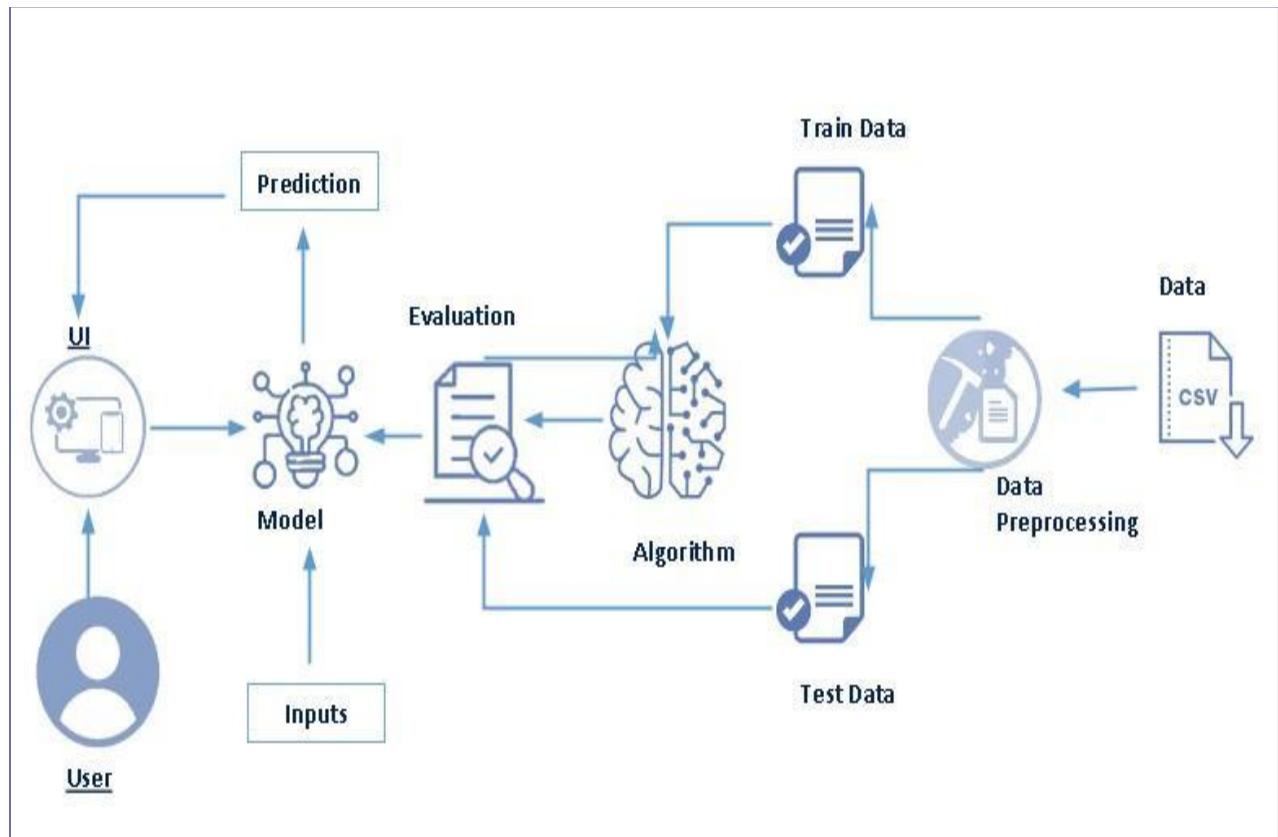


Employee Promotion Prediction Using IBM Watson Studio Machine Learning



SUBMITTED BY :
KODALI.RAKESH
PIN NO:- 19481A0351

Category: Machine Learning

Skills Required:

Python,Python For Data Visualization,Exploratory Data Analysis,Data Preprocessing Techniques,Machine Learning,Classification Algorithms,Python-Flask

