Demystify Employee Leaving Predictive Model using Machine Learning

Done by
Sangeeta Kamite
Nimish Ukey
Anitya Umare
Arshad Bagde

Mentored by

Mr. Gautam P

In partial fulfillment of the requirement for
POSTGRADUATE DIPLOMA IN DATA SCIENCE AND ANALYTICS

NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY [NIELIT]
CHENNAI
(An Autonomous Scientific Society of Ministry of Electronics & Information Technology)
Government of India
No. 25, Gandhi Mandapam Road, Chennai – 600025,
Tamilnadu, India.

DECLARATION

We hereby declare that the project work entitled

Demystify Employee Leaving Predictive Model using Machine Learning

Done for

NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY [NIELIT] CHENNAI
(An Autonomous Scientific Society of Ministry of Electronics & Information Technology)
Government of India
No. 25, Gandhi Mandapam Road, Chennai – 600025, Tamilnadu, India.

Under the Guidance of

Mr. Gautam P

NAME OF THE STUDENTS

Sangeeta Kamite
Nimish Ukey
Anitya Umare
Arshad Bagde

Place: Chennai
Date: 11/04/2021

ACKNOWLEDGEMENT

At first, we convey our acknowledgement to Dr. Sanjeev Kumar Jha, Joint Director (Tech.), NIELIT Chennai, for helping us to continue the course and do the project during the COVID19 Pandemic situation.We are very much happy to have him and given us many supports and valuable guidance during our study at NIELIT Chennai.

Then, we would also like to express our deep gratefulness to faculty, for providing us with the necessary technical support and motivation. They allowed us to work freely with the topic by giving proper guidance and resources, which was an inevitable space that we receive to develop our knowledge and skills from the beginning of the work till the completion of the Project.

We are highly pleased for getting an opportunity to study in NIELIT Chennai for completing the course.
Finally, we want to thank our parents, colleagues, and friends for all the love and support

Table Of Contents

Demystify Employee Leaving Predictive Model using Machine Learning

1 INTODUCTION

Now a day's data science predictions are used in IT industries, for the improvement in market investment, employee management etc. Retention of valuable employees within an organization has become an important issue as it is hard to find out the reasons that why employees are leaving an organization and keep them satisfied is a big challenge, for this a report is made to predict the retention of an employee in an organization using the python programming with data science methods.

The main idea of this report is to find out that which valuable employee will leave the company and the features which are affecting him/her to making this decision like salary level, no. of hours spending in a week, promotion, no. of work accident etc.

The application was developed in python programming language and prediction is made with the help of data science and machine learning models. The design criteria and the implementation details are presented in this report. Data mining is the next big in the world of Information Technology, usage of data extraction is increasing day by day. Data science is the process of mining of useful insights from larger amount of data to use it for the development purpose. To extract data several algorithms, methods and analyzing processes are used depending upon the kind of data we have and what the analyst intended to do with the data. The data we get is in the form of raw data, it needs to get preprocessed to make it in the form to apply algorithm on it.

Preprocessing techniques includes collection, noise removal, data reduction, transformation etc. data science methodologies are mainly classified in two categories as making prediction and pattern discovery, prediction making is the process of producing estimate result by analyzing previous results known as regression or supervised learning and pattern discovery is that method when we apply different approaches to find out similarities and dissimilarities in the given data by assigning class notations which is known as clustering or unsupervised learning.

Data Science is a blend of various tools, algorithms, and machine learning principles with the goal to discover hidden patterns from the raw data. A Data Analyst as a rule clarifies what is happening by handling history of the information. Then again, Data Scientist not exclusively does the exploratory investigation to find bits of knowledge from it, yet in addition utilizes different propelled machine learning calculations to recognize the event of a specific occasion later on.

A Data Scientist will take a gander at the information from numerous edges, at times edges not known before. Along these lines, Data Science is essentially used to settle on choices and forecasts making utilization of prescient causal examination, prescriptive investigation (prescient in addition to choice science) and machine learning. We know that larger companies contain more than thousand employees working for them, so taking care of the needs and satisfaction of each employee is a challenging task to do, it results in valuable and talented employees leave the company without giving the proper reason.

This paper provides solution for the given problem as it gives a prediction model that can be used to predict which employee will leave the company and which will not leave. It also helps in finding the exact reasons which are motivating the employees for shifting companies like lower salary, less promotions or heavy work load etc. To find the result in the form of yes or no, we have used logistic regression method, which predicts result in binary values that are 0 or 1, 0 means employee will not leave the company and 1 means he/she will.

1.1 Losing an employee is a problem for various following reasons:

- It is difficult to find suitable replacements for employees, particularly those with high experience and special skills.
- It takes time, effort and money to recruit new employees.
- Loss of an employee adversely affects ongoing projects and services, which leads to dissatisfaction among customers and other stake-holders.
- It takes time and efforts for new employees to achieve the same levels of expertise and productivity.
- Loss of an employee costs money. Employee churn rates can be as high as 12–15% annually.
- Loss of company knowledge
- Loss of Goodwill of the company.

1.2 Employee Retention:

- Employee retention is a process in which the employees are encouraged to remain with the organization for the maximum period of time or until the completion of the project.
- Employee retention is beneficial for the organization as well as the employee.

1.3 Importance of Employee Retention:

- Reduced cost of turnover

- Reduced loss of company knowledge
- Reduced interruption of Customer Service
- Intervention as turnover leads to more turnover
- Goodwill of the company
- Regaining efficiency

We will be using a step-by-step systematic approach using a method that could be used for a variety of Machine Learning problems. This project would fall under what is commonly known as HR Analytics or People Analytics.

2 PROBLEM DEFINITION

In this study, we will attempt to solve the following problem statement is:

- What is the likelihood of an active employee leaving the company?
- What are the key indicators of an employee leaving the company?
- What strategies can be adopted based on the results to improve employee retention?
- 'Even better, the results will help for proactive planning of new hiring in advance.

3 PREVIOUS WORK

Retention of valuable employee within an organization is a major issue in the companies, so several efforts aremade to find out the proper employee management policies in the companies, we are discussing some work from them – Piotr Płoński (MLJAR) et.al [1] proposed the analytic methods those can improve Human Resources (HR) management for companies with large number of employees by providing approaches to predict employee attrition with machine learning. They used 1200 employee's data for training datasets, which contains description, but the retention is unknown, which is predicted using binary classification.

Le Zhang and Graham Williams et.al [2] proposed that employee retention is the biggest challenge for a company, so it is important for company to recognize behavioral patterns to understand their employees better. They used R for predictions by feature extraction methods as word-to-vector, term frequency, or term frequency and inverse document frequency, R packages such as tm etc. They finally concluded that ensemble techniques can be deployed to effectively boost model performance.

Ashish Mishra et.al [3] proposed that it is first important to recruit right person to do talent management, the easily available data source for present and past candidates is their resume. This paper provides a method to calculate the employee score using his educational and business experience scores. They concluded that information like number of years of education, number of organizations worked for, number of positions held in the past, and age can be easily translated into a score for every employee which can be used for predicting retention.

Rupesh Khare, Dimple Kaloya and Gauri Gupta et.al [4] proposed that a risk equation can be develop, which can be used assess attrition risk with current set of employees that a company is having. They concluded by stating that among the various attrition predictive techniques available in the market, Logistic Regression and Discriminant Analysis are the closest to give a solution which produced highly accurate results.

Randy Lao et.al [5] states that a company which make healthy environment and provide equal opportunities for employees to glow, grows rapidly. Their goal is to create a model that help in improving retention strategies on targeted employees. He used R programming language and, they concluded by saying that employees having higher satisfaction and evaluation rate will have fewer chance to leave the company.

4 TOOLS AND PACKAGES USED

Why We Use Python For Employee Attrition?

Here are some of the important factors, for that we preferred use Python over other data science tools:

- Open source
- Easy to learn
- Scalability
- Choice of data science libraries
- Python community
- Graphics and visualization

4.1 Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects.

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

4.2 Some Important Packages Used In Python Are As Followed

4.2.1 NumPy

NumPy is the fundamental package for scientific computing with Python. It contains a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and FORTRAN code, useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data- types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

4.2.2 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

The name is derived from the term "panel data", an econometric term for data sets that include

observations over multiple time periods for the same individuals. some of the features of pandas are dataframe object for data manipulation with integrated indexing, tools for reading and writing data between in-memory data structures and different file formats, data alignment and integrated handling of missing data, reshaping and pivoting of data sets, label-based slicing, fancy indexing, and subsetting of large data sets, data structure column insertion and deletion, group by engine allowing split- apply-combine operations on data sets, data set merging and joining, hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.

Time series-functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging and provides data filtration.

4.2.3 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural “pylab” interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

4.2.4 Scikit-learn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

5 MACHINE LEARNING

Machine learning is the process of making the machine To learn itself through patterns and training data sets. Training data sets are data which is given to machine for understanding the hidden patterns within data and make relations for own understanding. It helps in working of machines efficiently by making them processed like a human brain. Pattern recognition is the most challenging task for developers to use such algorithms that allows different machines to work according to the requirement.

This Project emphasizes on making prediction of retention of an employee within an organization such that whether the employee will leave the company or continue with it. It uses the data of previous employees which have worked for the company and by finding pattern it predicts the retention in the form of yes or no. It uses various parameters of employees such as salary, number of years spent in the company, promotions, number of hours, work accident, financial background etc.

Considering new processing innovations, machine adapting today isn't care for machine learning of the past. It was conceived from design acknowledgment and the hypothesis that PCs can learn without being customized to perform assignments; specialists intrigued by manmade brainpower et.al [6] needed to check whether PCs could gain from information. The iterative part of machine learning is essential claiming as models are presented to new information, they can freely adjust. They gain from past calculations to deliver solid, repeatable choices and results. It's a science that is not new – but rather one that is increasing crisp energy. While numerous machine learning calculations have been around for quite a while, the capacity to naturally apply complex scientific computations to huge information again and again, quicker and speedier is a current advancement.

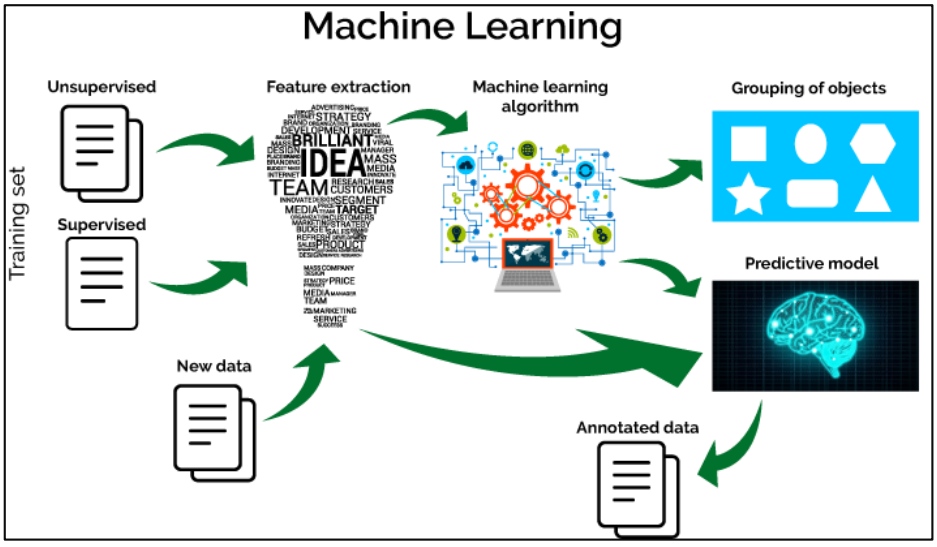


Fig 5.1. Machine Learning Approach

5.1 Machine Learning Algorithms

Machine learning algorithms are differentiated as supervised or unsupervised.

A. Supervised machine learning calculations can apply what has been realized in the past to new information utilizing marked cases to anticipate future occasions. Beginning from the examination of a known preparing dataset, the learning calculation creates a surmised capacity to make expectations about the yield esteems. The framework can give focuses to any new contribution after adequate preparing. The learning calculation can likewise contrast its yield and the right, planned yield and discover mistakes to adjust the model appropriately.

B. In differentiate, unsupervised machine learning calculations are utilized when the data used to prepare is neither grouped nor named. Unsupervised learning contemplates how frameworks can induce a capacity to portray a concealed structure from unlabeled information. The framework doesn't make sense of the correct yield; however, it investigates the information and can attract derivations from datasets to depict concealed structures from unlabeled information.

C. Semi-directed et.al [7] machine learning calculations fall some place in the middle of regulated and unsupervised learning, since they utilize both marked and unlabeled information for preparing – ordinarily a little measure of named information and a lot of unlabeled information. The frameworks that utilization this strategy can significantly enhance learning precision. For the most part, semi-administered learning is picked when the procured named information requires gifted and significant assets to prepare it/gain from it. Something else, obtaining unlabeled information by and large doesn't require extra assets.

D. Reinforcement machine learning calculations is a learning technique that interfaces with its condition by creating activities and finds mistakes or rewards. Experimentation seek and postponed compensate are the most pertinent attributes of fortification learning. This technique enables machines and programming operators to naturally decide the perfect conduct inside a setting to augment its execution. Basic reward input is required for the specialist to realize which activity is ideal; this is known as the support flag.

5.2 ML Algorithms Used For Prediction

- Logistic Regression
- Random Forest
- Artificial Neural Network

5.2.1 Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for classification problems.

