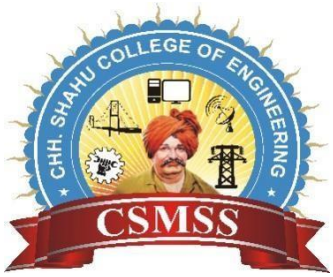
CSMSS
CHH. SHAHU COLLEGE OF ENGINEERING
KANCHANWADI, PAITHAN ROAD, CHH. SAMBHAJINAGAR – 431011



A REPORT OF PROJECT PHASE-II

ON

"HR ANALYTICS & PREDICTION OF EMPLOYEE ATTRITION WEB APPLICATION"

SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENT FOR

AWARD OF THE DEGREE

BACHELOR OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE & DATA SCIENCE ENGINEERING

Submitted By

Ms. Shital Madhukar Rananavare

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE ENGINEERING

CONTENTS

List of Figures

List of Abbreviations

Sr.No.	Symbol	Illustrations
1	HR	Human Resource
2	ROC	Receiver Operating Characteristics
3	CART	Classification And Regression Tree
4	sd	Standard Deviation
5	EDA	Exploratory Data Analysis
6	SVC	Support Vector Classifier

ABSTRACT

Employee Attrition analytics is specifically focused on identifying no. of employee's voluntary leaves, what might have prevented them from leaving and how we can use data to predict attrition risk.

Employee retention strategies are -integral to the success and well-being of a company, there are often many reasons why employees leave an organization or company, and in this case study, I explore some of the key drivers employed attrition. Employee attrition measures how many workers have left a company and is a common metric companies use to assess their performance while turnover rates vary from industry to industry. The Bureau of labor statistics reported that among voluntary separations the overall turnover rate was 25% in 2020.

In this project, I will explore IBM dataset on HR Analytics using Python programming language. The data consists of nearly 1,500 current and former employees with information related to their job satisfaction, work life balance, tenure experience, salary and demographic data Below is a brief overview and summary statistics of the data. Employee Attrition can help HR leader find the root cause of the problem and predict when employees will leave and why with this data, employers can make changes to improve attrition rate.

Chapter 1

INTRODUCTION

1.1 Introduction

Attrition is the departure of employees from the organization for any reason (voluntary or involuntary), including resignation, termination, death or retirement. Attrition rate is the rate at which employees leave an organization divided by the average number of employees at the organization over a given period of time. Employee attrition can happen for several reasons. These include unhappiness about employee benefits or the pay structure, a lack of employee development opportunities, and even poor conditions in the workplace.

Employee attrition means " employees leave an organization or company. There are many reason. say employees leave an organization like poor working conditions, retirement, unfair pay lack of professional growth and other personal reasons: compensation & job profile is highly responsible for employee attrition. In this project, I will explore IBM dataset on HR Analytics using python programming language. The data consists of nearly 1500 current and former employees which information related to the job satisfaction, salary, experience and demographic data, Employee Attrition can help. HR leaders find the root cause of the problem and predict when employees will leave & Why.

1.2 Objective of Project

The main objective of this research work is to develop a model that can help to predict whether an employee will leave the company or not. The essential idea is to measure the effectiveness of employee appraisal and satisfaction rates within the company, which can help to reduce the attrition rate of employees.

1.2 Problem Definition

The problem of HR analytics and prediction of employee attrition involves analyzing relevant data to gain insights into the factors that contribute to employee turnover within an organization. The goal is to develop predictive models that can accurately forecast the likelihood of employees leaving the organization, allowing HR departments to take proactive measures to retain valuable talent.

Finding the Root Cause of Attrition:-

Attrition, by definition, doesn't just consider employees who leave an organization voluntarily. It also includes employees who exit the business through other ways such as involuntary dismissal, retirement, change of role, promotions or advancements.

While even the best HR teams can't prevent some of these events nor would they want to in the case of career progressions a good talent management strategy should nevertheless plan for these movements to occur at certain points in time. Part of that planning entails forecasting, which, in turn, requires understanding of weak or vulnerable areas within the organization.

HR leaders should seek to uncover patterns or trends in departures across the organization to find the potential root cause of their attrition. For instance, bad hiring decisions that is, on boarding workers who are not a good fit for the role or the company often cause high turnover.

Attrition analytics can help HR leaders find the root cause of the problem and predict when employees will leave and why. With this data, employers can make changes to improve attrition rates.

By addressing the problem of HR analytics and prediction of employee attrition, organizations can proactively identify employees who are at risk of leaving and implement targeted retention strategies, thereby reducing turnover rates and retaining valuable talent.

Chapter 2

LITERATURE SURVEY

2.1 Related Work

This use case takes HR data and uses machine learning models to predict what employees will be more likely to leave given some attributes. Such model would help an organization predict employee attrition and define a strategy to reduce such costly problem.

The input dataset is an CSV file with information about 1470 employees. For each employee, in addition to whether the employee left or not (attrition), there are attributes / features such as age, employee role, daily rate, job satisfaction, years at the company, years in current role, etc.

2.2 Requirement Analysis

The steps we will go through are:

1. Data preprocessing:

Data processing is the entire process of data collection, filtering, sorting, calculation, and other logical operations.

2. Data analysis:

Data analysis is the process of inspecting, cleansing, transforming and modeling data with the goal of discovering useful information, informing conclusions and supporting decision-making.

3.Model training:

A machine learning training model is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from.

4.Model validation:

Model validation is the process that is carried out after **Model Training** where the trained model is evaluated with a testing data set. The testing data may or may not be a chunk of the same data set from which the training set is procured.

two types of Model Validation techniques are namely,

- In-sample validation - Testing data from the same dataset that is used to build the model.
- Out-of-sample validation - Testing data from a new dataset that isn't used to build the model.

5.Model predictions:

Predictive modeling is commonly used statistical technique to predict future behavior. Predictive modeling solutions are a form of data mining technology that works by analyzing historical and current data and generating a model to help predict future outcomes.

Most widely used prediction models are:

Linear Regression & Decision Tree

6.Visualization of results:

Predictive Data visualization helps to better understand and analyze complex data sets by presenting them in an easily understandable format. Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps,

data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

Chapter 3

SYSTEM ANALYSIS & DESIGN

In software design, as in mathematics, the representation schemes used are of fundamental importance. At least three levels of design notations exist: external design specifications, which describe the external characteristics of a software system; architectural design specifications, which describe the structure of the system; and detailed design specifications, which describe control flow, data representation, and other algorithmic details within the modules.

3.1 Phases of SDLC Model Used:

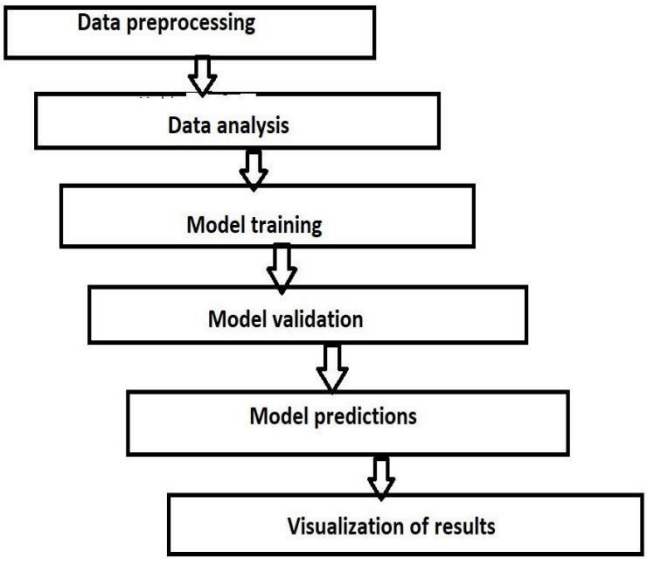


Fig. 3.1 Waterfall Model

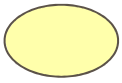
3.2 UML Diagrams

Actor:

A coherent set of roles that users of use cases play when interacting with the use `cases.

Use case:

A description of sequence of actions, including variants, that a system performs that yields an observable result of value of an actor.



UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

There are various kinds of methods in software design:

They are as follows:

1. Use case Diagram
2. Sequence Diagram
3. Collaboration Diagram
4. Activity Diagram
5. State chat Diagram
6. Data Flow Diagram

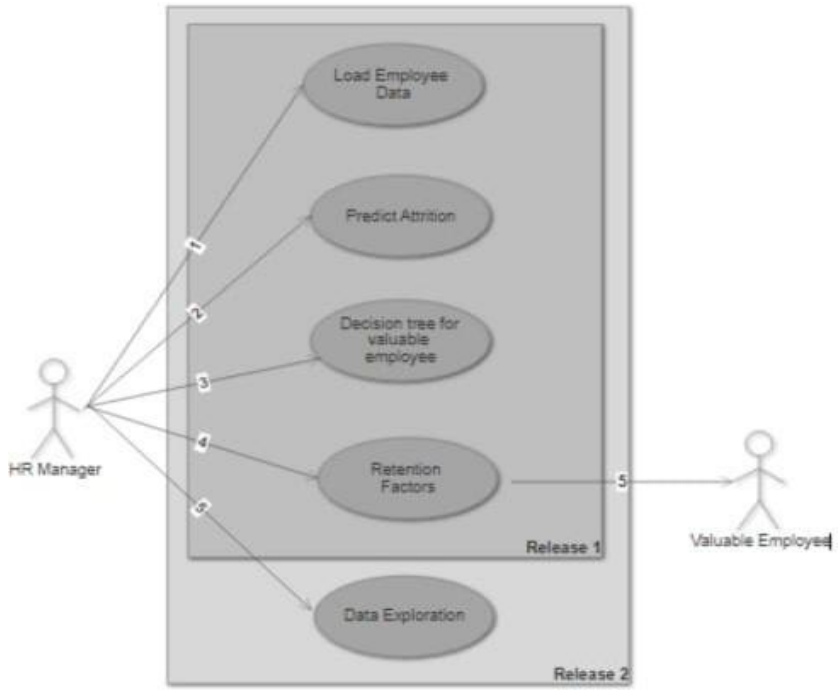
USE CASE DIAGRAM:

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what's called an actor.

Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can't do.
Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

- The purpose is to show the interactions between the use case and actor.
- To represent the system requirements from user's perspective.
- An actor could be the end-user of the system or an external system.

A Use case is a description of set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship. It is an association between the use cases and actors. An actor represents a real-world object. Primary Actor – Sender, Secondary Actor Receiver.



3.2 User Case Diagram

Chapter 4

SYSTEM IMPLEMENTATION

4.1 Implementation Details

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively. The system can be implemented only after thorough testing is done and if it is found to work according to the specification.

It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover and an evaluation of change over methods a part from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system. The more complex the system being implemented, the more involved will be the systems analysis and design effort required just for implementation.

The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

The Process for Classification :

1.Create an estimation sample and two validation samples by splitting the data into two groups.

2.Set up the dependent variable, employee attrition (as a categorical 0-1 variable)

3.Estimate the classification model using the estimation data, and interpret the results.

4.Assess the accuracy of classification in the first validation sample, possibly repeating steps 2-5 a few times changing the classifier in different ways to increase performance.

5.Finally, assess the accuracy of classification in the second validation sample. You should use and report all relevant performance measures and plots on this second validation sample only.

Step 1: Split the data

We split the data using a randomized splitting technique. The second validation data mimic out-of-sample data, and the performance on this validation set is a better approximation of the performance one should expect in practice from the selected classification method. The split used is 80% training data, 20% testing data- for example, when there is a lot of data, you may only keep a few hundreds of them for the validation and test sets.

Step 2: Set up the dependent variable

The data original file was not organized as a categorical Valuable, so we changed the column "Attrition" to 0 and 1 values.

In our data the number of 0/1's in our estimation sample is as follows:

Attrition	No Attrition
186	990

Step 3: Simple Analysis

We are running a simple table to visualize the Data of those values that are attired

	min	percent	median	mean	percent	max	std
Age	18	28.00	32.0	34.11	40.00	58	9.98
BusinessTravel	1	2.00	3.0	2.61	3.00	3	0.59
EducationField	1	2.00	3.0	3.33	4.00	6	1.41
EnvironmentSatisfaction	1	1.00	3.0	2.45	3.75	4	1.17
Gender	1	1.00	2.0	1.61	2.00	2	0.49
HourlyRate	31	50.00	65.5	65.29	83.75	100	20.38
JobInvolvement	1	2.00	3.0	2.50	3.00	4	0.77
JobLevel	1	1.00	1.0	1.63	2.00	5	0.95
JobRole	1	3.00	7.0	5.71	8.00	9	2.63
JobSatisfaction	1	1.00	3.0	2.48	3.00	4	1.10
MaritalStatus	1	2.00	2.5	2.35	3.00	3	0.72
MonthlyIncome	1081	2363.00	3090.5	4805.40	6098.75	19859	3681.86

Table 4.1.1 Simple Analysis

Step 4: Classification and Interpretation

Given our decisions, we decide to use a number of classification methods to develop a model that discriminates the different classes.In this paper we will consider: logistic regression and classification and regression trees (CART). Logistic Regression: Logistic Regression is a method similar to linear regression except that the dependent variable is discrete (e.g., 0 or 1). Linear logistic regression estimates the coefficients of a linear model using the selected independent variables while optimizing a classification criterion. For example, this is the logistic regression parameters for our data:

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.7	1.3	1.3	0.2
Age	0.0	0.0	-1.3	0.2
BusinessTravel	0.0	0.1	-0.1	0.9
EducationField	0.0	0.1	0.5	0.6
EnvironmentSatisfaction	-0.4	0.1	-4.3	0.0
Gender	0.2	0.2	1.1	0.3
HourlyRate	0.0	0.0	-0.5	0.6
JobInvolvement	-0.5	0.1	-4.1	0.0
JobLevel	-0.1	0.3	-0.2	0.8
JobRole	0.0	0.0	0.1	0.9
JobSatisfaction	-0.4	0.1	-4.3	0.0
MaritalStatus	0.5	0.2	3.0	0.0
MonthlyIncome	0.0	0.0	-0.8	0.4
MonthlyRate	0.0	0.0	0.3	0.7
NumCompaniesWorked	0.2	0.0	4.1	0.0
OverTime	1.8	0.2	9.2	0.0
PercentSalaryHike	0.0	0.0	-0.8	0.4
PerformanceRating	-0.2	0.4	-0.4	0.7
RelationshipSatisfaction	-0.3	0.1	-3.3	0.0
StockOptionLevel	-0.2	0.2	-1.3	0.2
TotalWorkingYears	-0.1	0.0	-2.4	0.0
TrainingTimesLastYear	-0.1	0.1	-1.7	0.1
WorkLifeBalance	-0.2	0.1	-1.6	0.1
YearsAtCompany	0.1	0.0	2.1	0.0
YearsInCurrentRole	-0.1	0.0	-3.0	0.0
YearsSinceLastPromotion	0.2	0.0	3.8	0.0
YearsWithCurrManager	-0.1	0.1	-2.2	0.0

Table 4.1.2 Classification and Interpretation

Step 5: Validation accuracy

Using the predicted class probabilities of the validation data, as outlined above, we can generate some measures of classification performance.

1. Confusion matrix:

The confusion matrix shows for each class the number (or percentage) of the data that are correctly classified for that class. For example, for the method above with the highest hit rate in the validation data (among logistic regression and the 2 CART models), and for probability threshold 45%, the confusion matrix for the validation data is:

	Predicted 1 (Attrition)	Predicted 0 (No Attrition)
Actual 1 (Attrition)	32.00	68.00
Actual 0 (No Attrition)	6.56	93.44

2. ROC curve:

The ROC curves for the validation data for the logistic regression as well as both the CARTs above are as follows:

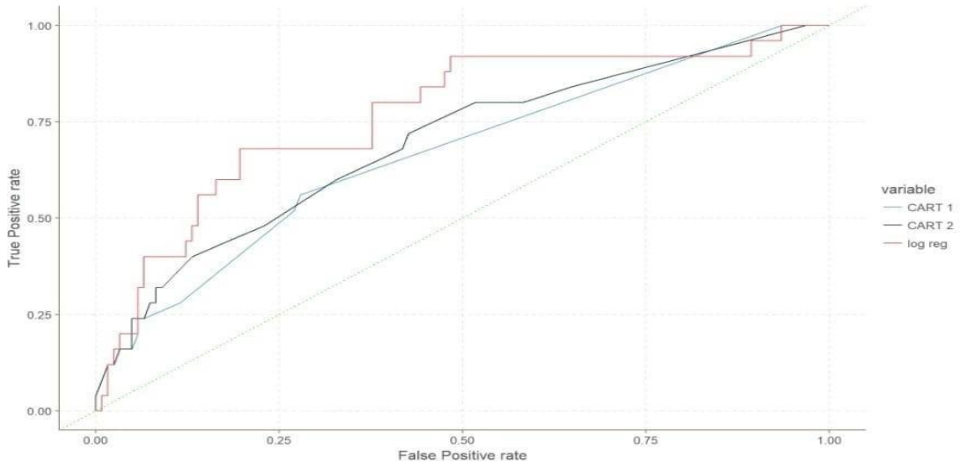


Fig-4.1.3 ROC Curve

3. Gains chart:

Gain Chart is an evaluation curve that assesses the performance of your model. The Gain chart plots the total positive rate in percent versus the percent of total counts. The gains charts for the validation data for our three classifiers are the following

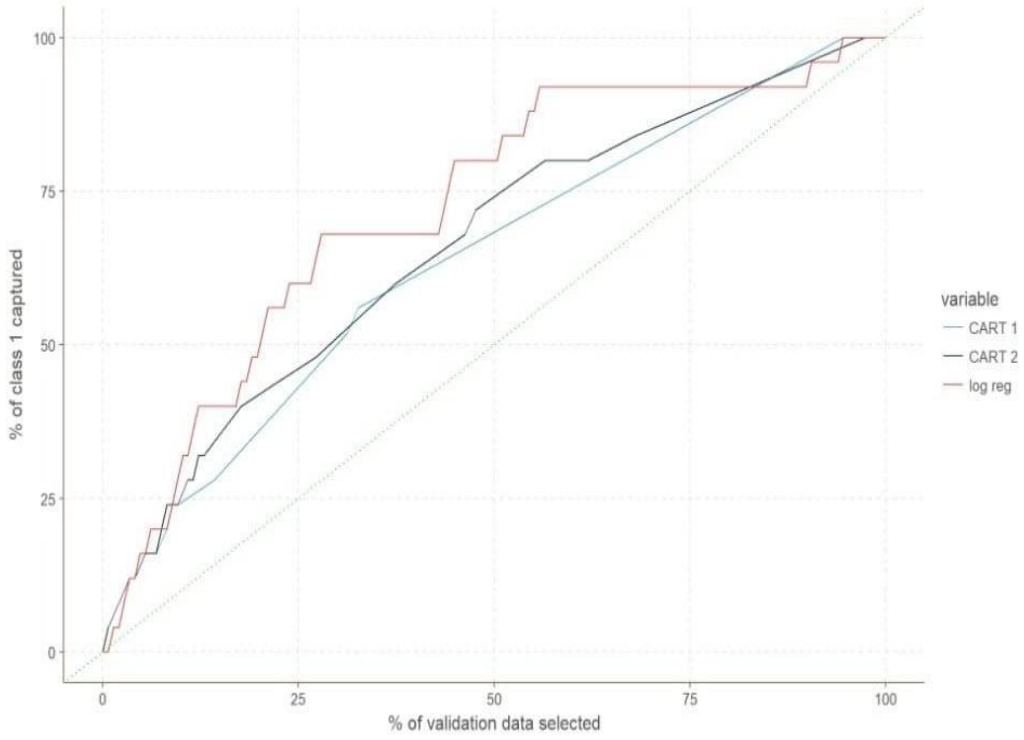


Fig-4.1.4 Gain Chart

Step 6. Test Accuracy

Having iterated steps 2-5 until we are satisfied with the performance of our selected model on the validation data, in this step the performance analysis outlined in step 5 needs to be done with the test sample.

