Employee Promotion Prediction Using Machine Learning

1.INTRODUCTION

Overview

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.

Proposed System

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.

For better training results we make use of IBM to train the model to deploy the model on IBM.

2.LITERATURE SURVEY

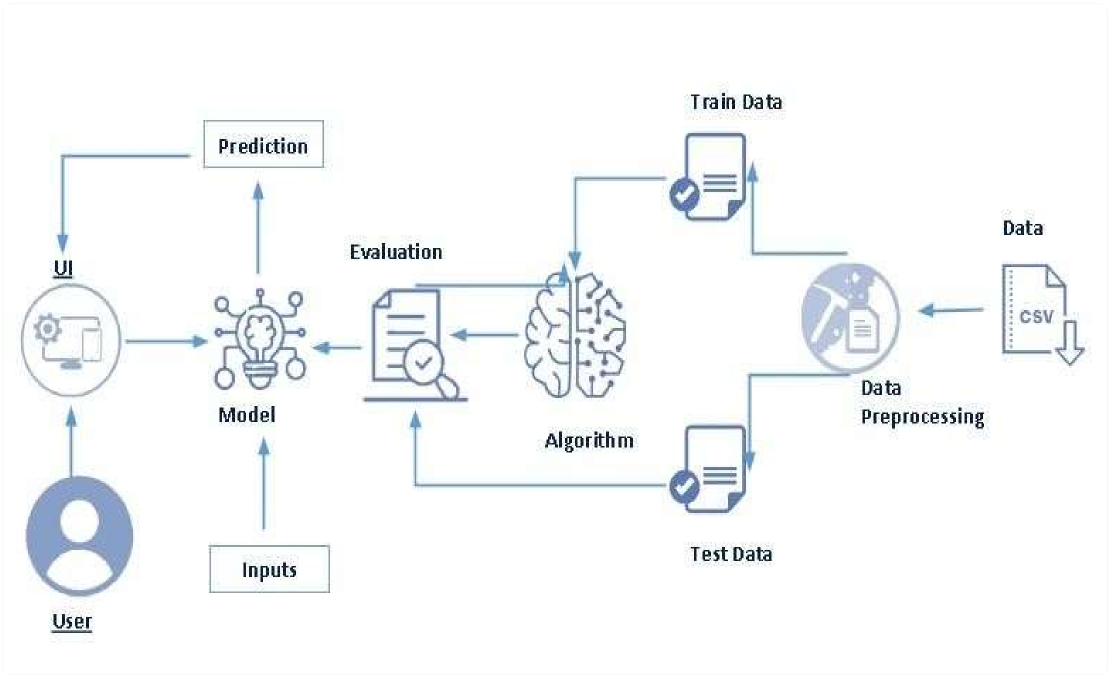
2.1Existing Problem

In [1], computer scientist was once quoted as spoken language, "You remove our prime twenty workers and that we [Microsoft] become a mediocre company". This statement by computer scientist took our attention to 1 of the main issues of worker attrition at workplaces. worker attrition (turnover) causes a major price to any organization which can afterward impact its overall potency. As per CompData Surveys, over the past 5 years, total turnover has accumulated from fifteen.1 p.c to one8.5 percent. For any organization, finding a well trained and experienced worker may be a complicated task, however it's even additional complicated to interchange such workers. This not solely will increase the many Human Resource (HR) price, however additionally impacts the market price of a corporation. Despite these facts and ground reality, there's very little attention to the literature, that has been seeded to several misconceptions between time unit and workers. Therefore, the aim of this paper is to supply a framework for predicting the worker churn by analyzing the employee's precise behaviors and attributes mistreatment classification techniques

3.THEORETICAL ANALYSIS

3.1 Block Diagram

Technical Architecture:



3.2 Hardware/software Desgining

Hardware requriement

1. 2 GB ram or above
2. Dual core processor or above
3. Internet connection

Software requirements

1. Anaconda Navigator
2. Python Packages
3. IBM Watson Studio

4.EXPERIMENTAL INVESTIGATION

Project Objectives

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data preprocessing techniques.
- You will be able to analyze or get insights into data through visualization.
- Applying different algorithms according to the dataset and based on visualization.
- Have knowledge of data/capping techniques on outliers and some visualization concepts.
- You will be able to know how to build a web application using the Flask framework.

5.FLOWCHART

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 we 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.

For this project, we will be using a Jupyter notebook and Spyder

To install the Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

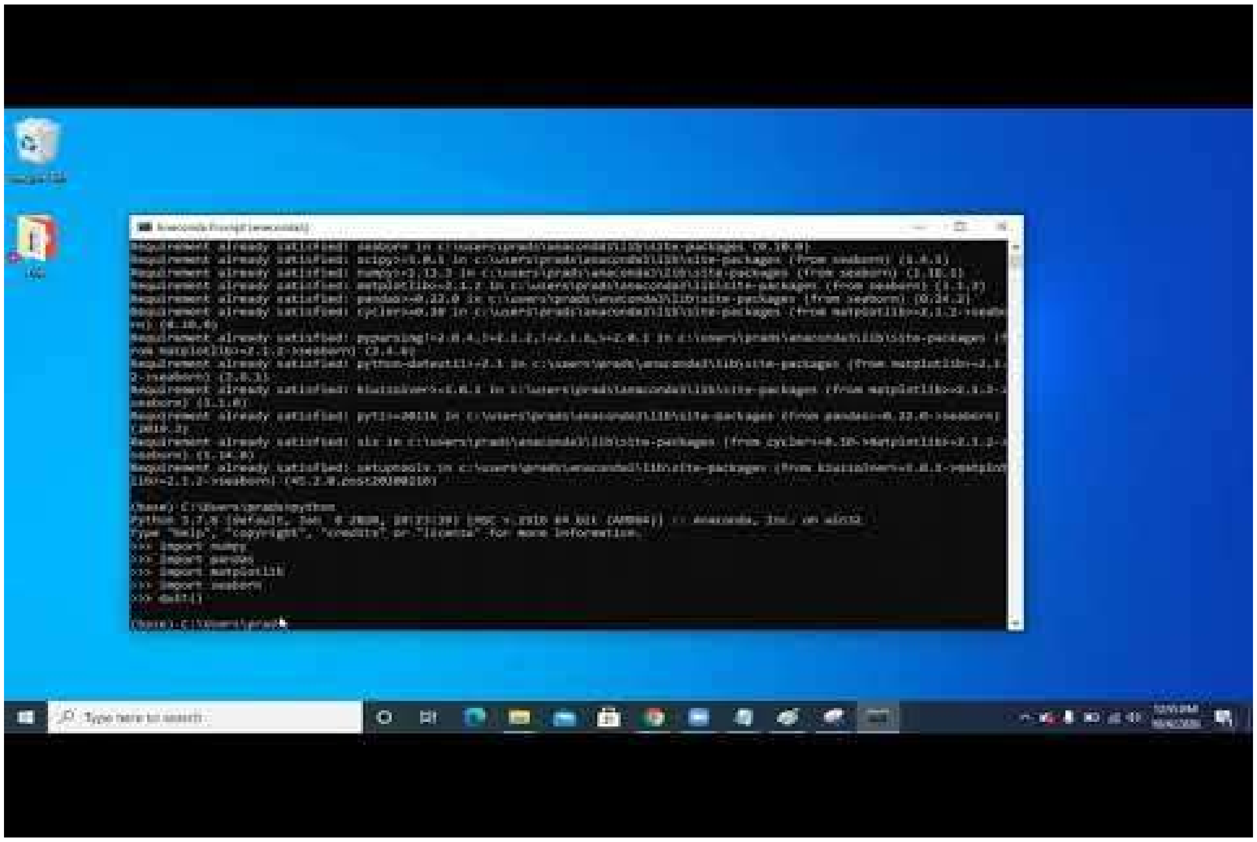
Step 2: Python packages

To build Machine learning models you 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 toolkits

Step 3: Flask - Web framework used for building Web applications.

Watch the video below to learn how to install packages.



```
Requirement already satisfied: seaborn in c:\users\prads\anaconda3\lib\site-packages (0.10.0)
Requirement already satisfied: scipy>=1.0.1 in c:\users\prads\anaconda3\lib\site-packages (from seaborn) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\prads\anaconda3\lib\site-packages (from seaborn) (1.18.1)
Requirement already satisfied: matplotlib>=2.1.2 in c:\users\prads\anaconda3\lib\site-packages (from seaborn) (3.1.3)
Requirement already satisfied: pandas>=0.22.0 in c:\users\prads\anaconda3\lib\site-packages (from seaborn) (0.24.2)
Requirement already satisfied: cycler>=0.10 in c:\users\prads\anaconda3\lib\site-packages (from matplotlib>=2.1.2->seaborn) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\prads\anaconda3\lib\site-packages (from matplotlib>=2.1.2->seaborn) (2.4.6)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\prads\anaconda3\lib\site-packages (from matplotlib>=2.1.2->seaborn) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\prads\anaconda3\lib\site-packages (from matplotlib>=2.1.2->seaborn) (1.1.0)
Requirement already satisfied: pytz>=2011k in c:\users\prads\anaconda3\lib\site-packages (from pandas>=0.22.0->seaborn) (2019.3)
Requirement already satisfied: six in c:\users\prads\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib>=2.1.2->seaborn) (1.14.0)
Requirement already satisfied: setuptools in c:\users\prads\anaconda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>=2.1.2->seaborn) (45.2.0.post20200210)

(base) C:\Users\prads>python
Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> import pandas
>>> import matplotlib
>>> import seaborn
>>> quit()

(base) C:\Users\prads>
```

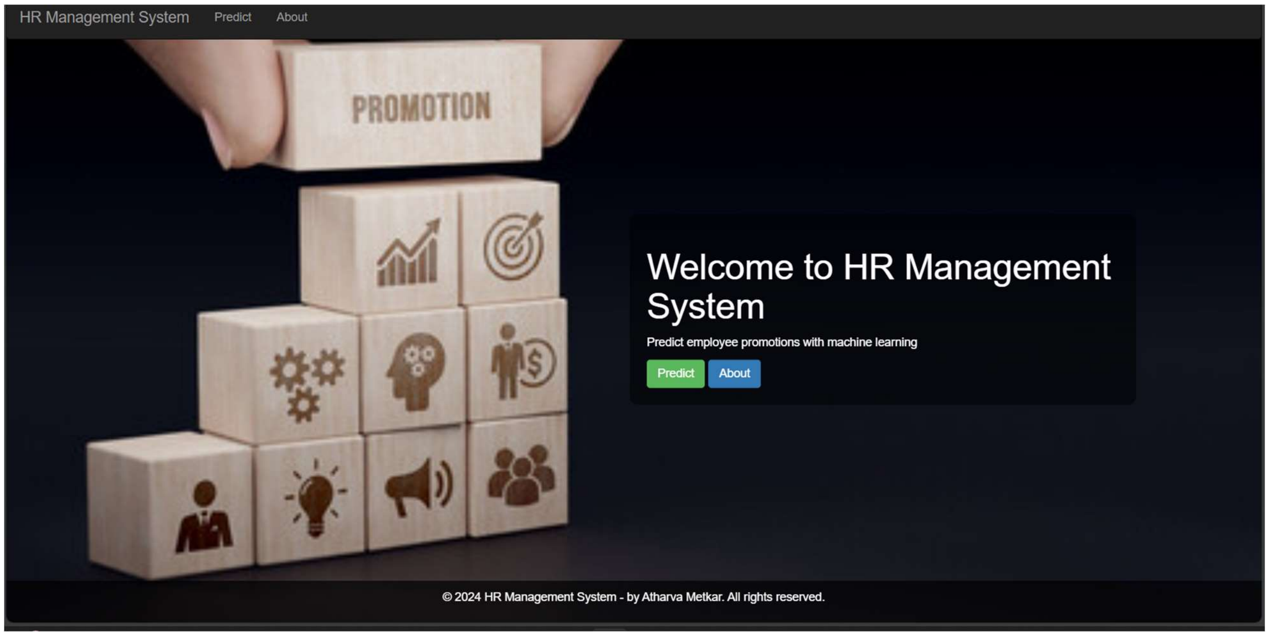
If you are using anaconda navigator, follow the below steps to download the required packages:

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.

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

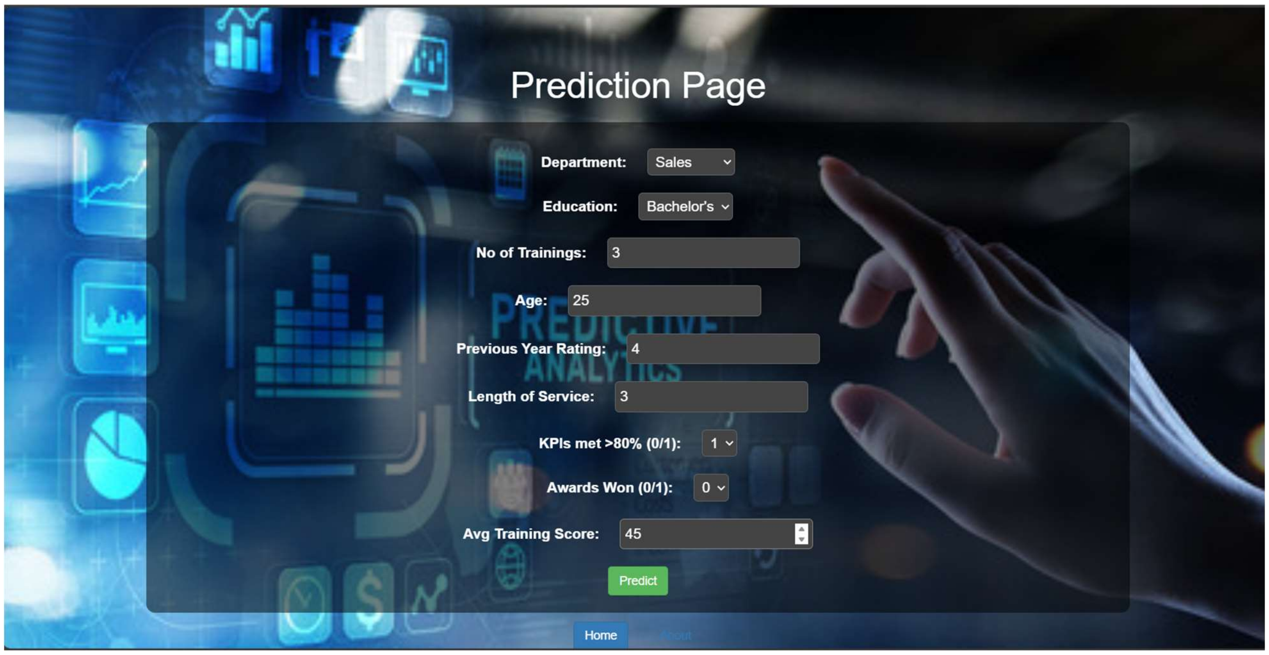
6.RESULT



Welcome to HR Management System

Predict employee promotions with machine learning

© 2024 HR Management System - by Atharva Metkar. All rights reserved.



Prediction Page

Department: Sales

Education: Bachelor's

No of Trainings: 3

Age: 25

Previous Year Rating: 4

Length of Service: 3

KPIs met >80% (0/1): 1

Awards Won (0/1): 0

Avg Training Score: 45

7. ADVANTAGES AND DISADVANTAGES

Advantages

- Efficient program for Employee promotion prediction

•Accurate output is produced

•Will predict Employee promotion with extreme accuracy

•Relatively inexpensive and fast

Disadvantages

• It will work in all condition but some condition it may not give correct output

8. APPLICATION

• Employee information

9. CONCLUSION

Promotions have a favorable, significant and beneficial impact on employee work performance in human resources process. In this study a prediction model for employee promotion is proposed by using RF method.

10. FUTURE SCOPE

This program allows users to predict if the Employee is promoted or not. By the help of this prediction, we can use this program and we can ensure that   the is promoted or not. It helps reduce the stress on user identifying important promoted employee easily.

11. BIBLIOGRAPHY

• https://towardsdatascience.com/will-your-employee-leave-a-machine-learning-model-8484c2a6663e

• https://medium.com/digitaladoption101/5-ways-predictive-analytics-can-contribute-toemployee-engagement-e37c0ea542a

12.APPENDIX

Prior Knowledge

One should have knowledge of the following Concepts

Please refer to the videos below to gain sufficient required knowledge to complete the project.

- **Supervised and unsupervised learning:**

- **Regression Classification and Clustering :**

- **Random Forest Classifier:**

- **Ensemble Technique:**

- **Decision Tree Classifier:**

- **KNN**: Refer the link
- **Xgboost**: Refer the link

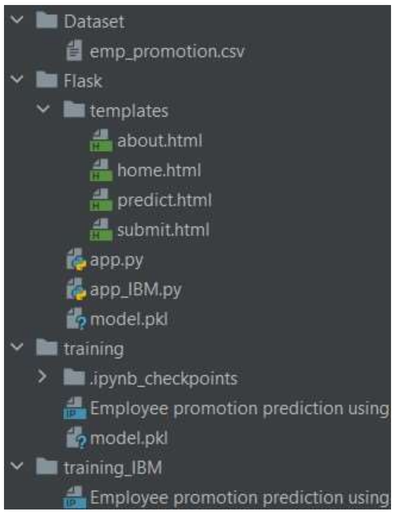
- **Evaluation metrics:** Refer the link

Congrats! You have completed the basic fundamentals of various topics.

Let's start building the project.

Project Structure

Create the Project folder which contains files as shown below



```
Dataset
    emp_promotion.csv
Flask
    templates
        about.html
        home.html
        predict.html
        submit.html
    app.py
    app_IBM.py
    model.pkl
training
    .ipynb_checkpoints
    Employee promotion prediction using
    model.pkl
training_IBM
    Employee promotion prediction using
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Model.pkl is our saved model. Further we will use this model for flask integration.

Data Collection

ML depends heavily on data, It is most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Download Dataset

Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used drug200.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link

Visualizing And Analysing The Data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

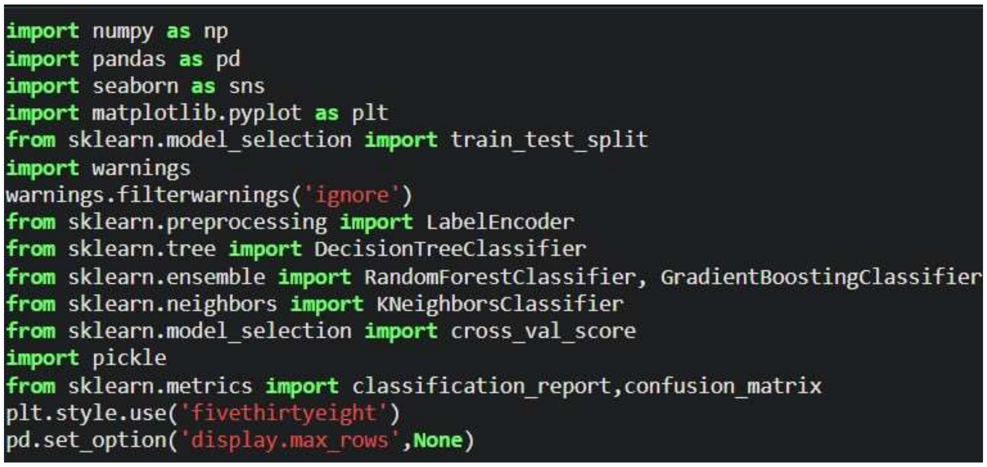
Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

8.APPENDIX

Importing The Libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualization style as fivethirtyeight.

To know about the packages refer the link given on pre requisites.

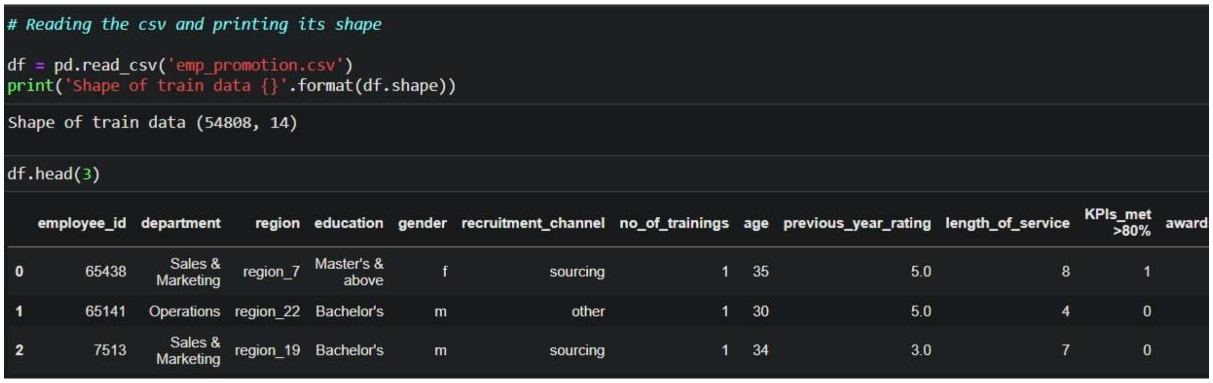


```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
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)
```

Read The Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of csv file.



```python
# Reading the csv and printing its shape

df = pd.read_csv('emp_promotion.csv')
print('Shape of train data {}'.format(df.shape))
```

```
Shape of train data (54808, 14)
```

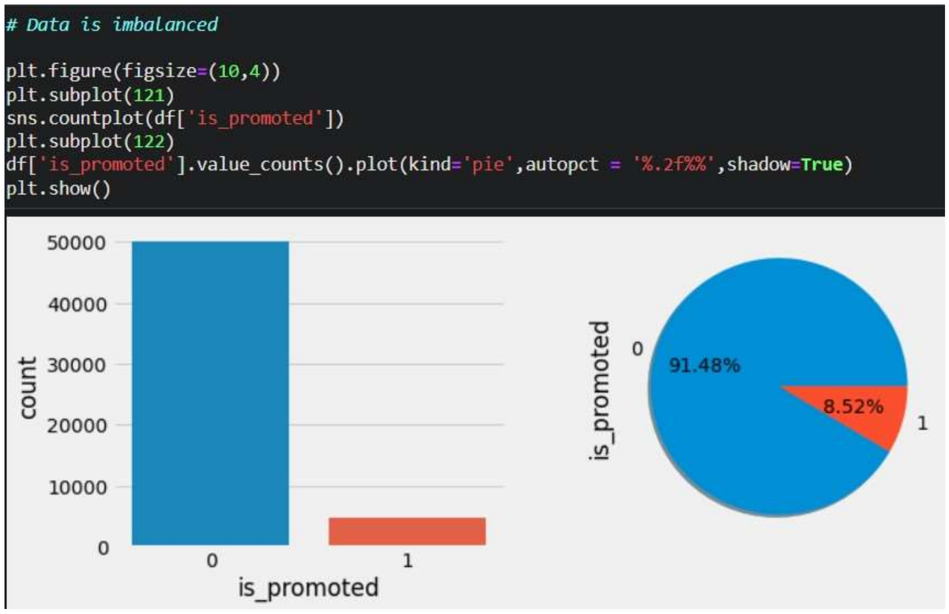
```python
df.head(3)
```

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	award
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	5.0	8	1	
1	65141	Operations	region_22	Bachelor's	m	other	1	30	5.0	4	0	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	3.0	7	0	

Univariate Analysis

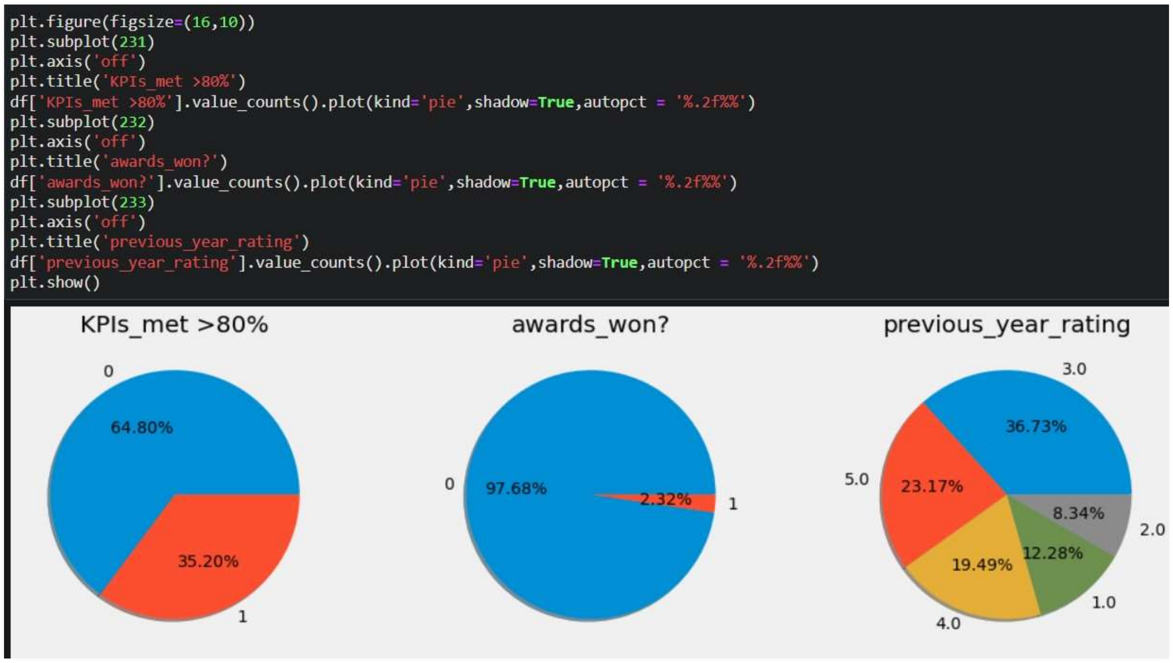
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as pie plot, box plot and count plot.

- 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.



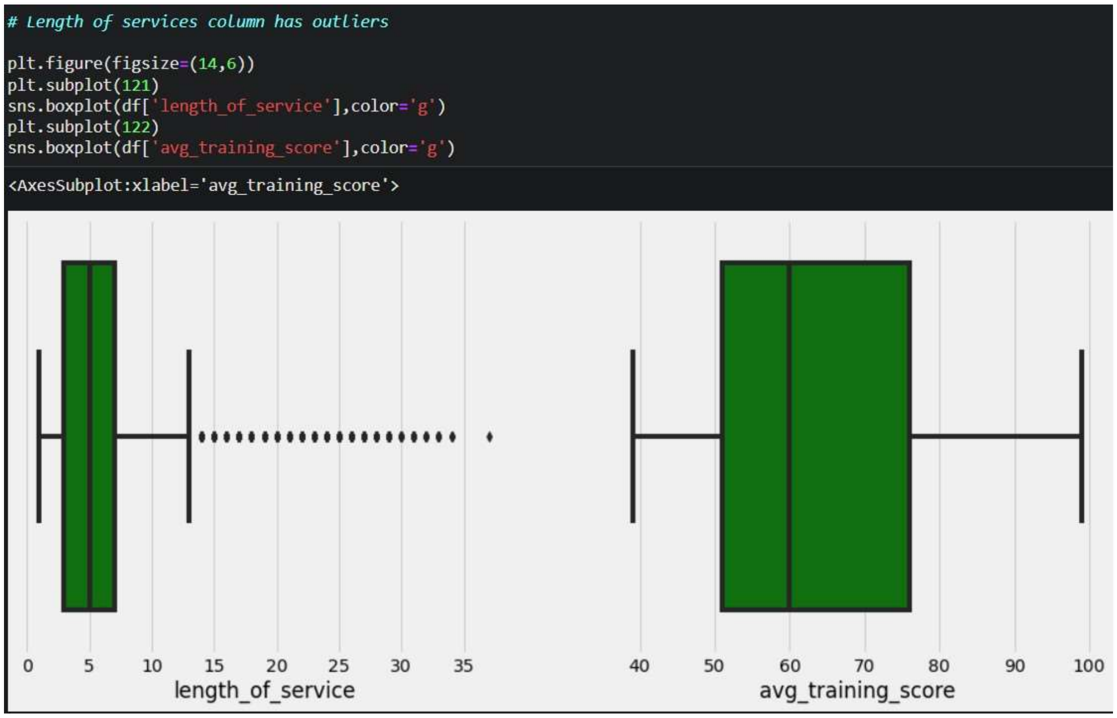
```
# Data is imbalanced

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()
```

- 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.



```
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()
```

- 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.



```
# Length of services column has outliers

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')
```

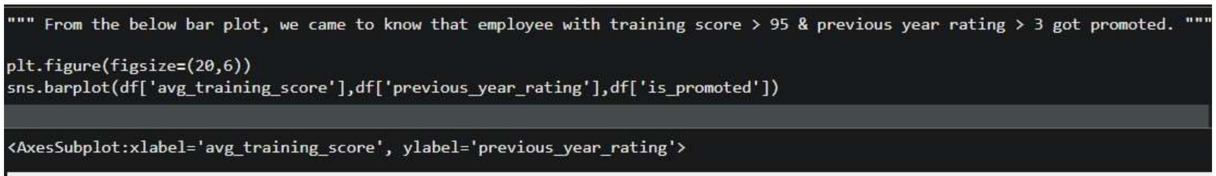
```
<AxesSubplot:xlabel='avg_training_score'>
```

Multivariate Analysis

In simple words, multivariate analysis is to find the relation between multiple features.

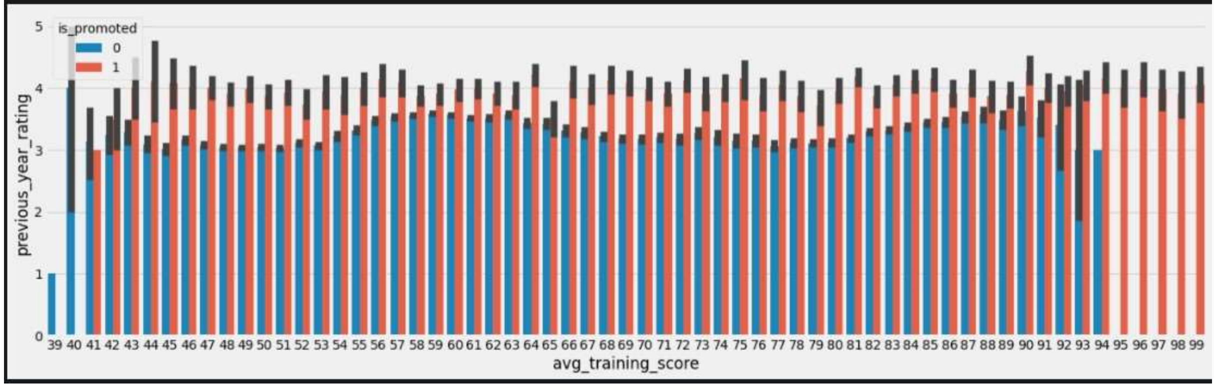
Here we have used barplot from seaborn package.

- 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%).



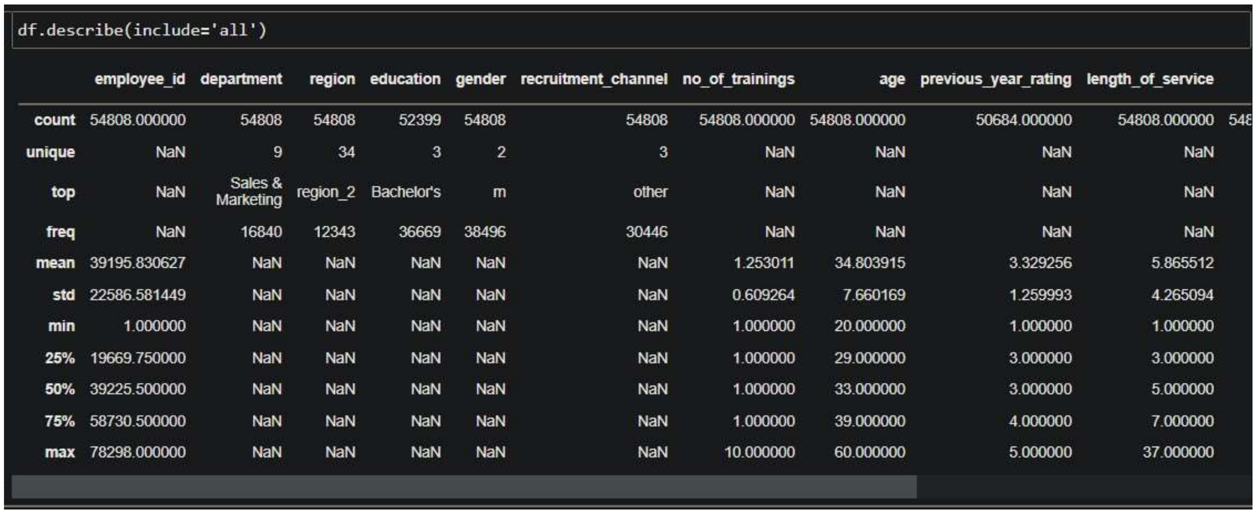
```
""" From the below bar plot, we came to know that employee with training score > 95 & previous year rating > 3 got promoted. """

plt.figure(figsize=(20,6))
sns.barplot(df['avg_training_score'],df['previous_year_rating'],df['is_promoted'])
```

```
<AxesSubplot:xlabel='avg_training_score', ylabel='previous_year_rating'>
```



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.



```
df.describe(include='all')
```

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service
count	54808.000000	54808	54808	52399	54808	54808	54808.000000	54808.000000	50684.000000	54808.000000
unique	NaN	9	34	3	2	3	NaN	NaN	NaN	NaN
top	NaN	Sales & Marketing	region_2	Bachelor's	m	other	NaN	NaN	NaN	NaN
freq	NaN	16840	12343	36669	38496	30446	NaN	NaN	NaN	NaN
mean	39195.830627	NaN	NaN	NaN	NaN	NaN	1.253011	34.803915	3.329256	5.865512
std	22586.581449	NaN	NaN	NaN	NaN	NaN	0.609264	7.660169	1.259993	4.265094
min	1.000000	NaN	NaN	NaN	NaN	NaN	1.000000	20.000000	1.000000	1.000000
25%	19669.750000	NaN	NaN	NaN	NaN	NaN	1.000000	29.000000	3.000000	3.000000
50%	39225.500000	NaN	NaN	NaN	NaN	NaN	1.000000	33.000000	3.000000	5.000000
75%	58730.500000	NaN	NaN	NaN	NaN	NaN	1.000000	39.000000	4.000000	7.000000
max	78298.000000	NaN	NaN	NaN	NaN	NaN	10.000000	60.000000	5.000000	37.000000

Data Pre-Processing

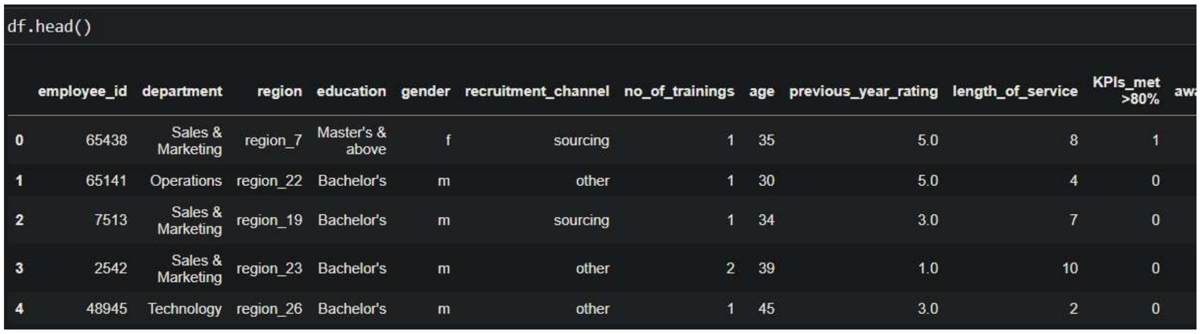
As we have understood how the data is. Lets pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

In the data frame, head() function is used to display the first 5 data. Our dataset has employee id (unique values), department (totally 9 dept.), region (location), education, gender, recruitment channel, age, no. of trainings, previous year ratings, length of service,
KPIs, award won, average training score and is_promoted (target variable) columns.

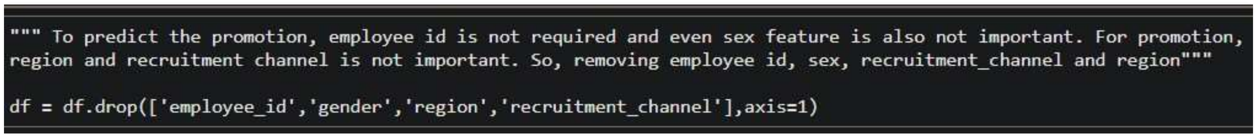


```
df.head()
```

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	aw
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	5.0	8	1	
1	65141	Operations	region_22	Bachelor's	m	other	1	30	5.0	4	0	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	3.0	7	0	
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	2	39	1.0	10	0	
4	48945	Technology	region_26	Bachelor's	m	other	1	45	3.0	2	0	

Drop Unwanted Features

We are building the model to predict the promotion of employees. Employee id is not useful for predicting employee promotion. Generally, based on the performance promotion is given. No organizations will promote their employees by gender, region, and recruitment channel. So, these features are removed from the dataset

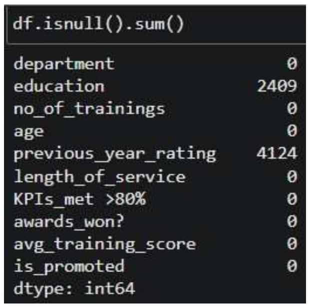


```
""" To predict the promotion, employee id is not required and even sex feature is also not important. For promotion,
region and recruitment channel is not important. So, removing employee id, sex, recruitment_channel and region"""

df = df.drop(['employee_id','gender','region','recruitment_channel'],axis=1)
```

Checking For Null Values

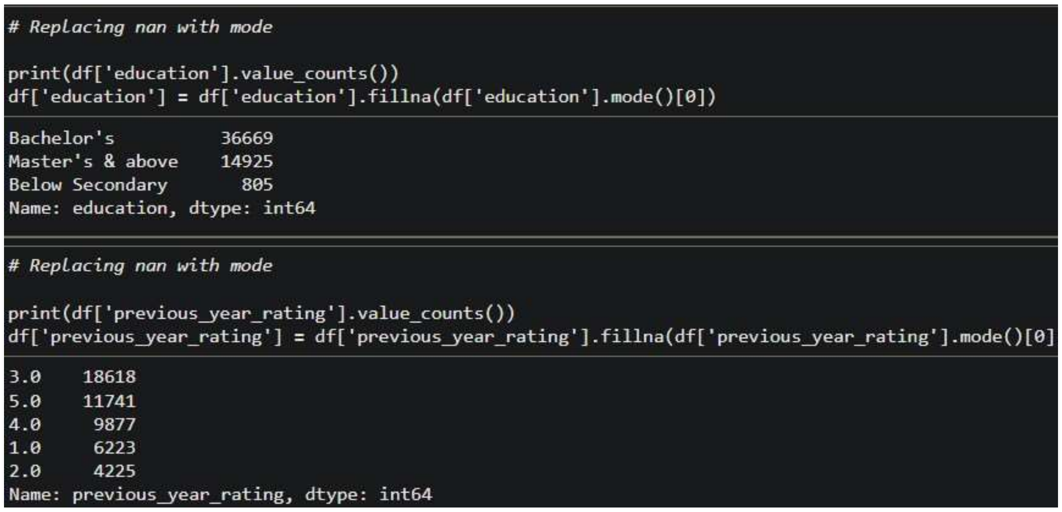
For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it.

From the below image we found that education column and previous year rating column has null values.



```
df.isnull().sum()

department                0
education              2409
no_of_trainings           0
age                       0
previous_year_rating   4124
length_of_service         0
KPIs_met >80%             0
awards_won?               0
avg_training_score        0
is_promoted               0
dtype: int64
```

- Let’s handle the null values.
- For the education feature and previous year rating feature, null values are replaced with their respective mode[0] values. These two features don’t have continuous values. So, the mode value is replaced. The most frequent repeated value for education column is bachelor’s and for previous year rating is 3.



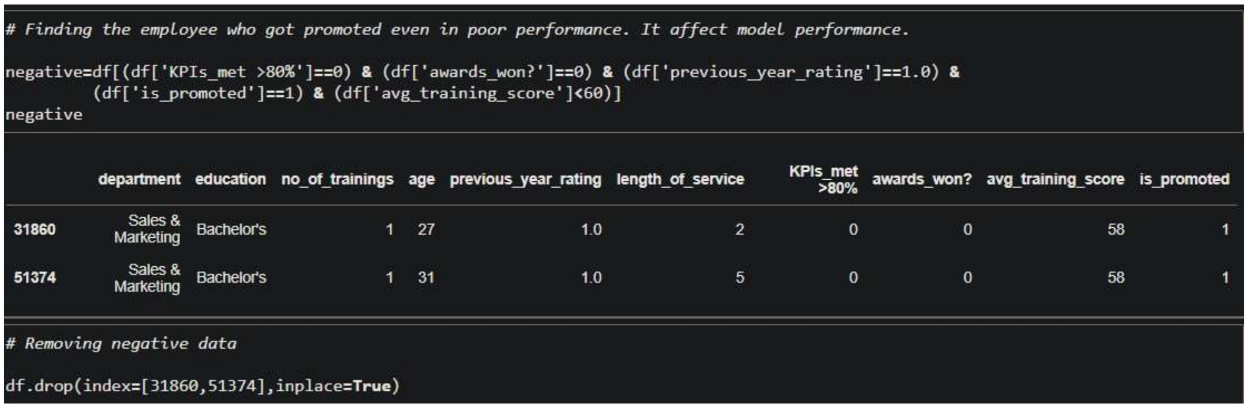
```
# Replacing nan with mode

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

Bachelor's          36669
Master's & above    14925
Below Secondary       805
Name: education, dtype: int64
```

```
# Replacing nan with mode

print(df['previous_year_rating'].value_counts())
df['previous_year_rating'] = df['previous_year_rating'].fillna(df['previous_year_rating'].mode()[0]

3.0    18618
5.0    11741
4.0     9877
1.0     6223
2.0     4225
Name: previous_year_rating, dtype: int64
```

Remove Negative Data

Employees with poor performance got promoted. It affects model performance. So, negative value should be removed.

- Here list comprehension is used to find the negative data.
- Negative data: Employees with no awards, previous year rating was 1.0, KPIs less than 80% and average training score is less than 60.
- Now, negative data is removed.



```
# 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
```

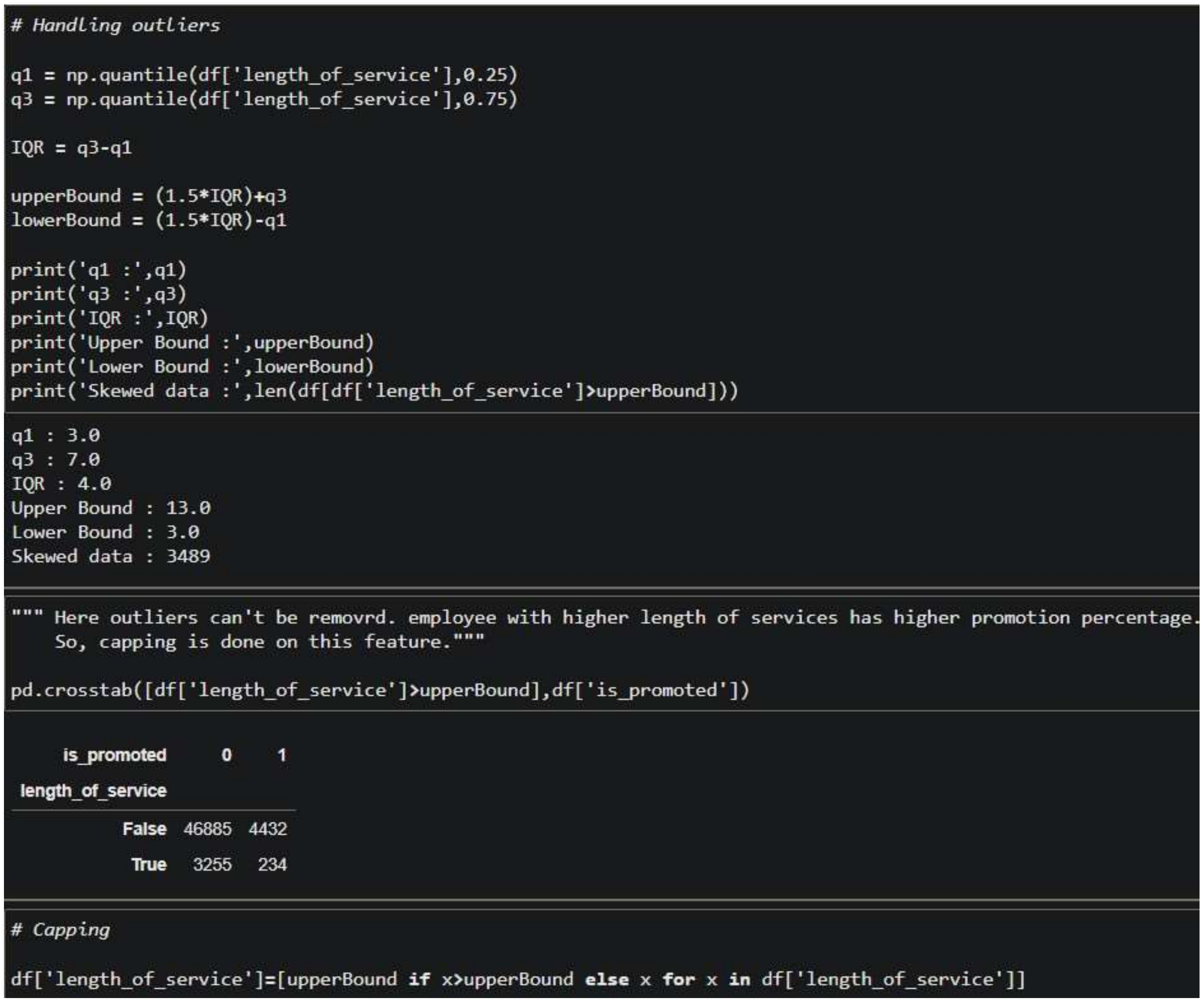
	department	education	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	awards_won?	avg_training_score	is_promoted
31860	Sales & Marketing	Bachelor's	1	27	1.0	2	0	0	58	1
51374	Sales & Marketing	Bachelor's	1	31	1.0	5	0	0	58	1

```
# Removing negative data

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

Handling Outliers

With the help of boxplot, outliers are visualized (refer activity 3 univariate analysis). And here we are going to find upper bound and lower bound of Na_to_K feature with some mathematical formula.

- To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.
- If outliers are removed, we lose more data. It will impact model performance.
- Here removing outliers is impossible. So, the capping technique is used on outliers.
- Capping: Replacing the outliers with upper bound values.



```
# Handling outliers

q1 = np.quantile(df['length_of_service'],0.25)
q3 = np.quantile(df['length_of_service'],0.75)

IQR = q3-q1

upperBound = (1.5*IQR)+q3
lowerBound = (1.5*IQR)-q1

print('q1 :',q1)
print('q3 :',q3)
print('IQR :',IQR)
print('Upper Bound :',upperBound)
print('Lower Bound :',lowerBound)
print('Skewed data :',len(df[df['length_of_service']>upperBound]))
```

```
q1 : 3.0
q3 : 7.0
IQR : 4.0
Upper Bound : 13.0
Lower Bound : 3.0
Skewed data : 3489
```

```
""" Here outliers can't be removrd. employee with higher length of services has higher promotion percentage.
    So, capping is done on this feature."""

pd.crosstab([df['length_of_service']>upperBound],df['is_promoted'])
```

is_promoted	0	1
length_of_service		
False	46885	4432
True	3255	234

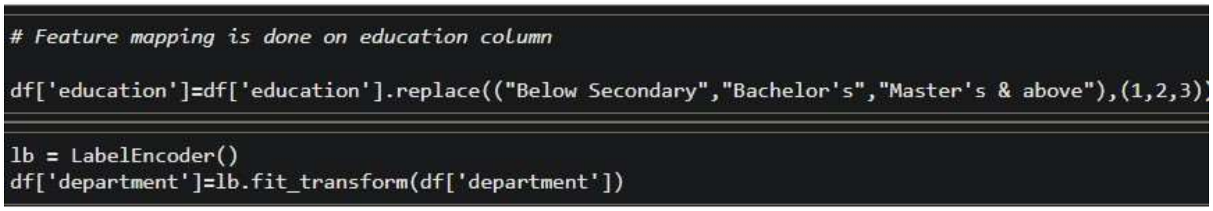
```
# Capping

df['length_of_service']=[upperBound if x>upperBound else x for x in df['length_of_service']]
```

Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using feature mapping and label encoding.

- In our project, categorical features are education and department feature. Feature mapping on education is done by replace() function.
- Label encoder is initialized and department feature is passed as parameter for fit_transform() function. Label encoding uses alphabetical ordering. In department feature we have 9 categories. Those categories are labelled in alphabetical order.

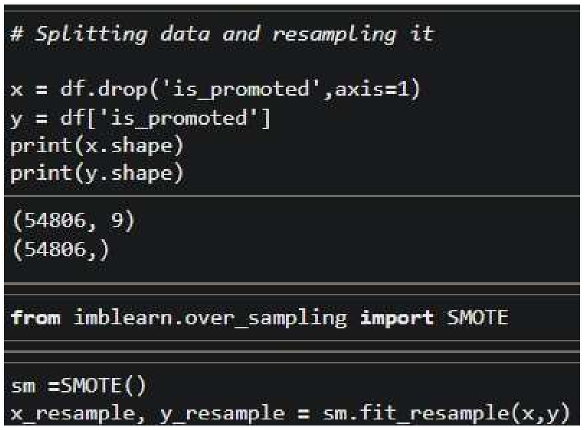


```
# Feature mapping is done on education column

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

lb = LabelEncoder()
df['department']=lb.fit_transform(df['department'])
```

Handling Imbalanced Data

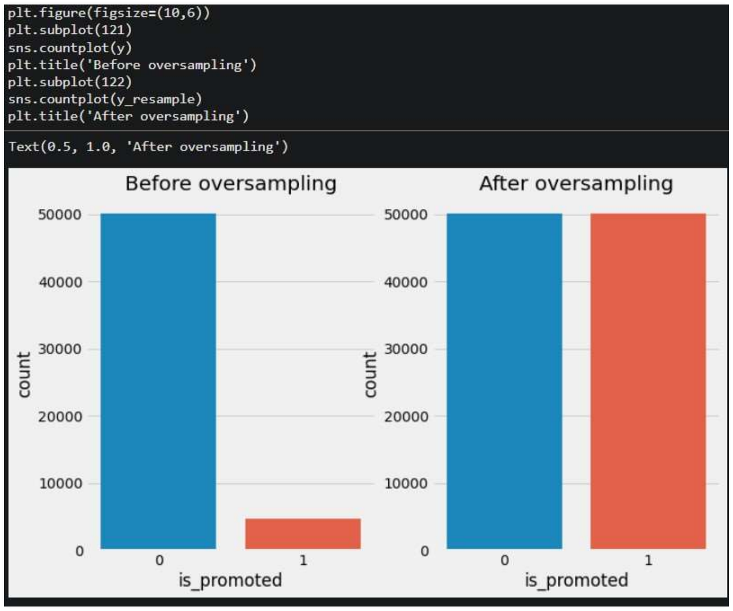
From the activity - univariate analysis we found our data is imbalanced. Now let's split the dataset into x and y. Independent features are passed to x variable and dependent feature is passed to y variable. Then, to handle imbalanced data resampling are done with SMOTE.

- 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 to lean more about SMOTE



```
# Splitting data and resampling it

x = df.drop('is_promoted',axis=1)
y = df['is_promoted']
print(x.shape)
print(y.shape)

(54806, 9)
(54806,)

from imblearn.over_sampling import SMOTE

sm =SMOTE()
x_resample, y_resample = sm.fit_resample(x,y)
```

- Refer the below diagram to visualize the result of smote technique.



```
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')
```

```
Text(0.5, 1.0, 'After oversampling')
```

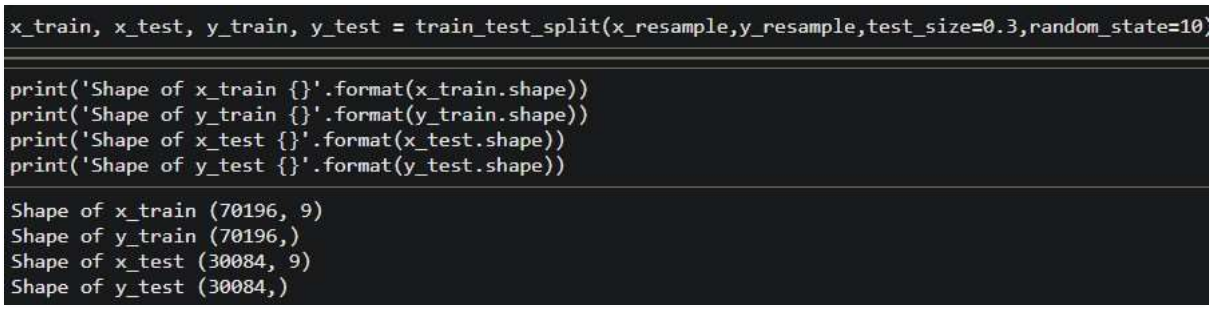
Splitting Data Into Train And Test

Now let’s split the Dataset into train and test sets. For splitting training and testing data we are using

train_test_split() function from sklearn. As parameters, we are passing x_resample, y_resample, test_size,

random_state.

For deep understanding refer this link



```
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))
```

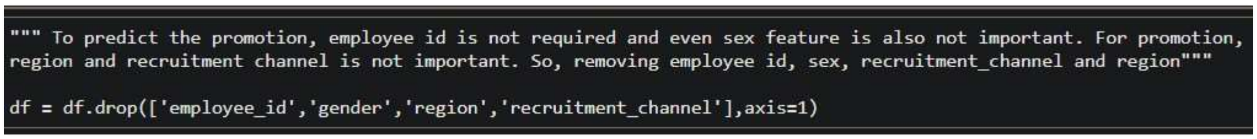
```
Shape of x_train (70196, 9)
Shape of y_train (70196,)
Shape of x_test (30084, 9)
Shape of y_test (30084,)
```

Model Building

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance. To evaluate the performance confusion matrix and classification report is used.

Drop Unwanted Features

We are building the model to predict the promotion of employees. Employee id is not useful for predicting employee promotion. Generally, based on the performance promotion is given. No organizations will promote their employees by gender, region, and recruitment channel. So, these features are removed from the dataset

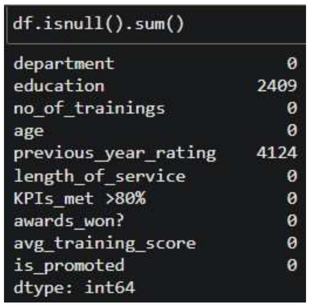


```
""" To predict the promotion, employee id is not required and even sex feature is also not important. For promotion,
region and recruitment channel is not important. So, removing employee id, sex, recruitment_channel and region"""

df = df.drop(['employee_id','gender','region','recruitment_channel'],axis=1)
```

Checking For Null Values

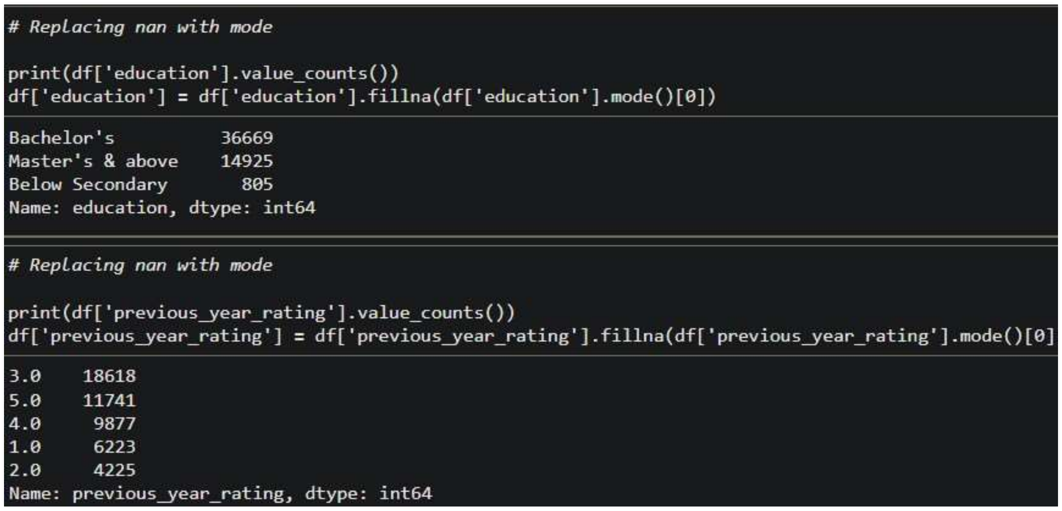
For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it.

From the below image we found that education column and previous year rating column has null values.



```
df.isnull().sum()

department                0
education              2409
no_of_trainings           0
age                       0
previous_year_rating   4124
length_of_service         0
KPIs_met >80%             0
awards_won?               0
avg_training_score        0
is_promoted               0
dtype: int64
```

- Let’s handle the null values.
- For the education feature and previous year rating feature, null values are replaced with their respective mode[0] values. These two features don’t have continuous values. So, the mode value is replaced. The most frequent repeated value for education column is bachelor’s and for previous year rating is 3.



```
# Replacing nan with mode

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

```
Bachelor's          36669
Master's & above    14925
Below Secondary       805
Name: education, dtype: int64
```

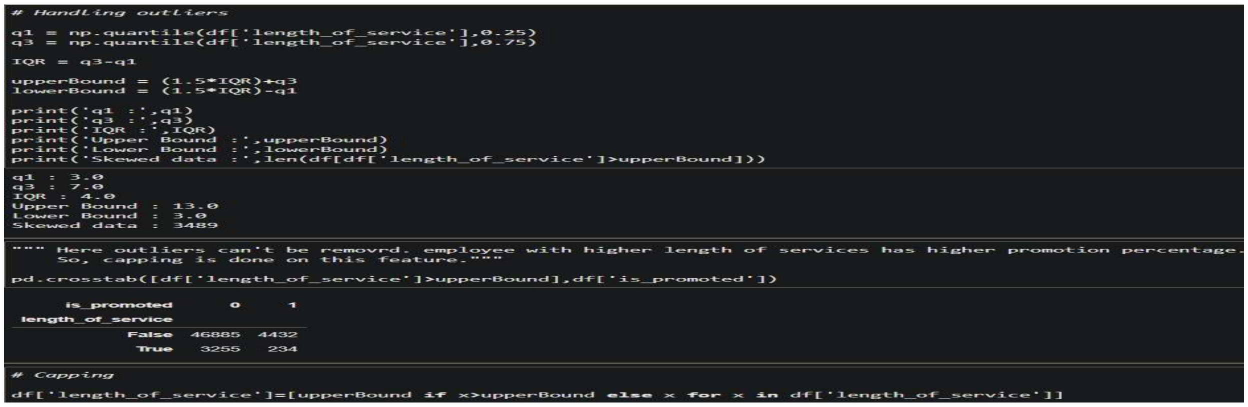
```
# Replacing nan with mode

print(df['previous_year_rating'].value_counts())
df['previous_year_rating'] = df['previous_year_rating'].fillna(df['previous_year_rating'].mode()[0]
```

```
3.0    18618
5.0    11741
4.0     9877
1.0     6223
2.0     4225
Name: previous_year_rating, dtype: int64
```

Remove Negative Data

Employees with poor performance got promoted. It affects model performance. So, negative value should be removed.

- Here list comprehension is used to find the negative data.
- Negative data: Employees with no awards, previous year rating was 1.0, KPIs less than 80% and average training score is less than 60. ● Now, negative data is removed.



```
# Handling outliers

q1 = np.quantile(df['length_of_service'],0.25)
q3 = np.quantile(df['length_of_service'],0.75)

IQR = q3-q1

upperBound = (1.5*IQR)+q3
lowerBound = (1.5*IQR)-q1

print('q1 :',q1)
print('q3 :',q3)
print('IQR :',IQR)
print('Upper Bound :',upperBound)
print('Lower Bound :',lowerBound)
print('Skewed data :',len(df[df['length_of_service']>upperBound]))
```

```
q1 : 3.0
q3 : 7.0
IQR : 4.0
Upper Bound : 13.0
Lower Bound : 3.0
Skewed data : 3489
```

```
""" Here outliers can't be removrd. employee with higher length of services has higher promotion percentage.
    So, capping is done on this feature."""

pd.crosstab([df['length_of_service']>upperBound],df['is_promoted'])
```

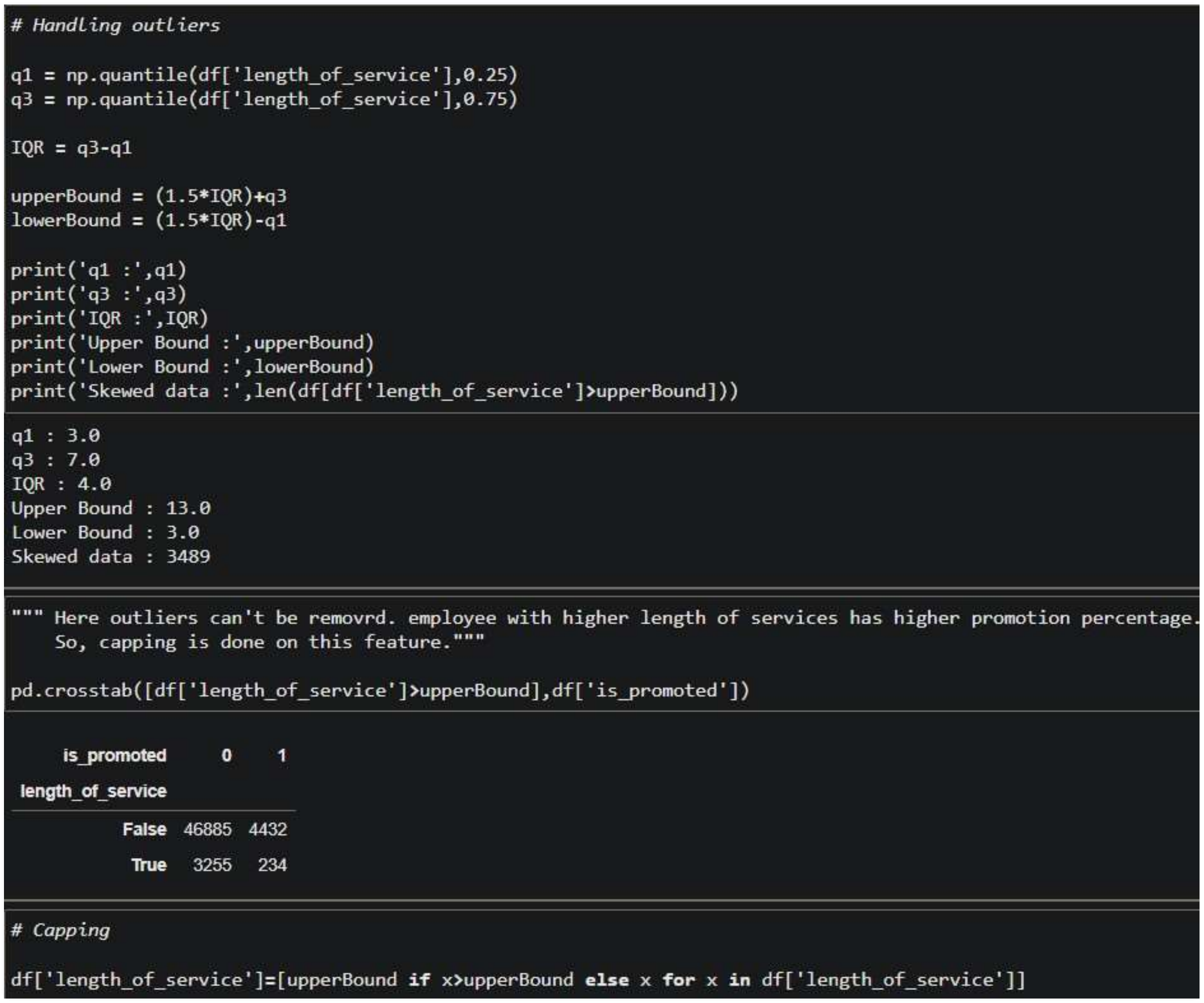
is_promoted	0	1
length_of_service		
False	46885	4432
True	3255	234

```
# Capping
df['length_of_service']=[upperBound if x>upperBound else x for x in df['length_of_service']]
```

Handling Outliers

With the help of boxplot, outliers are visualized (refer activity 3 univariate analysis). And here we are going to find upper bound and lower bound of Na_to_K feature with some mathematical formula.

- To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.
- If outliers are removed, we lose more data. It will impact model performance.
- Here removing outliers is impossible. So, the capping technique is used on outliers.
- Capping: Replacing the outliers with upper bound values.



```
# Handling outliers

q1 = np.quantile(df['length_of_service'],0.25)
q3 = np.quantile(df['length_of_service'],0.75)

IQR = q3-q1

upperBound = (1.5*IQR)+q3
lowerBound = (1.5*IQR)-q1

print('q1 :',q1)
print('q3 :',q3)
print('IQR :',IQR)
print('Upper Bound :',upperBound)
print('Lower Bound :',lowerBound)
print('Skewed data :',len(df[df['length_of_service']>upperBound]))
```

```
q1 : 3.0
q3 : 7.0
IQR : 4.0
Upper Bound : 13.0
Lower Bound : 3.0
Skewed data : 3489
```

```
""" Here outliers can't be removrd. employee with higher length of services has higher promotion percentage.
    So, capping is done on this feature."""

pd.crosstab([df['length_of_service']>upperBound],df['is_promoted'])
```

is_promoted	0	1
length_of_service		
False	46885	4432
True	3255	234

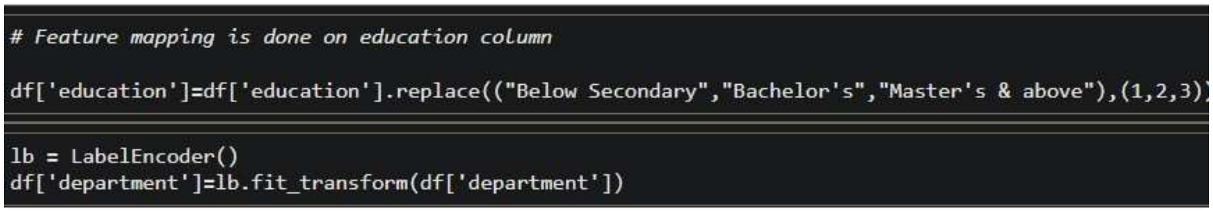
```
# Capping

df['length_of_service']=[upperBound if x>upperBound else x for x in df['length_of_service']]
```

Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using feature mapping and label encoding.

- In our project, categorical features are education and department feature. Feature mapping on education is done by replace() function.
- Label encoder is initialized and department feature is passed as parameter for fit_transform() function. Label encoding uses alphabetical ordering. In department feature we have 9 categories. Those categories are labelled in alphabetical order.



```
# Feature mapping is done on education column

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

lb = LabelEncoder()
df['department']=lb.fit_transform(df['department'])
```

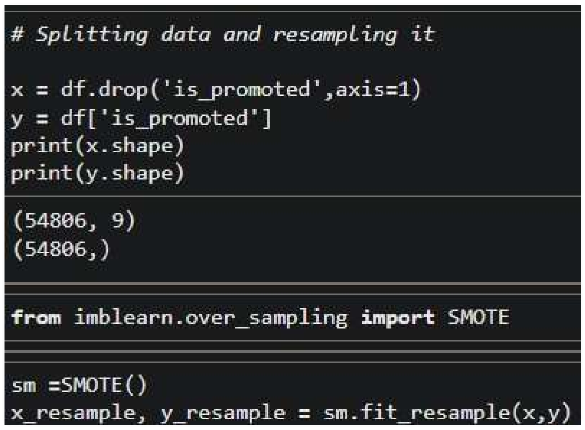
Handling Imbalanced Data

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 to lean more about SMOTE



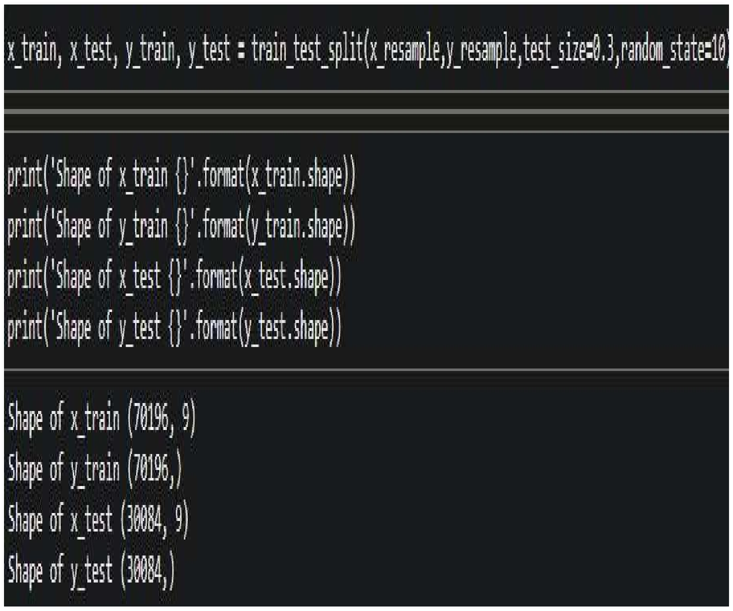
```
# Splitting data and resampling it

x = df.drop('is_promoted',axis=1)
y = df['is_promoted']
print(x.shape)
print(y.shape)
```

```
(54806, 9)
(54806,)
```

```
from imblearn.over_sampling import SMOTE
```

```
sm =SMOTE()
x_resample, y_resample = sm.fit_resample(x,y)
```

- Refer the below diagram to visualize the result of smote technique.



```
x_train, x_test, y_train, y_test = train_test_split(x_resample,y_resample,test_size=0.3,random_state=10)
```

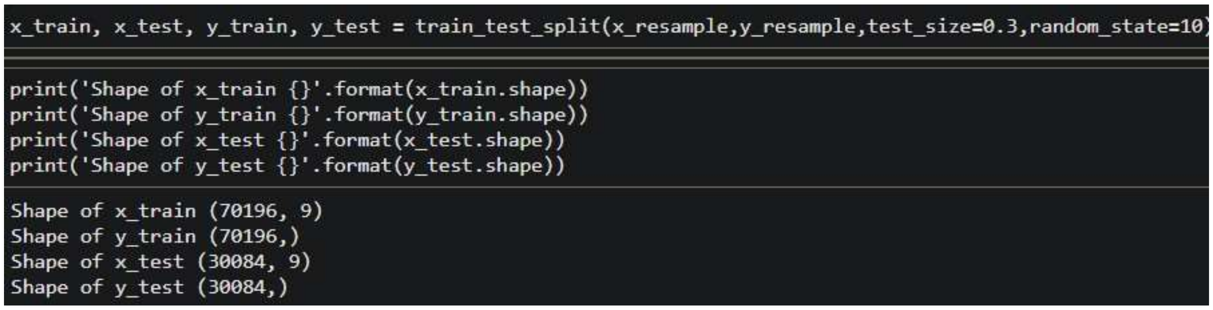
```
print('Shape of x_train {}'.format(x_train.shape))
print('Shape of y_train {}'.format(y_train.shape))
print('Shape of x_test {}'.format(x_test.shape))
print('Shape of y_test {}'.format(y_test.shape))
```

```
Shape of x_train (70196, 9)
Shape of y_train (70196,)
Shape of x_test (30084, 9)
Shape of y_test (30084,)
```

Splitting Data Into Train And Test

Now let's split the Dataset into train and test sets. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x_resample, y_resample, test_size, random_state.

For deep understanding refer this link



```
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))
```

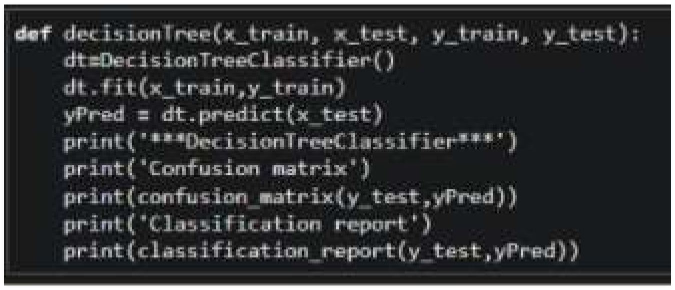
```
Shape of x_train (70196, 9)
Shape of y_train (70196,)
Shape of x_test (30084, 9)
Shape of y_test (30084,)
```

Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance. To evaluate the performance confusion matrix and classification report is used.

Decision Tree Model

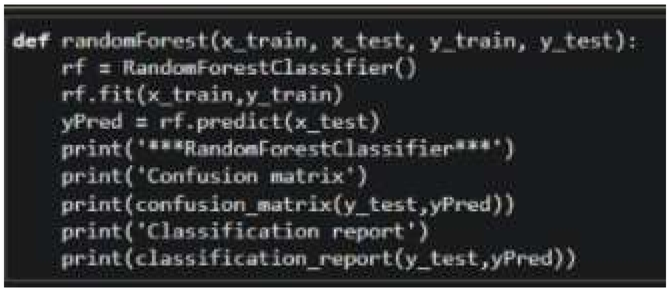
A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.



```
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))
```

Random Forest Model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

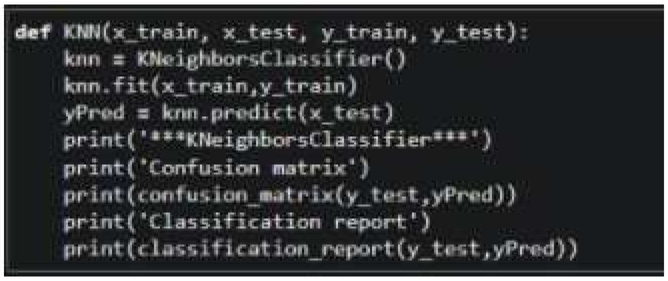


```
def randomForest(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

KNN Model

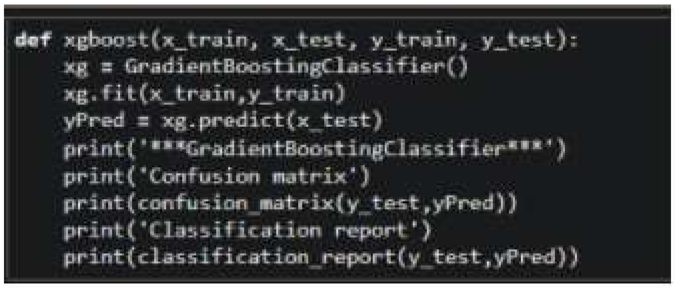
A function named KNN is created and train and test data are passed as the parameters.

Inside the function, KNeighborsClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.


```
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))
```

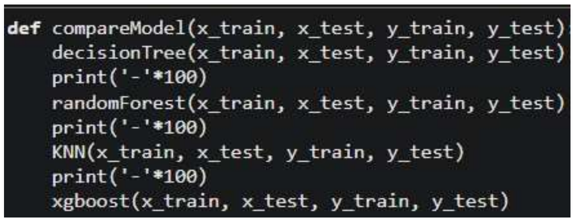
Xgboost Model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, GradientBoostingClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.


```
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))
```

Compare The Model

For comparing the above four models compareModel function is defined.


```
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)
```

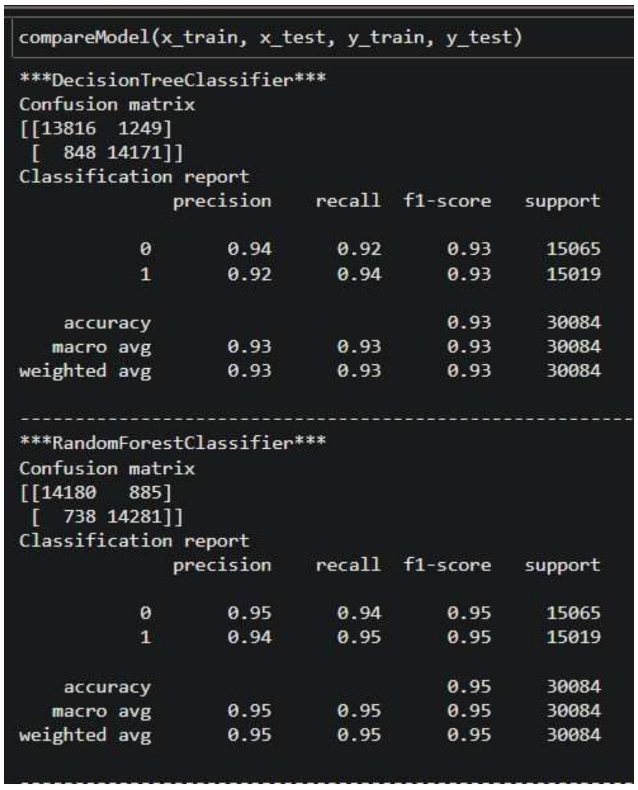
After calling the function, the results of models are displayed as output. From the four model random forest and decision tree is performing well. From the below image, we can see the accuracy of the models. Both models have 95% and 93% accuracy. Random forest model accuracy is high. And from confusion matrix random forest has higher number of true positive and true negative. So, here random forest is selected and evaluated with cross validation. Additionally, we can tune the model with hyper parameter tuning techniques.

But here we have not used it.

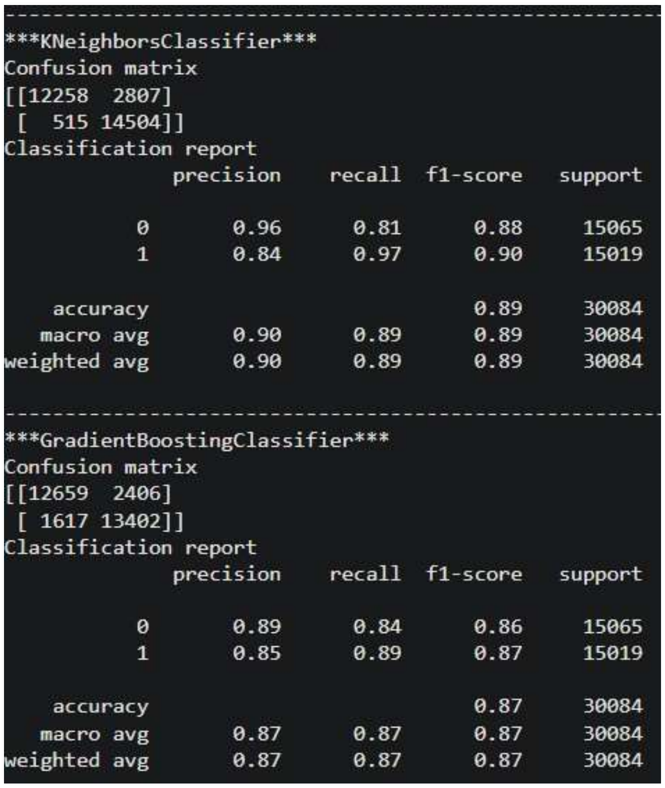
To get deep knowledge in confusion matrix and classification report refer the below links:

Link1

Link2



```
compareModel(x_train, x_test, y_train, y_test)

***DecisionTreeClassifier***
Confusion matrix
[[13816  1249]
 [  848 14171]]
Classification report
              precision    recall  f1-score   support

           0       0.94      0.92      0.93     15065
           1       0.92      0.94      0.93     15019

    accuracy                           0.93     30084
   macro avg       0.93      0.93      0.93     30084
weighted avg       0.93      0.93      0.93     30084

-------------------------------------------------------
***RandomForestClassifier***
Confusion matrix
[[14180   885]
 [  738 14281]]
Classification report
              precision    recall  f1-score   support

           0       0.95      0.94      0.95     15065
           1       0.94      0.95      0.95     15019

    accuracy                           0.95     30084
   macro avg       0.95      0.95      0.95     30084
weighted avg       0.95      0.95      0.95     30084

-------------------------------------------------------
```

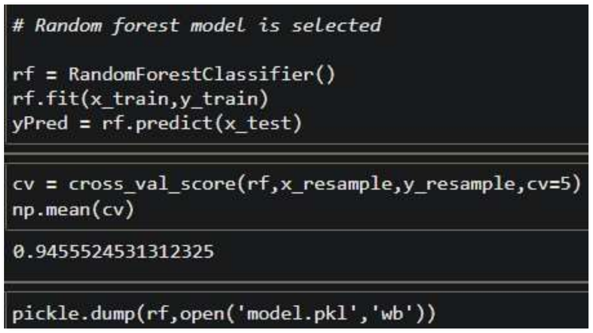


```
--------------------------------------------------
***KNeighborsClassifier***
Confusion matrix
[[12258  2807]
 [  515 14504]]
Classification report
              precision    recall  f1-score   support

           0       0.96      0.81      0.88     15065
           1       0.84      0.97      0.90     15019

    accuracy                           0.89     30084
   macro avg       0.90      0.89      0.89     30084
weighted avg       0.90      0.89      0.89     30084

--------------------------------------------------
***GradientBoostingClassifier***
Confusion matrix
[[12659  2406]
 [ 1617 13402]]
Classification report
              precision    recall  f1-score   support

           0       0.89      0.84      0.86     15065
           1       0.85      0.89      0.87     15019

    accuracy                           0.87     30084
   macro avg       0.87      0.87      0.87     30084
weighted avg       0.87      0.87      0.87     30084
```

Evaluating Performance Of The Model And Saving The Model

From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x_resample, y_resample, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

Note: To understand cross validation, refer this link.



```
# Random forest model is selected

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)
```

```
0.9455524531312325
```

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

Application Building

In this section, we will be 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

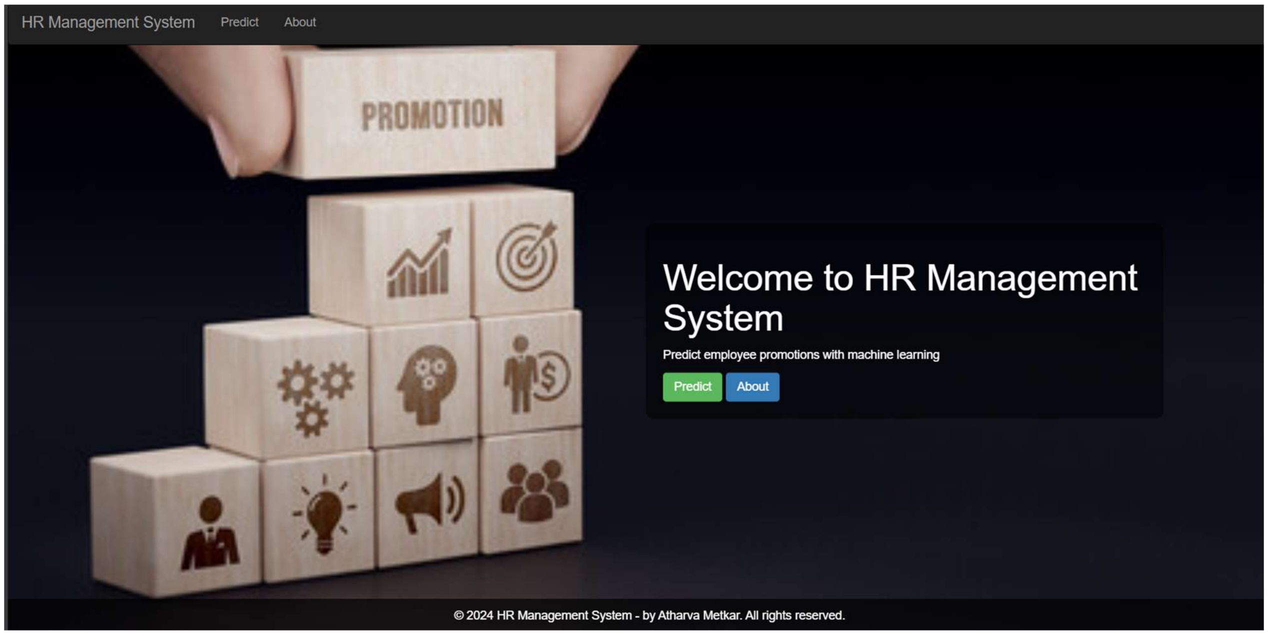
Building Html Pages

For this project, create three HTML files namely

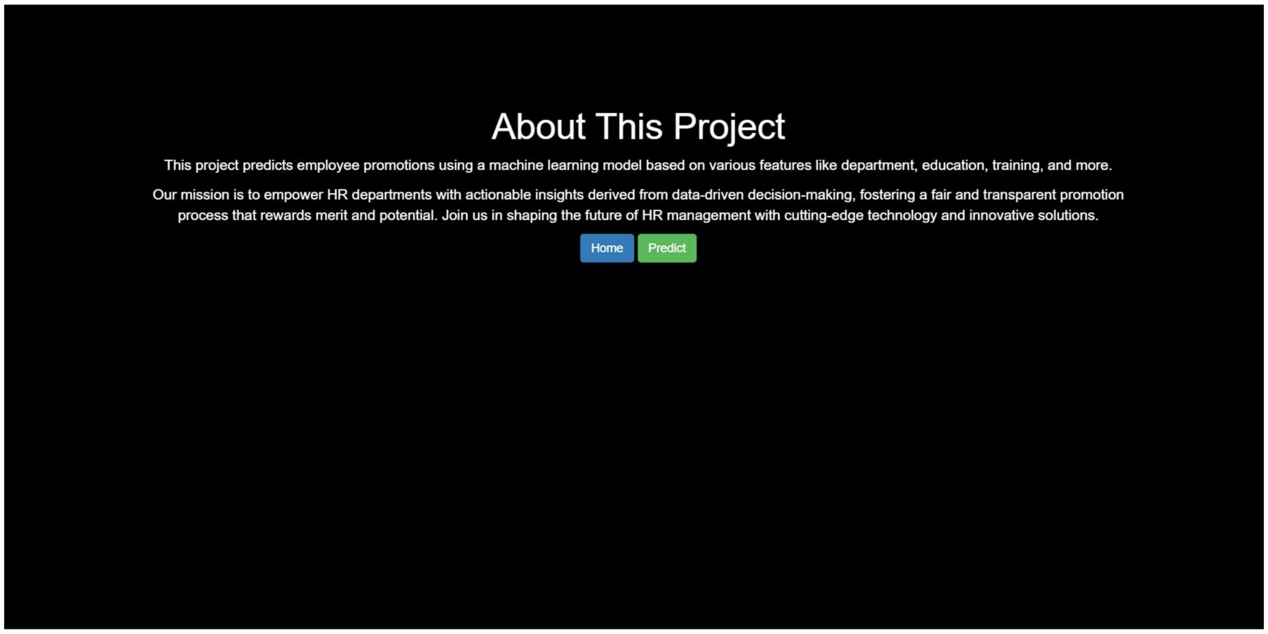
- home.html ● about.html
- predict.html
- result.html

and save them in templates folder.

Let’s see how our home.html page looks like:


HR Management System
Predict
About
PROMOTION
Welcome to HR Management System
Predict employee promotions with machine learning
Predict
About
© 2024 HR Management System - by Atharva Metkar. All rights reserved.

Let’s see how our about.html page looks like:



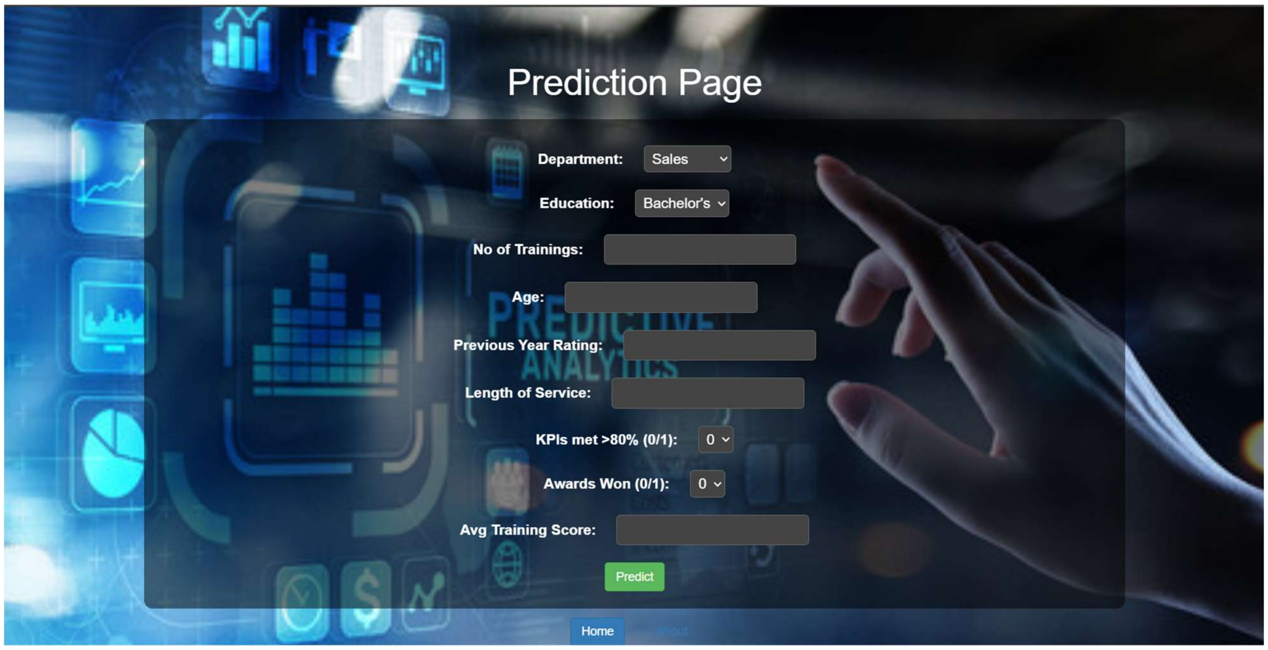
About This Project

This project predicts employee promotions using a machine learning model based on various features like department, education, training, and more.

Our mission is to empower HR departments with actionable insights derived from data-driven decision-making, fostering a fair and transparent promotion process that rewards merit and potential. Join us in shaping the future of HR management with cutting-edge technology and innovative solutions.

Now when you click on predict button you will get redirected to predict.html

Lets look how our predict.html file looks like:

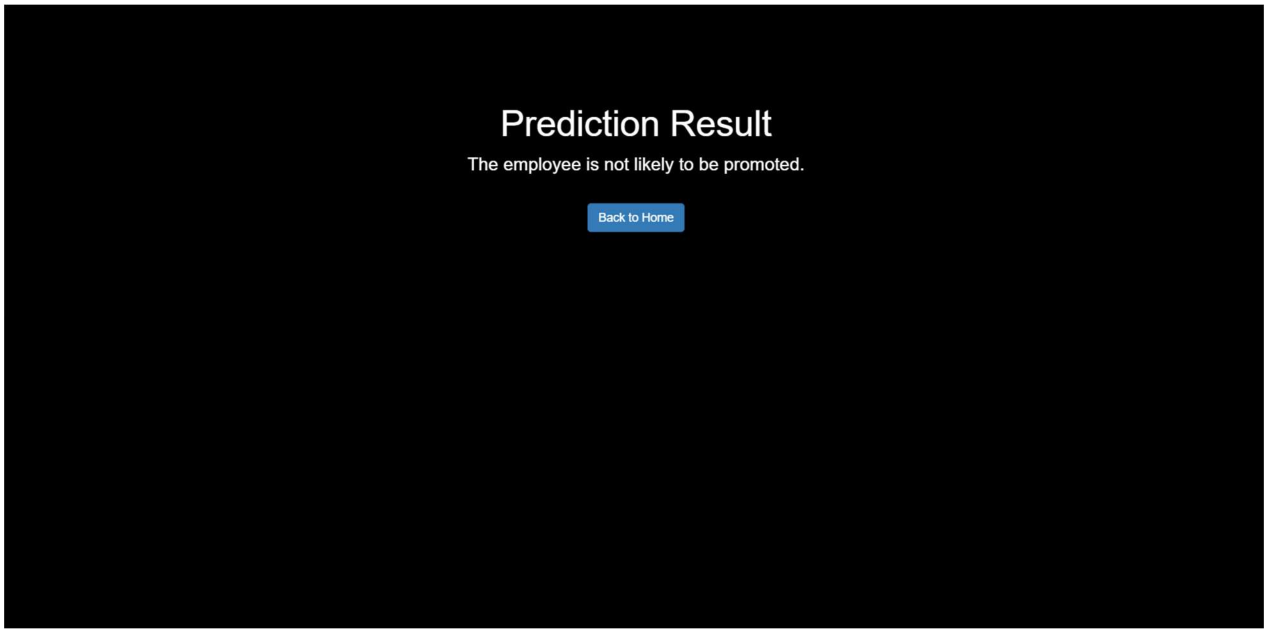


Prediction Page

Department:	Sales
Education:	Bachelor's
No of Trainings:	
Age:	
Previous Year Rating:	
Length of Service:	
KPIs met >80% (0/1):	0
Awards Won (0/1):	0
Avg Training Score:	

Now when you click on predict button you will get redirected to result.html

Lets look how our result.html file looks like:


Prediction Result

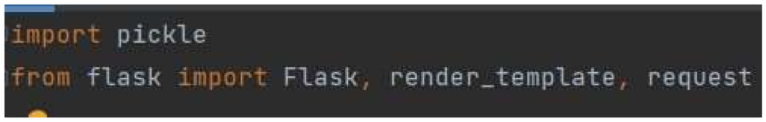
The employee is not likely to be promoted.

Build Python Code

Import the libraries

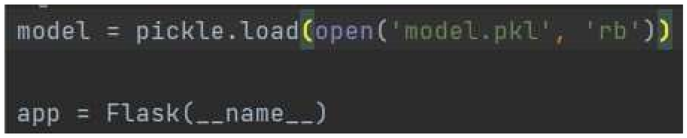
Pickle: Pickle is a module in Python used for serializing and de-serializing Python objects. Flask: Refer prior

knowledge section mentioned above.

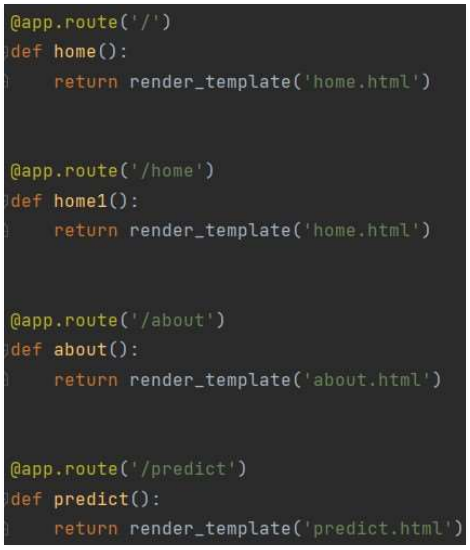

```
import pickle
from flask import Flask, render_template, request
```

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI

application. Flask constructor takes the name of the current module (__name__) as argument.


```
model = pickle.load(open('model.pkl', 'rb'))

app = Flask(__name__)
```

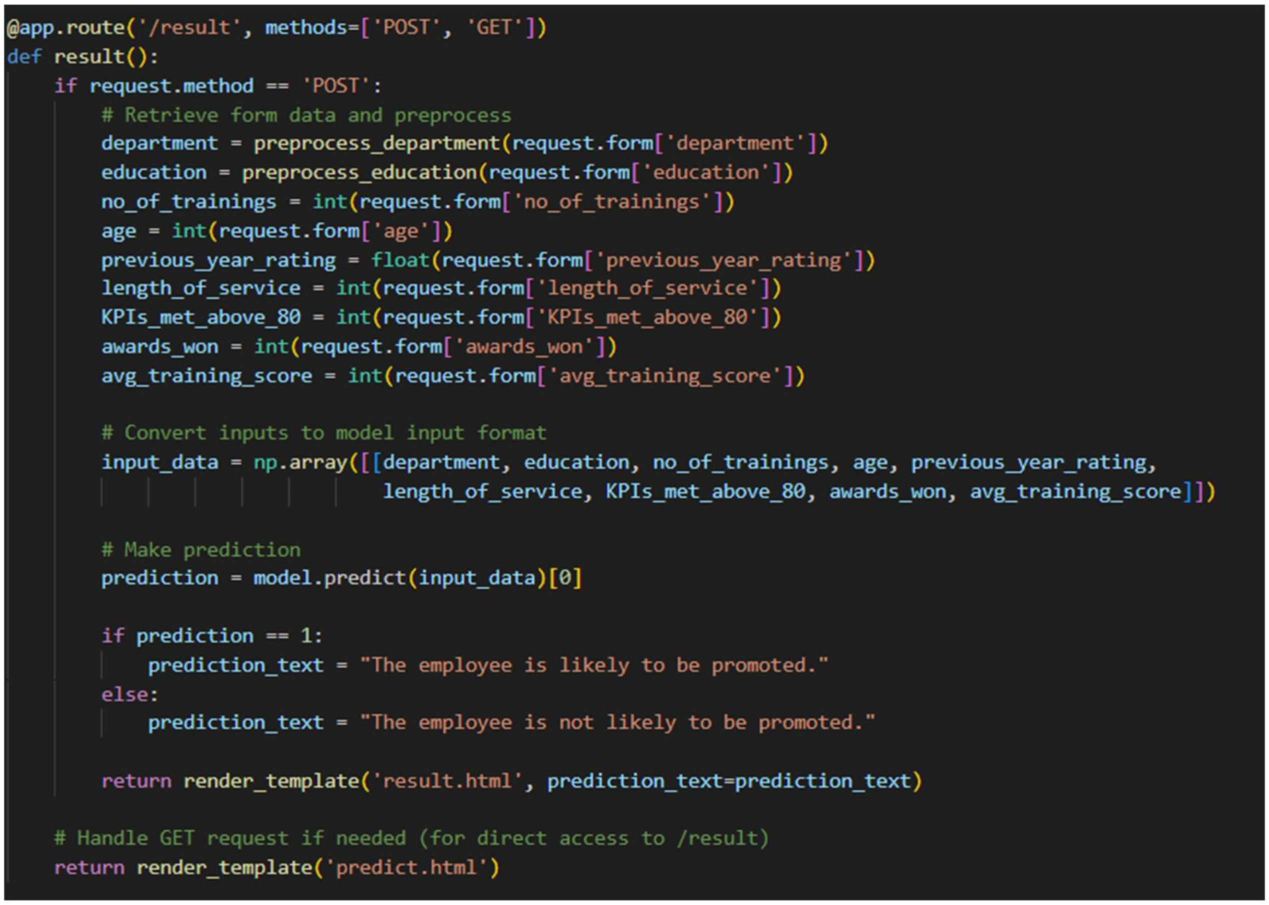
Render HTML page:



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


@app.route('/home')
def home1():
    return render_template('home.html')


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


@app.route('/predict')
def predict():
    return render_template('predict.html')
```

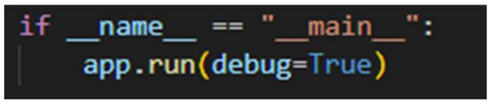
Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the predict html page the values can be retrieved using POST Method.


```python
@app.route('/result', methods=['POST', 'GET'])
def result():
    if request.method == 'POST':
        # Retrieve form data and preprocess
        department = preprocess_department(request.form['department'])
        education = preprocess_education(request.form['education'])
        no_of_trainings = int(request.form['no_of_trainings'])
        age = int(request.form['age'])
        previous_year_rating = float(request.form['previous_year_rating'])
        length_of_service = int(request.form['length_of_service'])
        KPIs_met_above_80 = int(request.form['KPIs_met_above_80'])
        awards_won = int(request.form['awards_won'])
        avg_training_score = int(request.form['avg_training_score'])

        # Convert inputs to model input format
        input_data = np.array([[department, education, no_of_trainings, age, previous_year_rating,
                                length_of_service, KPIs_met_above_80, awards_won, avg_training_score]])

        # Make prediction
        prediction = model.predict(input_data)[0]

        if prediction == 1:
            prediction_text = "The employee is likely to be promoted."
        else:
            prediction_text = "The employee is not likely to be promoted."

        return render_template('result.html', prediction_text=prediction_text)

    # Handle GET request if needed (for direct access to /result)
    return render_template('predict.html')
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the submit.html page earlier.

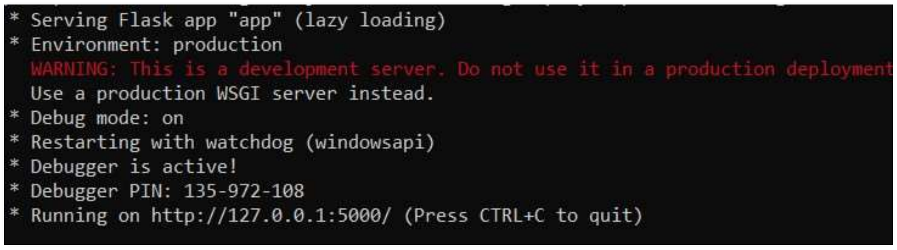
Main Function:


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

Run The Application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.

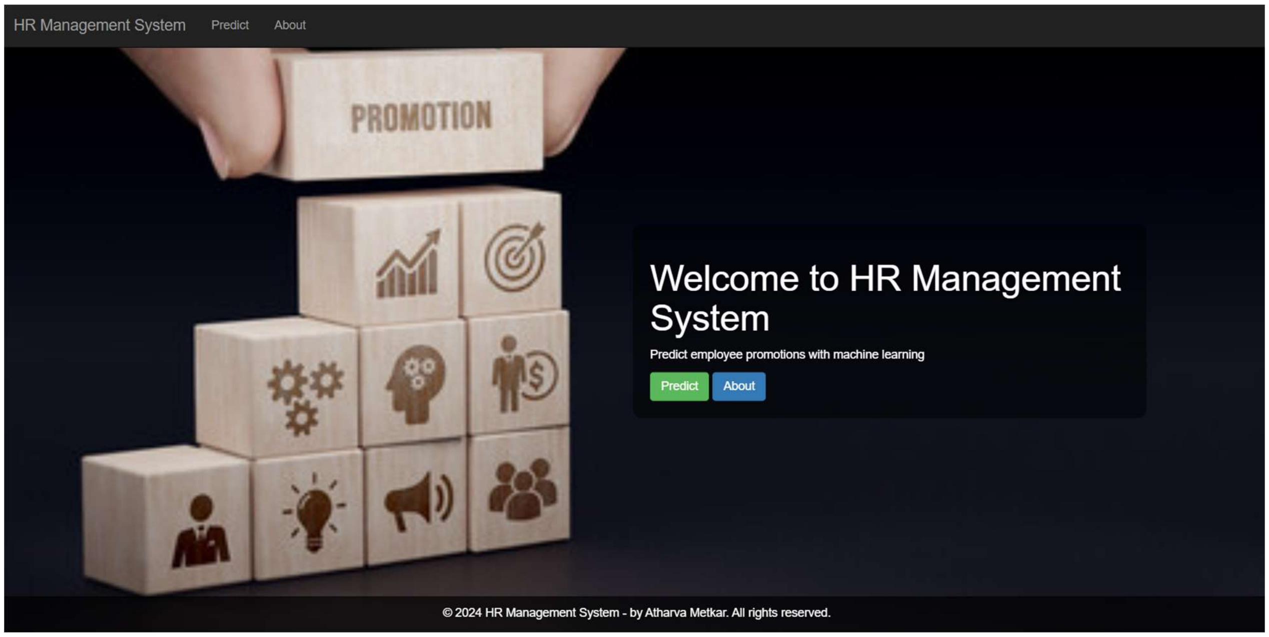
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 135-972-108
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now paste the URL on the browser, you will redirect to home.html page. Let's look our home page

9.CONCLUSION Output



HR Management System Predict About

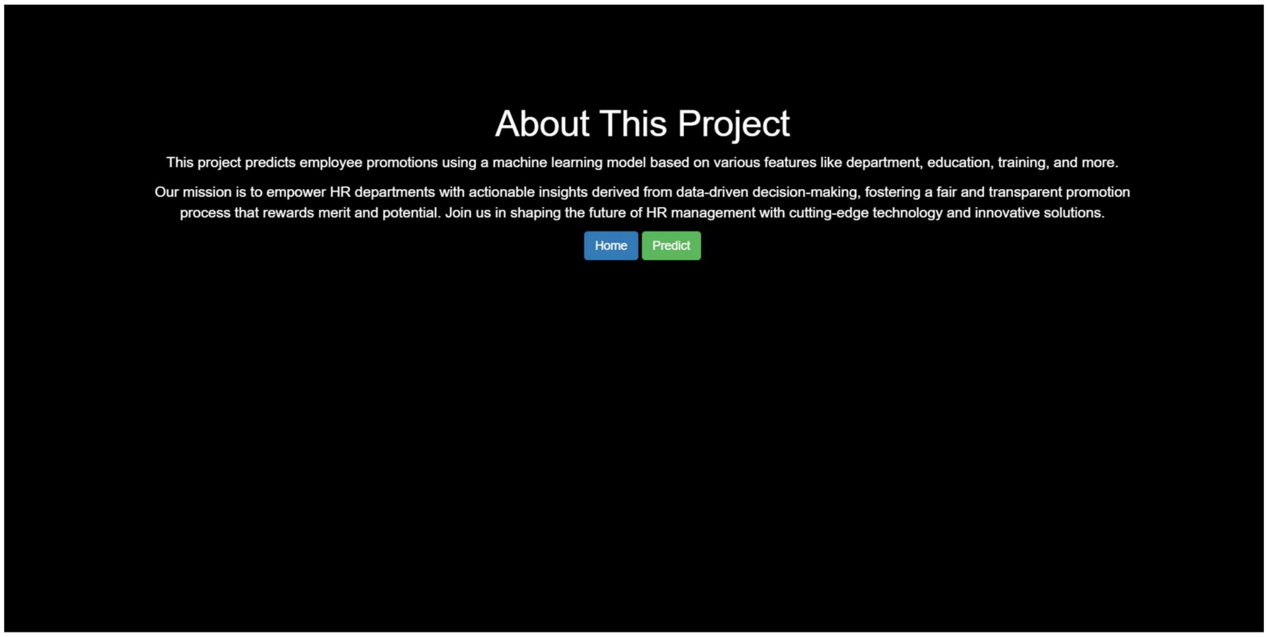
Welcome to HR Management System

Predict employee promotions with machine learning

Predict About

© 2024 HR Management System - by Atharva Metkar. All rights reserved.

To know about the project click on About button .Now it will redirect to about.html page



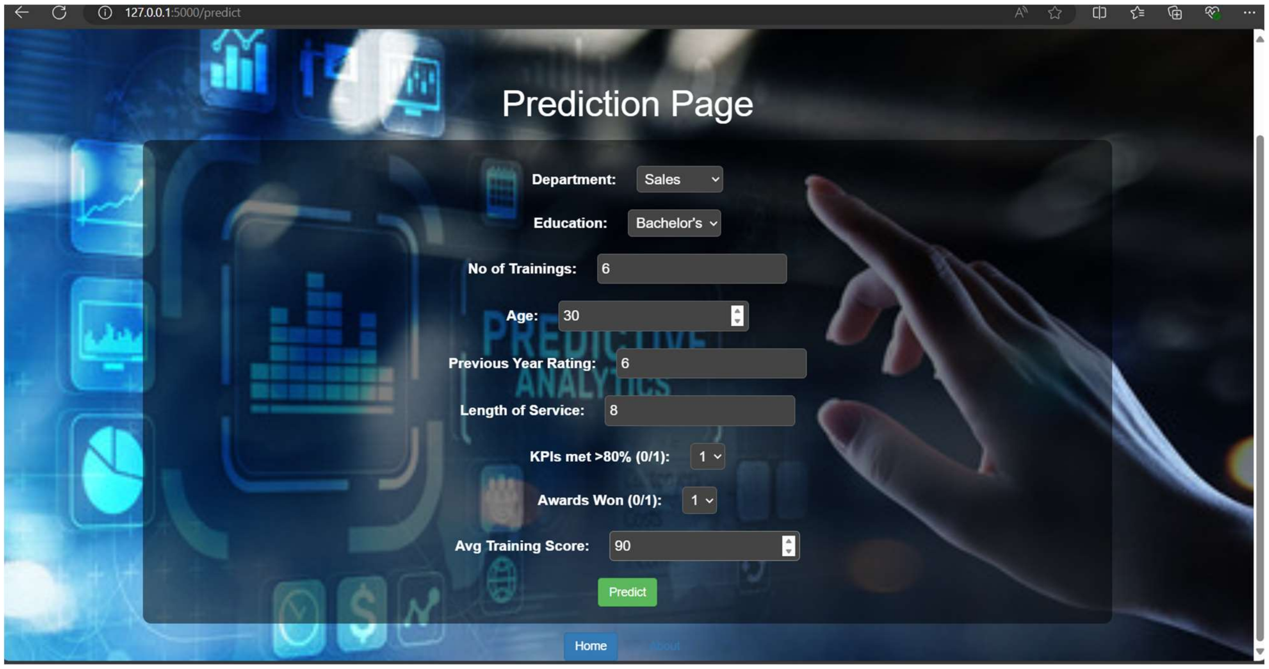
About This Project

This project predicts employee promotions using a machine learning model based on various features like department, education, training, and more.

Our mission is to empower HR departments with actionable insights derived from data-driven decision-making, fostering a fair and transparent promotion process that rewards merit and potential. Join us in shaping the future of HR management with cutting-edge technology and innovative solutions.

To predict your promotion click on predict. It will redirect to predict.html page. Now give your inputs and click on predict button. Output will be displayed in result.html page.

Input 1:

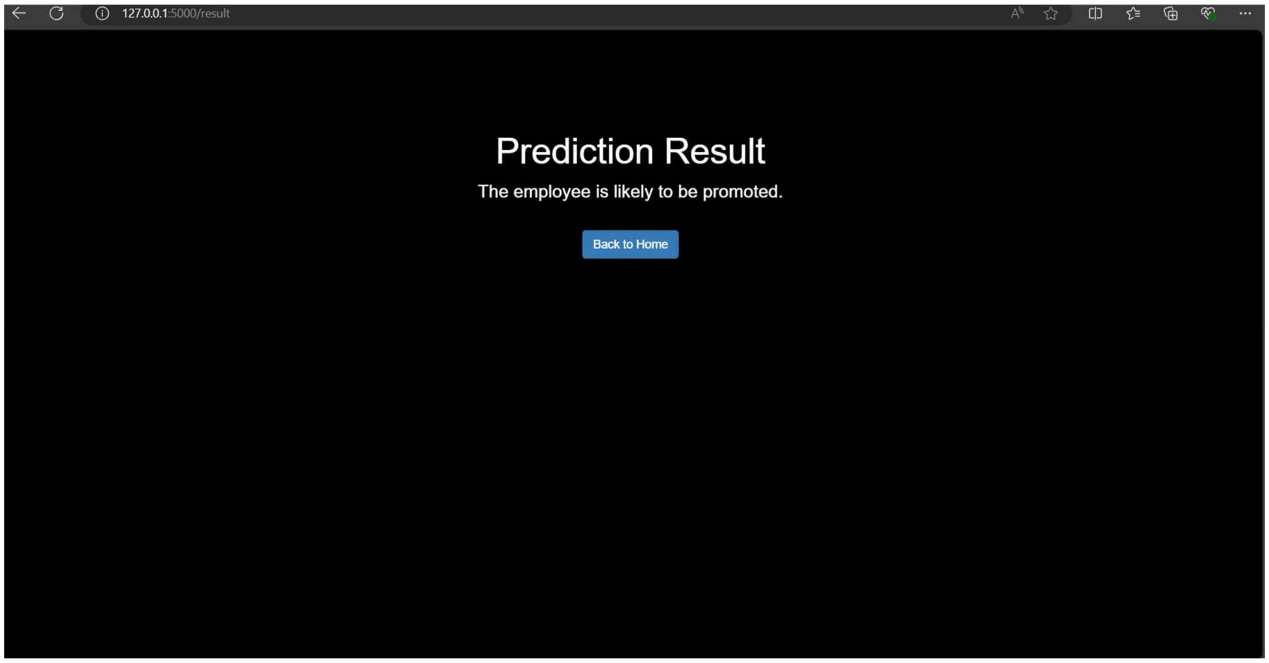


127.0.0.1:5000/predict

Prediction Page

Department:	Sales
Education:	Bachelor's
No of Trainings:	6
Age:	30
Previous Year Rating:	6
Length of Service:	8
KPIs met >80% (0/1):	1
Awards Won (0/1):	1
Avg Training Score:	90

Output 1:

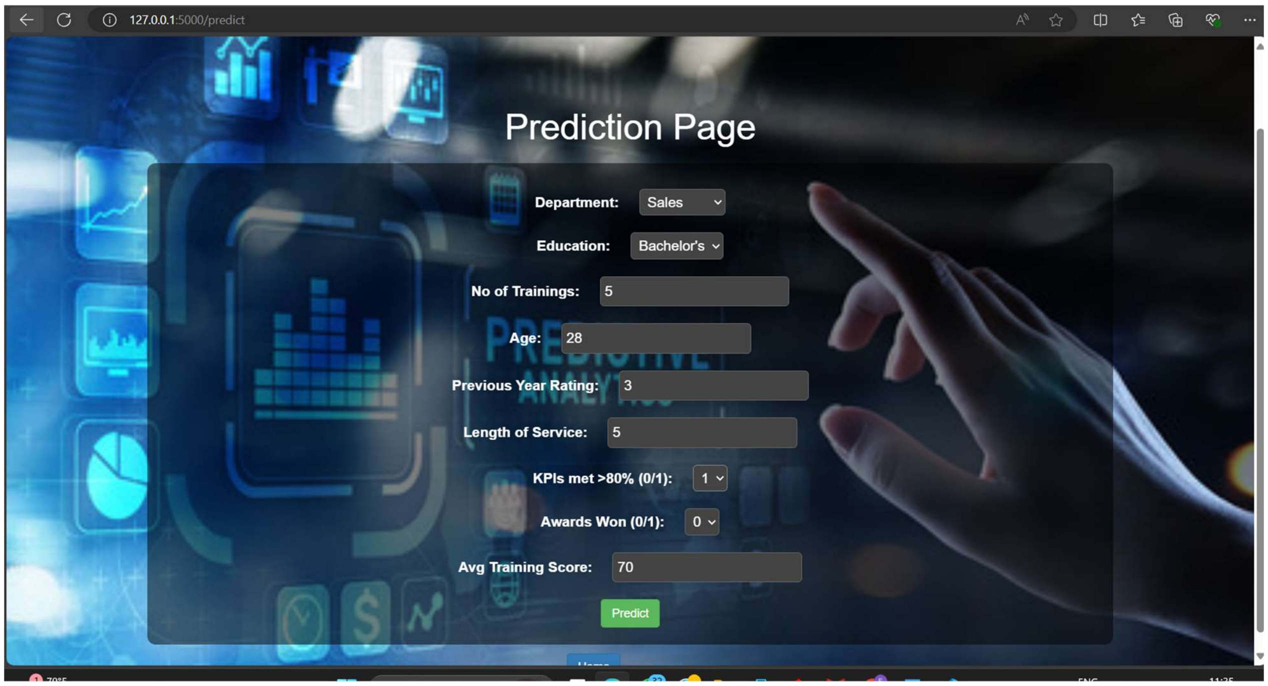


127.0.0.1:5000/result

Prediction Result

The employee is likely to be promoted.

Input 2:



127.0.0.1:5000/predict

Prediction Page

Department: Sales

Education: Bachelor's

No of Trainings: 5

Age: 28

Previous Year Rating: 3

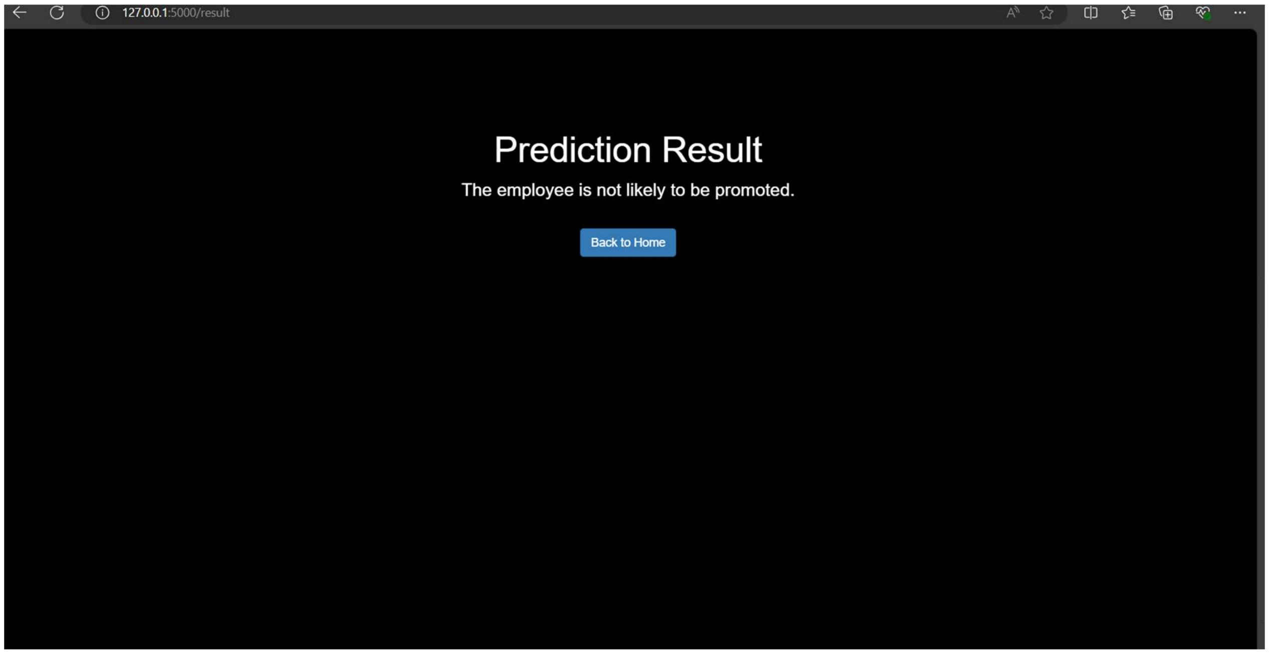
Length of Service: 5

KPIs met >80% (0/1): 1

Awards Won (0/1): 0

Avg Training Score: 70

Output 2:



Prediction Result

The employee is not likely to be promoted.