The logistic function is defined as: $\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$

And it looks like this:
It outputs numbers between 0 and 1. At input 0, it outputs 0.5

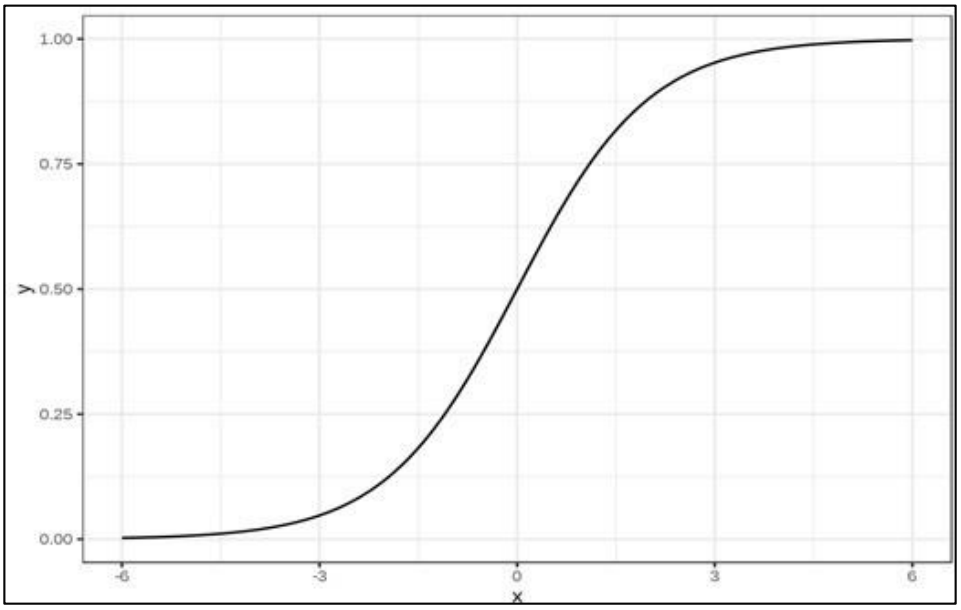


Figure 5.2 Logistic Regression Functions

5.2.2 Random Forest Algorithm

Random forest is a supervised learning algorithm that can also be used for regression and classification. Random Forest is popular because it's the algorithms that are the most reliable, flexible, and easy to use. Random forest is also an ensemble algorithm. Ensemble algorithms have the ability to combine with a number of other algorithms that either have the same or different type of classifying object. For example, Random Forest uses a number of different decision tree classifiers that fit subsets of the input data and averages those trees in order to improve accuracy.

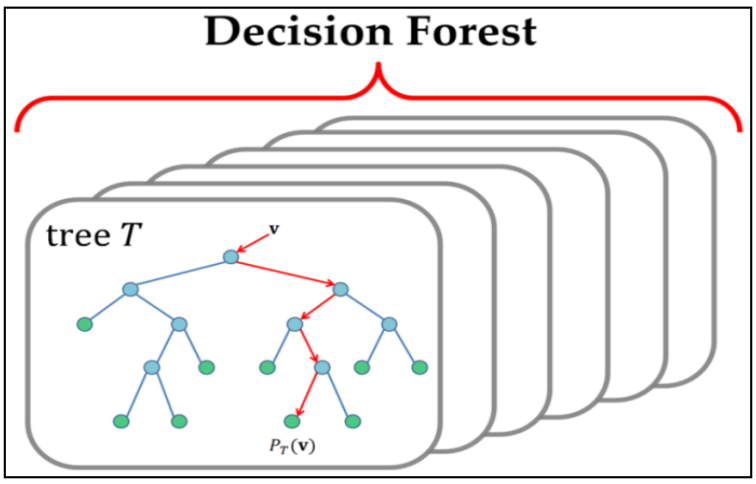


Fig 5.3 Decision Forest

Random Forest algorithm attempts fits a number of decision tree classifiers on various sub samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [sklearn]. The general process of the algorithm is easy to understand in 4 steps:

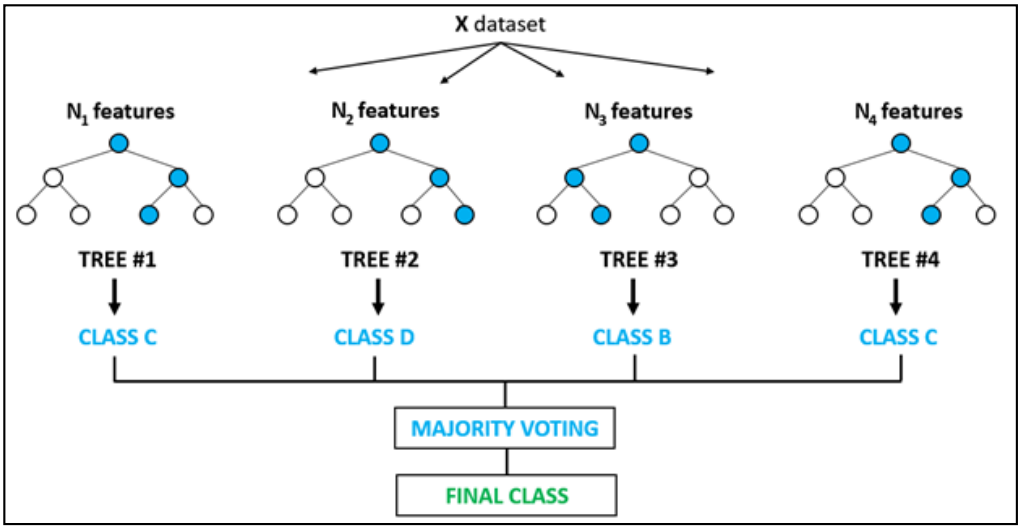


Fig 5.4 Random Forest Classification

1. Split the input dataset into random sub samples
2. Each subsample gets its own decision tree classifier that will give a prediction to that specific sub sample only.
3. Vote on each result predicted from decision trees
4. Finally select the result with the highest vote and present it as output

It is one of the most accurate learning algorithms available. It runs efficiently on large databases. It can handle thousands of input variables without deletion. It gives estimates of what variables are important in the classification

5.2.3 Artificial Neural Network

ANN architecture is based on the structure and function of the biological neural network. Similar to neurons in the brain, ANN also consists of neurons which are arranged in various layers. Feed forward neural network is a popular neural network which consists of an input layer to receive the external data to perform pattern recognition, an output layer which gives the problem solution, and a hidden layer is an intermediate layer which separates the other layers. The adjacent neurons from the input layer to output layer are connected through acyclic arcs. The ANN uses a training algorithm to learn the datasets which modifies the neuron weights depending on the error rate between target and actual output. In general, ANN uses the back propagation algorithm as a training algorithm to learn the datasets. The general structure of ANN is shown in Fig. 5.5.

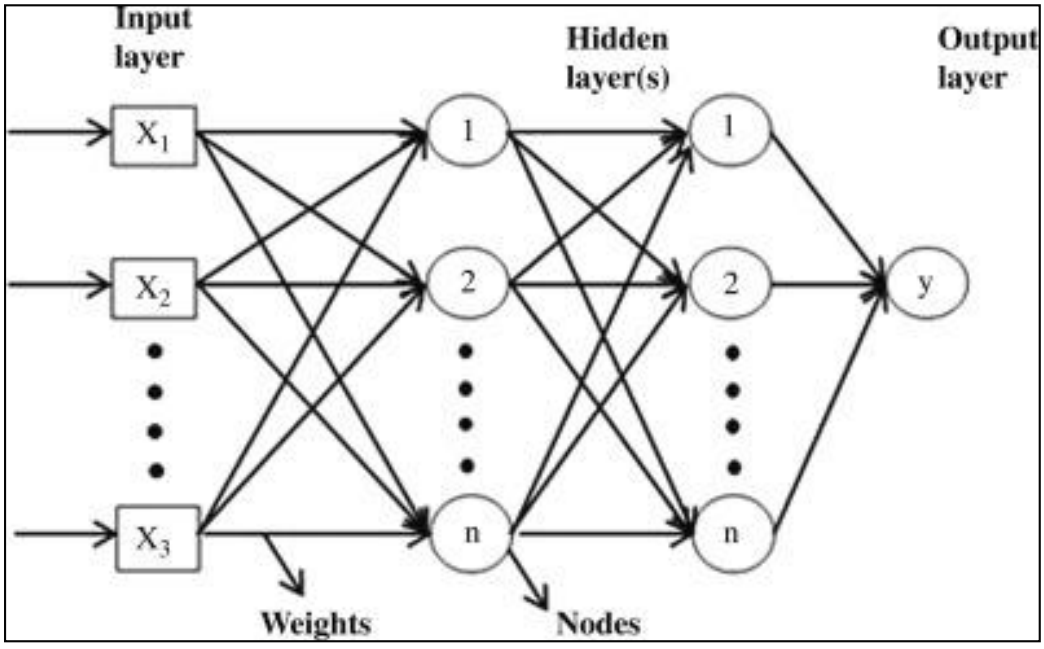


Figure 5.5. General structure of ANN.

6 TECHNOLOGY USED

We have utilized Python programming dialect, which is a translated, progressively written dialect and least difficult in grammar. Python is utilized for every one of the applications like in IOT advancement, information science field, web improvement, scripting reason and so forth. Consequently, now it is being utilized generally over the globe.

Python contains various number of libraries accessible in it, this makes it simple to use for each application like for web rejecting delightful cleanser, for GUI improvement TKinter, for web network urlib2, for machine learning sklearn et.al [8], numpy, pandas and so on. Python is one of the for the most part utilized dialect for Data Science applications since it gives libraries, for example, Pandas, nltk which can oversee substantial number of datasets into fitting way, it gives representation libraries like Matplotlib, Bokeh, Seaborn and so on that are exceedingly expressive regarding charts and plots portrayals.

The sklearn library is one which gives bigger number of machine learning calculations, for example, direct and various relapse, polynomial relapse, choice tree characterization and so on., to make expectations, bunching and grouping of information in number of billions. Machine learning is a branch in software engineering that reviews the outline of calculations that can learn. Run of the mill errands are idea learning, work learning or "prescient demonstrating", bunching and finding prescient examples. These undertakings are found out through accessible information that were seen through encounters or directions, for instance. The expectation that accompanies this teach is that including the experience into its assignments will in the end enhance the learning. However, this change needs to occur such that the learning itself ends up programmed with the goal that people like ourselves don't have to meddle any longer is a definitive objective.

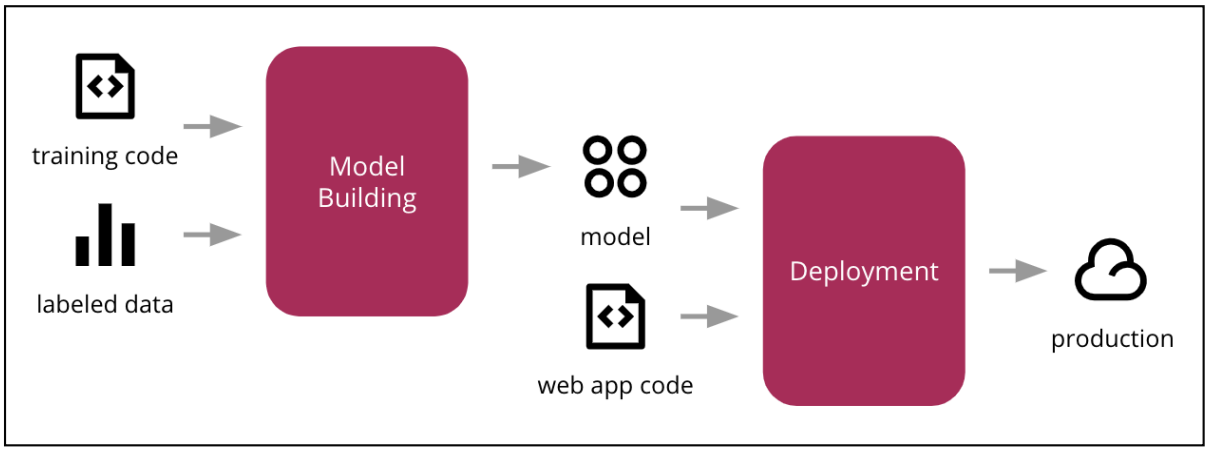


Fig. 6.1. Prediction Methodolog

Scikit-learn is the most helpful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a great deal of efficient devices for machine learning and factual displaying including arrangement, relapse, bunching and dimensionality lessening. Scikit-learn gives a scope of directed and unsupervised learning calculations through a reliable interface in Python.

7 METHODOLOGY USED FOR PREDICTION

- **Data Analysis.**
- **Data Engineering**
- **Data Processing**
- **Model Creation & Evaluation**

7.1 Data Analysis

In this case study, a IBM dataset was sourced from
https://www.kaggle.com/pavansubhasht/ibmhranalytics-attrition-dataset/activity
The dataset used here is IBM HR Analytics dataset from Kaggle. There are 1470 records with 35 features. Check the video below to have a real touch on the raw dataset. This is a fictional data set created by data scientists. Its main purpose was to demonstrate the IBM Watson Analytics tool for employee attrition.