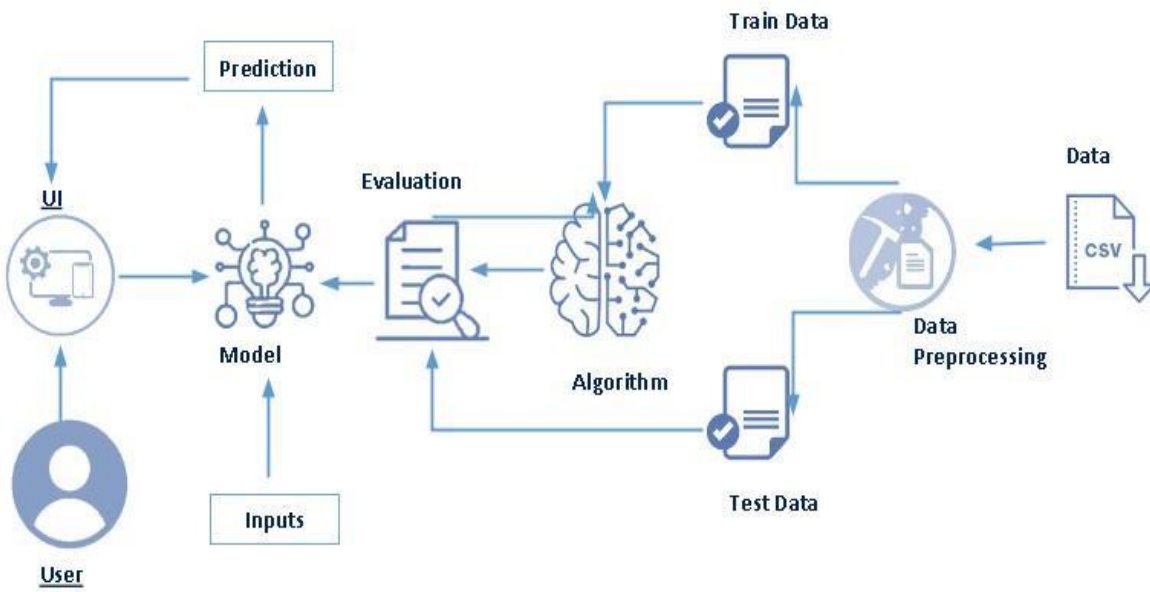
Promotion or career advancement is a process through which an employee of a company is given a higher share of duties, a higher pay scale, or both. Promotion is not just beneficial for employees but is also highly crucial for the employer or business owners. It boosts the morale of promoted employees,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

The client is facing problem while selecting to promot an employe.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.

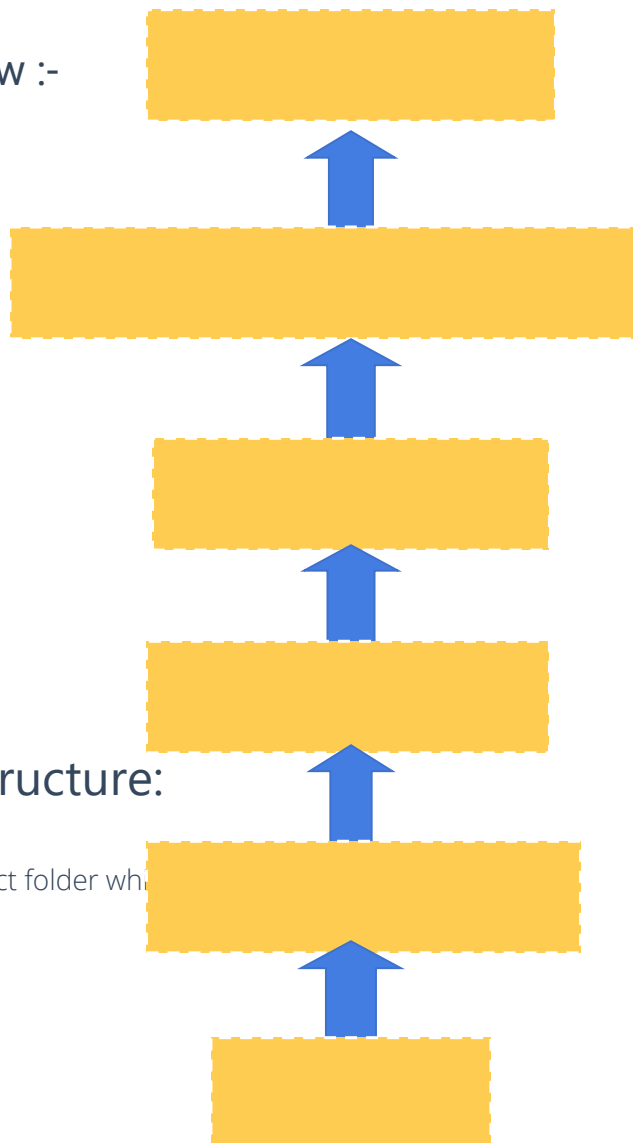
For better training results we make use of IBM to train the model to deploy the model on IBM.

client need right person who is eligible for promotion.For this he need some special criteria to select the employe for promotion.

