HUMAN RESOURCE MANAGEMENT: PREDICTING EMPLOYEE PROMOTIONS USING MACHINE LEARNING

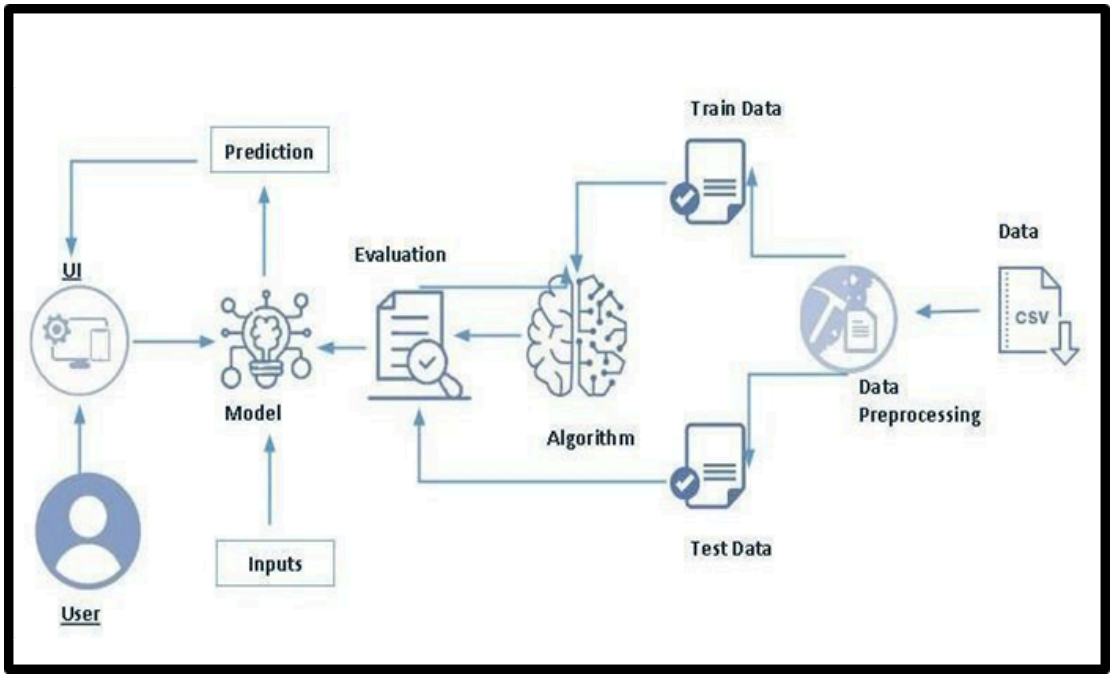
SmartInternz

www.smartinternz.com

Human Resource Management: Predicting Employee Promotions Using MachineLearning

Promotions are granted to recognize an employee's contribution and to assign higher responsibilities. A promotion is a key part of employee growth and company development, but many times, these decisions are made subjectively or inconsistently. It becomes problematic when deserving employees are overlooked or when promotions are influenced by bias. This leads to dissatisfaction and a decline in performance. So, it is important to have a system that can objectively predict promotion eligibility. Many factors impact such decisions, and manual evaluation often lacks fairness. The main purpose of the Employee Promotion Prediction system is to forecast if an employee is eligible for promotion based on various input parameters and performance metrics.

Technical Architecture:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analysesthe input the prediction is showcased on the UI

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

- Define Problem / Problem Understanding
    - Specify the businessproblem
    - Business requirements
    - Literature Survey
    - Social or Business Impact.
- Data Collection &Preparation
    - Collect the dataset
    - Data Preparation
- Exploratory Data Analysis
    - Descriptive statistical
    - Visual Analysis
- Model Building
    - Training the model in multiplealgorithms
    - Testing the model
- Performance Testing & Hyperparameter Tuning
    - Testing model with multiple evaluation metrics
    - Comparing model accuracybefore & after applying hyperparameter tuning
- Model Deployment
    - Save the best model
    - Integrate with Web Framework

- Project Demonstration &Documentation
    - Record explanation Videofor project end to end solution
    - Project Documentation-Step by step projectdevelopment procedure

Milestone 1: Define Problem/ Problem Understanding

Activity 1: Specify the business problem

Refer ProjectDescription

Activity 2: Business requirements

An employee promotion prediction project can have several business requirements, based on the overall goals and outcomes expected. Some key requirements may include:

- Accurate and up-to-date information:The system should be built using recent and valid employee records to ensure the predictions are aligned with real performance indicators.
- Scalability: The model should be adaptable to changes in employee roles, new promotion criteria, and organizational restructuring.
- Compliance: The system must align with HR policies, labor laws, and ethical standards related to employee evaluation and promotion.
- User-friendly interface: The prediction system should be simple and accessible for HR teams and decision-makers to understand and utilize effectively.