7.2 Data Engineering

Feature engineering involves creating new features and relationships from current features. To start off, let's segregate the categorical variables from numerical ones. We can use the **datatype *method*** to find categorical variables, as their **dtype** would be *'object'*. You may notice data types are already shown when using **employee_df.info()**.Then, we can encode categorical variables. Two methods are available. One is to use **OneHotEncoder *from sklearn***, and the other is **get_dummies()** from ***pandas***. I prefer the latter, as it returns a dataframe which makes the following step easy. Specifically,

```
employee_df_cat = pd.get_dummies(employee_df_cat)
```

Then, concatenate the encoded categorical and numerical variables together. Specifically,

```
X_all = pd.concat([X_cat, X_numerical], axis = 1)
```

One final step is to generate the target variable

```
employee_df['Attrition'] = employee_df['Attrition'].apply(lambda x: 1
if x == 'Yes' else 0)
y = employee_df['Attrition']
```

7.3 Data Processing

Now we are ready to process the data, including data split, scaling, and balancing. To make the data ready for training, we need to scale the features to avoid any variables dominating over other variables, namely taking higher weights and a strong influence on model learning. Specifically,

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X = scaler.fit_transform(X_all)
```

Now let's partition the dataset into a training set and a test set. To split data,

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25)
```

We have noted that severe imbalance between employees leaves and stays. So let's implement the **SMOTE** method to oversample the minority classes. Specifically,

```
oversampler = SMOTE(random_state=0)
X_smote_train, y_smote_train =
oversampler.fit_sample(X_train,y_train)
```

Now the data is ready for the model.

7.4 Model Creation And Evaluation

As alluded at the beginning of the post, we aim to evaluate and compare the performance of a handful of models.

7.4.1 Logistic Regression

Simply put, logistic regression uses a logarithmic transformation on a linear combination of independent variables which allows us to model a nonlinear problem in a linear manner. It is commonly used for a binary classification problem where some correlation between predictors and response variables is assumed.
To create a logistic regression classifier, we use sklearn as below.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_smote_train, y_smote_train)
```

As indicated in Fig. 7.1, the logistic regression classifier gives an accuracy of 0.75 and an F1 score of 0.52.

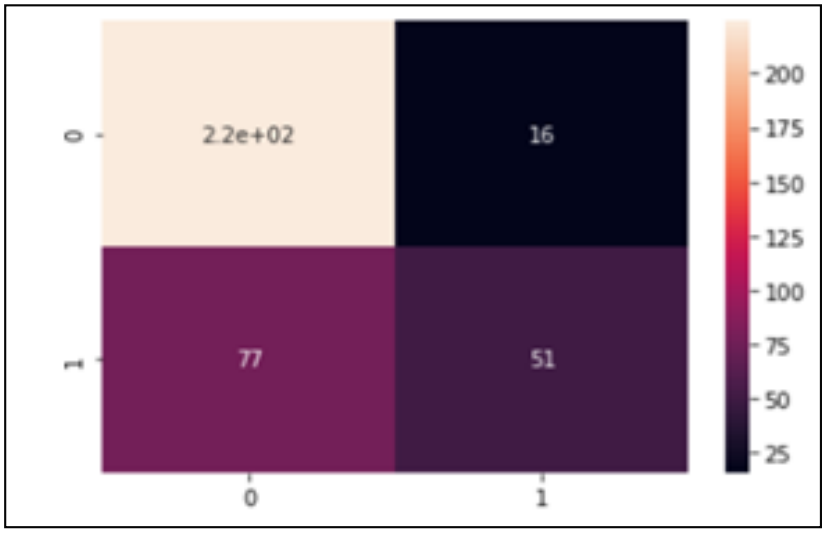


Fig.7.1 Confusion matrix for logistic regression model

7.4.2 Random Forest

Random Forest is a type of ensemble model with a decision tree as its build block. It creates a group of decision trees and uses their collective predictive power to obtain relatively strong performance. For a really good read that drives home the basics of the Random Forest, To create a Random Forest classifier,

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_smote_train, y_smote_train)
```

Using the same method to evaluate the performance, we obtained an accuracy of 0.85 and an F1 score of 0.39.

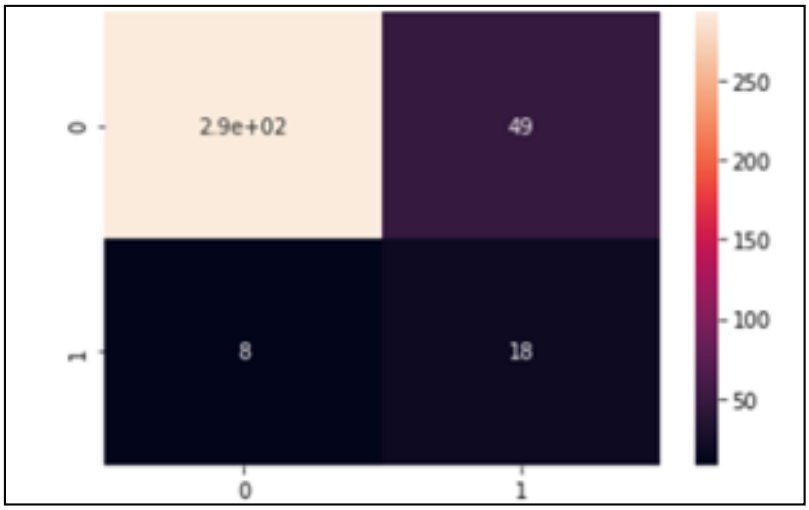


Fig.7.2 Confusion matrix for random forest model

7.4.3 Artificial Neural Network

The final attempt is to create and train an artificial neural network. Here we will build a sequential model with a few dense layers and dropout technique to reduce overfitting. Specifically,

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
model = Sequential()
model.add(Dense(units = 50, activation = 'relu', input_shape = (50,
)))
model.add(Dense(units = 500, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 500, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 50, activation = 'relu'))
model.add(Dense(units = 1, activation = 'sigmoid'))
```

To compile the neural network, we use *'adam'* optimizer and binary cross-entropy as the loss function.

```
model.compile(optimizer='adam', loss = 'binary_crossentropy', metrics
= ['accuracy'])
epochs_hist = model.fit(X_smote_train, y_smote_train, epochs = 50,
batch_size = 50)
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
```

Note above, we set a threshold probability for sigmoid function output at 0.5. So, any output greater than 0.5 is taken as 'leave', and otherwise as 'stay'. Fig. 7.3 shows the model loss during training. It seems the model reached a convergence with 20 epochs.

Finally, the confusion matrix heat map as shown in Fig. 7.3 gives an accuracy of 0.88 and an F1 score of 0.41.

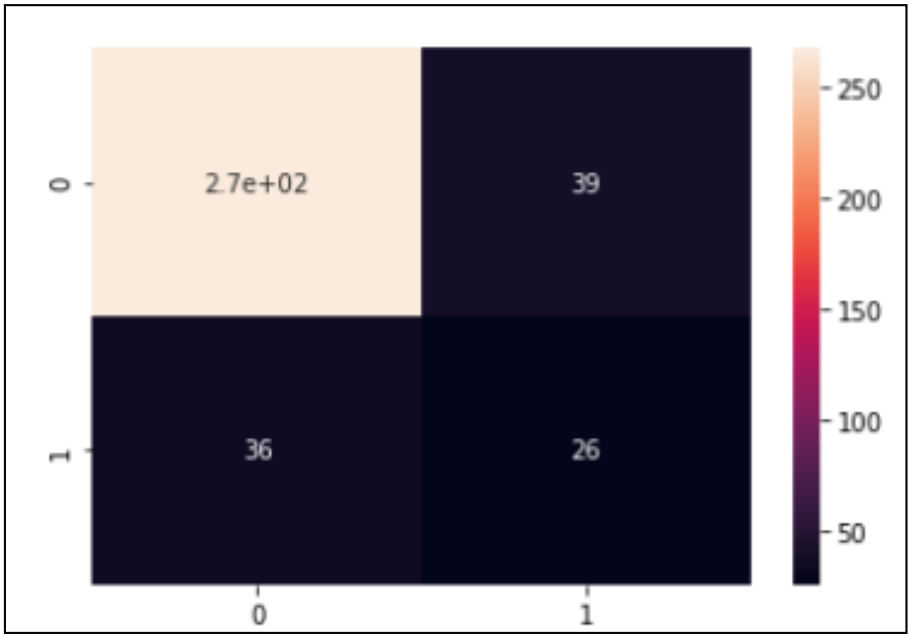


Fig.7.3 Confusion matrix for artificial neural network

8 PREDICTIVE ANALYSIS

Predictive Analytics is the field of study that employs statistical analysis, data mining techniques and machine learning to predict the future events with accuracy based on past and current situation.

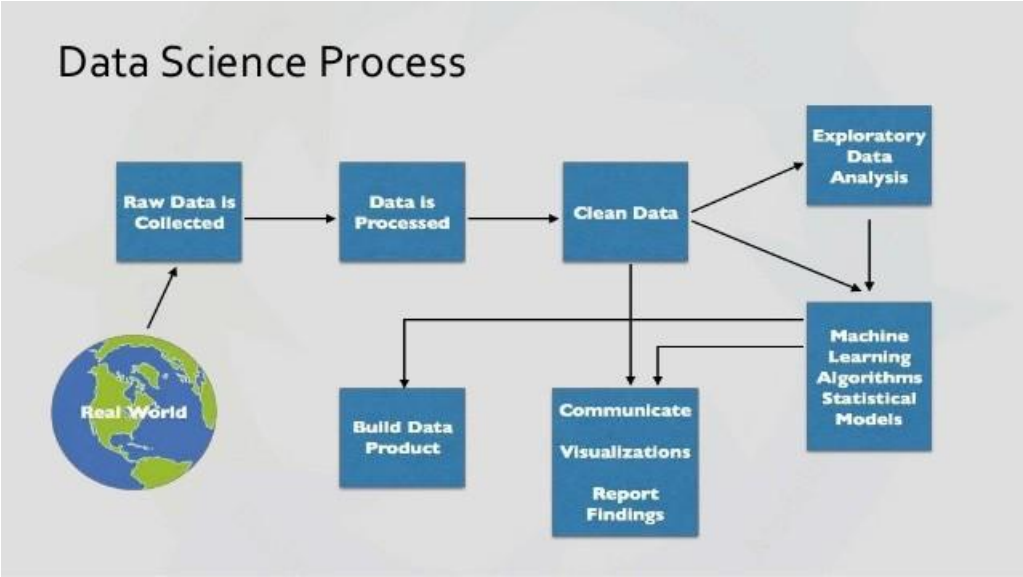


Fig 8.1 Data Science Process

8.1 Steps in Predictive Analysis

- Distribution Analysis
- Independent Variable correlation Analysis
- Response Variable correlation analysis
- Model Fitting

8.1.1 Distribution Analysis

We use the histogram as a simple visualization for numerical variable distribution analysis. First, let's convert binary categorical data into numerical. Use a one-line function with lambda to apply to the whole column. Then, pandas built-in plot function to create a histogram. Specifically,

```
employee_df['Attrition'] = employee_df['Attrition'].apply(lambda x: 1
if x == 'Yes' else 0)
employee_df['OverTime'] = employee_df['OverTime'].apply(lambda x: 1
if x == 'Yes' else 0)
employee_df['Over18'] = employee_df['Over18'].apply(lambda x: 1 if x
== 'Y' else 0)
employee_df.hist(bins = 30, figsize= (15, 15), color = 'r')
```

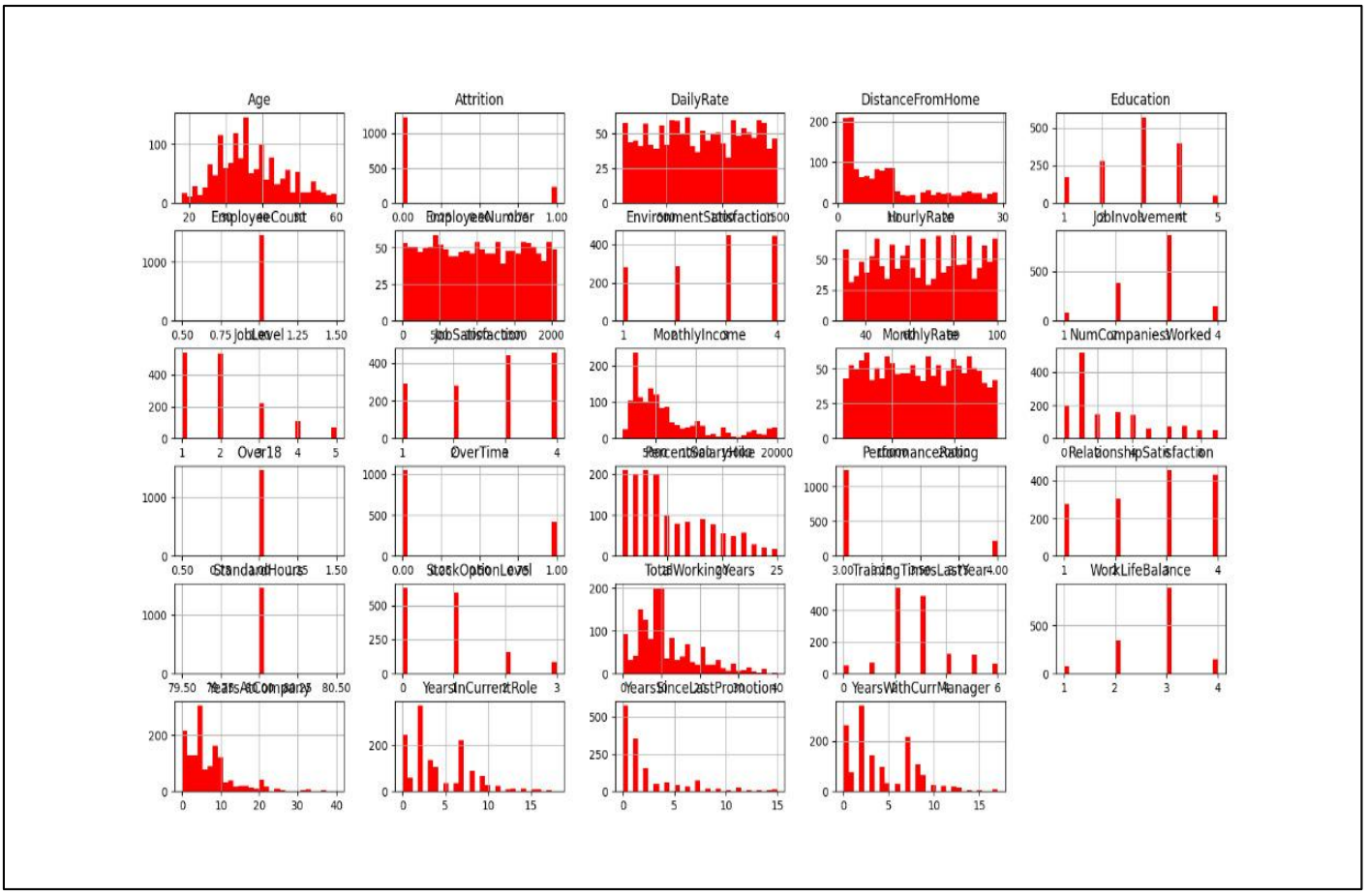


Fig 8.2. Distribution analysis for numerical variables

Reviewing Fig., you may find column **'EmployeeCount'**, ***'Over18'***, ***and*** **'StandardHours'** is of no value because these remain the same for all employees**.** So remove them**:**