Let's see in our case how the hit ratio, confusion matrix, ROC curve, gains chart, and profit curve look like for our test data. For the hit ratio and the confusion matrix we use 45% as the probability threshold for classification.

	Predicted 1 (Attrition)	Predicted 0 (No Attrition)
Actual 1 (Attrition)	42.31	57.69
Actual 0 (No Attrition)	0.83	99.17

ROC curves for the test data:

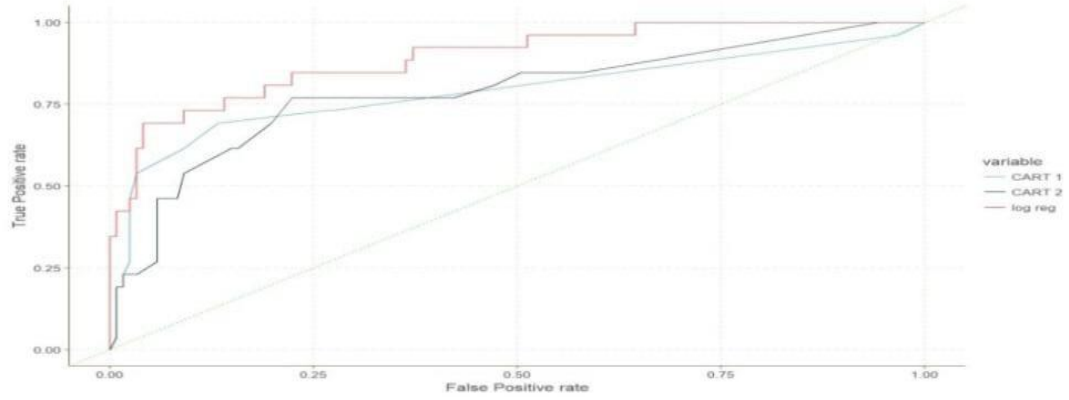


Fig-4.1.5 Test Accurancy

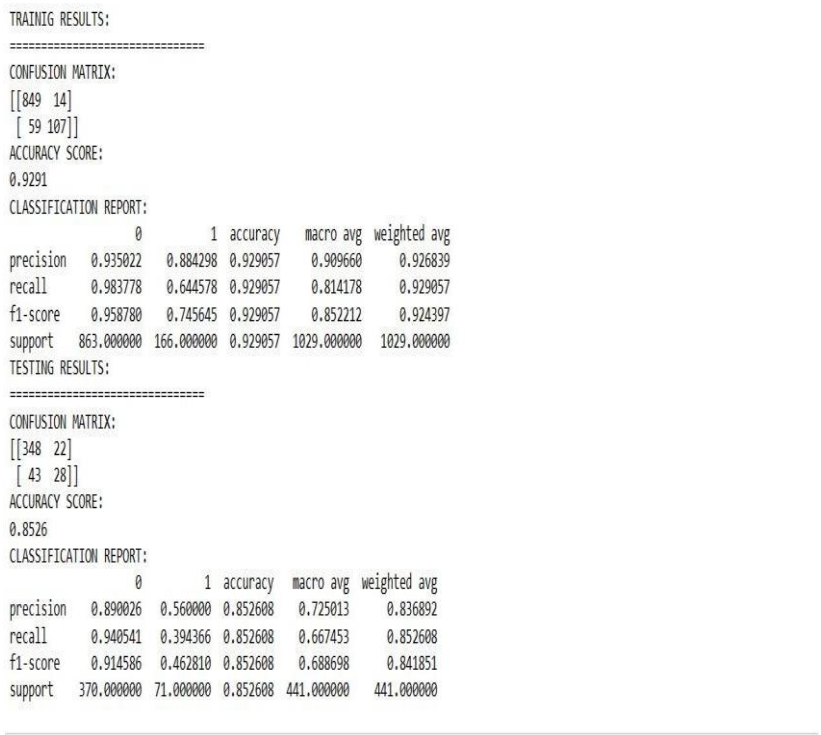
Step 7. Data Analysis

After we ran the model multiple times and iterate to find the best value, we came with some conclusions:

Model is biased towards predicting non attrition.There is a tension between probability threshold and the number of employees who are accurately predicted as potential churners. A high probability threshold would end in a high number of errors. The business relevance is predict attrition well, rather than non attrition hence a lower probability threshold is chosen.

The confusion matrix shows that of all the people who are going to leave the company, our algorithm identifies about 42% of them accurately. While not ideal, this is a huge improvement on random sampling where we could have predicted only about 16% (the actual attrition rate). On the other hand, there is a cost of wrongly identifying attrition of non-leaving employees resulting in inefficiencies in resource allocation.

Log Regression is the best model, as it always predict a higher area under the curve and a better confusion matrix



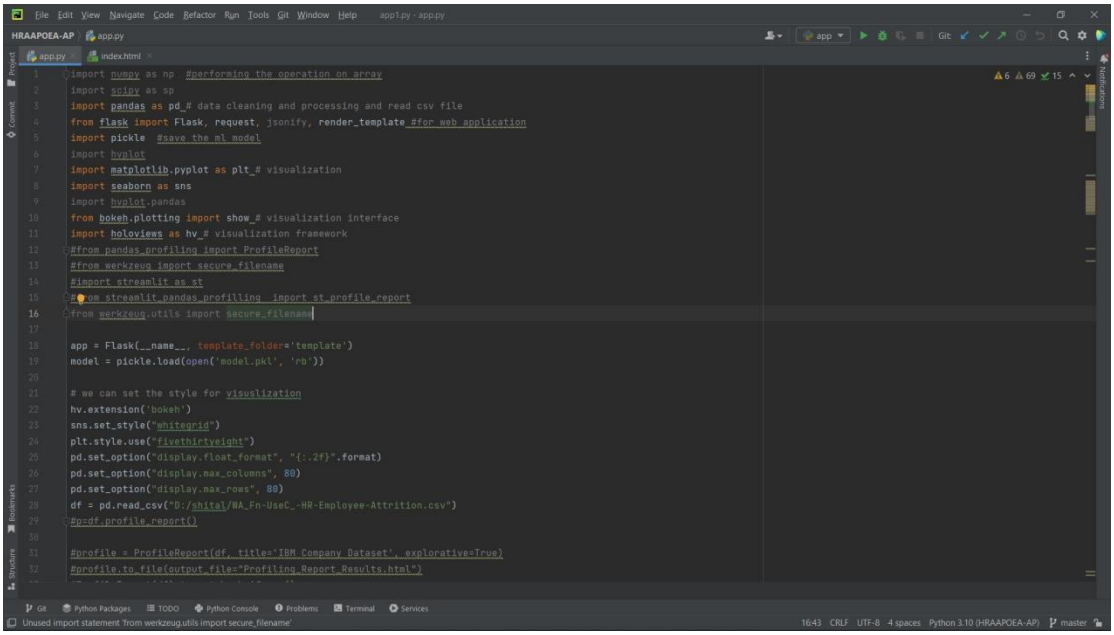
```
TRAINIG RESULTS:
==============================
CONFUSION MATRIX:
[[849  14]
 [ 59 107]]
ACCURACY SCORE:
0.9291
CLASSIFICATION REPORT:
                  0         1  accuracy   macro avg  weighted avg
precision  0.935022  0.884298  0.929057    0.909660      0.926839
recall     0.983778  0.644578  0.929057    0.814178      0.929057
f1-score   0.958780  0.745645  0.929057    0.852212      0.924397
support  863.000000 166.000000  0.929057 1029.000000   1029.000000
TESTING RESULTS:
==============================
CONFUSION MATRIX:
[[348  22]
 [ 43  28]]
ACCURACY SCORE:
0.8526
CLASSIFICATION REPORT:
                  0        1  accuracy   macro avg  weighted avg
precision  0.890026  0.560000  0.852608    0.725013      0.836892
recall     0.940541  0.394366  0.852608    0.667453      0.852608
f1-score   0.914586  0.462810  0.852608    0.688698      0.841851
support  370.000000 71.000000  0.852608  441.000000    441.000000
```

Fig 4.1.6 Evaluation of result

Sep 8. Creating API and GUI for the model

Flask API: Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts (POCCO). It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework .[4]

We will be using flask to connect our model to the GUI that we will create using HTML.



