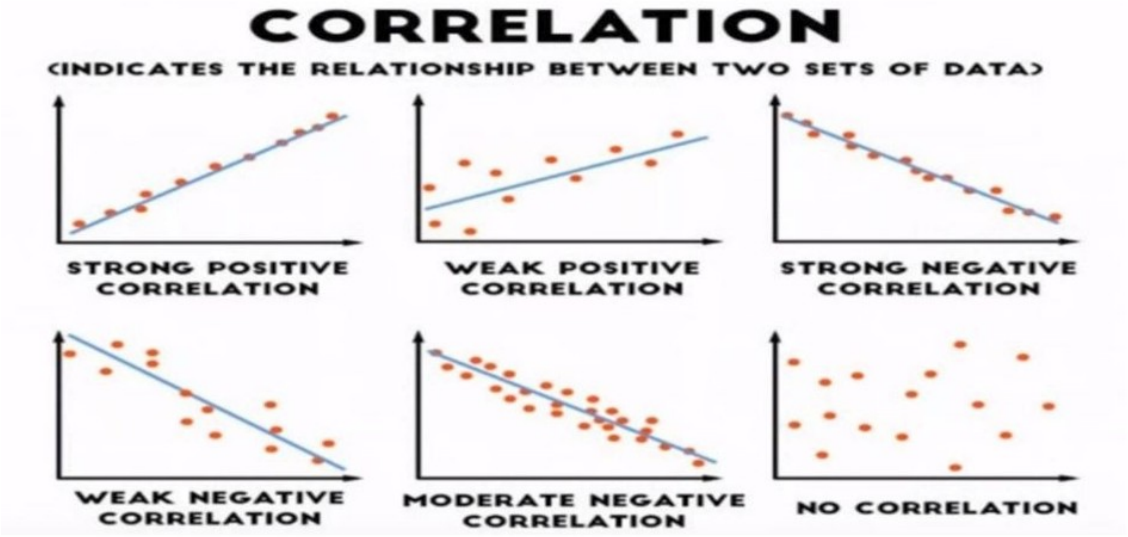

CORRELATION
(INDICATES THE RELATIONSHIP BETWEEN TWO SETS OF DATA)
STRONG POSITIVE CORRELATION
WEAK POSITIVE CORRELATION
STRONG NEGATIVE CORRELATION
WEAK NEGATIVE CORRELATION
MODERATE NEGATIVE CORRELATION
NO CORRELATION