```
employee_df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18',
'StandardHours'], axis = 1, inplace = True)
```

8.1.2 Independent Variable Correlation Analysis

in general, correlated features don't improve model performance. It is wise to remove correlated features considering the curse of dimensionality and the interpretability of the model. To perform correlation analysis, use:

```
correlations = employee_df.corr()
plt.subplots(figsize = (20, 20))
sns.heatmap(correlations, annot = True)
```

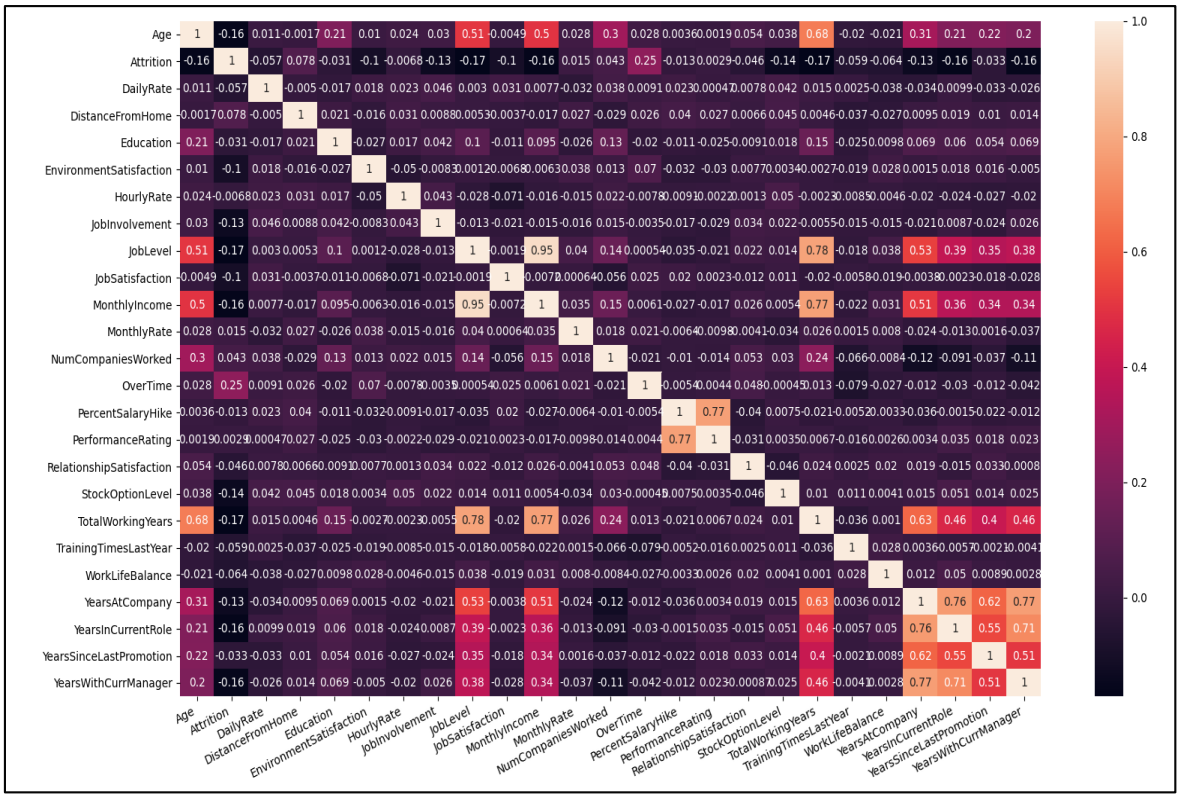


Fig 8.3. Correlation analysis for predictors

Fig.8.3 shows some sensible correlations. For instance, 'TotalWorkingYears' is strongly correlated with 'JobLevel' and 'MonthlyIncome'. In general, we can see quite a lot of variables are poorly correlated. Great, because it is desirable to train a predictive model with features that are not highly correlated with each other.

8.1.3 Response Variable Correlation Analysis

To analyze how each feature impacts employee leaves, we can inspect each variable's distribution with respect to the response variable. First, for employee 'Age', let's use count plot**,**

```
plt.figure(figsize = (25, 10), dpi = 300)
sns.countplot(x = 'Age', hue = 'Attrition', data = employee_df )
```

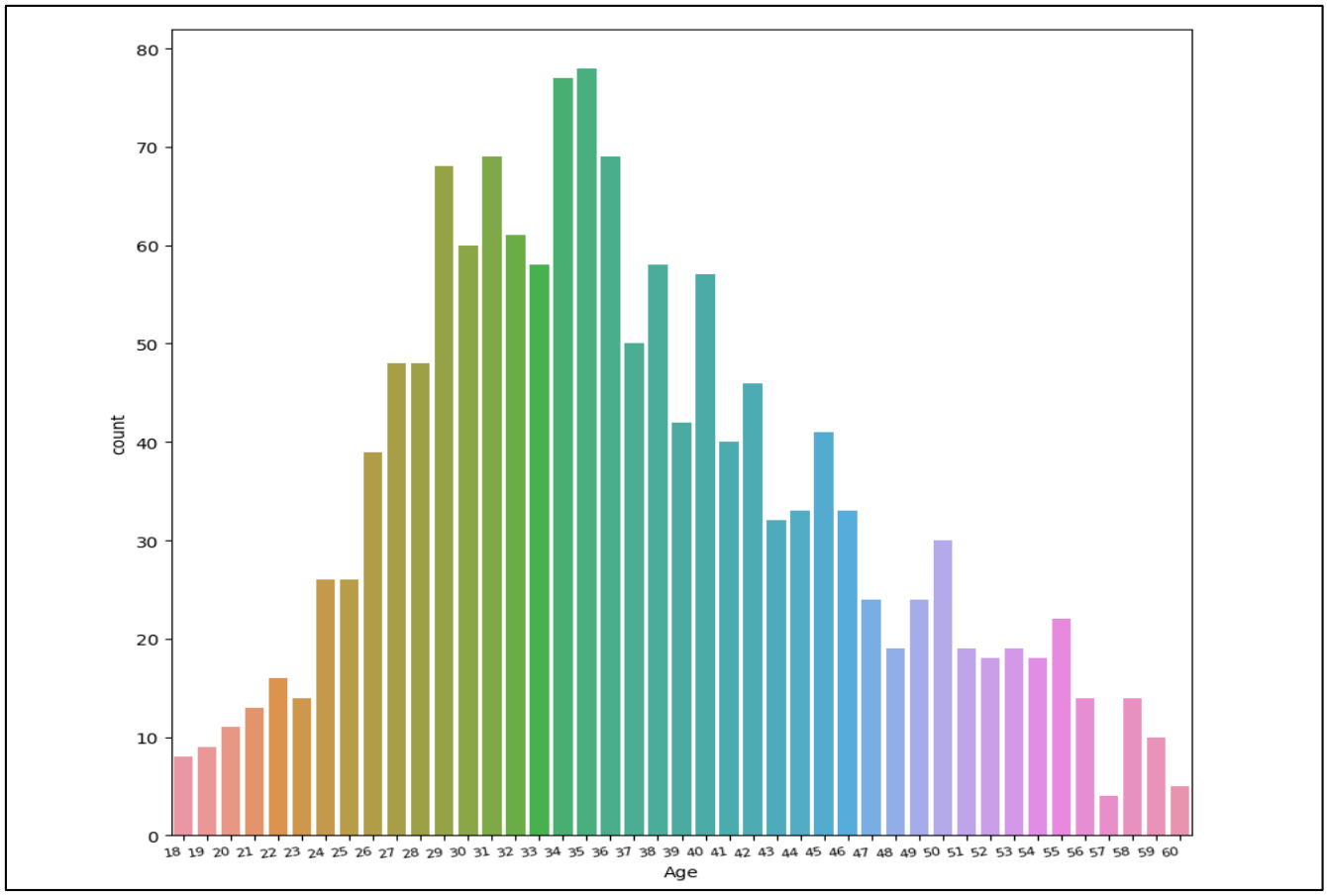


Fig 8.4. Distribution analysis of employee Age

Fig.8.4 shows employees between 25–35, show a high leave ratio compared to the rest ages. **Below 21,** a very high leave percentage can be spotted, indicating young professionals are more likely to leave the firm

Second, for *'JobRole'*, count plot is:

```
sns.countplot(x = 'JobRole', hue = 'Attrition', data = employee_df)
```

Fig.8.5 shows ***'sales executive', 'sales representative'*, and *'lab technician'* are more likely to leave compared to other roles**

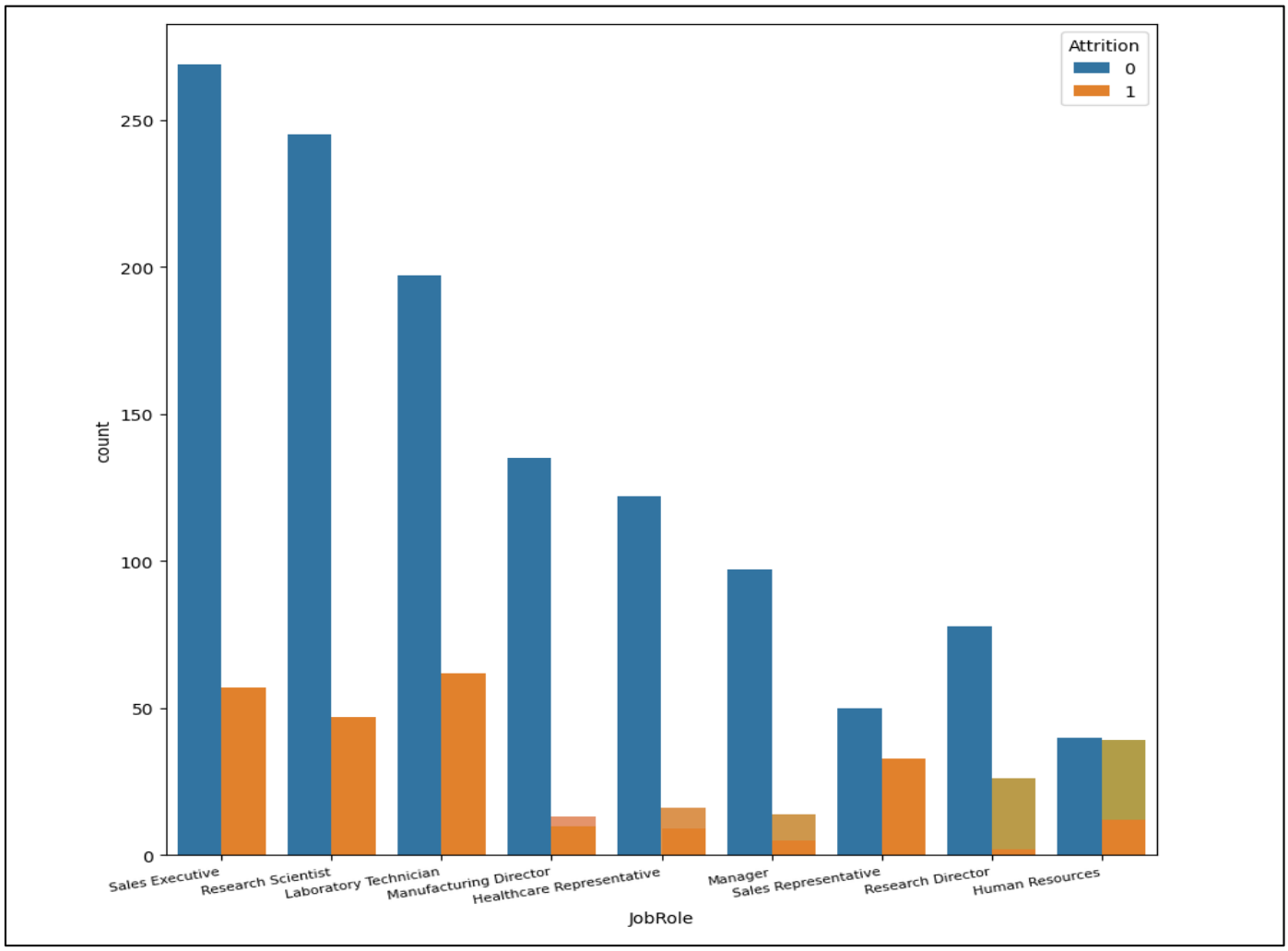


Fig 8.5 Employee Leave Countplot vs. Jobrole

9 SOURCE CODE & SNAPS

- **Import Python Libraries & Packages**

```
#import NumPy
import numpy as np
#import Pandas
import pandas as pd
#import Seaborn
import seaborn as sns
#import matplotlib
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
#import KMeans module
from sklearn.cluster import KMeans
#import train_test_split function
from sklearn.model_selection import train_test_split
#import resample function
from sklearn.utils import resample
#import scikit-learn metrics for accuracy calculation
from sklearn import metrics
#import sklearn modules for preprocessing & LabelEncoder
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
#import Common sklearn Model Helpers
from sklearn import feature_selection
from sklearn import model_selection
#import Classification report
from sklearn.metrics import classification_report
# import Libraries for data modelling
from sklearn import svm, tree, linear_model, neighbors,gaussian_process
from sklearn import naive_ bayes, ensemble,discriminant_analysis,
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

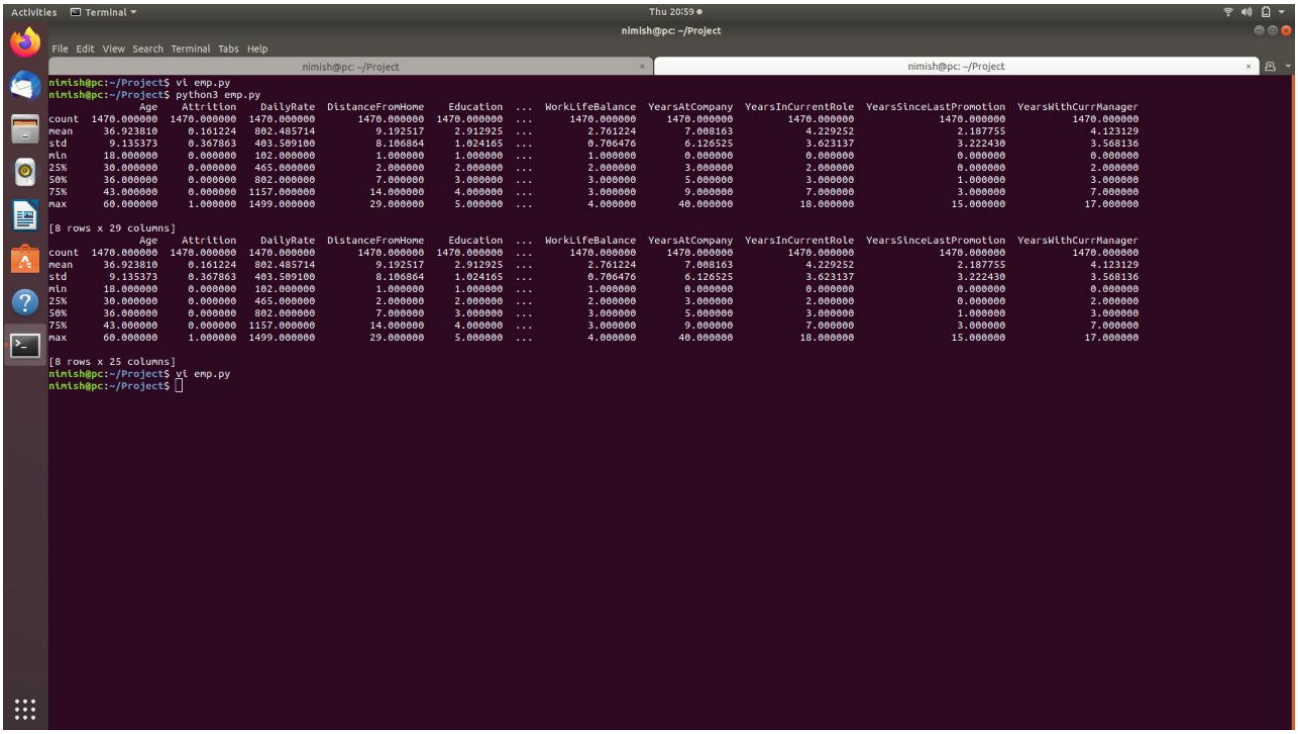
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
# import sklearn modules for ML model selection
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import Randomized SearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
#import sklearn modules for performance metrics
from sklearn.metrics import confusion_matrix, classification_ report,
        precision_recall_curve
from sklearn.metrics import auc, roc_ auc_score, roc_curve, recall_score, log_loss
from sklearn.metrics import f1_score, accura-cy_score, roc_auc_ score, make_scorer
from sklearn.metrics import average_precision_score
#import warnings
from warnings import filterwarnings
sns.set_context("notebook")
plt.style. use("fivethirtyeight")
filterwarnings("ignore")
```

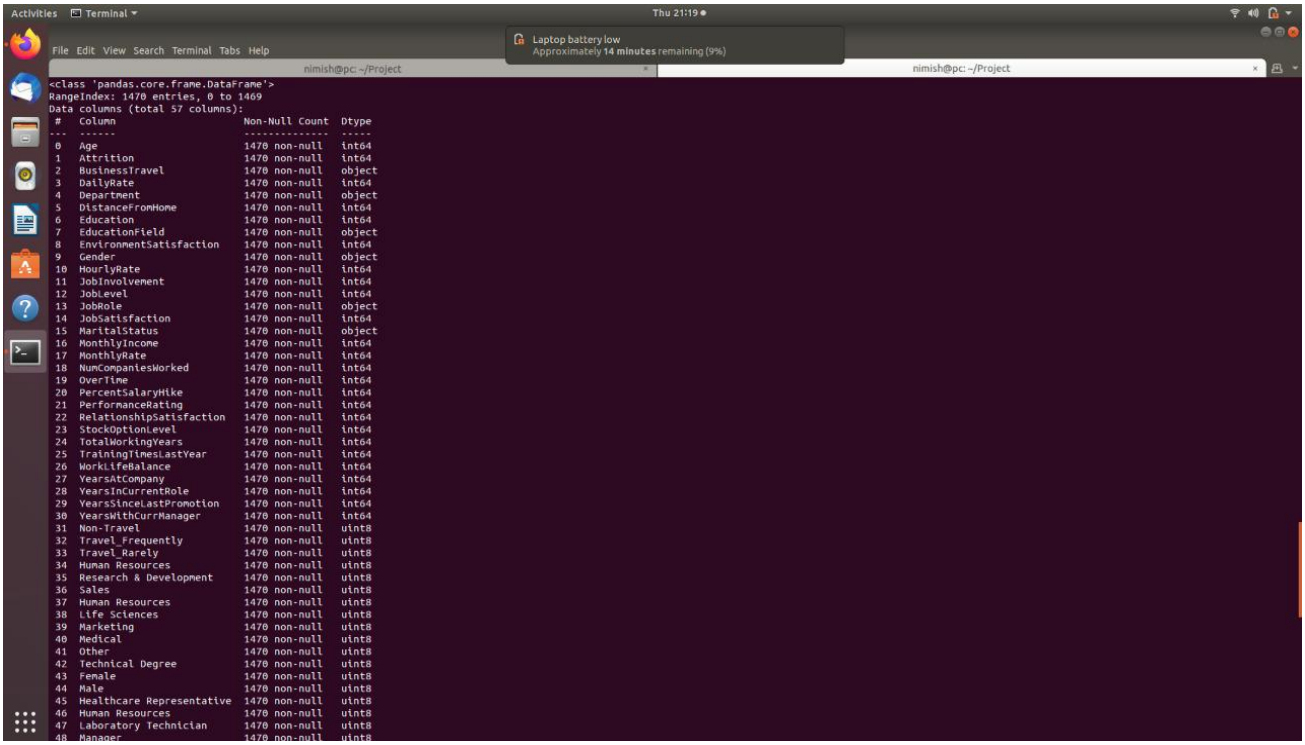
➢ Loading or Import the data in Python

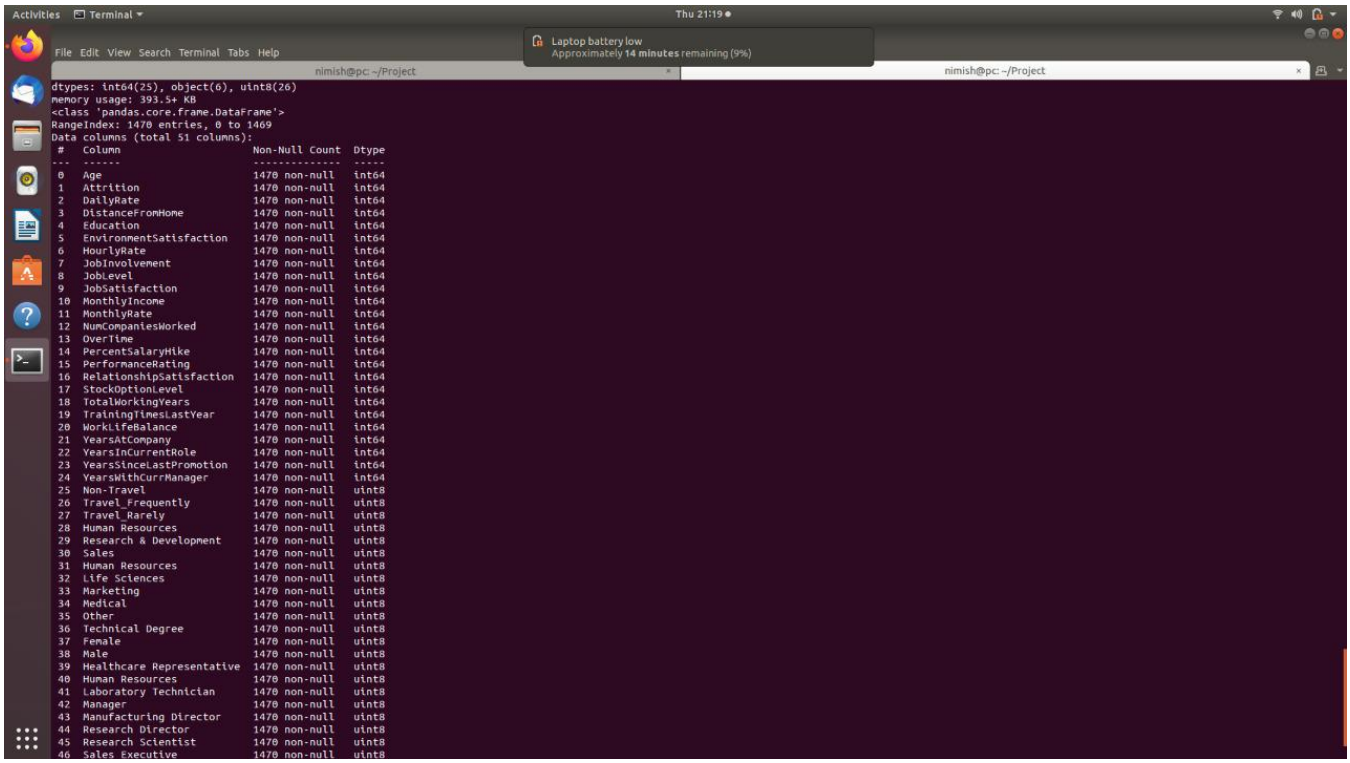
Loading data in python environment is the most initial step of analyzing data by using pandas.

The prerequisite for doing any data related operations in Python, such as data cleansing, data aggregation, data transformation, and data visualisation, is to load data into Python. Depends on the types of data files (e.g. .csv, .txt, .xls .tsv, .html, .json and some relational databases etc.) and their size, different methods should be applied to deal with this initial operation accordingly

