# PROFESSIONAL TRANING REPORT

**(DOCUMENTATION )**

**At**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(DEEMED TO BE UNIVERSITY)**

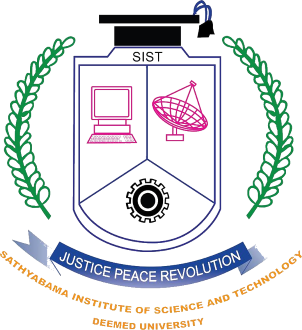
Submitted in partial fulfilment of the required for the award of

Bachelor of Engineering Degree in Computer Science and Engineering

By

Bhargavi Reddy

**Reg No:39110166**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI-600119, TAMILNADU**

**APRIL 2022**

|  |  |  |
| --- | --- | --- |
|  | **SATHYABAMA**  **INSTITUTE OF SCIENCE AND TECHNOLOGY** (DEEMED TO BE UNIVERSITY) **Accredited with Grade “A” by NAAC**  (Established under Section 3 of UGC Act, 1956)  JEPPIAAR NAGAR, RAJIV GANDHI SALAI  CHENNAI– 600119  [**www.sathyabama.ac.in**](http://www.sathyabama.ac.in) |  |

**DEPARTEMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIED CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **Bhargavi reddy (Reg. No: 39110166)** who carried out the project entitled “**Employee Promotion Prediction ”** under my supervision from March 2022 to April 2022.

**Internal Guide**

**Ms.Lavanya.G**

## **Submitted for Viva voice Examination held on**

**Internal Examiner External Examiner**

**DECLARATION**

I,Bhargavi Reddyhereby declare that the project report entitled **Employee Promotion Prediction** done by me under the guidance of **Ms.Lavanya.G** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

**DATE :**

**PLACE:CHENNAI Signature of The Candidate**

**ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. Sasikala M.E., Ph. D**, **Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **MS.lavanya.G** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project

**ABSTRACT**

Employee turnover has been identified as a key issue for organizations because of its adverse impact on work place productivity and long term growth strategies. To solve this problem, organizations use machine learning techniques to predict employee turnover.

Accurate predictions enable organizations to take action for retention or succession planning of employees. However, the data for this modeling problem comes from HR Information Systems (HRIS); these are typically under-funded compared to the Information Systems of other domains in the organization which are directly related to its priorities.

This leads to the prevalence of noise in the data that renders predictive models prone to over-fitting and hence inaccurate. This is the key challenge that is the focus of this paper, and one that has not been addressed historically. The novel contribution of this paper is to explore the application of Extreme Gradient Boosting (XGBoost) technique which is more robust because of its regularization formulation. Data from the HRIS of a global retailer is used to compare XGBoost against six historically used supervised classifiers and demonstrate its significantly higher accuracy for predicting employee turnover.

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | Abstract |  |
| 1 | INTRODUCTION | 1 |
|  |  |  |
| 2 | AIM, SCOPE OF INVESTIGATION | 3 |
| 3 | MATERIALS, METHODS AND ALGORITHMS USED | 4 |
|  | 3.1. PROJECT DESCRIPTION | 4 |
|  | 3.2. LIBRARIES USED | 5 |
|  | 3.3. Importing the data set from .csv file | 5 |
|  | 3.4. *Data Pre-Processing* | 7 |
|  | 3.5. Splitting data for training and testing | 8 |
|  | 3.6. Project Flow | 9 |
| 4 | RESULT, PEFORMANCE ANALYSIS | 16 |
| 5 | SUMMARY AND CONCLUSION | 16 |
|  | APPENDIX | 17 |
|  | REFERENCES | 21 |

**CHAPTER \_ 1**

**INTRODUCTION**

Promotion or career advancement is a process through which an employee of a company is given a higher share of duties, a higher pay scale, or both. Promotion is not just beneficial for employees but is also highly crucial for the employer or business owners. It boosts the morale of promoted employees, increases their productivity, and hence improves upon the overall profits earned by the organization.

The client is facing a problem in identifying the right people for promotion. The company needs help in identifying the eligible candidates at a particular checkpoint so that they can expedite the entire promotion cycle. This problem can be solved by building a machine learning that automates the process of promoting an employee. we make use of employee datasets to build different classification ML models such as Decision tree, Random forest, KNN, and xgboost. The best model is selected and saved for integration with the flask application.