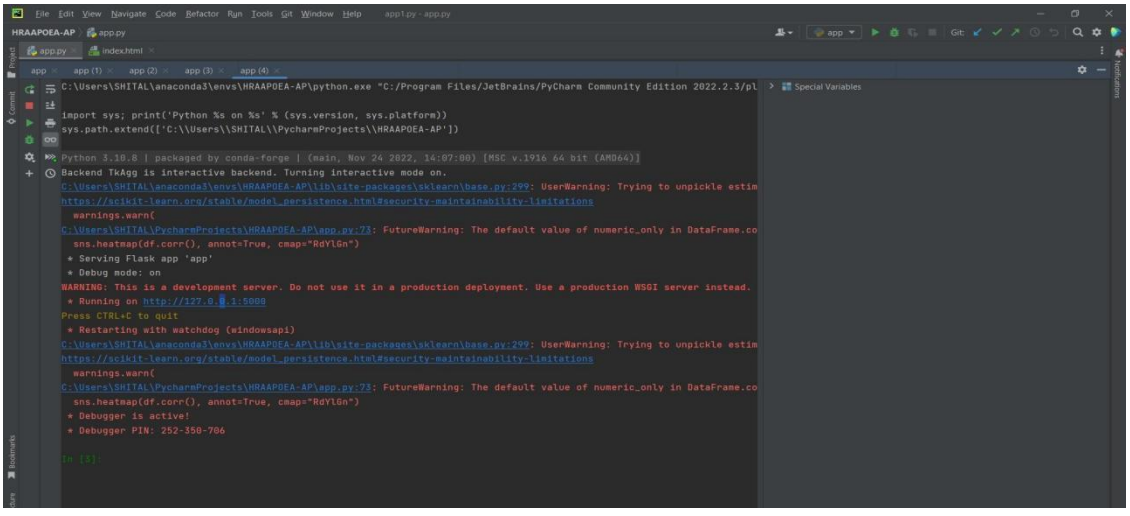


Fig-4.1.7 Flask Building in Pycharm

4.2Snapshots

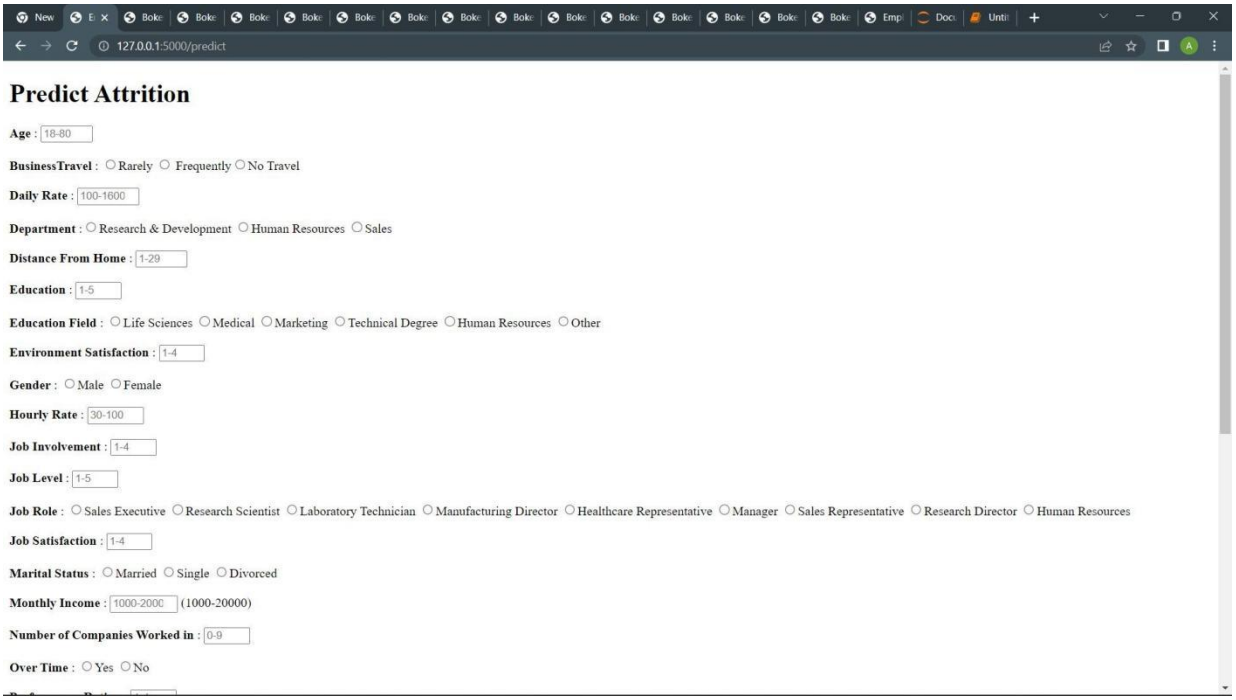


Fig-4.2.1 Flask Result

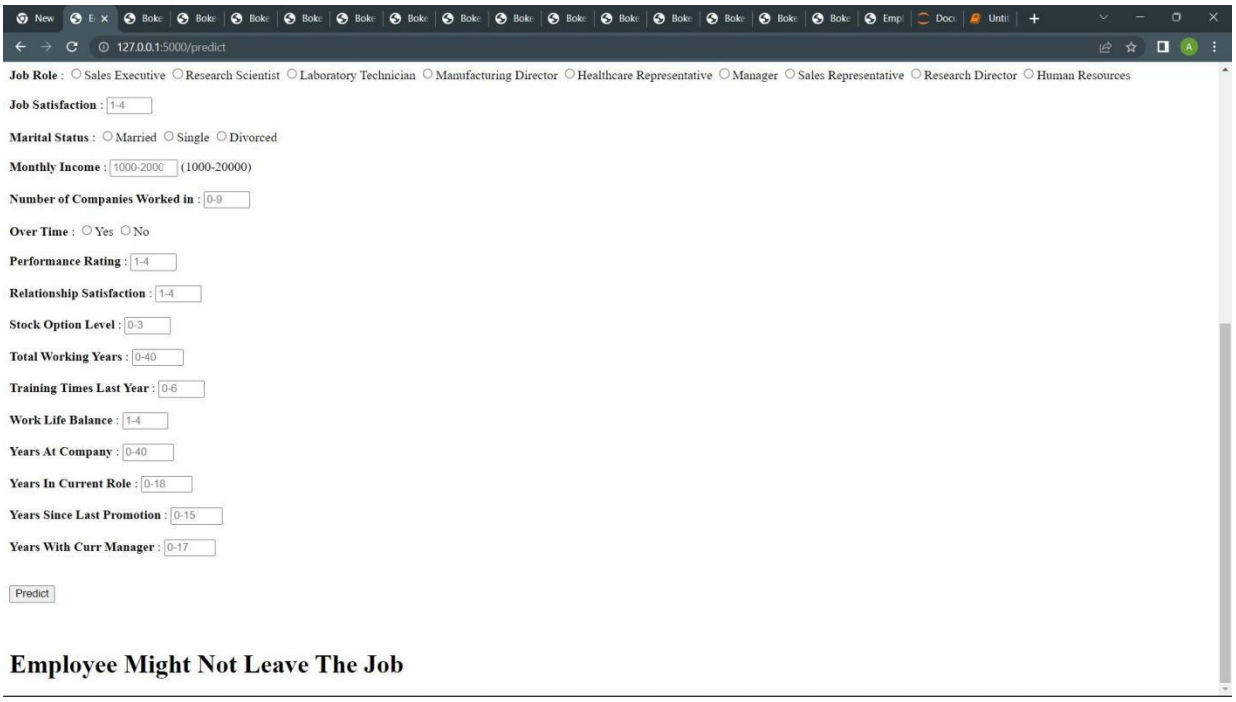


Fig-4.2.2 Prediction Of Employee Attrition

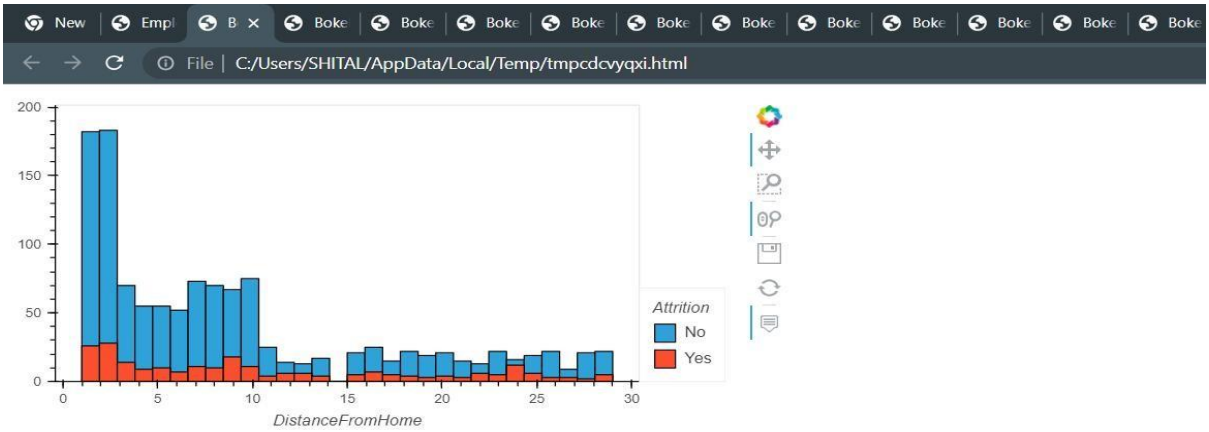


Fig-4.2.3 Visualization using Bokeh interface

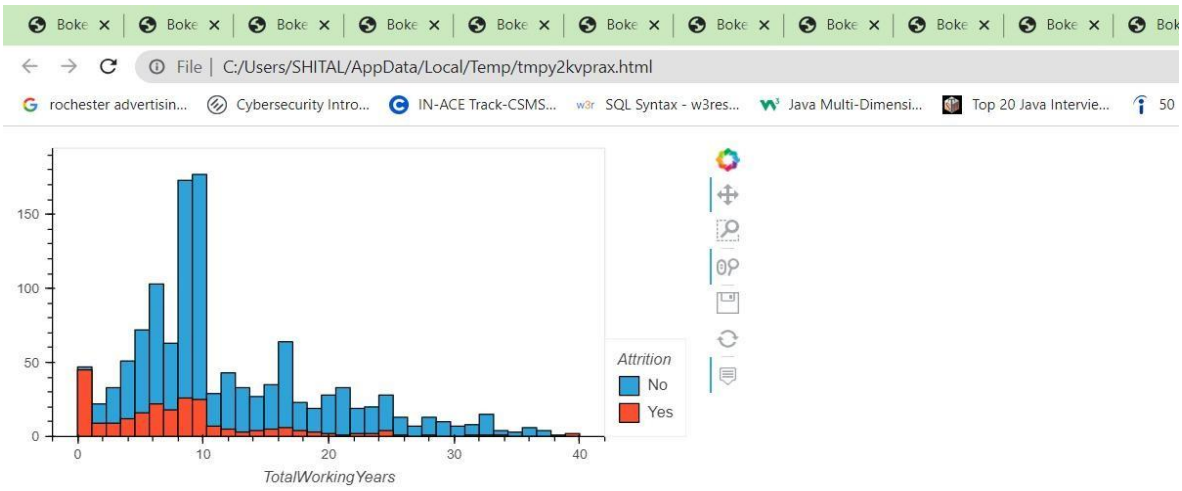


Fig-4.2.4 Visualization using Bokeh interface

Fig-4.2.4 Visualization using Bokeh interface

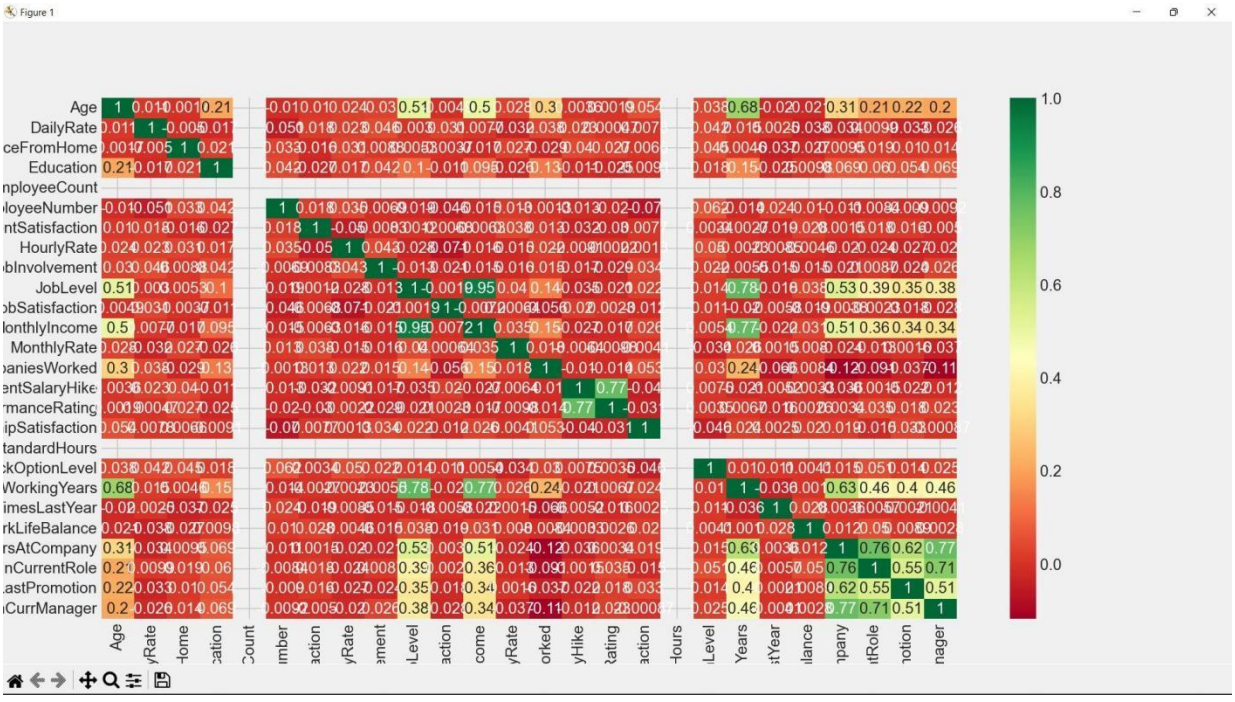


Fig-4.2.4 Heat Map

4.3 Coding

```
!pip install hvplot
import hvplot#interactive visualization library for web
import pandas as pd#data cleaning and analysis
import numpy as np#array function
import matplotlib.pyplot as plt#graphical plotting library
import seaborn as sns# visualization
import hvplot.pandas

%matplotlib inline#for backkend function
sns.set_style("whitegrid")#background
plt.style.use("fivethirtyeight")#style

pd.set_option("display.float_format", "{:.2f}".format)
pd.set_option("display.max_columns", 80)
pd.set_option("display.max_rows", 80)

df = pd.read_csv("D:/shital/WA_Fn-UseC_-HR-Employee-Attrition.csv")
df.head()

df.info()

df.describe()

for column in df.columns:
    print(f"{column}: Number of unique values {df[column].nunique()}")
    print("============================================================")

df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis="columns",
inplace=True)

object_col = []
for column in df.columns:
    if df[column].dtype == object and len(df[column].unique()) <= 30:
        object_col.append(column)
        print(f"{column} : {df[column].unique()}")
```

```
        print(df[column].value_counts())
        print("====================================")
object_col.remove('Attrition')

len(object_col)

from sklearn.preprocessing import LabelEncoder#converts the variables into the categorical

label = LabelEncoder()
df["Attrition"] = label.fit_transform(df.Attrition)

disc_col = []#this loop is used to find the unique values after performing labelencoder
for column in df.columns:
    if df[column].dtypes != object and df[column].nunique() < 30:
        print(f"{column} : {df[column].unique()}")
        disc_col.append(column)
        print("====================================")
disc_col.remove('Attrition')

cont_col = []
for column in df.columns:
    if df[column].dtypes != object and df[column].nunique() > 30:
        print(f"{column} : Minimum: {df[column].min()}, Maximum: {df[column].max()}")
        cont_col.append(column)
        print("====================================")

df.hvplot.hist(y='DistanceFromHome', by='Attrition', subplots=False, width=600, height=300,
bins=30)#hvplot is uesd to
#represent interactive visualization of web on bokeh and holoviews platform

df.hvplot.hist(y='Education', by='Attrition', subplots=False, width=600, height=300)
df.hvplot.hist(y='RelationshipSatisfaction', by='Attrition', subplots=False, width=600, height=300)
df.hvplot.hist(y='EnvironmentSatisfaction', by='Attrition', subplots=False, width=600,
height=300)