```
#Load the data
employee_df=pd.read_csv('/home/nimish/Downloads/WA_Fn-UseC_-HR-Employee-Attrition.csv')#
```


Activities
Terminal
Thu 20:59
nimish@pc: ~/Project
File Edit View Search Terminal Tabs Help
nimish@pc: ~/Project
nimish@pc: ~/Project
nimish@pc:~/Project$ vi emp.py
nimish@pc:~/Project$ python3 emp.py
Age Attrition DailyRate DistanceFromHome Education ... WorkLifeBalance YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager
count 1470.000000 1470.000000 1470.000000 1470.000000 1470.000000 ... 1470.000000 1470.000000 1470.000000 1470.000000 1470.000000
mean 36.923810 0.161224 802.485714 9.192517 2.912925 ... 2.761224 7.008163 4.229252 2.187755 4.123129
std 9.135373 0.367863 403.509100 8.106864 1.024165 ... 0.706476 6.126525 3.623137 3.222430 3.568136
min 18.000000 0.000000 102.000000 1.000000 1.000000 ... 1.000000 0.000000 0.000000 0.000000 0.000000
25% 30.000000 0.000000 465.000000 2.000000 2.000000 ... 2.000000 3.000000 2.000000 0.000000 2.000000
50% 36.000000 0.000000 802.000000 7.000000 3.000000 ... 3.000000 5.000000 3.000000 1.000000 3.000000
75% 43.000000 0.000000 1157.000000 14.000000 4.000000 ... 3.000000 9.000000 7.000000 3.000000 7.000000
max 60.000000 1.000000 1499.000000 29.000000 5.000000 ... 4.000000 40.000000 18.000000 15.000000 17.000000
[8 rows x 29 columns]
Age Attrition DailyRate DistanceFromHome Education ... WorkLifeBalance YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager
count 1470.000000 1470.000000 1470.000000 1470.000000 1470.000000 ... 1470.000000 1470.000000 1470.000000 1470.000000 1470.000000
mean 36.923810 0.161224 802.485714 9.192517 2.912925 ... 2.761224 7.008163 4.229252 2.187755 4.123129
std 9.135373 0.367863 403.509100 8.106864 1.024165 ... 0.706476 6.126525 3.623137 3.222430 3.568136
min 18.000000 0.000000 102.000000 1.000000 1.000000 ... 1.000000 0.000000 0.000000 0.000000 0.000000
25% 30.000000 0.000000 465.000000 2.000000 2.000000 ... 2.000000 3.000000 2.000000 0.000000 2.000000
50% 36.000000 0.000000 802.000000 7.000000 3.000000 ... 3.000000 5.000000 3.000000 1.000000 3.000000
75% 43.000000 0.000000 1157.000000 14.000000 4.000000 ... 3.000000 9.000000 7.000000 3.000000 7.000000
max 60.000000 1.000000 1499.000000 29.000000 5.000000 ... 4.000000 40.000000 18.000000 15.000000 17.000000
[8 rows x 25 columns]
nimish@pc:~/Project$ vi emp.py
nimish@pc:~/Project$


Activities
Terminal
Thu 21:19
Laptop battery low
Approximately 14 minutes remaining (9%)
File Edit View Search Terminal Tabs Help
nimish@pc: ~/Project
nimish@pc: ~/Project
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 57 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 Age 1470 non-null int64
1 Attrition 1470 non-null int64
2 BusinessTravel 1470 non-null object
3 DailyRate 1470 non-null int64
4 Department 1470 non-null object
5 DistanceFromHome 1470 non-null int64
6 Education 1470 non-null int64
7 EducationField 1470 non-null object
8 EnvironmentSatisfaction 1470 non-null int64
9 Gender 1470 non-null object
10 HourlyRate 1470 non-null int64
11 JobInvolvement 1470 non-null int64
12 JobLevel 1470 non-null int64
13 JobRole 1470 non-null object
14 JobSatisfaction 1470 non-null int64
15 MaritalStatus 1470 non-null object
16 MonthlyIncome 1470 non-null int64
17 MonthlyRate 1470 non-null int64
18 NumCompaniesWorked 1470 non-null int64
19 OverTime 1470 non-null int64
20 PercentSalaryHike 1470 non-null int64
21 PerformanceRating 1470 non-null int64
22 RelationshipSatisfaction 1470 non-null int64
23 StockOptionLevel 1470 non-null int64
24 TotalWorkingYears 1470 non-null int64
25 TrainingTimesLastYear 1470 non-null int64
26 WorkLifeBalance 1470 non-null int64
27 YearsAtCompany 1470 non-null int64
28 YearsInCurrentRole 1470 non-null int64
29 YearsSinceLastPromotion 1470 non-null int64
30 YearsWithCurrManager 1470 non-null int64
31 Non-Travel 1470 non-null uint8
32 Travel_Frequently 1470 non-null uint8
33 Travel_Rarely 1470 non-null uint8
34 Human Resources 1470 non-null uint8
35 Research & Development 1470 non-null uint8
36 Sales 1470 non-null uint8
37 Human Resources 1470 non-null uint8
38 Life Sciences 1470 non-null uint8
39 Marketing 1470 non-null uint8
40 Medical 1470 non-null uint8
41 Other 1470 non-null uint8
42 Technical Degree 1470 non-null uint8
43 Female 1470 non-null uint8
44 Male 1470 non-null uint8
45 Healthcare Representative 1470 non-null uint8
46 Human Resources 1470 non-null uint8
47 Laboratory Technician 1470 non-null uint8
48 Manager 1470 non-null uint8



➢ TRAINING AND TEST DATASETS

➢ Training Dataset

used to fit the parameters (e.g., weights) of, for example, a classifier.

For classification tasks, a supervised learning algorithm looks at the training dataset to determine, or learn, the optimal combinations of variables that will generate a good predictive model.

The goal is to produce a trained (fitted) model that generalizes well to new, unknown data. The fitted model is evaluated using “new” examples from the held-out datasets (validation and test datasets) to estimate the model’s accuracy in classifying new data.

To reduce the risk of issues such as overfitting, the examples in the validation and test datasets should not be used to train the model. Most approaches that search through training data for empirical relationships tend to overfit the data, meaning that they can identify and exploit apparent relationships in the training data that do not hold in general.

➢ Test Dataset

Test dataset is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. If a model fit to the training dataset also fits the test dataset well, minimal overfitting has taken place. A better fitting of the training dataset as opposed to the test dataset usually points to overfitting.

A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier. To do this, the final model is used to predict classifications of examples in the test set. Those predictions are compared to the examples' true classifications to assess the model's accuracy.

In a scenario where both validation and test datasets are used, the test dataset is typically used to assess the final model that is selected during the validation process.

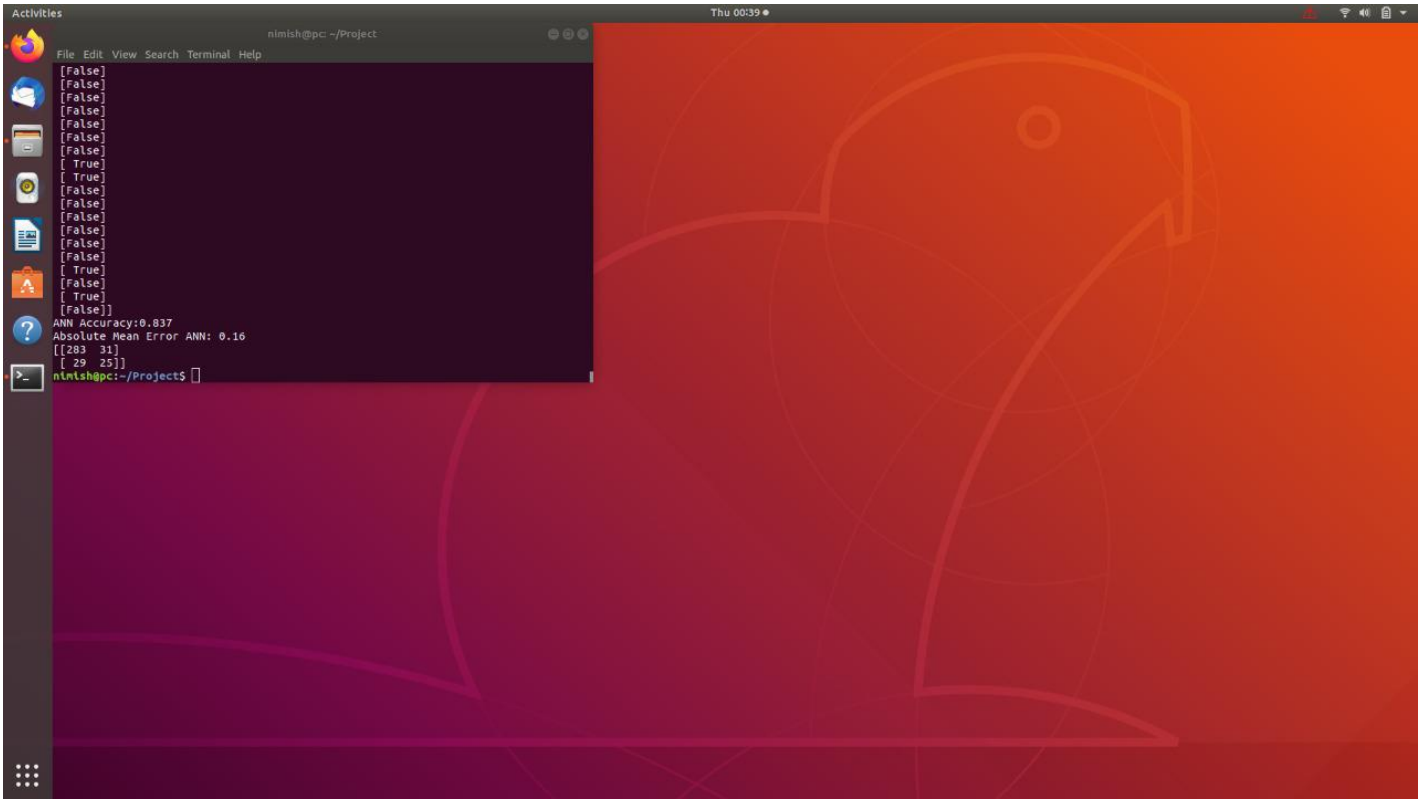
In the case where the original dataset is partitioned into two subsets (training and test datasets), the test dataset might assess the model only once.

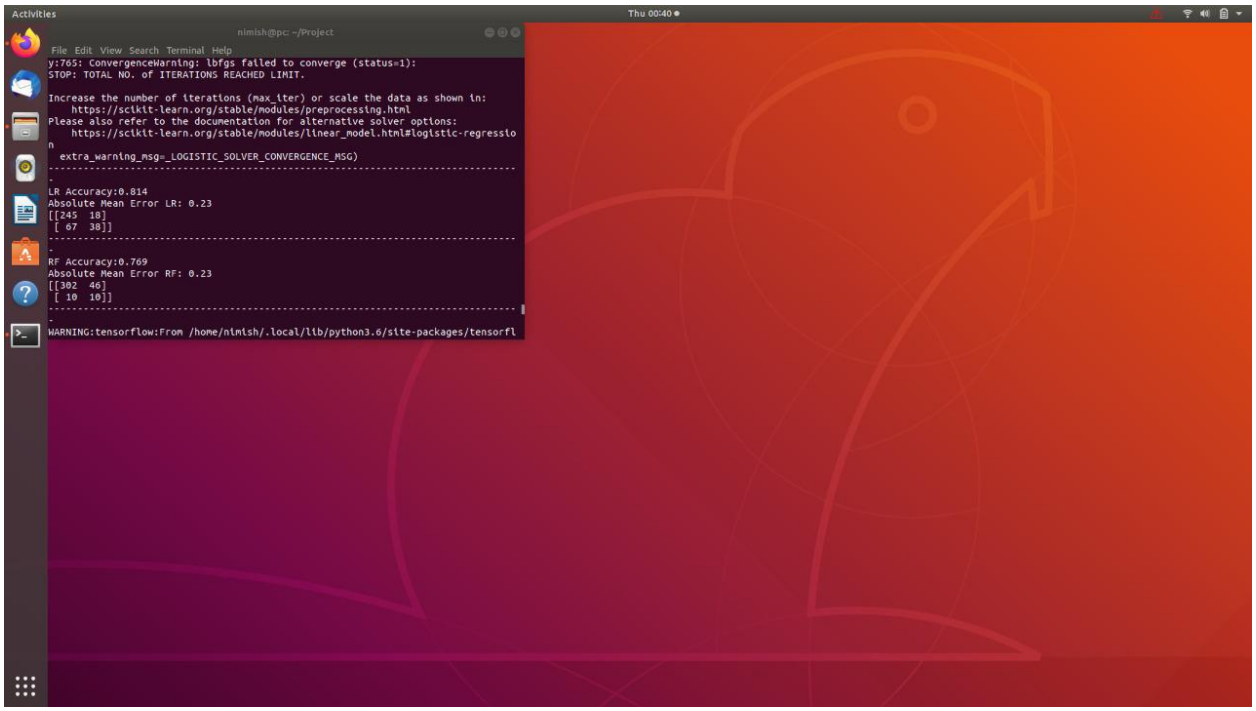
- Emp_leaving.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
employee_df=pd.read_csv('/home/nimish/Downloads/WA_Fn-UseC_-HR-Employee-
Attrition.csv')
#binary categorical data into numerical
employee_df['Attrition'] = employee_df['Attrition'].apply(lambda x: 1 if x == 'Yes' else 0)
employee_df['OverTime'] = employee_df['OverTime'].apply(lambda x: 1 if x == 'Yes'else 0)
employee_df['Over18'] = employee_df['Over18'].apply(lambda x: 1 if x == 'Y' else 0)
employee_df.hist(bins = 30, figsize= (15, 15),color='r')
#print(employee_df.describe())
employee_df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis = 1,
inplace = True)
#print(employee_df.describe())
#Correlated features don't improve model performance. It is wise to remove correlated features
correlations = employee_df.corr()
plt.subplots(figsize = (20, 20))
```

```
sns.heatmap(correlations, annot = True)
#Age Variable Correlation Analysisi
plt.figure(figsize = (15, 10), dpi = 300)
age = sns.countplot(x = 'Age', hue = 'Attrition', data = employee_df )
age.set_xticklabels(age.get_xticklabels(),fontsize=7, rotation=40, ha="right")
#plt.tight_layout()
#Job Role
sns.countplot(x = 'JobRole', hue = 'Attrition', data = employee_df)
#distance from home
#sns.kdeplot(left_df['DistanceFromHome'], label = 'Employee left', shade = True, color = 'r')
#sns.kdeplot(stay_df['DistanceFromHome'], label = 'Employee stay', shade = True, color = 'b')
plt.show()
```