The problem of employee turnover has shot to prominence in organizations because of its negative impacts on issues ranging from work place morale and productivity, to disruptions in project continuity and to long term growth strategies. One way organizations deal with this problem is by predicting the risk of attrition of employees using machine learning techniques thus giving organizations leaders and Human Resources (HR) the foresight to take pro-active action for retention or plan for succession.

However, the machine learning techniques historically used to solve this problem fail to account for the noise in the data in most HR Information Systems (HRIS). Most organizations have not prioritized investments in efficient HRIS solutions that would capture an employee’s data during his/her tenure.

One of the major factors is the limited understanding of benefits and cost. It is still difficult to measure the return of investment in HRIS . This leads to noise in the data, which in turn attenuates the generalization capability of these algorithms.

**CHAPTER\_2**

**AIM AND SCOPE OF PRESENT INVESTIGATION**

**AIM** : To analyze and visualize the data using .csv or .tsv files and by using graphs to support our analysis from a data set.

**SCOPE:** The client is facing a problem in identifying the right people for promotion. The company needs help in identifying the eligible candidates at a particular checkpoint so that they can expedite the entire promotion cycle. This problem can be solved by building a machine learning that automates the process of promoting an employee. we make use of employee datasets to build different classification ML models such as Decision tree, Random forest, KNN, and xgboost. The best model is selected and saved for integration with the flask application.

**CHAPTER \_3**

**METHODS , MATERIALS AND ALGORITHMS USED**

**3.SYSTEM SPECIFICATION**

**Hardware Requirements:**

1.System : Pentium i3 Processor.

2.Hard Disk : 500 GB.

3.Monitor : 15’’ LED

4.Input Devices : Keyboard, Mouse

5.Ram : 2 GB

**Software Requirements :**

1.Operating system : Windows 10.

2.Coding Language : Python

3.Jupyter notebook

**3.1 PROJECT DESCRIPTION:**

We are going to analyze the data of Employee which contains information like Employee ID, Date of Joining, Gender, Company Type, WFH Setup Available, Resource Allocation, Years of Experience, No. of Awards gained . We have got this dataset from Kaggle. We are going to analyze the information using Linear Regression in this dataset and try to present in a visual manner using different libraries in python programming language like NumPy, pandas, matplotlib and seaborn.

**3.2 LIBRARIES USED:**

We will generally be working on this project using the standard libraries such as:

* Sklearn
* Pandas
* Numpy
* Matplotlib
* Seaborn

Sklearn is the root important library from which we are going to import and make use of the algorithms or the metrics or the train\_test\_split for making predictions using the model generated using these.

Pandas is the standard library one would be dealing with while doing a machine learning project. As I have mentioned above that I will be using the data set which is stored in an excel file, we can use pandas to read the excel file.

Similarly, we will use Numpy to make use of the NumPy arrays, while we are calculating the errors or handling the data of some kind.

Matplotlib is one of the major necessities as we are going to use it for creating plots and to analyze our predictions based on that.

Seaborn is also a library used for visualizing the data using the plots. It was my personal opinion to choose this library out of other visualizing modules and libraries present. Moving forward there may be some other libraries that are used offhand but all serve for the same use.

We will break down more about the libraries and their usage once we start using them. I will try to describe that particular usage and its overall description too once we get into the coding part.

As we have seen that the above are the necessary libraries which we will import as full but for some we will only import the necessary modules (just to have a quick knowledge of exactly what modules we will be using).

We will import a few specific functions and algorithms from modules of Sklearn library, they are namely:

* train\_test\_split from model\_selection module of sklearn
* from sklearn. linear model import LinearRegression
* metrics module of sklearn

Model\_selection is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction.

train\_test\_split is a function in sklearn model selection for splitting data arrays into two subsets: for training data and for testing data.

With this function you don’t need to divide the dataset manually. By default, sklearn train\_test\_split will make random partitions for the two subsets. However, you can also specify a random state for the operation.

Linear Regression

Linear Regression is a linear model that assumes a linear relationship between input variables (independent variables ‘x’) and output variable(dependent variable-’y’) such that ‘y’ can be calculated from a linear combination of input variables(x).For single input variable,method is referred to as Simple Linear Linear Regression whereas for multiple input variables it is referred to as Multiple Linear Regression.