Activity 3: Literature Survey(Student Will Write)

A literature survey for an employee promotion prediction project would involve studying and reviewing existing research papers, articles, and publications on the topic of promotion systems. The goal of the survey is to collect insights on current evaluation practices, their benefits and limitations, and to identify any gaps that this project can address. The review would also explore the various machine learning methods used in earlier promotion prediction models, along with the relevant datasets, algorithms, and outcomes that could help shape the development and design of the current system.

Activity 4: Social or Business Impact.

Social Impact:- Improved workplace transparency: By offering data-driven and unbiased promotion predictions, the project can support fair decision-making in organizations, leading to better employee satisfaction, motivation, and trust in the evaluation process.

Business Model/Impact :- Optimized talent management: By analyzing key performance metrics and employee history, the project can assist organizations in identifying high-potential individuals, improving workforce planning, and enhancing overall productivity.

Milestone 2: Data Collection & Preparation

ML depends heavilyon data. It is the most crucialaspect that makesalgorithm training possible.

So, this section allows you to download the required dataset.

Activity 1: Collect 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: https://drive.google.com/file/d/14eQR1VWHwuomPaXdKulkZEpSstvav8Db/view?usp=sharing

As the dataset is downloaded. Let us read and understand the data properlywith the help of some visualisation techniques and some analysing techniques.

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

Activity 1.1: Importing the libraries

Import the necessary librariesas shown in the image

```
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)
```

Activity 1.2: Read the Dataset

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

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

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.

```
#Checking for null values
df.isnull().sum()
```

	0
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

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitablefor training the machine learningmodel as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Drop unwanted Features
- Remove Negative Data
- Handling missing values
- Handling Outliers

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

Activity 2.1: 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.

```
[ ]  #Dropping unwanted features
     """ 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)
```

Activity 2.2: 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.

```
[ ]  #Removing Negative Data
     #Finding the employee who got promoted even in poor performance. It affects the 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)
```

Activity 2.3: Handling missingvalues

- 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])
```

```
education
Bachelor's          36669
Master's & above    14925
Below Secondary       805
Name: count, 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])
```

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

Activity 2.4: Handling Outliers

With the help of boxplot, outliers are visualized (refer activity 3 univariate analysis). And here we are going t0 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 removed. 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']]
```

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

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.

Note: In this case it was done before Preprocessing.

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

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	awards_won?	avg_training_score	is_promoted
count	54808.000000	54808	54808	52399	54808	54808	54808.000000	54808.000000	50684.000000	54808.000000	54808.000000	54808.000000	54808.000000	54808.000000
unique	NaN	9	34	3	2	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	Sales & Marketing	region_2	Bachelor's	m	other	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	16840	12343	36669	38496	30446	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	39195.830627	NaN	NaN	NaN	NaN	NaN	1.253011	34.803915	3.329256	5.865512	0.351974	0.023172	63.386750	0.085170
std	22586.581449	NaN	NaN	NaN	NaN	NaN	0.609264	7.660169	1.259993	4.265094	0.477590	0.150450	13.371559	0.279137
min	1.000000	NaN	NaN	NaN	NaN	NaN	1.000000	20.000000	1.000000	1.000000	0.000000	0.000000	39.000000	0.000000
25%	19669.750000	NaN	NaN	NaN	NaN	NaN	1.000000	29.000000	3.000000	3.000000	0.000000	0.000000	51.000000	0.000000
50%	39225.500000	NaN	NaN	NaN	NaN	NaN	1.000000	33.000000	3.000000	5.000000	0.000000	0.000000	60.000000	0.000000
75%	58730.500000	NaN	NaN	NaN	NaN	NaN	1.000000	39.000000	4.000000	7.000000	1.000000	0.000000	76.000000	0.000000
max	78298.000000	NaN	NaN	NaN	NaN	NaN	10.000000	60.000000	5.000000	37.000000	1.000000	1.000000	99.000000	1.000000