Figure 26

Visual representation of correlation using heatmap

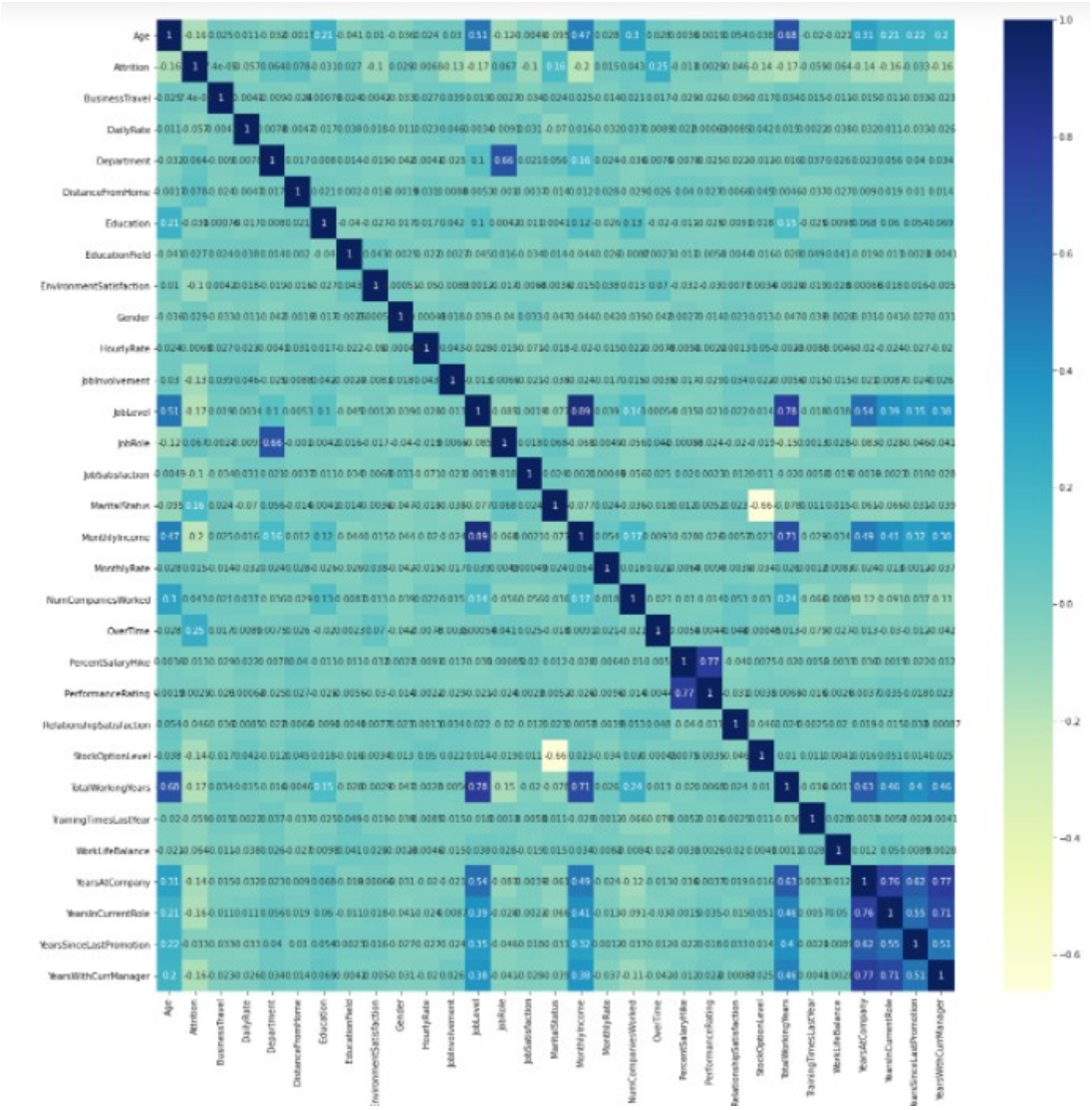


Figure 27: visualization of correlation

Smote technique

Smote to oversample due to the skewness in target.Since we have already noted the severe imbalance in the values within the target variable,implementing the SMOTE method in the dealing with this skewed value via the imblearn Python package.

```
: from imblearn.combine import SMOTETomek
  smk=SMOTETomek(random_state=100)
  X,y=smk.fit_sample(df.drop(['Attrition'],axis=1),df['Attrition'])
```

```
y.value_counts()
```

```
1    1181
0    1181
Name: Attrition, dtype: int64
```

```
sns.countplot(y)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1790aa92408>
```