**3.3 Importing the data set from .csv file**

As we discussed earlier we will import the data set which was saved in csv (.csv extension). We will use pandas to do it, I will use two different dataframe to import data set into. One for the dataset used for training, testing, model and predictions and the other for visualization.

**3.4 Data Pre-Processing**

Data processing or data handling is one of the important steps needed to be taken before using the data front he given data set. Data processing is the task of converting data from a given form to a much more usable and desired form i.e., making it more meaningful and informative. Using Machine learning algorithms, mathematical modeling, and statistical knowledge, this entire process can be automated. The output of the process can be in any desired form like graphs, videos, charts, tables, images, and many more, depending on the task we are performing and the requirements of the machine.

The classic steps for data processing are as follows:

* Collection
* Preparation
* Input
* Processing
* Output
* Storage

Collection:

The most crucial step is to have the data of good quality and accuracy. The data we will be using is collected from UCL dataset for airfoil self-noise. Good data ensures that the results of the model are valid and can be trusted upon.

Preparation:

The collected data can be in a raw form which can’t be directly fed to the machine. So, this is a process of collecting data sets from different sources, analyzing the data sets and then constructing a new data set for further processing and exploration. This preparation can be performed either manually or from the automatic approach. Data can also be prepared in numeric forms also which would fasten the model’s learning.

For this preparation we have to first check for null values, for our self-noise data set we are lucky that we do not have any null values.

Processing:

In this stage, results are procured by the machine in a meaningful manner which can be inferred easily by the user. Output can be in the form of reports, graphs, videos, etc. We will go through this phase in the next part.

**3.5 Splitting data for training and testing:**

As already discussed in chapter 3.2, we will use train\_test\_split that is imported from the model\_selection of sklearn library. We have already discussed about the model\_selection module and its use, now we will understand about the train\_test\_split function:

Sklearn train\_test\_split has several parameters. A basic example of syntax will look like: [ train\_test\_split (X, y, train\_size = 0. \*, test\_size = 0. \*, random\_state = \*)].

* X, y - it is the dataset you are selecting to use.
* train\_size - this parameter sets the size of the training data set. There are three options: None - which is default, Int - which requires exact number of samples and Float - which ranges from 0.1 to 1.0.
* test\_size - this parameter specifies the size of the testing data set. The default state suits the training size. It will be set to 0.25 if the training size is set to default.

random\_state - the default mode performs a random split using np. random. Alternatively, you can add an integer using an exact number

**3.6 Project Flow:**

**Project Flow:**

* The user interacts with the UI to enter the input.
* Entered input is analyzed by the model which is integrated.
* Once the model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

* Data collection
  + Collect the dataset or create the dataset
* Visualizing and analyzing data
  + Univariate analysis
  + Multivariate analysis
  + Descriptive analysis
* Data pre-processing
  + Drop unwanted features
* Checking for null values
* Remove negative data
* Handling outlier
* Handling categorical data
* Handling Imbalanced data
* Splitting data into train and test
* Model building
* Import the model building libraries
* Initializing the model
* Training and testing the model
* Evaluating performance of the model
* Save the model
* Application Building
* Create an HTML file
* Build python code
* Run the Application

### **Pre-Requisites**

### In order to develop this project need to install the following software/packages:

**Step 1: Anaconda Navigator**

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning-related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform,  package management system. Anaconda comes with great tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code.

**Step 2: Python packages**

To build Machine learning models must require the following packages

* Sklearn: Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.
* NumPy: NumPy is a Python package that stands for 'Numerical Python. It is the core library for scientific computing, which contains a powerful n-dimensional array of object
* Pandas: pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language.
* Matplotlib: It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkit

Open anaconda prompt as administrator

* Type **“pip install numpy ”** and click enter
* Type **“pip install pandas ”** and click enter
* Type **“pip install scikit-learn”** and click enter.
* Type **”pip install matplotlib”** and click enter.
* Type **”pip install scipy"** and click enter.
* Type **”pip install pickle-mixin”** and click enter.
* Type **”pip install seaborn”** and click enter.
* Type **“pip install Flask ”** and click enter.