Output





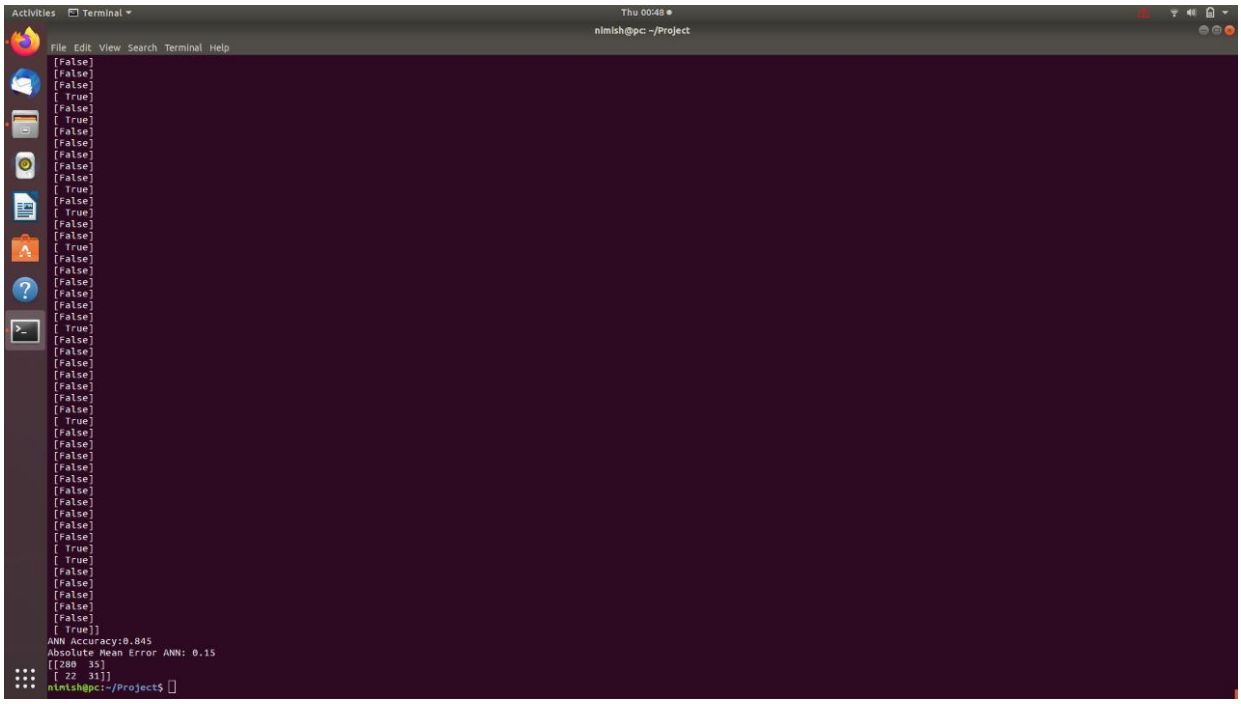
- emp_leaving_ml.py

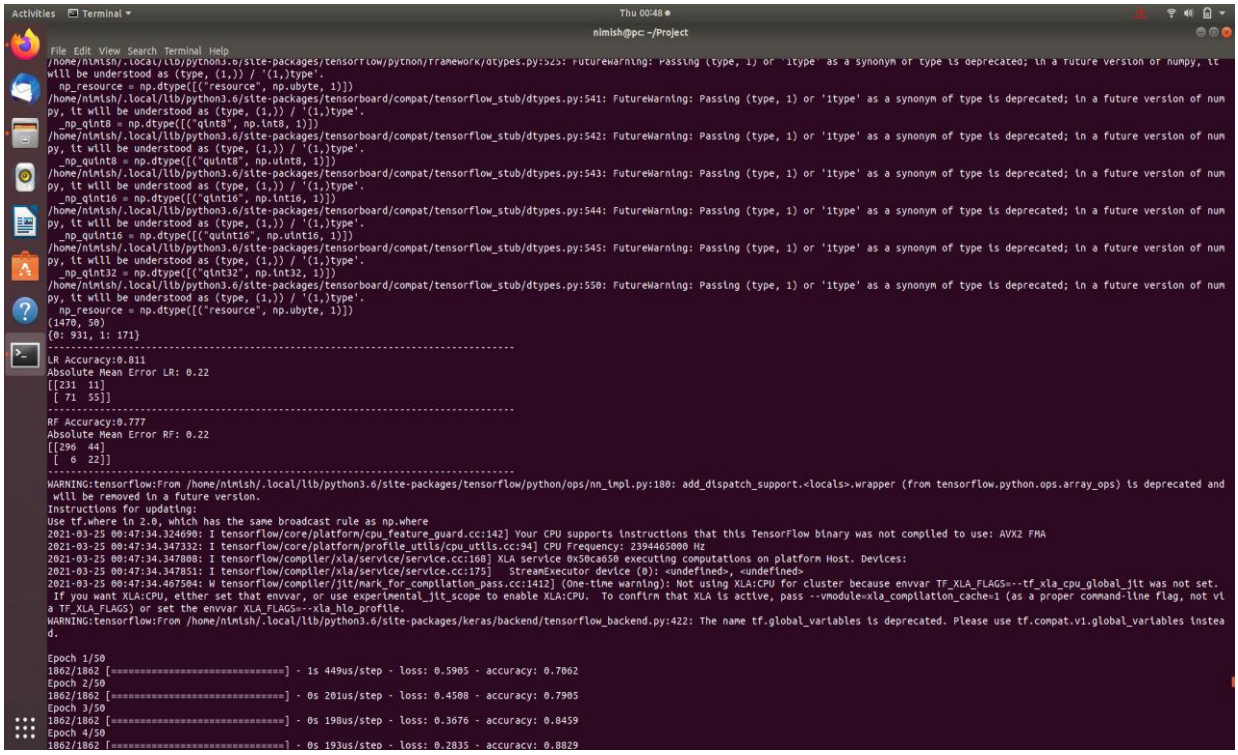
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
#import imblearn
from imblearn.over_sampling import SMOTE
#from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTENC
from sklearn.metrics import confusion_matrix,mean_absolute_error,accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
from keras.models import Sequential
from keras.layers import Dense, Dropout
employee_df=pd.read_csv('/home/nimish/Downloads/WA_Fn-UseC_-HR-Employee-Attrition.csv')
employee_df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis = 1, inplace =
True)
```

```
#binary categorical data into numerical
employee_df['Attrition'] = employee_df['Attrition'].apply(lambda x: 1 if x == 'Yes' else 0)
employee_df['OverTime'] = employee_df['OverTime'].apply(lambda x: 1 if x == 'Yes'else 0)

#Encode categorical variables i.e. get_dummies()
bt = pd.get_dummies(employee_df['BusinessTravel'])
dpt = pd.get_dummies(employee_df['Department'])
edu = pd.get_dummies(employee_df['EducationField'])
gender = pd.get_dummies(employee_df['Gender'])
jrole = pd.get_dummies(employee_df['JobRole'])
mstatus = pd.get_dummies(employee_df['MaritalStatus'])
frames = [employee_df,bt,dpt,edu,gender,jrole,mstatus]
X_all = pd.concat(frames,axis=1)
X_all.drop(['BusinessTravel', 'Department', 'Gender', 'EducationField','JobRole','MaritalStatus'], axis = 1,
inplace = True)
y = employee_df['Attrition']
#X_all.info()
X_all.drop(['Attrition'], axis = 1, inplace = True)
scaler = MinMaxScaler()
X = scaler.fit_transform(X_all)
print(employee_df.shape)
print(employee_df.shape)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

#Check imbalance of dataset
unique, count = np.unique(y_train, return_counts=True)
y_train_count = { k:v for (k,v) in zip(unique,count)}
print(y_train_count)

#SMOTE method to oversample the minority classes
oversampler = SMOTE(random_state=0)
X_smote_train, y_smote_train = oversampler.fit_resample(X_train,y_train)

#Logistic Regression
lr = LogisticRegression()
scoring = 'accuracy'
lr.fit(X_smote_train, y_smote_train)
y_pred = lr.predict(X_test)
result = model_selection.cross_val_score(lr,X_smote_train,y_smote_train, scoring=scoring)
print('LR Accuracy:%.3f' % result.mean())
```

```
cm_lr = confusion_matrix(y_pred, y_test)
sns.heatmap(cm_lr, annot= True)
mae = mean_absolute_error(y_test, y_pred)
print("Absolute Mean Error LR: %.2f" % mae)
print(cm_lr)
RF = RandomForestClassifier()
RF.fit(X_smote_train, y_smote_train)
rf_y_pred = RF.predict(X_test)
cm_rf = confusion_matrix(rf_y_pred, y_test)
sns.heatmap(cm_rf, annot= True)
print(cm_rf)
mae_rf = mean_absolute_error(y_test, y_pred)
print("Absolute Mean Error RF: %.2f" % mae_rf)
rf_result = accuracy_score(y_pred,y_test)
print('RF Accuracy:%.3f' % rf_result.mean())
model = Sequential()
model.add(Dense(units = 50, activation = 'relu', input_shape = (50, )))
model.add(Dense(units = 500, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 500, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 50, activation = 'relu'))
model.add(Dense(units = 1, activation = 'sigmoid'))
model.compile(optimizer='adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
epochs_hist = model.fit(X_smote_train, y_smote_train, epochs = 50, batch_size = 50)
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
print(y_pred)
cm = confusion_matrix(y_pred, y_test)
sns.heatmap(cm, annot= True)
print(cm)
mae = mean_absolute_error(y_test, y_pred)
print("Absolute Mean Error ANN: %.2f" % mae)
ann_result = accuracy_score(y_pred,y_test)
print('ANN Accuracy:%.3f' % ann_result.mean())
plt.show()
```

Output


Activities
Terminal
Thu 00:48
nimish@pc: ~/Project
File Edit View Search Terminal Help
[False]
[False]
[False]
[True]
[False]
[True]
[False]
[False]
[False]
[False]
[True]
[False]
[True]
[False]
[False]
[True]
[False]
[False]
[False]
[False]
[False]
[False]
[True]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[False]
[True]
[True]
[False]
[False]
[False]
[False]
[False]
[True]]
ANN Accuracy:0.845
Absolute Mean Error ANN: 0.15
[[280 35]
[22 31]]
nimish@pc:~/Project$


Activities
Terminal
Thu 00:48
nimish@pc: ~/Project
File Edit View Search Terminal Help
/home/nimish/.local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing (type, 1) or 'itype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_resource = np.dtype([("resource", np.ubyte, 1)])
/home/nimish/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or 'itype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/nimish/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or 'itype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/nimish/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning: Passing (type, 1) or 'itype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/nimish/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning: Passing (type, 1) or 'itype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/nimish/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: FutureWarning: Passing (type, 1) or 'itype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_qint32 = np.dtype([("qint32", np.int32, 1)])
/home/nimish/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: FutureWarning: Passing (type, 1) or 'itype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
np_resource = np.dtype([("resource", np.ubyte, 1)])
(1470, 50)
(0: 931, 1: 171)
LR Accuracy:0.811
Absolute Mean Error LR: 0.22
[[231 11]
[71 55]]
RF Accuracy:0.777
Absolute Mean Error RF: 0.22
[[296 44]
[6 22]]
WARNING:tensorflow:From /home/nimish/.local/lib/python3.6/site-packages/tensorflow/python/ops/nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
2021-03-25 00:47:34.324690: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2021-03-25 00:47:34.347332: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2394465000 Hz
2021-03-25 00:47:34.347800: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x50ca650 executing computations on platform Host. Devices:
2021-03-25 00:47:34.347851: I tensorflow/compiler/xla/service/service.cc:175] StreamExecutor device (0): <undefined>, <undefined>
2021-03-25 00:47:34.467504: W tensorflow/compiler/jit/mark_for_compilation_pass.cc:1412] (One-time warning): Not using XLA:CPU for cluster because envvar TF_XLA_FLAGS=--tf_xla_cpu_global_jit was not set. If you want XLA:CPU, either set that envvar, or use experimental_jit_scope to enable XLA:CPU. To confirm that XLA is active, pass --vmodule=xla_compilation_cache=1 (as a proper command-line flag, not via TF_XLA_FLAGS) or set the envvar XLA_FLAGS=--xla_hlo_profile.
WARNING:tensorflow:From /home/nimish/.local/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
Epoch 1/50
1862/1862 [==========] - 1s 449us/step - loss: 0.5905 - accuracy: 0.7062
Epoch 2/50
1862/1862 [==========] - 0s 201us/step - loss: 0.4508 - accuracy: 0.7905
Epoch 3/50
1862/1862 [==========] - 0s 198us/step - loss: 0.3676 - accuracy: 0.8459
Epoch 4/50
1862/1862 [==========] - 0s 193us/step - loss: 0.2835 - accuracy: 0.8829

9.1 Performance Matrices

The most commonly used Performance metrics for classification problem are as follows,

- 'Accuracy
- Confusion Matrix
- Precision, Recall, and F1 score
- ROC AUC

9.1.1 Accuracy

Accuracy is the simple ratio between the number of correctly classified points to the total number of points.

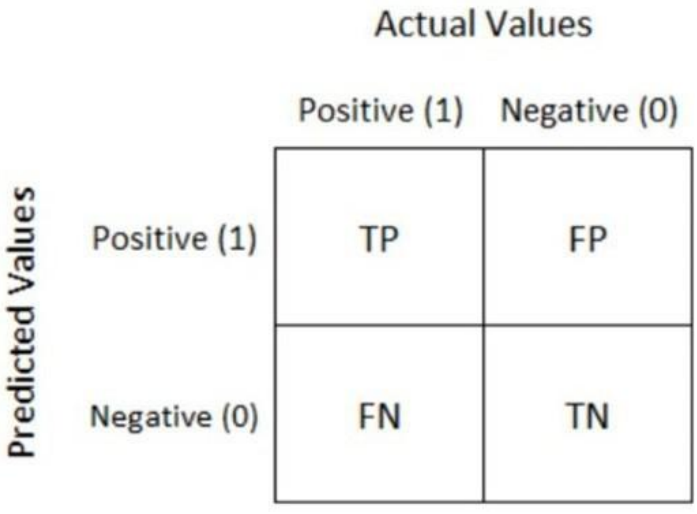
Limitations of Accuracy:

If the data set is highly imbalanced, and the model classifies all the data points as the majority class data points, the accuracy will be high. This makes accuracy not a reliable performance metric for imbalanced data.

From accuracy, the probability of the predictions of the model can be derived. So from accuracy, we can not measure how good the predictions of the model are.

9.1.2 Confusion Matrix

Confusion Matrix is a summary of predicted results in specific table layout that allows visualization of the performance measure of the machine learning model for a binary classification problem (2 classes) or



multiclass classification problem (more than 2 classes)

- TP means True Positive.

  It can be interpreted as the model predicted positive class and it is True .
- FP means False Positive.
  It can be interpreted as the model predicted positive class but it is False.
- FN means False Negative.
  It can be interpreted as the model predicted negative class but it is False.
- TN means True Negative.
  It can be interpreted as the model predicted negative class and it is True.

For a sensible model, the principal diagonal element values will be high and the off-diagonal element values will be below i.e., TP, TN will be high

Advantages of Confusion Matrix:

- The confusion matrix provides detailed results of the classification
- Derivatives of the confusion matrix are widely used.
- Visual inspection of results can be enhanced by using a heatmap**.**

9.1.3 Precision, Recall and F1 Score

- **Precision** is the fraction of the correctly classified instances from the total classified instances. Precision helps us understand how useful the results are.

$$Precision = \frac{TP}{TP+FP}$$

- **Recall** is the fraction of the correctly classified instances from the total classified instances. Recall helps us understand how complete the results are.

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

- **F1 score** is the harmonic mean of precision and recall. It is given as,

$$F1\ score = \frac{2*Precision*Recall}{Precision+Recall}$$

The F-score is often used in the field of information retrieval for measuring search, document classification, and query classification performance

The F-score has been widely used in the natural language processing literature, such as the evaluation of named entity recognition and word segmentation.

9.1.4 ROC AUC

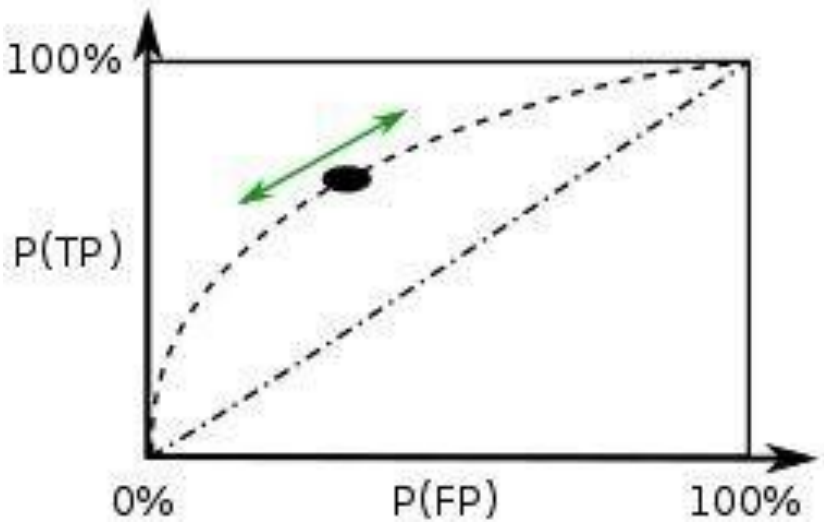
A Receiver Operating Characteristic curve or ROC curve is created by plotting the True Positive (TP) against the False Positive (FP) at various threshold settings. The ROC curve is generated by plotting the cumulative distribution function of the True Positive in the y-axis versus the cumulative distribu-tion function of the False Positive on the x-axis.

The dashed curved line is the ROC Curve

The area under the ROC curve (ROC AUC) is the single-valued metric used for evaluating the performance.

The higher the AUC, the better the performance of the model at distinguishing between the classes.

In general, an AUC of 0.5 suggests no discrimination, a value between 0.5–0.7 is acceptable and anything above 0.7 is good-to-go-model. However, medical diagnosis models, usually AUC of 0.95 or more, are considered to be good-to-go-model.



Advantages of ROC AUC :

- ROC curves are widely used to compare and evaluate different classification algorithms.
- ROC curve is widely used when the dataset is imbalanced.
- ROC curves are also used in verification of forecasting meteorology

9.2 CORRELATION

Correlation means association-more precisely it is a measure of the extent to which two variables are related.

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.

Statistics and data science are often concerned about the relationships between two or more variables (or features) of a dataset. Each data point in the dataset is an observation, and the features are the properties or attributes of those observations.