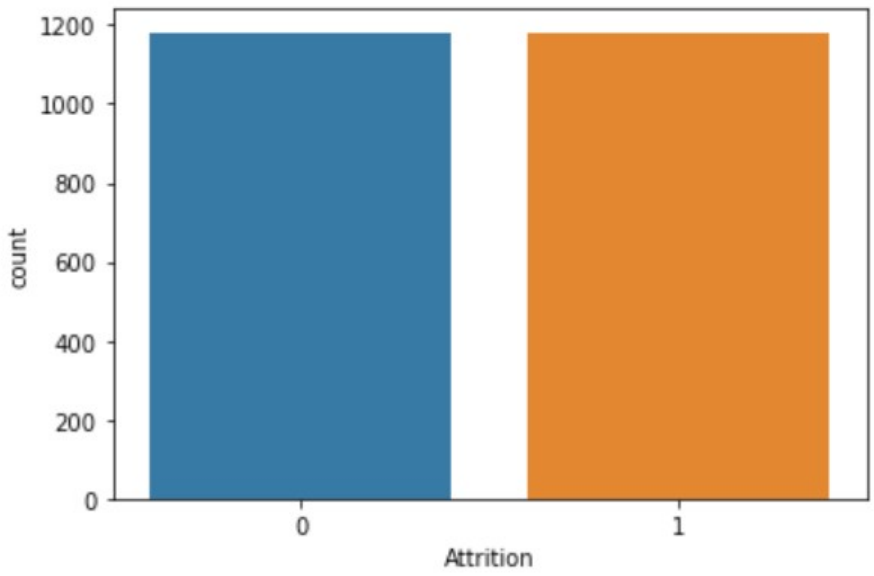


Figure 28

CHAPTER 5

BUILDING THE MODEL

5.1 Testing and training:

Train/Test is a method to measure the accuracy of your model.It is called Train/Test because you split the the data set into two sets: a training set and a testing set.You train the model using the training set.You test the model using the testing set.

- 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 (i.e. train data = 75%, test data =25% or train data = 80% , test data= 20%)
- 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(we need to mention the variables)

splitting the data into test and train

```
# Split the dataset into 75% Training set and 25% Testing set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

```
#printing the shapes of train and test data respectively
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1653, 30)
(709, 30)
(1653,)
(709,)
```

Figure 28

5.2 Implementing algorithms:

5.2.1 Random Forest Algorithm:

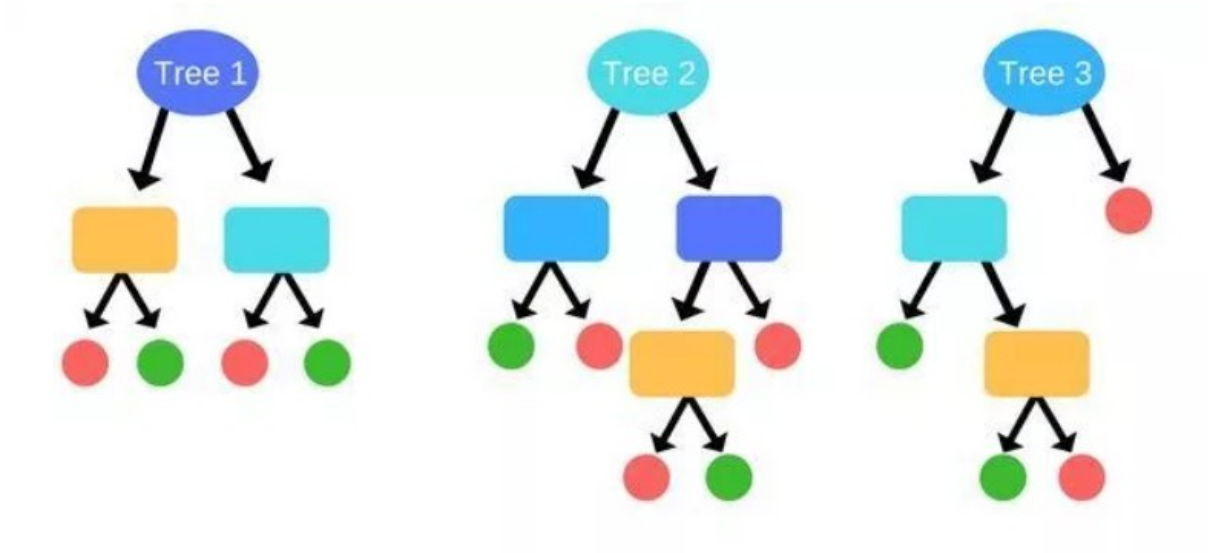


Figure 29

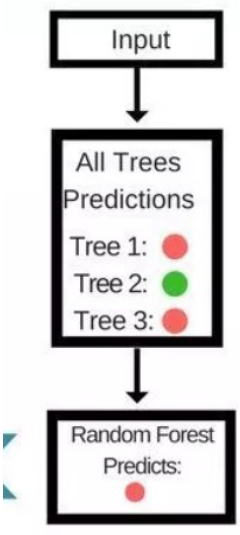


Figure 30

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Random decision forests correct for decision trees' habit of overfitting to their training set.

The first algorithm for random decision forests was created by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg

```
]: #Using Random Forest Classification algorithm
   from sklearn.ensemble import RandomForestClassifier
   from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
   rfc = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',random_state=0)## creating object for class
   rfc.fit(X_train, y_train)

]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)

]: y_pred_train=rfc.predict(X_train)
```

Figure 31

Predicting the values for the test data:

```
37]: #Showing  the confusion matrix and accuracy for  the model on the test data
     from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test,y_pred_test)
     TN = cm[0][0]
     TP = cm[1][1]
     FN = cm[1][0]
     FP = cm[0][1]
     print(cm)
     result=((TP + TN) / (TP + TN + FN + FP))
     print('Model testing accuracy=',result)

     [[263  30]
      [ 36 262]]
     Model testing accuracy= 0.8883248730964467

38]: print(accuracy_score(y_test,y_pred_test))## test value
     confusion_matrix(y_test, y_pred_test)

     0.8883248730964467

38]: array([[263,  30],
            [ 36, 262]], dtype=int64)
```

Figure 32

Printing the classification report for random forest testing algorithm with accuracy as the metric and visualizing the onfusion matrix using heatmap

```
print(classification_report(y_test,y_pred_test))
```

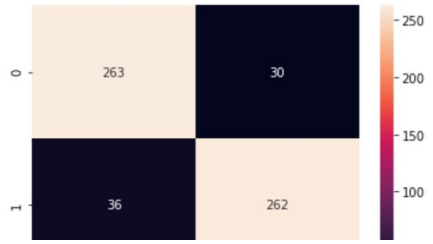
```
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       293
           1       0.90      0.88      0.89       298

    accuracy                           0.89       591
   macro avg       0.89      0.89      0.89       591
weighted avg       0.89      0.89      0.89       591
```

```
sns.heatmap(confusion_matrix(y_test,y_pred_test), annot=True, fmt='d') ## visual representation of classification report
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1790977c408>
```



The accuracy for the random forest algorithm was 89%.

Figure 33

5.2.2 Logistic Regression Algorithm:

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable(or output), y, can take only discrete values for given set of features(or inputs), X.

Contrary to popular belief, logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

$$g(z) = \frac{1}{1+e^{-z}}$$

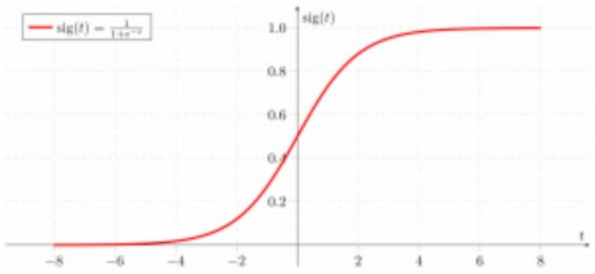


Figure 33

Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself.

Logistic regression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()## creating object
lr.fit(X_train,y_train)
y_train_pred = lr.predict(X_train)
```

```
accuracy_score(y_train, y_train_pred)## train data
```

```
0.7221908526256352
```

```
lr.fit(X_test, y_test)
y_test_pred = lr.predict(X_test);
```

```
print(accuracy_score(y_test, y_test_pred))## test value
confusion_matrix(y_test, y_test_pred)
```

```
0.7411167512690355
```

```
array([[224,  69],
       [ 84, 214]], dtype=int64)
```

Figure 34

Printing the classification report for random forest testing algorithm with accuracy as the metric and visualizing the onfusion matrix using heatmap

```
print(classification_report(y_test, y_test_pred))
sns.heatmap(confusion_matrix(y_test,y_test_pred),fmt='d', annot=True)## visual representation of classification report
```

```
              precision    recall  f1-score   support

           0       0.73      0.76      0.75       293
           1       0.76      0.72      0.74       298

    accuracy                           0.74       591
   macro avg       0.74      0.74      0.74       591
weighted avg       0.74      0.74      0.74       591
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1790a19b748>
```

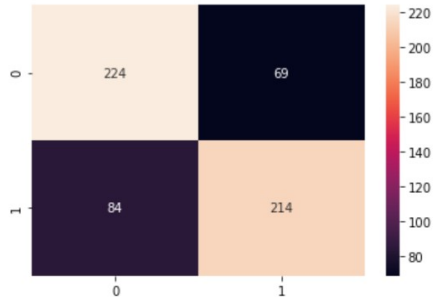


Figure 35

The accuracy for the logistic regression algorithm was 74%.