df.hvplot.hist(y='JobLevel', by='Attrition', subplots=False, width=600, height=300)
df.hvplot.hist(y='JobSatisfaction', by='Attrition', subplots=False, width=600, height=300)
df.hvplot.hist(y='NumCompaniesWorked', by='Attrition', subplots=False, width=600, height=300)
df.hvplot.hist(y='PercentSalaryHike', by='Attrition', subplots=False, width=600, height=300)
```

```
df.hvplot.hist(y='StockOptionLevel', by='Attrition', subplots=False, width=600, height=300)
df.hvplot.hist(y='TrainingTimesLastYear', by='Attrition', subplots=False, width=600, height=300)
df.hvplot.hist(y='Age', by='Attrition', subplots=False, width=600, height=300, bins=35)
df.hvplot.hist(y='MonthlyIncome', by='Attrition', subplots=False, width=600, height=300,
bins=50)
df.hvplot.hist(y='YearsAtCompany', by='Attrition', subplots=False, width=600, height=300,
bins=35)
df.hvplot.hist(y='TotalWorkingYears', by='Attrition', subplots=False, width=600, height=300,
bins=35)

plt.figure(figsize=(30, 30))#used to check the correlations
sns.heatmap(df.corr(), annot=True, cmap="RdYlGn", annot_kws={"size":15})

col = df.corr().nlargest(20, "Attrition").Attrition.index#heat map
plt.figure(figsize=(15, 15))
sns.heatmap(df[col].corr(), annot=True, cmap="RdYlGn", annot_kws={"size":10})

df.drop('Attrition', axis=1).corrwith(df.Attrition).hvplot.barh()#bar plot

# Transform categorical data into dummies
dummy_col = [column for column in df.drop('Attrition', axis=1).columns if df[column].nunique()
< 20]
data = pd.get_dummies(df, columns=dummy_col, drop_first=True, dtype='uint8')
data.info()

print(data.shape)

# Remove duplicate Features
data = data.T.drop_duplicates()
data = data.T

# Remove Duplicate Rows
data.drop_duplicates(inplace=True)

print(data.shape)

from sklearn.model_selection import train_test_split, GridSearchCV# cross validiation method
from sklearn.preprocessing import StandardScaler#used to standardization
```

```
X = data.drop('Attrition', axis=1)
y = data.Attrition
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42,
                                                    stratify=y)
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
X_std = scaler.transform(X)

from sklearn.metrics import precision_recall_curve, roc_curve

def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g--", label="Recall")
    plt.xlabel("Threshold")
    plt.legend(loc="upper left")
    plt.title("Precision/Recall Tradeoff")
def plot_roc_curve(fpr, tpr, label=None):
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], "k--")
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')

precisions, recalls, thresholds = precision_recall_curve(y_test, lr_clf.predict(X_test_std))
plt.figure(figsize=(14, 25))
plt.subplot(4, 2, 1)
plot_precision_recall_vs_threshold(precisions, recalls, thresholds)
plt.subplot(4, 2, 2)
plt.plot(precisions, recalls)
plt.xlabel("Precision")
plt.ylabel("Recall")
plt.title("PR Curve: precisions/recalls tradeoff");
plt.subplot(4, 2, 3)
fpr, tpr, thresholds = roc_curve(y_test, lr_clf.predict(X_test_std))
plot_roc_curve(fpr, tpr)
```

HTML Code:

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Employee Attrition Prediction</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>

</head>

<body>
 <div class="login">

   <h1>Predict Attrition</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}" method="POST" autocomplete="on">

        <strong>Age</strong> : <input type="number" name="Age" placeholder="18-80" required min="18" max="80"><br><br>

        <strong>BusinessTravel</strong> : <input type="radio" name="BusinessTravel" id="Rarely" value="Rarely">Rarely <input type="radio" name="BusinessTravel" id="Frequently" value="Frequently"> Frequently<input type="radio" name="BusinessTravel" id="No Travel" value="No Travel">No Travel<br><br>

        <strong>Daily Rate</strong> : <input type="number" name="Daily Rate" placeholder="100-1600" required min="100" max="1600" size="30"><br><br>

        <strong>Department</strong> :<input type="radio" name="Department" id="Research & Development" value="Research & Development">Research & Development <input type="radio" name="Department" id="Human Resources" value="Human Resources">Human Resources  <input type="radio" name="Department" id="Sales"  value="Sales">Sales<br><br>
```

```
        <strong>Distance From Home</strong> : <input type="number" name="Distance From Home" placeholder="1-29" required min="1" max="29"><br><br>

        <strong>Education</strong> : <input type="number" name="Education" placeholder="1-5" required min="1" max="5"><br><br>

        <strong>Education Field</strong> : <input type="radio" name="Education Field" id="Life Sciences" value="Life Sciences">Life Sciences <input type="radio" name="Education Field" id="Medical" value="Medical">Medical <input type="radio" name="Education Field" id="Marketing" value="Marketing">Marketing <input type="radio" name="Education Field" id="Technical Degree" value="Technical Degree">Technical Degree <input type="radio" name="Education Field" id="Human Resources"  value="Human Resources">Human Resources <input type="radio" name="Education Field" id="Other"  value="Other">Other  <br><br>

        <strong>Environment Satisfaction</strong> : <input type="number" name="Environment Satisfaction" placeholder="1-4" required min="1" max="4"><br><br>

        <strong>Gender</strong> : <input type="radio" name="Gender" id="Male" value="Male">Male <input type="radio" name="Gender" id="Female" value="Female">Female<br><br>

        <strong>Hourly Rate</strong> : <input type="number" name="Hourly Rate" placeholder="30-100" required min="30" max="100"><br><br>

        <strong>Job Involvement</strong> : <input type="number" name="Job Involvement" placeholder="1-4" required min="1" max="4"><br><br>

        <strong>Job Level</strong> : <input type="number" name="Job Level" placeholder="1-5" required min="1" max="5"><br><br>

        <strong>Job Role</strong> : <input type="radio" name="Job Role" id="Sales Executive" value="Sales Executives">Sales Executive <input type="radio" name="Job Role" id="Research Scientist" value="Research Scientist">Research Scientist  <input type="radio" name="Job Role" id="Laboratory Technician" value="Laboratory Technician">Laboratory Technician <input type="radio" name="Job Role" id="Manufacturing Director" value="Manufacturing Director">Manufacturing Director <input type="radio" name="Job Role" id="Healthcare Representative" value="Healthcare Representative">Healthcare Representative <input type="radio" name="Job Role" id="Manager" value="Manager">Manager  <input type="radio"
```

```
name="Job Role" id="Sales Representative" value="Sales Representative">Sales Representative
<input type="radio" name="Job Role" id="Research Director" value="Research Director">Research
Director <input type="radio" name="Job Role" id="Human Resource" value="Human
Resources">Human Resources  <br><br>

      <strong>Job Satisfaction</strong> : <input type="number" name="Job Satisfaction"
placeholder="1-4" required min="1" max="4"><br><br>

      <strong>Marital Status</strong> : <input type="radio" name="Marital Status"
id="Married" value="Married">Married  <input type="radio" name="Marital Status" id="Single"
value="Single">Single <input type="radio" name="Marital Status" id="Divorced"
value="Divorced">Divorced<br><br>

      <strong>Monthly Income</strong> : <input type="number" name="Monthly Income"
placeholder="1000-20000" required min="1000" max="20000" size="30"> (1000-
20000)<br><br>

      <strong>Number of Companies Worked in</strong> : <input type="number"
name="Number of Companies Worked in" placeholder="0-9" required min="1"
max="9"><br><br>

      <strong>Over Time</strong> : <input type="radio" name="Over Time" id="Yes"
value="Yes">Yes  <input type="radio" name="Over Time" id="No" value="No">No<br><br>

      <strong>Performance Rating</strong> : <input type="number" name="Performance
Rating" placeholder="1-4" required min="1" max="4"><br><br>

      <strong>Relationship Satisfaction</strong> : <input type="number" name="Relationship
Satisfaction" placeholder="1-4" required min="1" max="4"><br><br>

      <strong>Stock Option Level</strong> : <input type="number" name="Stock Option Level"
placeholder="0-3" required min="0" max="3"><br><br>

      <strong>Total Working Years</strong> : <input type="number" name="Total Working
Years" placeholder="0-40" required min="0" max="40"><br><br>

      <strong>Training Times Last Year</strong> : <input type="number" name="Training
Times Last Year" placeholder="0-6" required min="0" max="6"><br><br>
```

```
        <strong>Work Life Balance</strong> : <input type="number" name="Work Life Balance" placeholder="1-4" required min="1" max="4"><br><br>

        <strong>Years At Company</strong> : <input type="number" name="Years At Company" placeholder="0-40" required min="0" max="40"><br><br>

        <strong>Years In Current Role</strong> : <input type="number" name="Years In Current Role" placeholder="0-18" required min="0" max="18"><br><br>

        <strong>Years Since Last Promotion</strong> : <input type="number" name="Years Since Last Promotion" placeholder="0-15" required min="0" max="15"><br><br>

        <strong>Years With Curr Manager</strong> : <input type="number" name="Years With Curr Manager" placeholder="0-17" required min="0" max="17"><br><br>

        <br><button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
     <h1>{{ prediction_text }}</h1>

 </div>



</body>
</html>
```

FLASK Code:

```
import numpy as np  #performing the operation on array
import scipy as sp
import pandas as pd # data cleaning and processing and read csv file
from flask import Flask, request, jsonify, render_template #for web application
import pickle  #save the ml model
import hvplot
import matplotlib.pyplot as plt # visualization
import seaborn as sns
import hvplot.pandas
from bokeh.plotting import show # visualization interface
import holoviews as hv # visualization framework
#from pandas_profiling import ProfileReport
#from werkzeug import secure_filename
#import streamlit as st
#from streamlit_pandas_profilling  import st_profile_report
from werkzeug.utils import secure_filename

app = Flask(__name__, template_folder='template')
model = pickle.load(open('model.pkl', 'rb'))

# we can set the style for visuslization
hv.extension('bokeh')
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
pd.set_option("display.float_format", "{:.2f}".format)
pd.set_option("display.max_columns", 80)
pd.set_option("display.max_rows", 80)
df = pd.read_csv("D:/shital/WA_Fn-UseC_-HR-Employee-Attrition.csv")
#p=df.profile_report()

#profile = ProfileReport(df, title='IBM Company Dataset', explorative=True)
#profile.to_file(output_file="Profiling_Report_Results.html")
#ProfileReport(df).to_notebook_iframe()
#df=df.drop(columns=['Unnamed:27','Unnamed_27'])
#webbrowser.open("Profiling_Report_Results.html", new=2)
#from pandas_profiling import ProfileReport
#profile = ProfileReport(df, title='IBM Company Dataset', explorative=True)
```

```
#html_report = profile.to_html()
#st.title("pandas profiling in streamlit!")
#st.write(df)
#st_profile_report(p)


plot=df.hvplot.hist(y='DistanceFromHome', by='Attrition', subplots=False, width=600,
height=300, bins=30)
plot1=df.hvplot.hist(y='Education', by='Attrition', subplots=False, width=600, height=300)
plot2=df.hvplot.hist(y='RelationshipSatisfaction', by='Attrition', subplots=False, width=600,
height=300)
plot3=df.hvplot.hist(y='EnvironmentSatisfaction', by='Attrition', subplots=False, width=600,
height=300)
plot4=df.hvplot.hist(y='JobLevel', by='Attrition', subplots=False, width=600, height=300)
plot5=df.hvplot.hist(y='JobSatisfaction', by='Attrition', subplots=False, width=600, height=300)
plot6=df.hvplot.hist(y='NumCompaniesWorked', by='Attrition', subplots=False, width=600,
height=300)
plot7=df.hvplot.hist(y='PercentSalaryHike', by='Attrition', subplots=False, width=600,
height=300)
plot8=df.hvplot.hist(y='StockOptionLevel', by='Attrition', subplots=False, width=600, height=300)
plot9=df.hvplot.hist(y='TrainingTimesLastYear', by='Attrition', subplots=False, width=600,
height=300)
plot10=df.hvplot.hist(y='Age', by='Attrition', subplots=False, width=600, height=300, bins=35)
plot11=df.hvplot.hist(y='MonthlyIncome', by='Attrition', subplots=False, width=600, height=300,
bins=50)
plot12=df.hvplot.hist(y='YearsAtCompany', by='Attrition', subplots=False, width=600,
height=300, bins=35)
plot13=df.hvplot.hist(y='TotalWorkingYears', by='Attrition', subplots=False, width=600,
height=300, bins=35)
show(hv.render(plot))
show(hv.render(plot1))
show(hv.render(plot2))
show(hv.render(plot3))
show(hv.render(plot4))
show(hv.render(plot5))
show(hv.render(plot6))
show(hv.render(plot7))
show(hv.render(plot8))
```