**PROJECT STRUCTURE:**

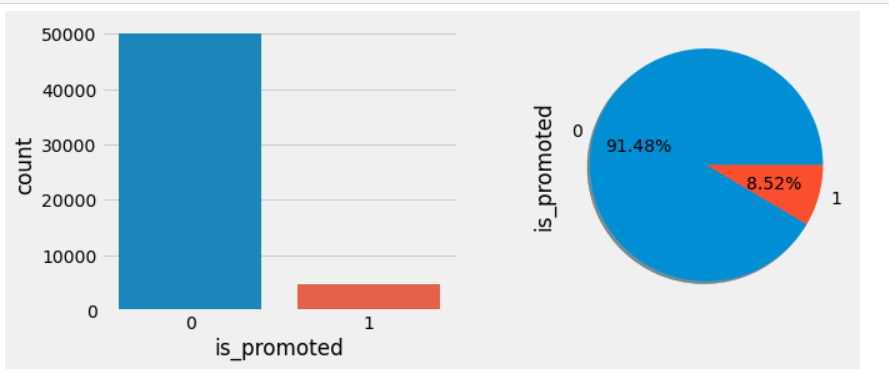
* Building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
* For IBM deployment app\_IBM.py file is used.
* Model.pkl is our saved model. Further we will use this model for flask integration.
* Training folder contains model training files and training\_ibm folder contains IBM model training files.

**VISUALISING AND ANALYSING THE DATA:**

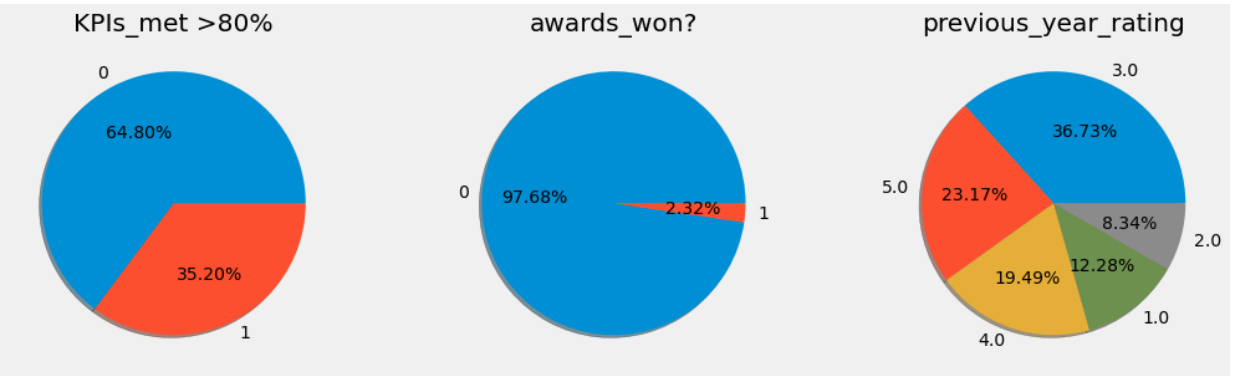
Univariate Analysis:

In simple words, univariate analysis is understanding the data with single feature.