5.2.3 Naive Bayes Algorithm:

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem**.** It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

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

Figure 36

where A and B are events and P(B) ? 0.

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.
- P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:

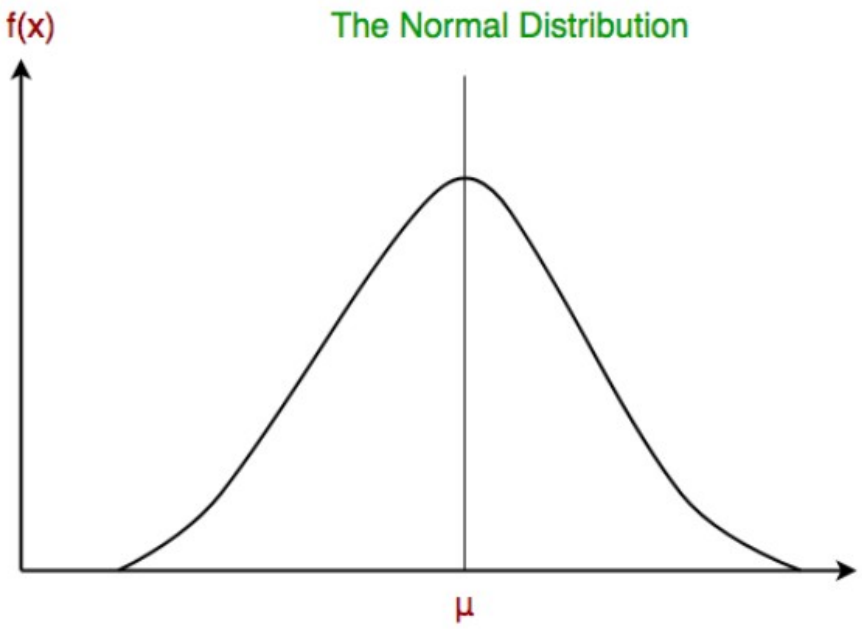


Figure 37

The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Now, we look at an implementation of Gaussian Naive Bayes classifier using scikit-learn.

```
from sklearn.naive_bayes import GaussianNB
gn = GaussianNB() ## creating object
gn.fit(X_train, y_train)
train_Pred = gn.predict(X_train)
```

```
confusion_matrix(y_train,train_Pred)
```

```
array([[599, 289],
       [182, 701]], dtype=int64)
```

```
accuracy_score(y_train,train_Pred)## train value
```

```
0.7340485601355167
```

```
test_Pred = gn.predict(X_test)
```

Printing the classification report for random forest testing algorithm with accuracy as the metric and visualizing the onfusion matrix using heatmap

```
accuracy_score(y_test,test_Pred) # test value
```

```
0.7529610829103215
```

```
sns.heatmap(confusion_matrix(y_test,test_Pred), annot=True,fmt='d')## visual representation of classification report
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1790c3fd108>
```

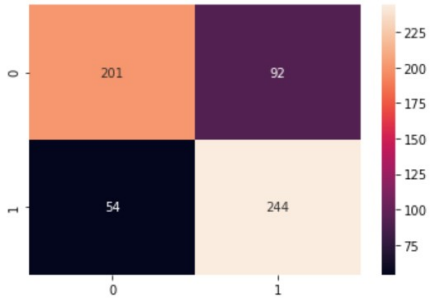


Figure 38

```
confusion_matrix(y_test,test_Pred)
print(classification_report(y_test,test_Pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.69      0.73       293
           1       0.73      0.82      0.77       298

    accuracy                           0.75       591
   macro avg       0.76      0.75      0.75       591
weighted avg       0.76      0.75      0.75       591
```