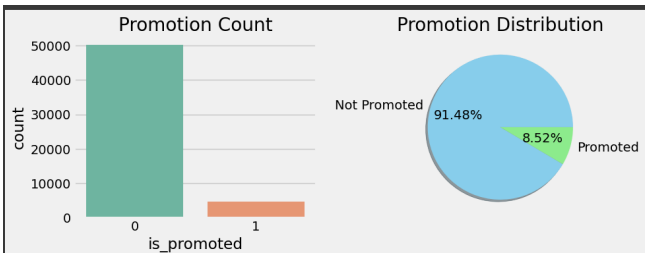
Activity 2: Visual analysis

Visual analysis is the processof using visualrepresentations, such as charts, plots, and graphs, to exploreand understand data.It is a way to quickly identifypatterns, trends, and outliersin the data, which can help to gain insightsand make informed decisions.

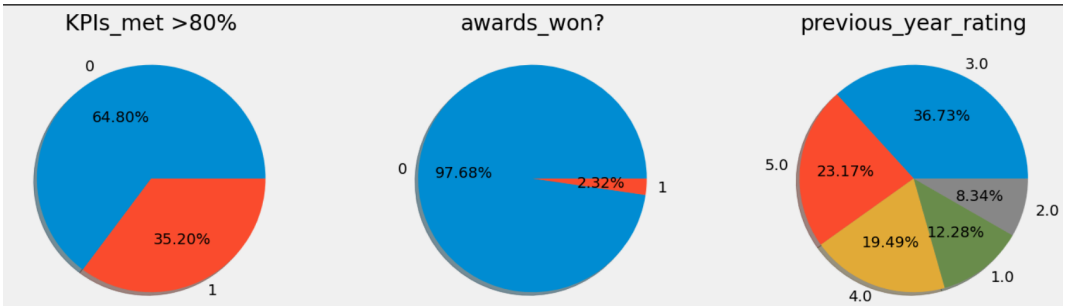
Activity 2.1: Univariate analysis

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

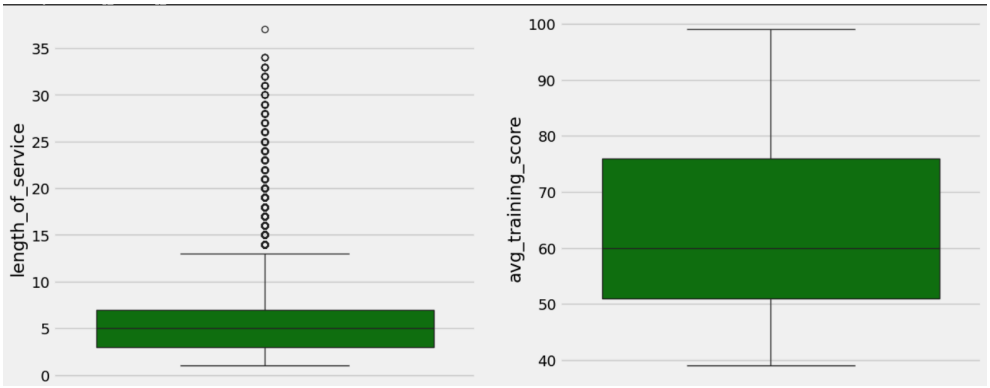
- Count plot and pie plot are used on the target variable. From the below image, we identified our data is imbalanced. 91% of the employees are not promoted. To get better model performance, imbalanced data should be converted to balanced data. Handling imbalanced data will be discussed on data pre processing.