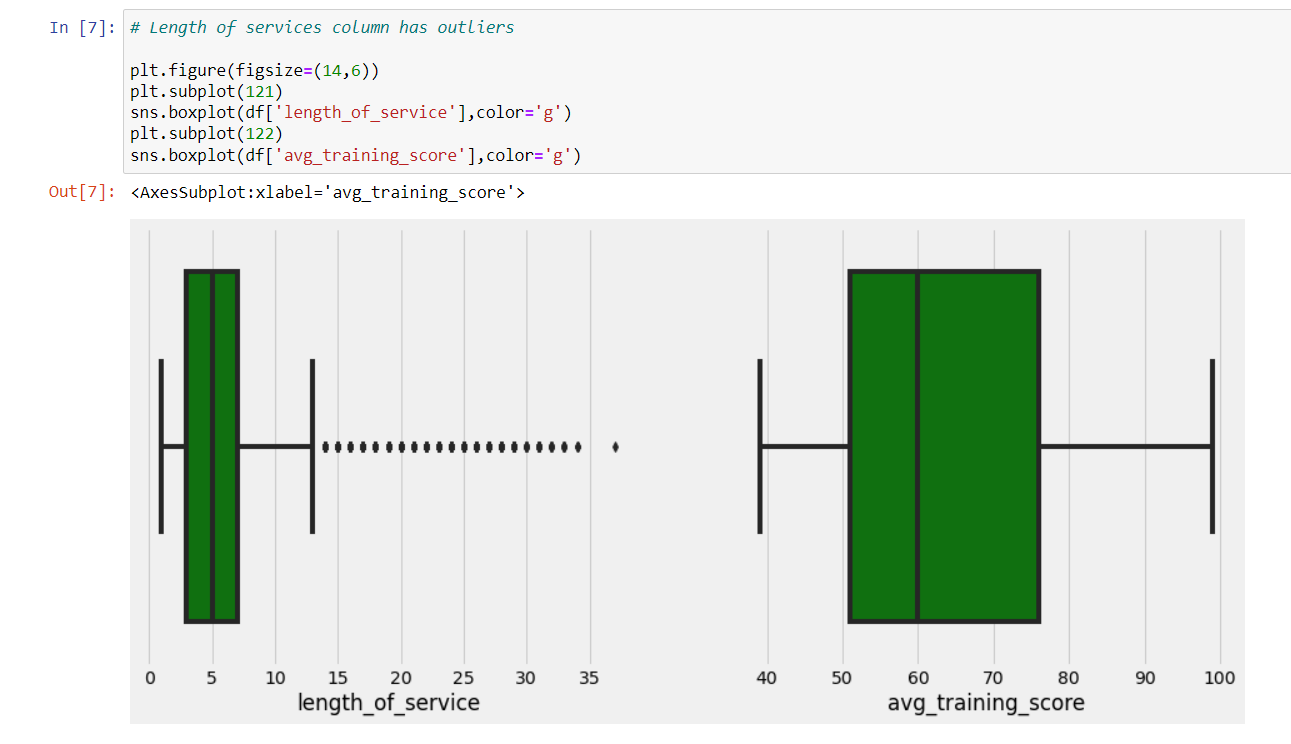
* Count plot and pie plot are used on the target variable. From the below image, we identified our data is imbalanced. 91% of the employees are not promoted. To get better model performance, imbalanced data should be converted to balanced data. Handling imbalanced data will be discussed on data pre processing.



* A pie plot is used on value counts() of the required features. From the below graph, we get a clear understanding that 97.68% of employees have not won any awards. Around 65% of employees have KPIs > 80%. More than 75% of employees have a previous year rating > 3.0. Instead of pie plot count plot can also be used.



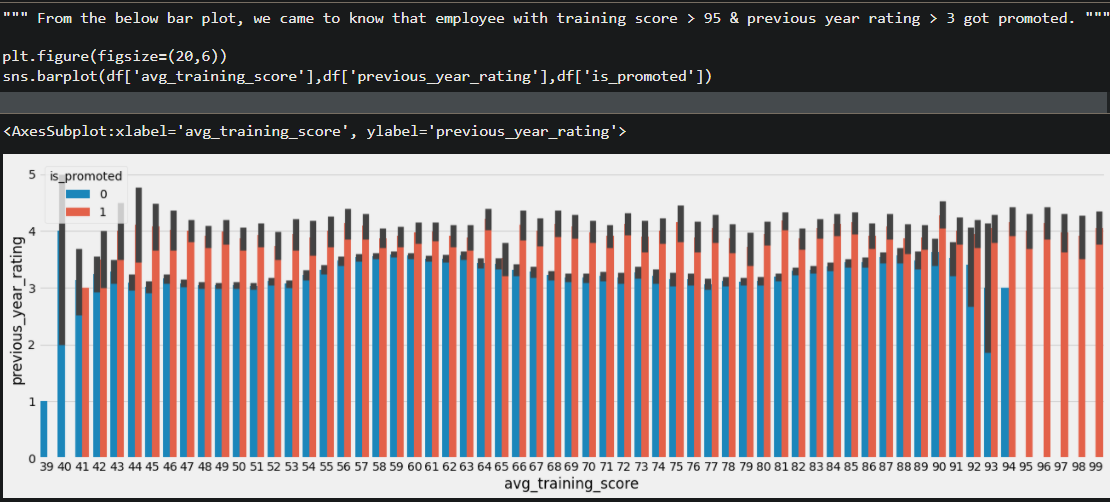
* Box plot is used on the length of service and average training score feature. Length of services feature has more outliers. The model should not be built without handling the outliers. Here, outliers are handled by the capping method. Capping will be discussed on data pre-processing.