The accuracy for Naïve Bayes is 75%

5.2.4 Comparing the algorithms:

```
models=['random forest','logistic regression','naive bayes']
accuracy=[0.89,0.74,0.75]
plt.bar(models,accuracy,color=['lightblue','pink','grey'])
plt.ylabel('Accuracy')
plt.title('accuracy of random forest vs logistic regression vs naive bayes' )
```

```
Text(0.5, 1.0, 'accuracy of random forest vs logistic regression vs naive bayes')
```

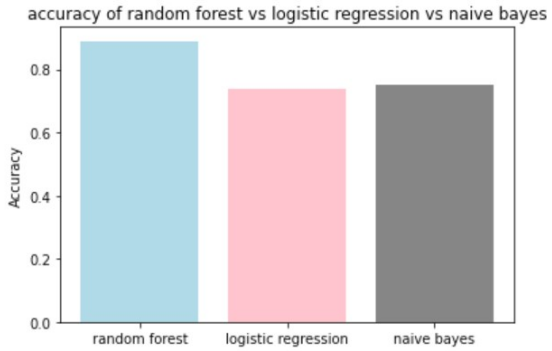


Figure 39

From the above graph it is seen that Random forest has the highest accuracy there by we will be considering random forest in the further prediction

Since the highest accuracy is around 89% I performed hyper parameter tuning using grid search and also implemented the xgboost .

CHAPTER 6

Hyper parameter Tuning:

A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data.

- They are often used in processes to help estimate model parameters.
- They are often specified by the practitioner.
- They can often be set using heuristics.
- They are often tuned for a given predictive modeling problem.

6.1 Grid searching of hyperparameters:

Grid search is an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

Here I have used Random forest algorithm therefore I will be passing the parameters of radom forest algorithm

```
from sklearn.model_selection import GridSearchCV ## importing the package
n_estimators = [100, 300, 500, 800, 1200]
max_depth = [5, 8, 15, 25, 30]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]

hyperF = dict(n_estimators = n_estimators, max_depth = max_depth,
              min_samples_split = min_samples_split,
             min_samples_leaf = min_samples_leaf)

gridF = GridSearchCV(rfc, hyperF, cv = 3, verbose = 1,
                     n_jobs = -1)
```

Figure 39

Predicting using the best parameters of the model after performing tuning using grid search

```
bestF = gridF.fit(X_train, y_train)
y_pred_f = bestF.predict(X_test)
print(classification_report(y_test, y_pred_f))
bestF.best_params_
```

```
Fitting 3 folds for each of 500 candidates, totalling 1500 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   46.7s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  3.1min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  8.0min
[Parallel(n_jobs=-1)]: Done 792 tasks      | elapsed: 15.9min
[Parallel(n_jobs=-1)]: Done 1242 tasks      | elapsed: 25.3min
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed: 30.3min finished
```

```
              precision    recall  f1-score   support

           0       0.94      0.90      0.92       293
           1       0.91      0.94      0.92       298

    accuracy                           0.92       591
   macro avg       0.92      0.92      0.92       591
weighted avg       0.92      0.92      0.92       591


{'max_depth': 25,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 1200}
```

```
sns.heatmap(confusion_matrix(y_test,y_pred_f), annot=True, fmt='d') ## visual representation of classification report
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1790c57c808>
```

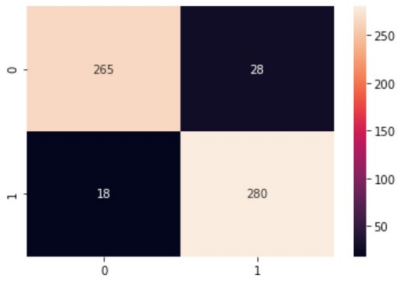


Figure 40

```
accuracy_score(y_test, y_pred_f) ## test score
```

```
0.922165820642978
```

Figure 41

As it is clearly visible that before hyper parameter tuning the accuracy was 89% and after hyper parameter tuning the accuracy increased to 92%

6.2 Boosting the algorithm:

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers.