```
# Calculate Correlations
corr = df_emp.corr()
mask = np.zeros_like(corr) mask[np.triu_indices_from(mask)] = True # Heatmap
plt.figure(figsize=(15, 10))
sns.heatmap(corr, vmax=.5, mask=mask, # annot=True,fmt='.2f', linewidths=.2, cmap="YlGnBu")
plt.show()
```

OUTPUT

There are three possible results of a correlational study: a positive correlation, a

negative correlation, and no correlation.

A positive correlation is a relationship between two variables in which both variables move in the same direction. Therefore, when one variable increases as the other variable increases, or one variable decreases while the other decreases. An example of positive correlation would be height and weight. Taller people tend to be heavier.

A negative correlation is a relationship between two variables in which an increase in one variable is associated with a decrease in the other. An example of negative correlation would be height above sea level and temperature. As you climb the mountain (increase in height) it gets colder (decrease in temperature).

A zero correlation exists when there is no relationship between two variables. For example there is no relationship between the amount of tea drunk and level of intelligence.

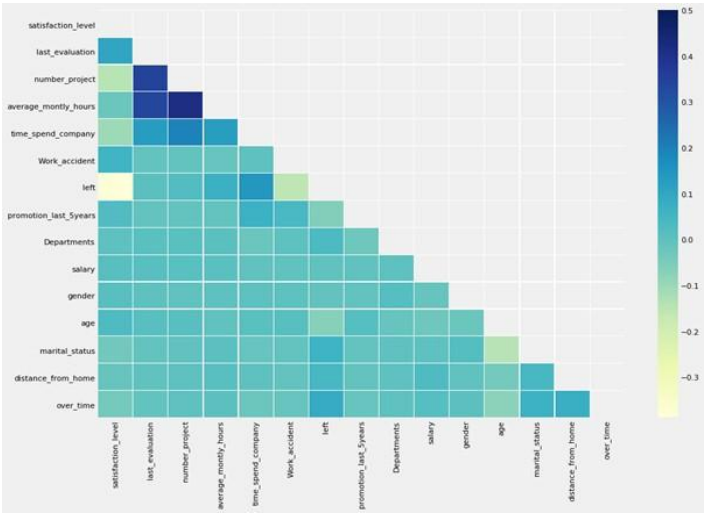


Figure 9.1: Correlation - Heat Map

```
# Find correlations with the target and sort
df_trans = df_emp.copy()
df_trans['Target'] = df_trans['left']
correlations = df_trans.corr()['Target'].sort_values()
print('Most Positive Correlations: \n', correlations.tail(5))
print('\nMost Negative Correlations: \n', correlations.head(5))
```

OUTPUT:

```
Most Positive Correlations:
 average_monthly_hours    0.071287
over_time                 0.089121
time_spend_company        0.144822
left                      1.000000
Target                    1.000000
Name: Target, dtype: float64

Most Negative Correlations:
 satisfaction_level      -0.388375
Work_accident            -0.154622
age                      -0.067498
promotion_last_5years    -0.061788
gender                   -0.009835
Name: Target, dtype: float64
```

10 MODEL COMPARISION

When you have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives.

The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize.

A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

Algorithm	ROC AUC Mean	ROC AUC STD	Accuracy Mean	Accuracy STD
Logistic Regression	82.00	1.81	75.49	1.27
Random Forest	99.06	0.43	96.83	0.41
ANN	85.82	1.96	83.76	1.04

Figure 10.1 Algorithm Accuracy Comparison

11 ANALYSING THE CAUSES OF TURNOVER:

Several observations:

Promotions:

Employees are far more likely to quit their job if they haven't received a promotion in the last 5 years

Time With Company:

Here, The three-year mark looks like a time to be a crucial point in an employee's career. Most of them quit their job around the three-year mark. Another important point is 6-years point, where the employee is very unlikely to leave.

Number of Projects:

Employee engagement is another critical factor to influence the employee to leave the company. Employees with 3-5 projects are less likely to leave the company. The employee with less and more number of projects are likely to leave.

12 STRATEGIC RETENTION PLAN OF OUR PROJECT ANALYSIS:

The stronger indicators of people leaving include:

Monthly Income: people on higher wages are less likely to leave the company. Hence, efforts should be made to gather information on industry benchmarks in the current local market to determine if the company is providing competitive wages.

Over Time: people who work overtime are more likely to leave the company. Hence efforts must be taken to appropriately scope projects upfront with adequate support and manpower so as to reduce the use of overtime.

Age: Employees in relatively young age bracket 25–35 are more likely to leave. Hence, efforts should be made to clearly articulate the long-term vision of the company and young employees fit in that vision, as well as provide incentives in the form of clear paths to promotion for instance.

Distance From Home: Employees who live further from home are more likely to leave the company. Hence, efforts should be made to provide support in the form of company transportation for clusters of employees leaving the same area, or in the form of Transportation Allowance. Initial screening of employees based on their home location is probably not recommended as it would be regarded as a form of discrimination as long as employees make it to work on time every day.

Total Working Years: Three-year mark looks like a time to be a crucial point in an employee's career. Most of them quit their job around the three-year mark. The more experienced employees are less likely to leave. Proper satisfaction measures should be taken.

13 RESULT AND DISSCUSSION

This report expects to foresee whether a worker will proceed or leave the association in view of the examination of the information of past representatives. The expectation factors incorporate fulfillment level, last assessment, normal month to month hours, pay, work mischance, advancement, time spent at the organization and division, in view of these parameters, distinctive machine learning models like strategic relapse, choice tree characterization and so on are connected to anticipate which worker will leave straightaway and the components that are most critical in this choice.

Through this project an organization can choose its strategies to keep great representatives from leaving the organization. Information science part that utilized as a part of this report is to take crude information from csv document and then apply diverse handling component to settle on information helpful in settling on choices from it like arrangement of dataset, Label Encoding, Onehot Encoding and include scaling. It at that point applies diverse relapse models to anticipate whether the worker will leave the organization or not as 0 and 1. If 0 comes in the outcome that implies that the worker will proceed with the organization, however if 1 comes then the representative will leave the organization. Here is given the example information that we utilized for making expectations, it is in an unthinkable frame which contains segments as fulfillment level, last assessment, number of undertakings, normal month to month hours, years spent in the organization, work mischance, advancement, office and pay.

When the accuracy of the result is being calculated from the previous analysed data with the help of confusion matrix and the accuracy score, this result is being compared with the available data to find the result accuracy and 97% of the predictions are made correct.

14 CONCLUSION

In this investigation, we become more acquainted with that maintenance of a representative inside an association can be discover utilizing strategic relapse procedure, which delivers an outcome with 97% exactness. It can likewise help in discovering the components that are influencing the representatives in the association like pay level, work stack, advancements and so forth. The future extent of information science is brilliant; consequently, this procedure can be utilized as a part of any association for better worker administration and for their fulfillment. This paper can be additionally reached out as it requires information as .csv records just, so this impediment can be expelled

15 References

[1] Piotr Płoński (MLJAR), “Human-first Machine Learning Platform,” Human Resource Analytics
Predict Employee Attrition.
[2] Le Zhang and Graham Williams (Data Scientist, Microsoft), “Employee Retention with R based
Data Science Accelarator”.
[3] Ashish Mishra (Data Scientist, Experfy), “Using Machine Learning to Predict and explain Employee Attrition”.
[4] Rupesh Khare, Dimple Kaloya and Gauri Gupta, “Employee Attrition Risk Assessment using Logistic Regression Analysis,” from 2nd IIMA International Conference on Advanced Data Analysis, Business Analytics and Intelligence.
[5] Randy Lao, “Predicting Employee KerneloverKaggle.
[6] Sandra W. Pyke & Peter M. Sheridan, “Logistic Regression Analysis of Graduate Student Retention,” from The Canadian Journal of Higher Education, Vol. XXIII-2, 1993.
[7] Prof. Dr. Vjollca Hasani and Prof. Dr. Alba Dumi, “Application of Logistic Regression in the Study
of Students’ Performance Level,” Journal of Educational and Social Research Italy.
[8] Dr. Jonathan Erhardt, “Artificial Intelligence: Opportunities and Risks,” Policy paper by the Effective Altruism Foundation.
[9] Sofia Stromberg’s, “Binary Logistic Regression and its application to data from a study of children's recognition of their own recorded voices” term paper in statically method.