MultiVariate Analysis:

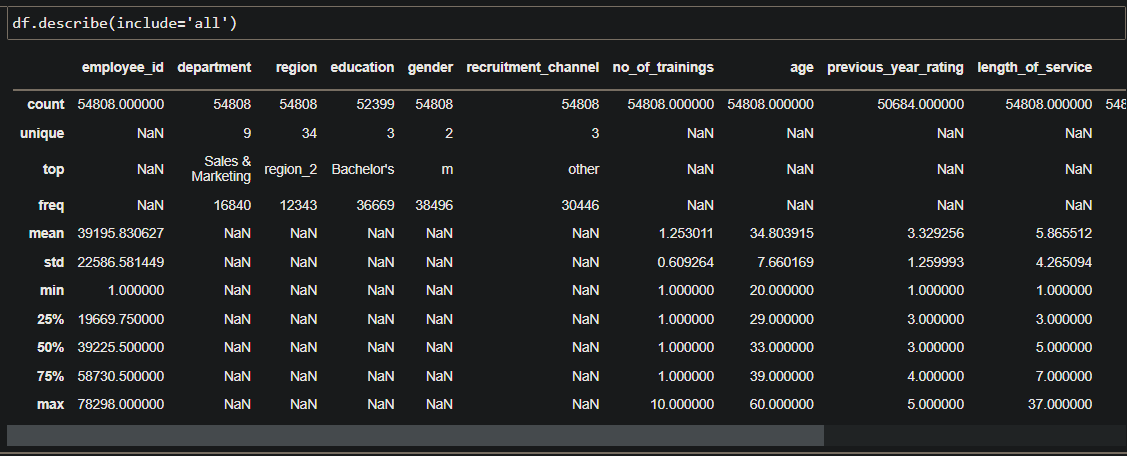
In simple words, multivariate analysis is to find the relation between multiple features. Here have used barplot from seaborn package.

* Three features are passed as parameters for barplot(). A clear pattern is understandable from the below plot. Employees with an average training score greater than 95 and a previous year rating greater than 3 got promotions (100%).



**Descriptive Analysis:**

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

****

**Handling Imbalanced Data:**

From the activity - univariate analysis we found our data is imbalanced. Now let’s split the dataset into x and y. Independent features are passed to x variable and dependent feature is passed to y variable. Then, to handle imbalanced data resampling are done with SMOTE.

* Import the SMOTE function from imblearn package.
* Create a variable and initialize smote() function. Now resampling is done with fit\_resample() function
* SMOTE : Refer this [link](https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/) to lean more about SMOTE

Application Building:

In this section, building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

* Building HTML Pages
* Building serverside script
* Run the application
* Output

**CHAPTER \_ 4**

**RESULT PERFORM ANALYSIS**

**RESULT:**

The application can be used to predict the employee promotion prediction By this application ,we can exclude human efforts and saves time. We have plotted the graphs in different formats like scatter plots no of employess got promoted using Linear Regression. Then we successfully done Data Visualization using seaborn.

**PERFORM ANALYSIS:**

Performance analysis is the technique of studying or comparing the performance of a specific situation in contrast to the aim and yet executed.

We distinguish three basic steps in the performance analysis process: data collection, data transformation, and data visualization. Data collection is the process by which we get data about program performance from an executing program.

I had used countplot technique to count the from data and plot the stripplot techniques to plot the graph in scatter graph values.

**CHAPTER \_5**

**CONCLUSION**

* The importance of predicting employee turnover in organizations and the application of machine learning in building turnover models was presented in this paper.
* The key challenge of noise in the data from HRIS that compromises the accuracy of these predictive models was also highlighted. Data from the HRIS of a global retailer was used to compare the XGBoost classifier against six other supervised classifiers that had been historically used to build turnover models.
* The results of this research demonstrate that the XGBoost classifier is a superior algorithm in terms of significantly higher accuracy, relatively low runtimes and efficient memory utilization for predicting turnover.
* The formulation of its regularization makes it a robust technique capable of handling the noise in the data from HRIS, as compared to the other classifiers, thus overcoming the key challenge in this domain.
* Because of these reasons it is recommended to use XGBoost for accurately predicting employee turnover, thus enabling organizations to take actions for retention or succession of employees.