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gödel Prize for their work. It can be used in conjunction with many other types of learning algorithms to improve performance.

AdaBoost is best used to boost the performance of decision trees on binary classification problems.

AdaBoost was originally called AdaBoost.M1 by the authors of the technique Freund and Schapire. More recently it may be referred to as discrete AdaBoost because it is used for classification rather than regression.

AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem.

The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. Because these trees are so short and only contain one decision for classification, they are often called decision stumps.

Each instance in the training dataset is weighted. The initial weight is set to:

$$weight(xi) = 1/n$$

Where xi is the i'th training instance and n is the number of training instances.

```
## boosting the algorithm
from sklearn.ensemble import AdaBoostClassifier
rf = RandomForestClassifier(max_depth= 25, min_samples_leaf= 1, min_samples_split= 2, n_estimators = 500)
rf_Boost = AdaBoostClassifier(base_estimator=rf,n_estimators=100,learning_rate=0.01,random_state=1)
bestrfb = rf_Boost.fit(X_train, y_train)
y_pred_rfb = bestrfb.predict(X_test)
print(classification_report(y_test, y_pred_rfb))
```

```
              precision    recall  f1-score   support

           0       0.94      0.91      0.93       293
           1       0.92      0.94      0.93       298

    accuracy                           0.93       591
   macro avg       0.93      0.93      0.93       591
weighted avg       0.93      0.93      0.93       591
```

```
accuracy_score(y_test, y_pred_rfb)
```

```
0.9272419627749577
```

Figure 42

Using roc curve to represent the result

```
from sklearn.metrics import roc_curve, roc_auc_score
fpr, tpr, thresholds = roc_curve(y_test, y_pred_rfb)
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.plot([0, 0], [1, 0], 'k--', lw=2, color = 'black')
plt.title('ROC curve for HR Attrition classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```

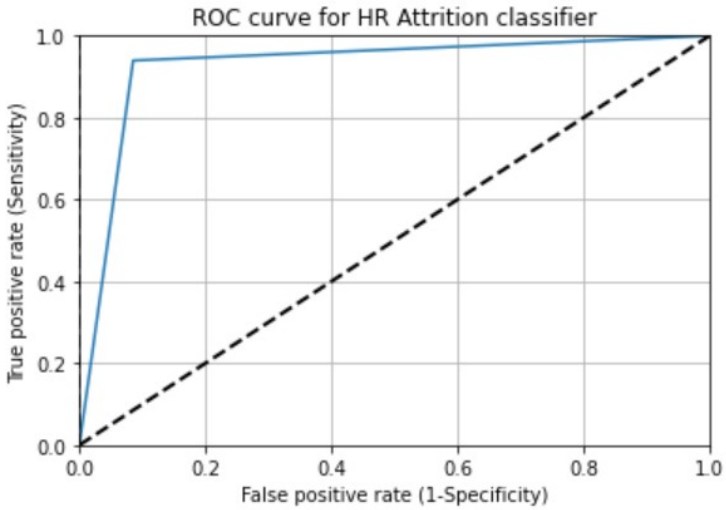


Figure 43

- accuracy before hyper parameter tuning of random forest algorithm=88%
- accuracy after hyper parameter tuning of random forest algorithm=92%

the **adaboost classifier** and the **hyper parameter tuning** gave the same accuracy score which is 92%

CONCLUSION:

Calculating employee turnover rate is not that simple as it seems to be. No common formula can be used by all the organizations. A formula had to be devised keeping in view the nature of the business and different job functions. After performing the operations on the data set it is concluded after performing thorough Exploratory Data analysis which include Stats models which are computed to get accuracy and also Heat maps which are computed to get a clear understanding of the data set (which parameter has most abundant effect on the study case) and is visible that amongst all the algorithms implemented Random Forest Algorithm has the highest accuracy and is best fit for prediction of employee attrition which gave us an accuracy of 92% post performing the hyper parameter tuning.

REFERENCES

https://www.geeksforgeeks.org

https://www.investopedia.com

https://www.furstperson.com/blog/causes-employee-attrition-cost-employee-attrition

https://scikit-learn.org/