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



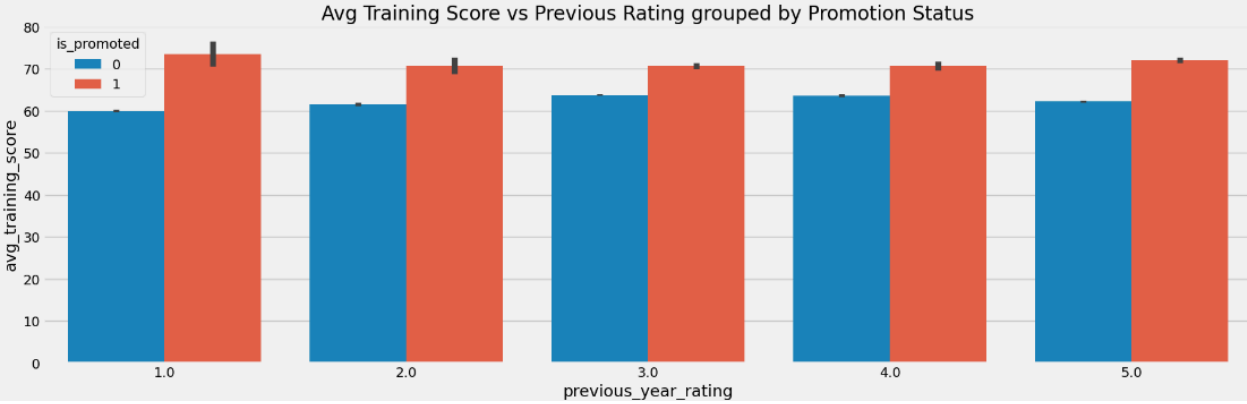
- Box plot is used on the length of service and average training score feature. Length of services feature has more outliers. The model should not be built without handling the outliers. Here, outliers are handled by the capping method. Capping will be discussed on data pre-processing.



Activity 2.3: 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%)



Encoding the Categorical Features:

- The categorical Features are can'tbe passed directlyto the Machine Learning Model.So we convert them into Numerical data based on their order. This Technique is called Encoding.
- Here we are importing Label Encoder from the SklearnLibrary.
- Here we are applying fit_transform to transform the categorical featuresto numerical features.
- 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.

```
[ ]  #Handling Categorical Values

     #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'])
```

Splitting data into trainand 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.

```
[ ]  #Splitting the data into train and test
     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 y_train {}'.format(x_test.shape))
     print('Shape of y_train {}'.format(y_test.shape))
```

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

Handling Imbalanced dataset

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.
- Creating a variable and initialize smote() function. Now resampling is done with fit_resample() function

```
[ ]  #SMOTE for Imbalanced Data
     #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)
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

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.

Activity 1.1: 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.

```
[ ]  #Decision Tree
     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))
```

Activity 1.2: 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.

```
[ ]  #Random Forest
     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))
```

Activity 1.3: 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.

```
[ ]  #KNN
     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))
```

Activity 1.4: 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.

```
[ ]  #XGBoost
     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))
```

Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with multipleevaluation metrics

Multiple evaluation metricsmeans evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

For comparing the above four models compareModel function is defined.

```
#Compare model function

def compareModel(x_train, x_test, y_train, y_test):
  decisionTree(x_train, x_test, y_train, y_test)
  print('-'*100)
  randomForest(x_train, x_test, y_train, y_test)
  print('-'*100)
  KNN(x_train, x_test, y_train, y_test)
  print('-'*100)
  xgboost(x_train, x_test, y_train, y_test)

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

After calling the function, the results of models are displayed as output.

```
***DecisionTreeClassifier***
Confusion matrix
[[13861  1204]
 [  885 14134]]
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
[[14207   858]
 [  782 14237]]
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
[[12293  2772]
 [  533 14486]]
Classification report
              precision    recall  f1-score   support

           0       0.96      0.82      0.88     15065
           1       0.84      0.96      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
[[12704  2361]
 [ 1673 13346]]
Classification report
              precision    recall  f1-score   support

           0       0.88      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
```

Activity 2: Comparing model accuracy before & after applying hyperparameter tuning (Hyperparameter tuning is optional.For this project it is not required.)

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.

```
[ ]  #Evaluating Performance of the model
     #Random Forest model is selected

     rf = RandomForestClassifier()
     rf.fit(x_train, y_train)
     yPred = rf.predict(x_test)
```

```
[ ]  #Cross Validation
     cv = cross_val_score(rf,x_resample,y_resample,cv=5)
     np.mean(cv)
```

```
     np.float64(0.9465396888711608)
```

Milestone 6: ModelDeployment

Activity 1: Save the best model

Saving the best model aftercomparing its performance using different evaluation metrics means selecting the model with the highest performance.This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future

```
[ ]  #Saving the model
     pickle.dump(rf, open('model.pkl','wb'))


[ ]  from google.colab import files
     files.download('model.pkl')
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is providedfor 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 server-side script
- Run the web application

Activity 2.1: BuildingHtml Page:

For this project, create three HTML files name

- home.html
- about.html
- predict.html
- submit.html

and save them in templates folder.

Activity 2.2: BuildPython code:

Import the libraries

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

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

Load the savedmodel. Importing the flask modulein 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.

Render HTML page:

```
@app.route('/')
@app.route('/home')
def home():
    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 usinga declared constructor to route to the HTML page whichwe have created earlier. In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser,the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```python
@app.route('/pred', methods=['POST'])
def pred():
    department = int(request.form['department'])
    education = int(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 = float(request.form['length_of_service'])
    KPIs = int(request.form['KPIs'])
    awards_won = int(request.form['awards_won'])
    avg_training_score = int(request.form['avg_training_score'])

    features = [[department, education, no_of_trainings, age, previous_year_rating,
                 length_of_service, KPIs, awards_won, avg_training_score]]

    prediction = model.predict(features)[0]

    if prediction == 0:
        text = "Sorry, you are not eligible for promotion"
    else:
        text = "Congratulations! You are eligible for promotion"

    return render_template('submit.html', predictionText=text)
```

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:

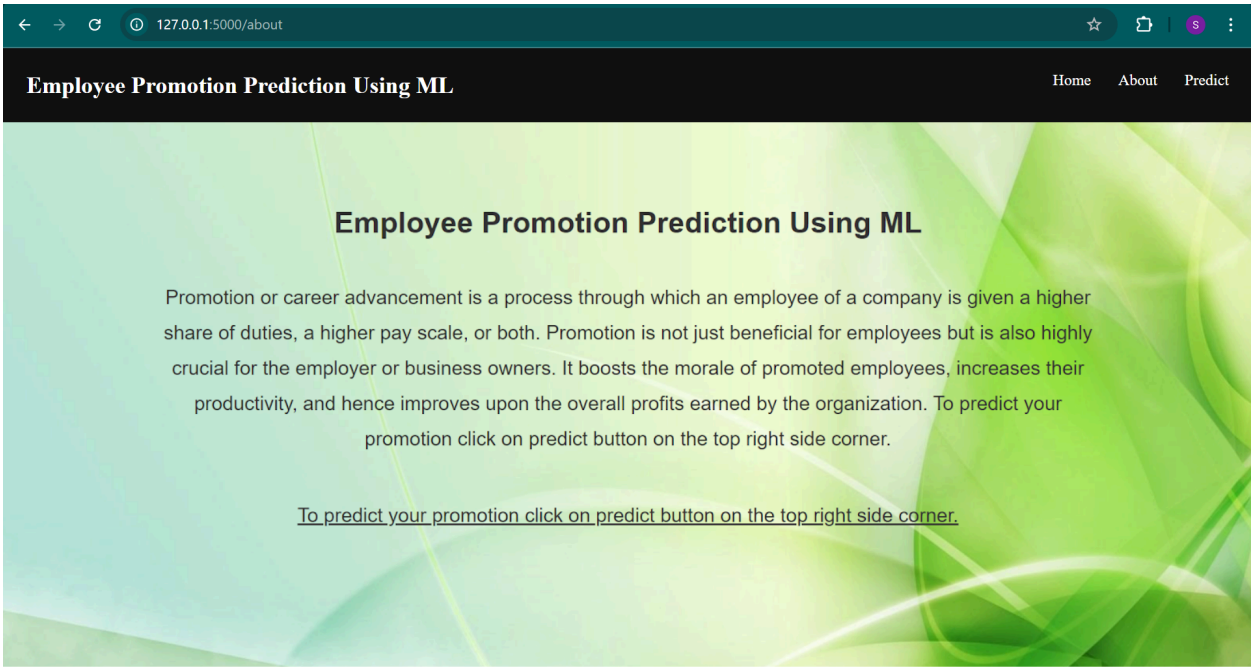
```python
if __name__ == '__main__':
    app.run(debug=True)
```

Activity 2.3: Run the web application

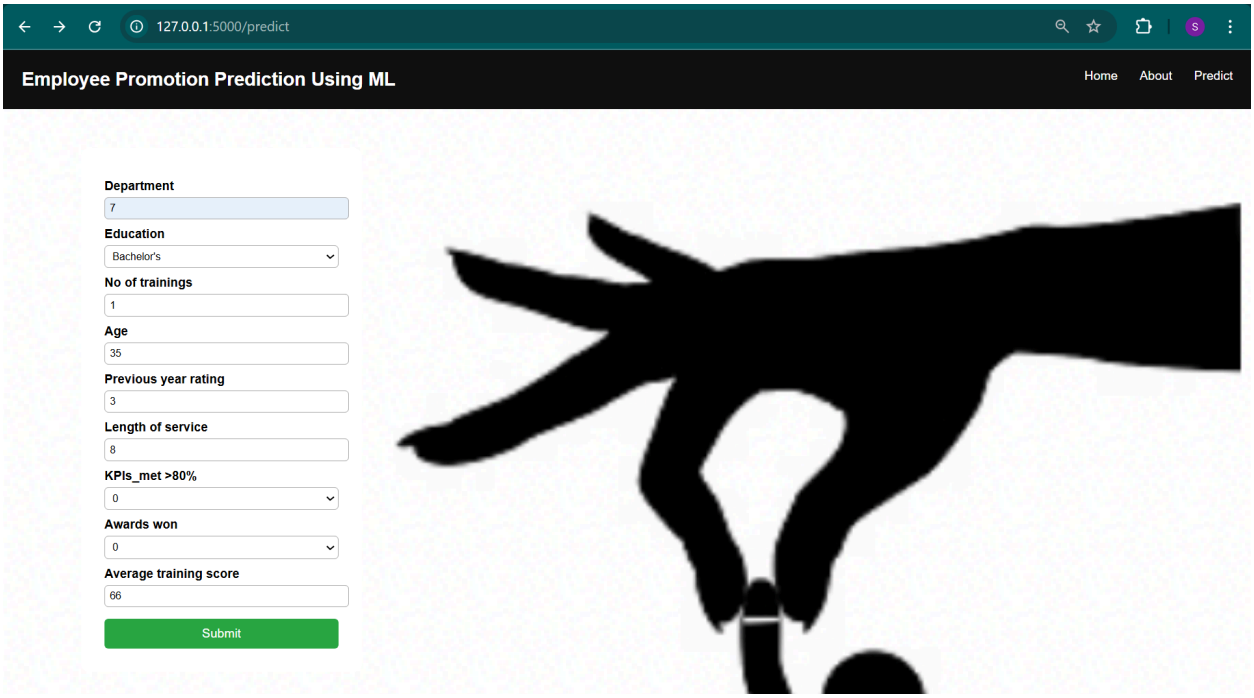
● Open VS Code from the desktop or start menu.
● Use the integrated terminal to navigate to the folder where your app.py file is located.
● Type the command python app.py and press Enter.
● Open a browser and go to http://127.0.0.1:5000/ to view the homepage of your web application.
● Click on the Predict button at the top right corner, enter the required input fields, click Submit, and view the promotion prediction on the screen.