**SOURCE CODE**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

import warnings

warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import cross\_val\_score

import pickle

from sklearn.metrics import classification\_report,confusion\_matrix

plt.style.use('fivethirtyeight')

pd.set\_option('display.max\_rows',None)

# Reading the csv and printing its shape

df = pd.read\_csv(r'C:\Users\HP\Desktop\Employee promotion prediction\Data set\emp\_promotion.csv')

print('Shape of train data {}'.format(df.shape))

df.head()

df['department'].unique()

plt.figure(figsize=(10,4))

plt.subplot(121)

sns.countplot(df['is\_promoted'])

plt.subplot(122)

df['is\_promoted'].value\_counts().plot(kind='pie',autopct = '%.2f%%',shadow=True)

plt.show()

""" From the below graph we get a clear understanding that 97.68% employees have not won any awards.

Around 65% of employees have KPIs > 80%. More than 75% of employees have previous year rating > 3.0 """

plt.figure(figsize=(16,10))

plt.subplot(231)

plt.axis('off')

plt.title('KPIs\_met >80%')

df['KPIs\_met >80%'].value\_counts().plot(kind='pie',shadow=True,autopct = '%.2f%%')

plt.subplot(232)

plt.axis('off')

plt.title('awards\_won?')

df['awards\_won?'].value\_counts().plot(kind='pie',shadow=True,autopct = '%.2f%%')

plt.subplot(233)

plt.axis('off')

plt.title('previous\_year\_rating')

df['previous\_year\_rating'].value\_counts().plot(kind='pie',shadow=True,autopct = '%.2f%%')

plt.show()

plt.figure(figsize=(14,6))

plt.subplot(121)

sns.boxplot(df['length\_of\_service'],color='g')

plt.subplot(122)

sns.boxplot(df['avg\_training\_score'],color='g')

plt.figure(figsize=(24,20))

plt.subplot(231)

sns.countplot(df['gender'],hue=df['is\_promoted'])

plt.title('Promotion based on gender')

plt.subplot(232)

plt.xticks(rotation=45)

sns.countplot(df['department'],hue=df['is\_promoted'])

plt.title('Promotion based on department')

plt.subplot(233)

sns.countplot(df['no\_of\_trainings'],hue=df['is\_promoted'])

plt.title('Promotion based on no\_of\_trainings')

plt.subplot(234)

sns.countplot(df['previous\_year\_rating'],hue=df['is\_promoted'])

plt.title('Promotion based on previous\_year\_rating')

plt.subplot(235)

sns.countplot(df['awards\_won?'],hue=df['is\_promoted'])

plt.title('Promotion based on awards\_won?')

plt.subplot(236)

sns.countplot(df['KPIs\_met >80%'],hue=df['is\_promoted'])

plt.title('Promotion based on department')

plt.show()

plt.figure(figsize=(20,6))

sns.barplot(df['avg\_training\_score'],df['previous\_year\_rating'],df['is\_promoted'])

df.describe(include='all')

df = df.drop(['employee\_id','gender','region','recruitment\_channel'],axis=1)

df.head()

df.isnull().sum()

print(df['education'].value\_counts())

df['education'] = df['education'].fillna(df['education'].mode()[0])

print(df['previous\_year\_rating'].value\_counts())

df['previous\_year\_rating'] = df['previous\_year\_rating'].fillna(df['previous\_year\_rating'].mode()[0])

df.info()

# Finding the employee who got promoted even in poor performance. It affect model performance.

negative=df[(df['KPIs\_met>80%']==0)&(df['awards\_won?']==0)& (df['previous\_year\_rating']==1.0) &

(df['is\_promoted']==1) & (df['avg\_training\_score']<60)]

Negative

df.drop(index=[31860,51374],inplace=True)

pd.crosstab([df['length\_of\_service']>upperBound],df['is\_promoted'])

df['length\_of\_service']=[upperBound if x>upperBound else x for x in df['length\_of\_service']]

df.select\_dtypes('object').head()

df['education']=df['education'].replace(("Below Secondary","Bachelor's","Master's & above"),(1,2,3))

lb = LabelEncoder()

df['department']=lb.fit\_transform(df['department'])

df.head()