```
show(hv.render(plot9))
show(hv.render(plot10))
show(hv.render(plot11))
show(hv.render(plot12))
show(hv.render(plot13))
plt.figure(figsize=(30, 30))
sns.heatmap(df.corr(), annot=True, cmap="RdYlGn")
plt.show()


@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST', 'GET'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    Age = request.form.get("Age")
    BusinessTravel = request.form['BusinessTravel']
    DailyRate = request.form.get('Daily Rate')
    Department = request.form['Department']
    DistanceFromHome = request.form.get("Distance From Home")
    Education = request.form.get("Education")
    EducationField = request.form['Education Field']
    EnvironmentSatisfaction = request.form.get("Environment Satisfaction")
    Gender = request.form['Gender']
    HourlyRate = request.form.get("Hourly Rate")
    JobInvolvement = request.form.get("Environment Satisfaction")
    JobLevel = request.form.get("Job Level")
    JobRole = request.form['Job Role']
    JobSatisfaction = request.form.get("Job Satisfaction")
    MaritalStatus = request.form['Marital Status']
    MonthlyIncome = request.form.get("Monthly Income")
    NumCompaniesWorked = request.form.get("Number of Companies Worked in")
    OverTime = request.form['Over Time']
    PerformanceRating = request.form.get("Performance Rating")
```

```
RelationshipSatisfaction = request.form.get("Relationship Satisfaction")
StockOptionLevel = request.form.get("Stock Option Level")
TotalWorkingYears = request.form.get("Total Working Years")
TrainingTimesLastYear = request.form.get("Training Times Last Year")
WorkLifeBalance = request.form.get("Work Life Balance")
YearsAtCompany = request.form.get("Years At Company")
YearsInCurrentRole = request.form.get("Years In Current Role")
YearsSinceLastPromotion = request.form.get("Years Since Last Promotion")
YearsWithCurrManager = request.form.get("Years With Curr Manager")

dict = {
    'Age': int(Age),
    'BusinessTravel': str(BusinessTravel),
    'DailyRate': int(DailyRate),
    'Department': Department,
    'DistanceFromHome': int(DistanceFromHome),
    'Education': Education,
    'EducationField': str(EducationField),
    'EnvironmentSatisfaction': int(EnvironmentSatisfaction),
    'Gender': str(Gender),
    'HourlyRate': int(HourlyRate),
    'JobInvolvement': int(JobInvolvement),
    'JobLevel': int(JobLevel),
    'JobRole': JobRole,
    'JobSatisfaction': int(JobSatisfaction),
    'MaritalStatus': str(MaritalStatus),
    'MonthlyIncome': int(MonthlyIncome),
    'NumCompaniesWorked': int(NumCompaniesWorked),
    'OverTime': str(OverTime),
    'PerformanceRating': int(PerformanceRating),
    'RelationshipSatisfaction': int(RelationshipSatisfaction),
    'StockOptionLevel':  StockOptionLevel,
    'TotalWorkingYears': int(TotalWorkingYears),
    'TrainingTimesLastYear': TrainingTimesLastYear,
    'WorkLifeBalance': int(WorkLifeBalance),
    'YearsAtCompany': int(YearsAtCompany),
    'YearsInCurrentRole': int(YearsInCurrentRole),
    'YearsSinceLastPromotion': int(YearsSinceLastPromotion),
```

```
        'YearsWithCurrManager': int(YearsWithCurrManager)
    }

    df = pd.DataFrame([dict])

    df['Total_Satisfaction'] = (df['EnvironmentSatisfaction'] +
                                df['JobInvolvement'] +
                                df['JobSatisfaction'] +
                                df['RelationshipSatisfaction'] +
                                df['WorkLifeBalance']) / 5

    # Drop Columns
    df.drop(
        ['EnvironmentSatisfaction', 'JobInvolvement', 'JobSatisfaction', 'RelationshipSatisfaction',
'WorkLifeBalance'],
        axis=1, inplace=True)# for permanent delete

    # Convert Total satisfaction into boolean
    df['Total_Satisfaction_bool'] = df['Total_Satisfaction'].apply(lambda x: 1 if x >= 2.8 else 0)
    df.drop('Total_Satisfaction', axis=1, inplace=True)

    # It can be observed that the rate of attrition of employees below age of 35 is high
    df['Age_bool'] = df['Age'].apply(lambda x: 1 if x < 35 else 0)
    df.drop('Age', axis=1, inplace=True)

    # It can be observed that the employees are more likey the drop the job if dailyRate less
than 800
    df['DailyRate_bool'] = df['DailyRate'].apply(lambda x: 1 if x < 800 else 0)
    df.drop('DailyRate', axis=1, inplace=True)

    # Employees working at R&D Department have higher attrition rate
    df['Department_bool'] = df['Department'].apply(lambda x: 1 if x == 'Research & Development'
else 0)
    df.drop('Department', axis=1, inplace=True)

    # Rate of attrition of employees is high if DistanceFromHome > 10
    df['DistanceFromHome_bool'] = df['DistanceFromHome'].apply(lambda x: 1 if x > 10 else 0)
    df.drop('DistanceFromHome', axis=1, inplace=True)
```

```
    # Employees are more likey to drop the job if the employee is working as Laboratory
Technician
    df['JobRole_bool'] = df['JobRole'].apply(lambda x: 1 if x == 'Laboratory Technician' else 0)
    df.drop('JobRole', axis=1, inplace=True)

    # Employees are more likey to the drop the job if the employee's hourly rate < 65
    df['HourlyRate_bool'] = df['HourlyRate'].apply(lambda x: 1 if x < 65 else 0)
    df.drop('HourlyRate', axis=1, inplace=True)

    # Employees are more likey to the drop the job if the employee's MonthlyIncome < 4000
    df['MonthlyIncome_bool'] = df['MonthlyIncome'].apply(lambda x: 1 if x < 4000 else 0)
    df.drop('MonthlyIncome', axis=1, inplace=True)

    # Rate of attrition of employees is high if NumCompaniesWorked < 3
    df['NumCompaniesWorked_bool'] = df['NumCompaniesWorked'].apply(lambda x: 1 if x > 3
else 0)
    df.drop('NumCompaniesWorked', axis=1, inplace=True)

    # Employees are more likey to the drop the job if the employee's TotalWorkingYears < 8
    df['TotalWorkingYears_bool'] = df['TotalWorkingYears'].apply(lambda x: 1 if x < 8 else 0)
    df.drop('TotalWorkingYears', axis=1, inplace=True)

    # Employees are more likey to the drop the job if the employee's YearsAtCompany < 3
    df['YearsAtCompany_bool'] = df['YearsAtCompany'].apply(lambda x: 1 if x < 3 else 0)
    df.drop('YearsAtCompany', axis=1, inplace=True)

    # Employees are more likey to the drop the job if the employee's YearsInCurrentRole < 3
    df['YearsInCurrentRole_bool'] = df['YearsInCurrentRole'].apply(lambda x: 1 if x < 3 else 0)
    df.drop('YearsInCurrentRole', axis=1, inplace=True)

    # Employees are more likely to the drop the job if the employee's YearsSinceLastPromotion <
1
    df['YearsSinceLastPromotion_bool'] = df['YearsSinceLastPromotion'].apply(lambda x: 1 if x < 1
else 0)
    df.drop('YearsSinceLastPromotion', axis=1, inplace=True)

    # Employees are more likely to the drop the job if the employee's YearsWithCurrManager <
```

```
1
    df['YearsWithCurrManager_bool'] = df['YearsWithCurrManager'].apply(lambda x: 1 if x < 1
else 0)
    df.drop('YearsWithCurrManager', axis=1, inplace=True)

    # Convert Categorical to Numerical
    # Buisness Travel
    if BusinessTravel == 'Rarely':
        df['BusinessTravel_Rarely'] = 1
        df['BusinessTravel_Frequently'] = 0
        df['BusinessTravel_No_Travel'] = 0
    elif BusinessTravel == 'Frequently':
        df['BusinessTravel_Rarely'] = 0
        df['BusinessTravel_Frequently'] = 1
        df['BusinessTravel_No_Travel'] = 0
    else:
        df['BusinessTravel_Rarely'] = 0
        df['BusinessTravel_Frequently'] = 0
        df['BusinessTravel_No_Travel'] = 1
    df.drop('BusinessTravel', axis=1, inplace=True)

    # Education
    if Education == 1:
        df['Education_1'] = 1
        df['Education_2'] = 0
        df['Education_3'] = 0
        df['Education_4'] = 0
        df['Education_5'] = 0
    elif Education == 2:
        df['Education_1'] = 0
        df['Education_2'] = 1
        df['Education_3'] = 0
        df['Education_4'] = 0
        df['Education_5'] = 0
    elif Education == 3:
        df['Education_1'] = 0
        df['Education_2'] = 0
        df['Education_3'] = 1
```

```
        df['Education_4'] = 0
        df['Education_5'] = 0
    elif Education == 4:
        df['Education_1'] = 0
        df['Education_2'] = 0
        df['Education_3'] = 0
        df['Education_4'] = 1
        df['Education_5'] = 0
    else:
        df['Education_1'] = 0
        df['Education_2'] = 0
        df['Education_3'] = 0
        df['Education_4'] = 0
        df['Education_5'] = 1
    df.drop('Education', axis=1, inplace=True)