Technical Architecture:

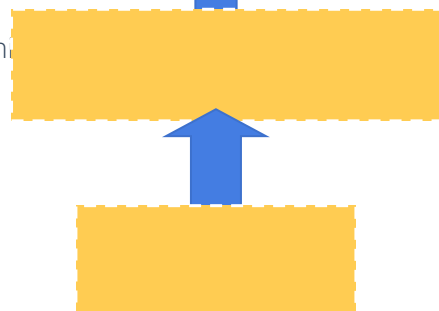


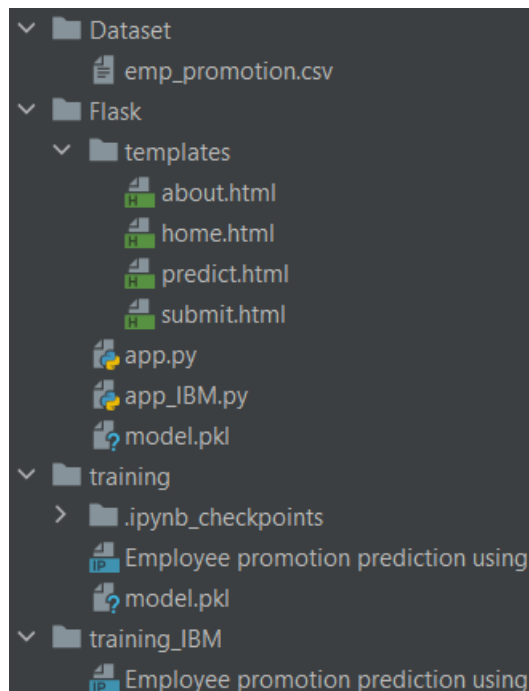
Project Flow :-



Project Structure:

Create the Project folder wh





Data Collection :

The Data set is collected from the given referral link

<https://drive.google.com/file/d/1l4qAYPpk3pctIYScWqw0Du2JEYF-rY80/view?usp=sharing>

Visualizing and Analysing The Data :

In visualizing and analysing Data we have to import the necessary libraries from the referral link. The next step we have to read the data set by using Pandas. **univariate analysis is understanding the data with single feature, multivariate analysis is to find the relation between multiple features.** Descriptive analysis is to study the basic features of data with the statistical process

Data Pre Processing :

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. 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
```

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% .

```
# 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'])
```

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.

```
# 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:

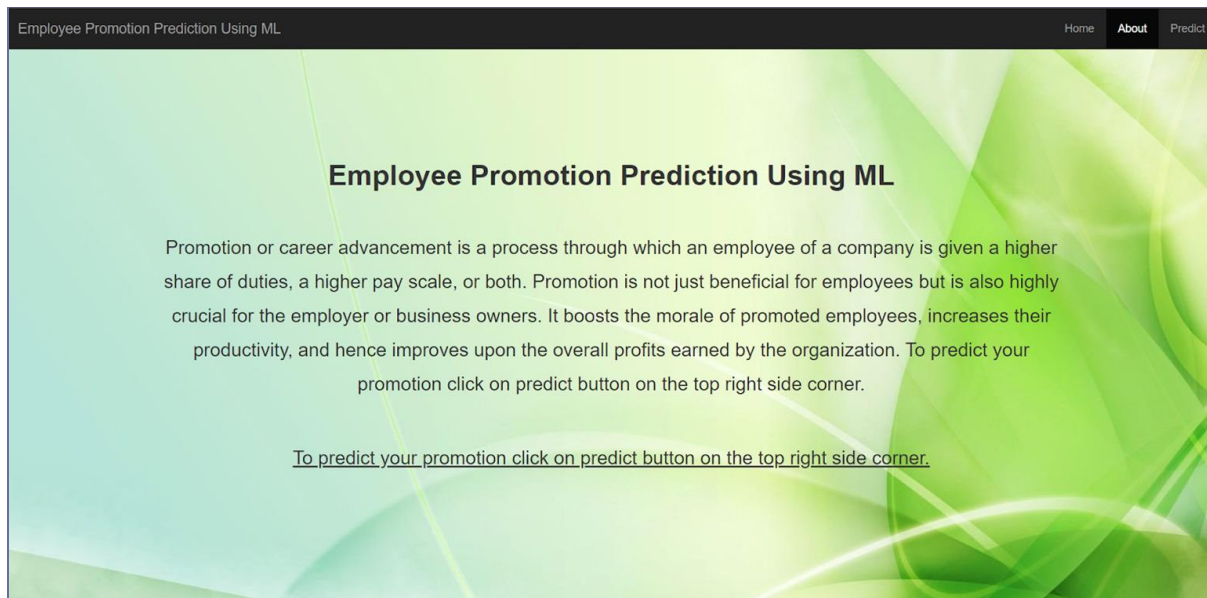
For this project, create three HTML files namely