7,3,1,35,5.0,8.0,1,0,49

x = df.drop('is\_promoted',axis=1)

y = df['is\_promoted']

print(x.shape)

print(y.shape)

from sklearn.metrics.pairwise import euclidean\_distances

from imblearn.over\_sampling import SMOTE

sm =SMOTE()

x\_resample, y\_resample = sm.fit\_resample(x,y)

plt.figure(figsize=(10,6))

plt.subplot(121)

sns.countplot(y)

plt.title('Before oversampling')

plt.subplot(122)

sns.countplot(y\_resample)

plt.title('After oversampling')

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_resample,y\_resample,test\_size=0.3,random\_state=10)

print('Shape of x\_train {}'.format(x\_train.shape))

print('Shape of y\_train {}'.format(y\_train.shape))

print('Shape of x\_test {}'.format(x\_test.shape))

print('Shape of y\_test {}'.format(y\_test.shape))

def decisionTree(x\_train, x\_test, y\_train, y\_test):

dt=DecisionTreeClassifier()

dt.fit(x\_train,y\_train)

yPred = dt.predict(x\_test)

print('\*\*\*DecisionTreeClassifier\*\*\*')

print('Confusion matrix')

print(confusion\_matrix(y\_test,yPred))

print('Classification report')

print(classification\_report(y\_test,yPred))

def randomForest(x\_train, x\_test, y\_train, y\_test):

rf = RandomForestClassifier()

rf.fit(x\_train,y\_train)

yPred = rf.predict(x\_test)

print('\*\*\*RandomForestClassifier\*\*\*')

print('Confusion matrix')

print(confusion\_matrix(y\_test,yPred))

print('Classification report')

print(classification\_report(y\_test,yPred))

def KNN(x\_train, x\_test, y\_train, y\_test):

knn = KNeighborsClassifier()

knn.fit(x\_train,y\_train)

yPred = knn.predict(x\_test)

print('\*\*\*KNeighborsClassifier\*\*\*')

print('Confusion matrix')

print(confusion\_matrix(y\_test,yPred))

print('Classification report')

print(classification\_report(y\_test,yPred))

def xgboost(x\_train, x\_test, y\_train, y\_test):

xg = GradientBoostingClassifier()

xg.fit(x\_train,y\_train)

yPred = xg.predict(x\_test)

print('\*\*\*GradientBoostingClassifier\*\*\*')

print('Confusion matrix')

print(confusion\_matrix(y\_test,yPred))

print('Classification report')

print(classification\_report(y\_test,yPred))

def compareModel(x\_train, x\_test, y\_train, y\_test):

decisionTree(x\_train, x\_test, y\_train, y\_test)

print('-'\*100)

randomForest(x\_train, x\_test, y\_train, y\_test)

print('-'\*100)

KNN(x\_train, x\_test, y\_train, y\_test)

print('-'\*100)

xgboost(x\_train, x\_test, y\_train, y\_test)

compareModel(x\_train, x\_test, y\_train, y\_test)

rf = RandomForestClassifier()

rf.fit(x\_train,y\_train)

yPred = rf.predict(x\_test)

cv = cross\_val\_score(rf,x\_resample,y\_resample,cv=5)

np.mean(cv)

pickle.dump(rf,open('model.pkl','wb'))

**REFERENCES**

* [1] S. Jahan, “Human Resources Information System (HRIS): A Theoretical Perspective”, Journal of Human Resource and Sustainability Studies, Vol.2 No.2, Article ID:46129, 2014.
* [2] M. Stoval and N. Bontis, “Voluntary turnover: Knowledge mana-gement – Friend or foe?”, Journal of Intellectual Capital, 3(3), 303-322, 2002.
* [3] J. L. Cotton and J. M. Tuttle, “Employee turnover: A meta-analysis and review with implications for research”, Academy of management Review, 11(1), 55-70, 1986.
* [4] L. M. Finkelstein, K. M. Ryanand E.B. King, “What do the young (old) people think of me? Content and accuracy of age-based metastereotypes”, European Journal of Work and Organizational Psychology, 22(6), 633-657, 2013.
* [5] B. Holtom, T. Mitchell, T. Lee, and M. Eberly, “Turnover and retention research: A glance at the past, a closer review of the present, and a venture into the future”, Academy of Management Annals, 2: 231-274, 2008