    # EducationField
    if EducationField == 'Life Sciences':
        df['EducationField_Life_Sciences'] = 1
        df['EducationField_Medical'] = 0
        df['EducationField_Marketing'] = 0
        df['EducationField_Technical_Degree'] = 0
        df['Education_Human_Resources'] = 0
        df['Education_Other'] = 0
    elif EducationField == 'Medical':
        df['EducationField_Life_Sciences'] = 0
        df['EducationField_Medical'] = 1
        df['EducationField_Marketing'] = 0
        df['EducationField_Technical_Degree'] = 0
        df['Education_Human_Resources'] = 0
        df['Education_Other'] = 0
    elif EducationField == 'Marketing':
        df['EducationField_Life_Sciences'] = 0
        df['EducationField_Medical'] = 0
        df['EducationField_Marketing'] = 1
        df['EducationField_Technical_Degree'] = 0
        df['Education_Human_Resources'] = 0
        df['Education_Other'] = 0
```

```
    elif EducationField == 'Technical Degree':
        df['EducationField_Life_Sciences'] = 0
        df['EducationField_Medical'] = 0
        df['EducationField_Marketing'] = 0
        df['EducationField_Technical_Degree'] = 1
        df['Education_Human_Resources'] = 0
        df['Education_Other'] = 0
    elif EducationField == 'Human Resources':
        df['EducationField_Life_Sciences'] = 0
        df['EducationField_Medical'] = 0
        df['EducationField_Marketing'] = 0
        df['EducationField_Technical_Degree'] = 0
        df['Education_Human_Resources'] = 1
        df['Education_Other'] = 0
    else:
        df['EducationField_Life_Sciences'] = 0
        df['EducationField_Medical'] = 0
        df['EducationField_Marketing'] = 0
        df['EducationField_Technical_Degree'] = 0
        df['Education_Human_Resources'] = 1
        df['Education_Other'] = 1
    df.drop('EducationField', axis=1, inplace=True)

    # Gender
    if Gender == 'Male':
        df['Gender_Male'] = 1
        df['Gender_Female'] = 0
    else:
        df['Gender_Male'] = 0
        df['Gender_Female'] = 1
    df.drop('Gender', axis=1, inplace=True)

    # Overtime
    if OverTime == 'Yes':
        df['OverTime_Yes'] = 1
        df['OverTime_No'] = 0
    else:
        df['OverTime_Yes'] = 0
```

```
    df['OverTime_No'] = 1
df.drop('OverTime', axis=1, inplace=True)

# Training Time Last Year
if TrainingTimesLastYear == 0:
    df['TrainingTimesLastYear_0'] = 1
    df['TrainingTimesLastYear_1'] = 0
    df['TrainingTimesLastYear_2'] = 0
    df['TrainingTimesLastYear_3'] = 0
    df['TrainingTimesLastYear_4'] = 0
    df['TrainingTimesLastYear_5'] = 0
    df['TrainingTimesLastYear_6'] = 0
elif TrainingTimesLastYear == 1:
    df['TrainingTimesLastYear_0'] = 0
    df['TrainingTimesLastYear_1'] = 1
    df['TrainingTimesLastYear_2'] = 0
    df['TrainingTimesLastYear_3'] = 0
    df['TrainingTimesLastYear_4'] = 0
    df['TrainingTimesLastYear_5'] = 0
    df['TrainingTimesLastYear_6'] = 0
elif TrainingTimesLastYear == 2:
    df['TrainingTimesLastYear_0'] = 0
    df['TrainingTimesLastYear_1'] = 0
    df['TrainingTimesLastYear_2'] = 1
    df['TrainingTimesLastYear_3'] = 0
    df['TrainingTimesLastYear_4'] = 0
    df['TrainingTimesLastYear_5'] = 0
    df['TrainingTimesLastYear_6'] = 0
elif TrainingTimesLastYear == 3:
    df['TrainingTimesLastYear_0'] = 0
    df['TrainingTimesLastYear_1'] = 0
    df['TrainingTimesLastYear_2'] = 0
    df['TrainingTimesLastYear_3'] = 1
    df['TrainingTimesLastYear_4'] = 0
    df['TrainingTimesLastYear_5'] = 0
    df['TrainingTimesLastYear_6'] = 0
elif TrainingTimesLastYear == 4:
    df['TrainingTimesLastYear_0'] = 0
```

```
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 1
        df['TrainingTimesLastYear_5'] = 0
        df['TrainingTimesLastYear_6'] = 0
    elif TrainingTimesLastYear == 5:
        df['TrainingTimesLastYear_0'] = 0
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 0
        df['TrainingTimesLastYear_5'] = 1
        df['TrainingTimesLastYear_6'] = 0
    else:
        df['TrainingTimesLastYear_0'] = 0
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 0
        df['TrainingTimesLastYear_5'] = 0
        df['TrainingTimesLastYear_6'] = 1
    df.drop('TrainingTimesLastYear', axis=1, inplace=True)

    prediction = model.predict(df)

    if prediction == 0:
        return render_template('index.html', prediction_text='Employee Might Not Leave The Job')

    else:
        return render_template('index.html', prediction_text='Employee Might Leave The Job')

if __name__ == "__main__":
    app.run(debug=True)
```

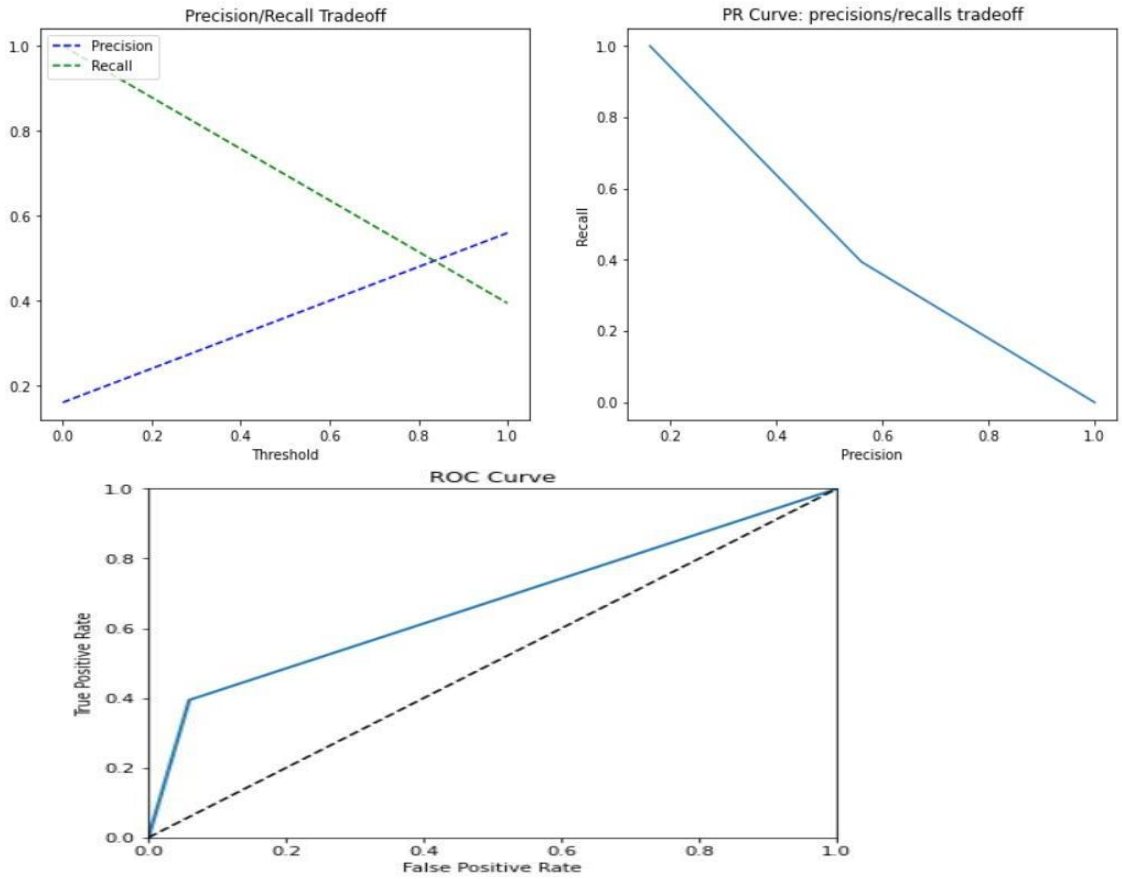
Chapter 5

RESULT

5.1 Comparative result to existing system

The specific comparative results of existing systems for HR analytics and prediction of employee attrition would depend on the algorithms, data sources, features, and methodologies used. Different systems may employ various machine learning techniques, such as logistic regression, decision trees, random forests, or neural networks, to predict employee attrition.

To assess the comparative results of existing systems, you would typically consider metrics such as accuracy, precision, recall, F1 score, and area under the curve (AUC) for the receiver operating characteristic (ROC) curve. These metrics indicate the performance and predictive power of the models.



Chapter 6

CONCLUSION

6.1 CONCLUSION

Using logistic regression algorithms, we built a model that helps predict employee attrition with an accuracy of 90%. We then completed an exploratory data analysis, creating heatmaps to show correlations between variables, and visualizations to understand the demographics and features of attrited employees. We then completed an exploratory data analysis, creating heatmaps to show correlations between variables, and visualizations to understand the demographics and features of attrited employees. Using logistic regression algorithms, we built a model that helps predict employee attrition with an accuracy of 95%. Then we concluded that there are some variables that are important factors in an employee's decision to attrition such as education fields, job roles, satisfaction, etc. We also developed implications for companies with actions to consider in order to prevent future loss of talent.

6.2 FUTURE SCOPE:

The Attrition Prediction model estimates the attrition risk for your employee populations in real-time, which is recalculated every time an employee submits feedback. The aggregated, segment-level view keeps the accuracy of your predictions high while protecting individual employee identity. If turnover rates are high, the immediate consequences are severe: loss of valuable knowledge and experience, loss of morale for those left, and loss of belief in the team's competence and ability to perform.