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

Now it look like This:



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



Lets look how our predict.html file looks like:

Employee Promotion Prediction Using ML

Home About Predict

Department

Education
Below Secondary ▾

No of trainings

Age

Previous year rating


Length of service

KPIs_met >80%
0 ▾

Awards won
0 ▾

Average training score

Submit




Lets look how our submit.html file looks like:

Employee Promotion Prediction Using ML

Home About Predict

Eligibility status: {{predictionText}}.



Build Python Code:

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



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

app = Flask(__name__)
```

```
@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')
```

Retrieves the value from UI:

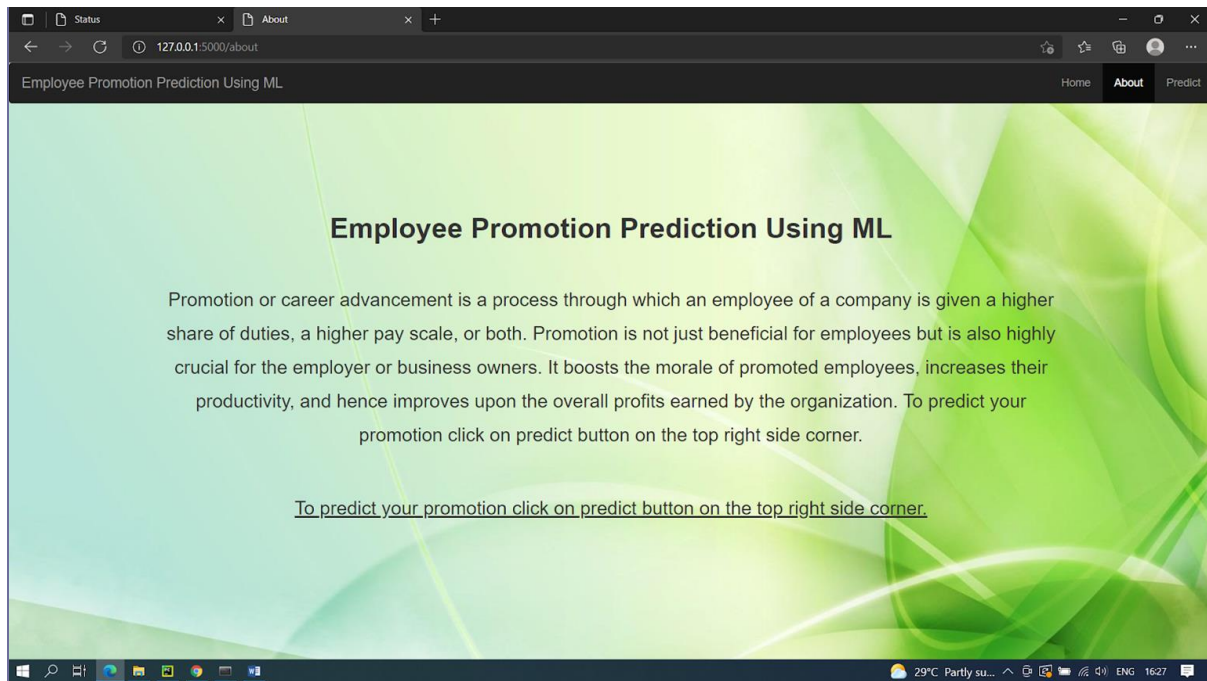
```
@app.route('/pred', methods=['POST'])
def pred():
    department = request.form['department']
    education = request.form['education']
    if education == '1':
        education = 1
    elif education == '2':
        education = 2
    else:
        education = 3
    no_of_trainings = request.form['no_of_trainings']
    age = request.form['age']
    previous_year_rating = request.form['previous_year_rating']
    length_of_service = request.form['length_of_service']
    KPIs = request.form['KPIs']
    if KPIs == '0':
        KPIs = 0
    else:
        KPIs = 1
    awards_won = request.form['awards_won']
    if awards_won == '0':
        awards_won = 0
    else:
        awards_won = 1
    avg_training_score = request.form['avg_training_score']
    total = [[department, education, no_of_trainings, age, float(previous_year_rating), float(length_of_service),
    KPIs, awards_won, avg_training_score]]
    prediction = model.predict(total)
    if prediction == 0:
        text = 'Sorry, you are not eligible for promotion'
    else:
```

Output

Home page



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

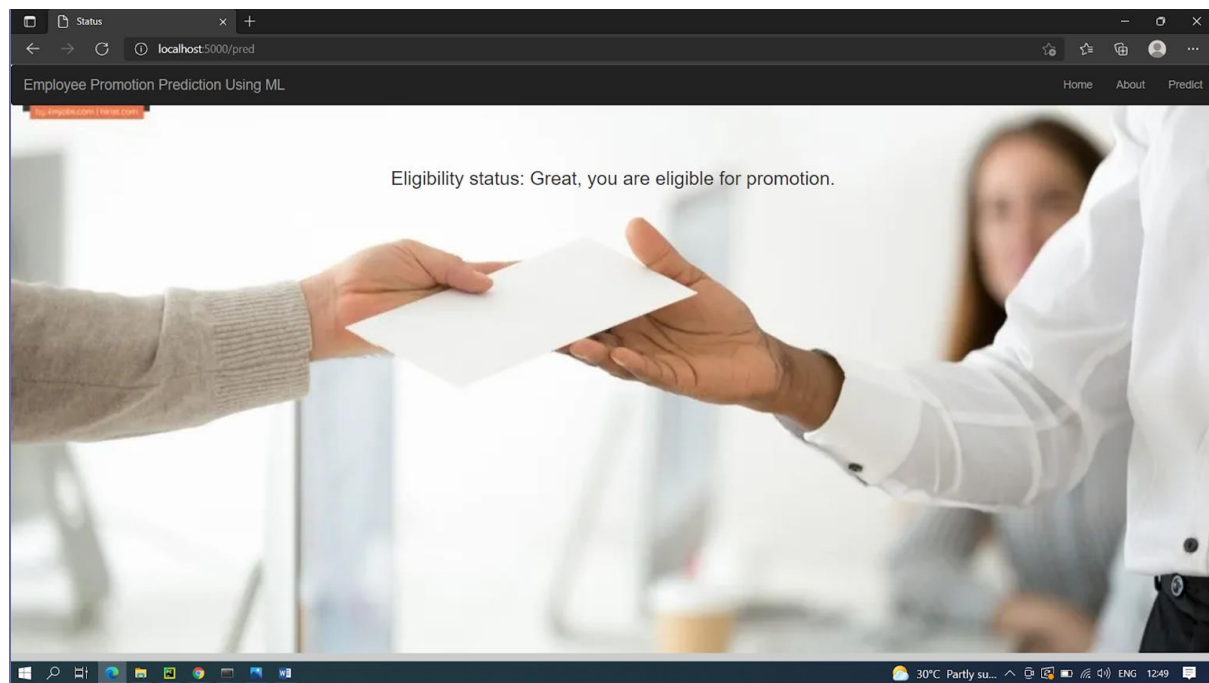


predict.html page. Now give your inputs and click on submit button. Output will be displayed in submit.html page.

Input 1:

A screenshot of the 'Predict' page of the same web application. The browser's address bar shows '127.0.0.1:5000/predict'. The page features a form with several input fields, each with a label and a corresponding input area. The inputs are: Department (7), Education (Bachelor's), No of trainings (1), Age (35), Previous year rating (3), Length of service (8), KPIs_met >80% (0), Awards won (0), and Average training score (65). A green 'Submit' button is located at the bottom left of the form. The background of the page shows a hand placing a wooden block with a red person icon on top of a stack of other blocks. The browser's taskbar at the bottom shows the system clock as 16:28 and the temperature as 29°C.

Output 1:



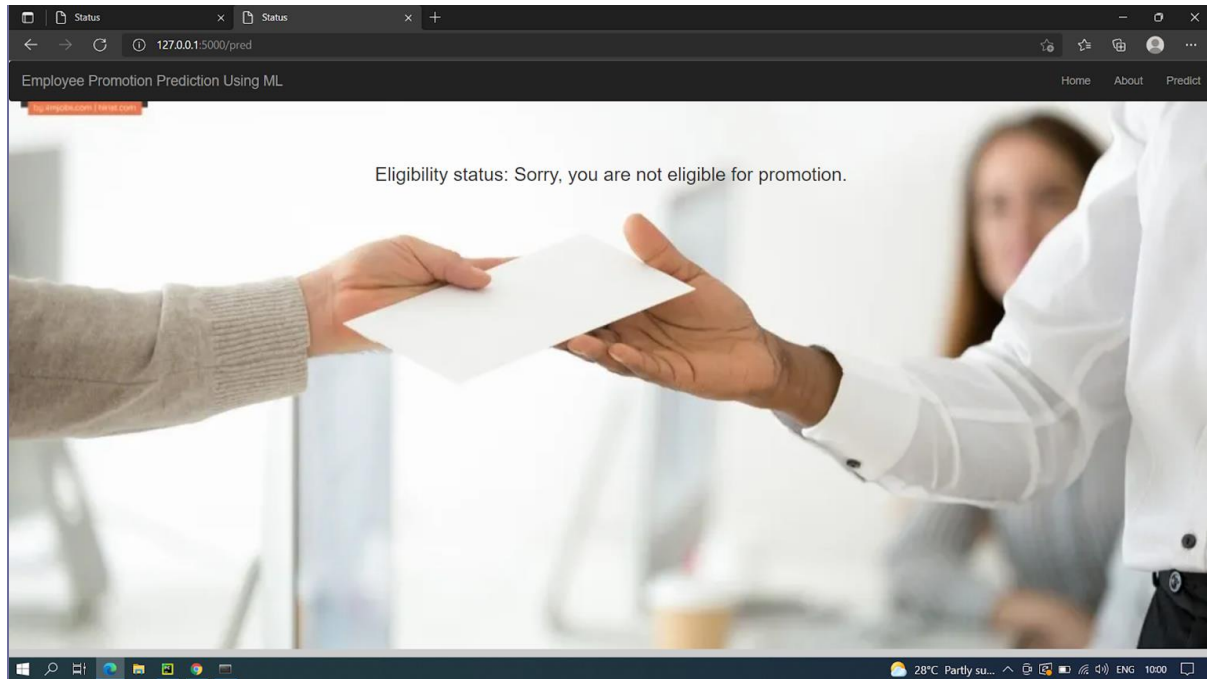
Input 2:

A screenshot of the same web application, but now showing the input form for the "Predict" page. The browser's address bar shows "127.0.0.1:5000/predict". The navigation bar includes "Home", "About", and "Predict" links. The form is set against a background image of wooden blocks, some of which have person icons. The form fields are as follows:

- Department:
- Education:
- No of trainings:
- Age:
- Previous year rating:
- Length of service:
- KPIs_met >80%:
- Awards won:
- Average training score:

A green "Submit" button is located at the bottom left of the form area. The Windows taskbar at the bottom shows the system time as 16:28 and the temperature as 29°C.

Output 2:



Train The ML Model On IBM:

We watched the below video to train machine learning in the IBM Watson

[Train The ML Model On IBM](#)

Integrate Flask With Scoring End Point:

We watched the below video to integrate the scoring endpoint to flask

[Integrate Flask With Scoring End Poin](#)

After watching the videos we train machine learning in the IBM Watson and integrated the scoring end point to the flask. After that we have to run the codes to get the same output while running in the local host and finally by giving required input to get the output. Such that Whether the employee is eligible for Promotion or not he

eligible for promotion. Thus, client can easily choose the right person for Promotion.

Advantages of A Project:

1. Automation of Everything.
2. Wide range of Applications.
3. Scope of improvement.
4. Efficient Handling of data.
5. Best for Education.

Disadvantages of A Project:

1. Possibility of High Error.
2. Algorithm Selection.
3. Data Acquisition.
4. Internet Issues.
5. Time And Space.

THANK YOU