```
C:\Users\Shyam Nagarajan\Desktop\venv\Lib\site-packages\sklearn\base.py:440: InconsistentVersionWarning: Trying
to unpickle estimator RandomForestClassifier from version 1.6.1 when using version 1.7.0. This might lead to bre
aking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server in
stead.
 * Running on http://127.0.0.1:5000
```

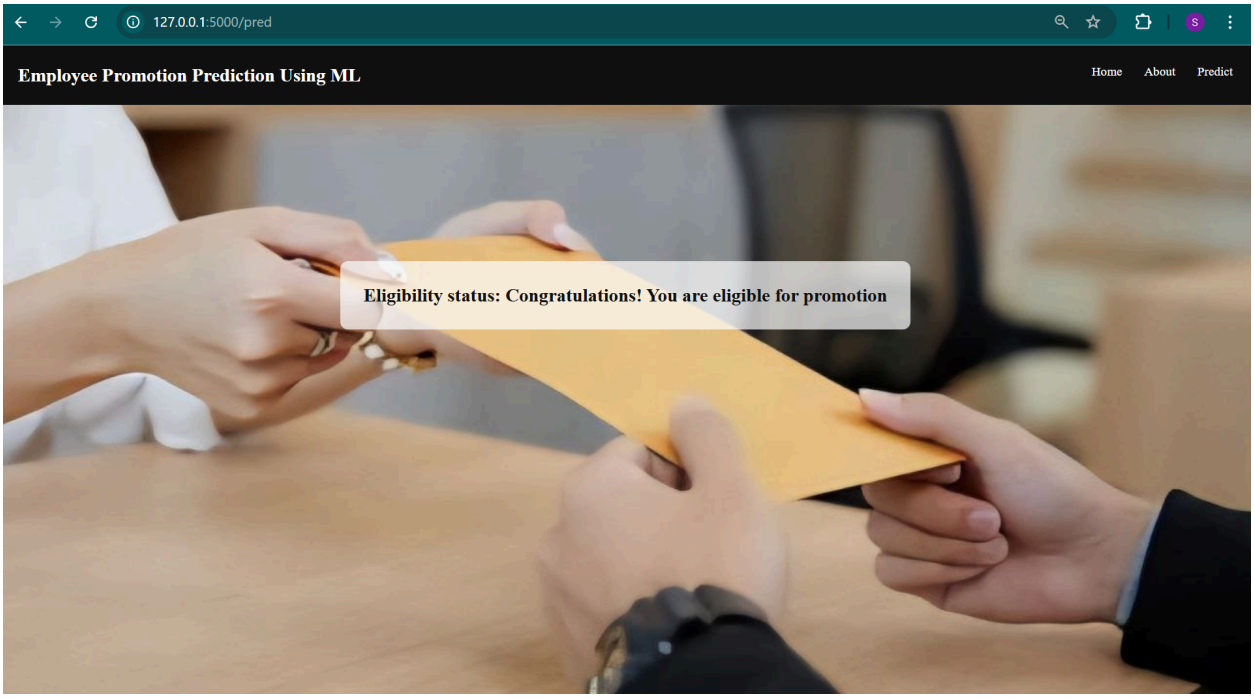
Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result


127.0.0.1:5000/about
Employee Promotion Prediction Using ML
Home
About
Predict
Employee Promotion Prediction Using ML
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. To predict your promotion click on predict button on the top right side corner.
To predict your promotion click on predict button on the top right side corner.

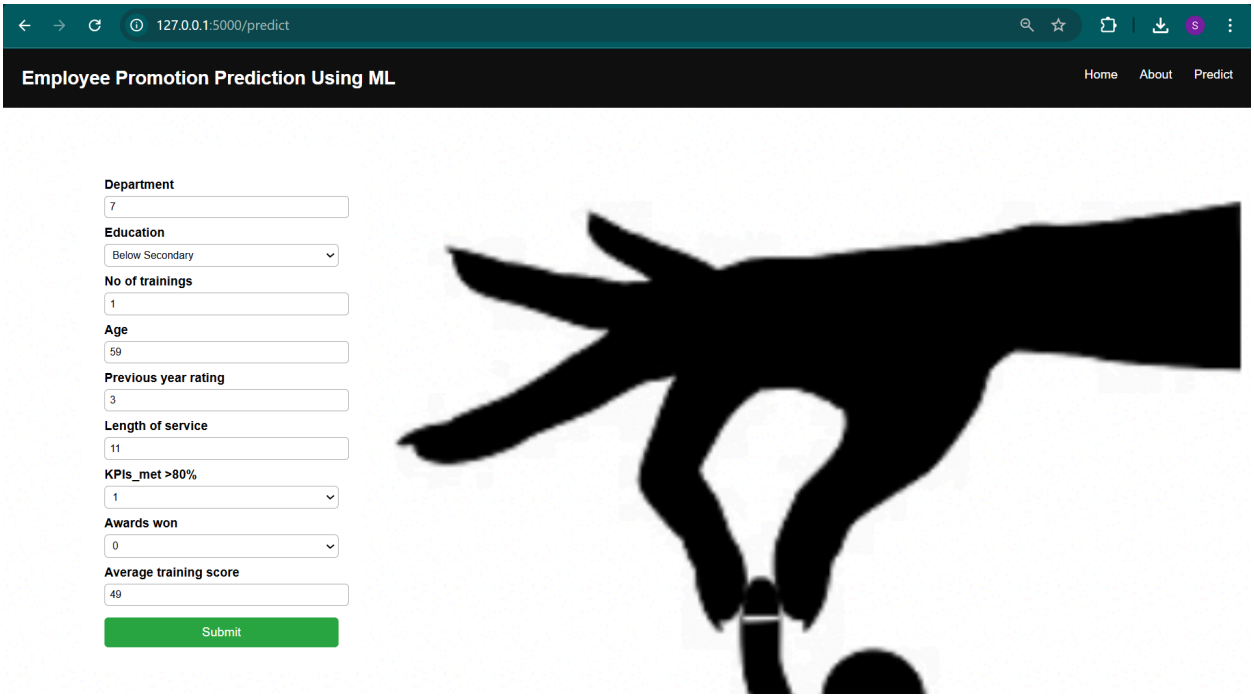
Input 1:



Output 1:



Input 2:



Output 2: