Employee Performance Prediction Solution Template

Project Description:

In this project we are going to analyse and predict the performance of employees in an organization on the basis of various factors, including, but not limited to, individual and domain specific characteristics, nature and level of schooling, socioeconomic status and different psychological factors.

Here we have used Supervised learning techniques namely Support Vector Machines, Random Forest, Naive Bayes, Neural Networks and Logistic Regression which considers these factors and provides insights into the performance and commitment of employees. The employees are classified into 3 output classes indicating the level of their performance from low to high. In this research paper, 10-fold validation technique is used to ensure the correctness of the prediction by the above-mentioned techniques. Support Vector Machines prove to be the most efficient in terms of accuracy. The result is accentuated by the high validation score obtained by the same.

Pre requisites:

To complete this project, you must required following software's, concepts and packages

- **Anaconda navigator and pycharm:**
  - Refer the link below to download anaconda navigator
  - Link : https://youtu.be/1ra4zH2G4o0

- **Python packages required :**
    - Open anaconda prompt as administrator
    - Type “pip install numpy” and click enter.
    - Type “pip install pandas” and click enter.
    - Type “pip install scikit-learn” and click enter.
    - Type ”pip install matplotlib” and click enter.
    - Type ”pip install scipy” and click enter.
    - Type ”pip install pickle-mixin” and click enter.
    - Type ”pip install seaborn” and click enter.
    - Type “pip install Flask” and click enter.

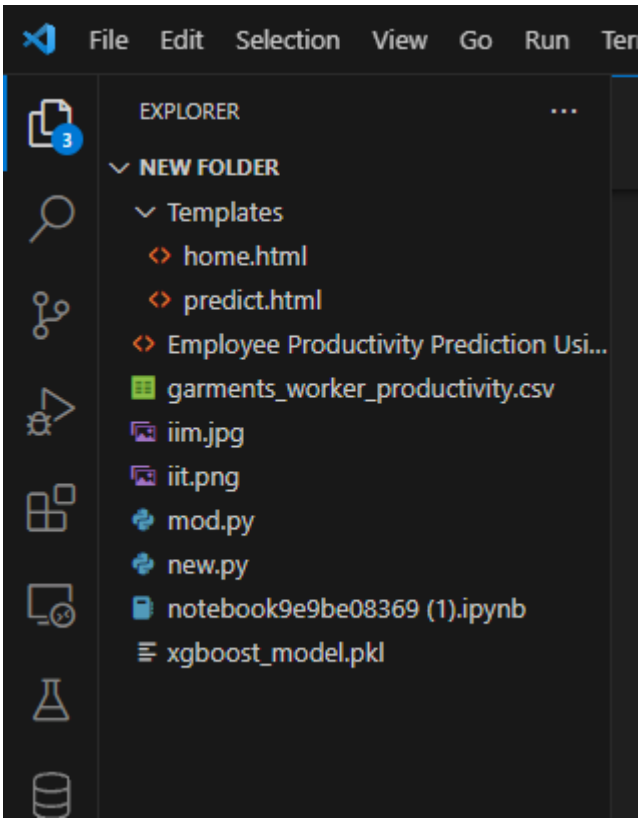
Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques and some visualization concepts.

Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once 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
    - Correlation analysis
    - Descriptive analysis

- Data pre-processing
    - Checking for null values
    - Handling Date & department column
    - Handling categorical 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 model
    - Save the model
- Application Building
    - Create an HTML file
    - Build python code

Project Structure:

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- gwp.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains Employee_Prediction.ipynb , model file.
- Ibm folder contains IBM deployment files.

Milestone 1: Data Collection

Acquiring the necessary dataset is a pivotal step in machine learning, as the algorithms heavily rely on the quality of the data for training. For this project, the dataset "garments_worker_productivity.csv" was obtained from kaggle.com. To access the dataset, you can visit the provided link or explore other popular sources like the UCI repositor

Activity 1: 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 garments_worker_productivity.csv data. This data is downloaded from kaggle.com. Please refer the link given below to download the dataset.

Link: Productivity Prediction of Garment Employees | Kaggle

Milestone 2: 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 analysing 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.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image.

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as ex
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
import plotly.offline as py
py.init_notebook_mode(connected=True)
```

Activity 2: 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.

```
data = pd.read_csv("/kaggle/input/productivity-prediction-of-garment-employees/garments_worker_productivity.csv")
data.head()
```

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_produ
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59.0	0.9
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8.0	0.8
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.8
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.8
4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0	0.8

Activity 3: Correlation analysis

In simple words, A correlation matrix is simply a table which displays the correlation coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values in a table. It is a powerful tool to summarize a large dataset and to identify and visualize patterns in the given data.

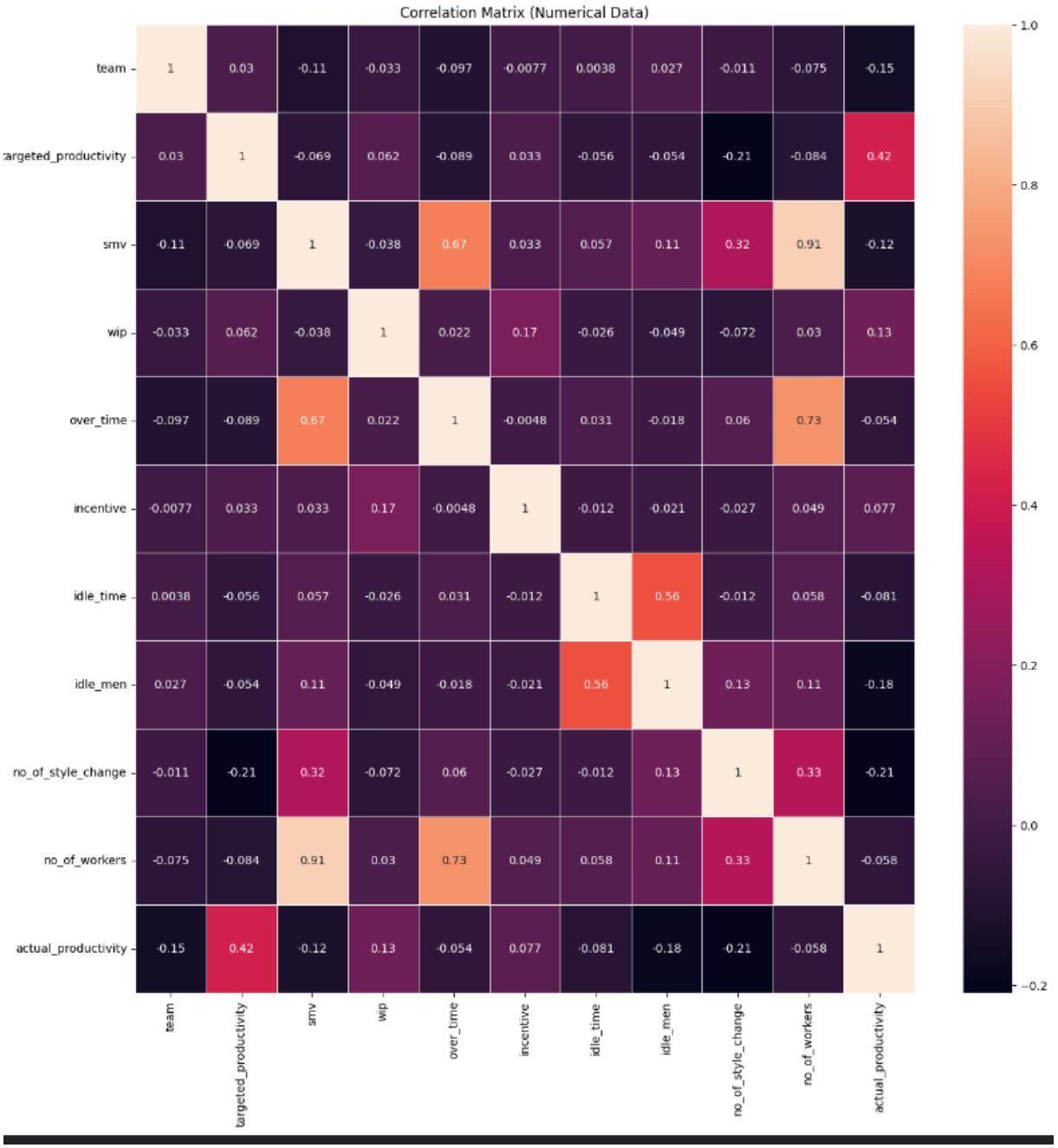
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
categorical_columns = data.select_dtypes(include=['object']).columns
```

```
numerical_data = data.select_dtypes(exclude=['object'])
corrMatrix = numerical_data.corr()
```

```
corrMatrix_categorical = data[categorical_columns].apply(lambda x: x.factorize()[0]).corr()
```

```
fig, ax = plt.subplots(figsize=(15, 15))
sns.heatmap(corrMatrix, annot=True, linewidths=.5, ax=ax)
plt.title("Correlation Matrix (Numerical Data)")
plt.show()
```



Activity 4: 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.

0]:
```
data.describe()
```

0]:

	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity
count	1197.000000	1197.000000	1197.000000	691.000000	1197.000000	1197.000000	1197.000000	1197.000000	1197.000000	1197.000000	1197.000000
mean	6.426901	0.729632	15.062172	1190.465991	4567.460317	38.210526	0.730159	0.369256	0.150376	34.609858	0.735091
std	3.463963	0.097891	10.943219	1837.455001	3348.823563	160.182643	12.709757	3.268987	0.427848	22.197687	0.174488
min	1.000000	0.070000	2.900000	7.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	0.233705
25%	3.000000	0.700000	3.940000	774.500000	1440.000000	0.000000	0.000000	0.000000	0.000000	9.000000	0.650307
50%	6.000000	0.750000	15.260000	1039.000000	3960.000000	0.000000	0.000000	0.000000	0.000000	34.000000	0.773333
75%	9.000000	0.800000	24.260000	1252.500000	6960.000000	50.000000	0.000000	0.000000	0.000000	57.000000	0.850253
max	12.000000	0.800000	54.560000	23122.000000	25920.000000	3600.000000	300.000000	45.000000	2.000000	89.000000	1.120437

Milestone 3: 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 Date & department column
- Handling categorical data
- 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.

Activity 1: Checking for null values

- Let’s find the shape of our dataset first, To find the shape of our data, data.shape method is used. To find the data type, data.info() function is used.

```
[11]: data.shape

[11]: (1197, 15)

[12]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1197 entries, 0 to 1196
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   date                   1197 non-null   object
 1   quarter                1197 non-null   object
 2   department             1197 non-null   object
 3   day                    1197 non-null   object
 4   team                   1197 non-null   int64
 5   targeted_productivity  1197 non-null   float64
 6   smv                    1197 non-null   float64
 7   wip                    691 non-null    float64
 8   over_time              1197 non-null   int64
 9   incentive              1197 non-null   int64
 10  idle_time              1197 non-null   float64
 11  idle_men               1197 non-null   int64
 12  no_of_style_change     1197 non-null   int64
 13  no_of_workers          1197 non-null   float64
 14  actual_productivity    1197 non-null   float64
dtypes: float64(6), int64(5), object(4)
memory usage: 140.4+ KB

+ Code   + Markdown
```

- For checking the null values, data.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that in our dataset there is one feature which has high number of null values. So we drop that feature.

Activity 2: Handling Date & department column

- Here what we are doing is converting the date column into datetime format.

```
data['date'] = pd.to_datetime(data['date'])
```

- Then converting date column to month (month index) & transferring the values into a new column called month. As we have the month column now we don’t need date, so we will drop it.

```
data.date
```

```
0       2015-01-01
1       2015-01-01
2       2015-01-01
3       2015-01-01
4       2015-01-01
           ...
1192    2015-03-11
1193    2015-03-11
1194    2015-03-11
1195    2015-03-11
1196    2015-03-11
Name: date, Length: 1197, dtype: datetime64[ns]
```

```
data.month
```

```
0       1
1       1
2       1
3       1
4       1
       ..
1192    3
1193    3
1194    3
1195    3
1196    3
Name: month, Length: 1197, dtype: int32
```

- From below image we can see that in department column the values are slit into 3 categories Sweing, finishing, finishing. Finishing class is repeating twice, so we will merge them into 1.

Activity 3: Handling Categorical Values

Remove leading or trailing spaces in column names:

python
Copy code
data.columns = data.columns.str.strip()
This line ensures that any leading or trailing spaces in the column names of the DataFrame data are removed. This step is a good practice to prevent potential issues with column names.

Label Encoding:

python
Copy code
label_encoder = LabelEncoder()
data['department_encoded'] = label_encoder.fit_transform(data['department'])
data['day_encoded'] = label_encoder.fit_transform(data['day'])
LabelEncoder is a preprocessing technique in machine learning to convert categorical labels into numerical representations.
For each unique value in the 'department' column, fit_transform assigns a unique numerical label to it and creates a new column 'department_encoded' in the DataFrame.
Similarly, for the 'day' column, a new column 'day_encoded' is created with numerical labels.
The purpose of this encoding is to provide a way for machine learning algorithms to work with categorical data, as many algorithms require numerical input. However, it's important to note that label encoding introduces an implicit ordinal relationship between the categories, which may not always be appropriate for all types of categorical data.

So, after this encoding, the 'department' and 'day' columns are replaced with their corresponding numerical representations in the new columns 'department_encoded' and 'day_encoded'.

```
from sklearn.preprocessing import LabelEncoder

# Remove possible leading or trailing spaces in column names
data.columns = data.columns.str.strip()

label_encoder = LabelEncoder()
data['department_encoded'] = label_encoder.fit_transform(data['department'])
data['day_encoded'] = label_encoder.fit_transform(data['day'])
```

```
data['quarter_encoded'] = data['quarter'].astype('category').cat.codes
data['department_encoded'] = data['department'].astype('category').cat.codes
data['day_encoded'] = data['day'].astype('category').cat.codes
```

Activity 4: Splitting data into train and test

Now let’s split the Dataset into train and test sets. First split the dataset into x and y and then split the data

set. After that x is converted into array format then passed into a new variable called X.

Here X and y variables are created. On X variable, data is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing X, y, test_size, random_state.

```
[25]: x=data.drop(['actual_productivity'],axis=1)
      y=data['actual_productivity']

[26]: X=x.to_numpy()

[27]: X

[27]: array([[Timestamp('2015-01-01 00:00:00'), 8, 0.8, ..., True, False,
              False],
             [Timestamp('2015-01-01 00:00:00'), 1, 0.75, ..., True, False,
              False],
             [Timestamp('2015-01-01 00:00:00'), 11, 0.8, ..., True, False,
              False],
             ...,
             [Timestamp('2015-03-11 00:00:00'), 7, 0.65, ..., False, False,
              True],
             [Timestamp('2015-03-11 00:00:00'), 9, 0.75, ..., False, False,
              True],
             [Timestamp('2015-03-11 00:00:00'), 6, 0.7, ..., False, False,
              True]], dtype=object)
```

+ Code + Markdown

Milestone 4: 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 three Regression algorithms. The best model is saved based on its performance.

Activity 1: Linear Regression model

Linear Regression has been initialized with the name model_lr. Then predictions are taken from x_test given to a variable named pred_test. After that Mean absolute error, mean squared error & r2_scrores are obtained.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load your dataset
data = pd.read_csv("/kaggle/input/productivity-prediction-of-garment-employees/garments_worker_productivity.csv")

# Convert the 'date' column to datetime format
data['date'] = pd.to_datetime(data['date'])

# Extract day, month, and year from the 'date' column
data['day'] = data['date'].dt.day
data['month'] = data['date'].dt.month
data['year'] = data['date'].dt.year

# Perform one-hot encoding for categorical features
data = pd.get_dummies(data, columns=['quarter', 'department'], drop_first=True)

# Define your features (X) and target variable (y)
X = data.drop(['actual_productivity', 'date'], axis=1)
y = data['actual_productivity']

# Split the dataset into a training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Linear Regression model
model = LinearRegression()
```

Activity 2: Random Forest model

Random Forest has been initialized with the name model_rf. Then predictions are taken from x_test given to a variable named pred. After that Mean absolute error, mean squared error & r2_scrores are obtained.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load your dataset
data = pd.read_csv("/kaggle/input/productivity-prediction-of-garment-employees/garments_worker_productivity.csv")

# Convert the 'date' column to datetime format
data['date'] = pd.to_datetime(data['date'])

# Extract day, month, and year from the 'date' column
data['day'] = data['date'].dt.day
data['month'] = data['date'].dt.month
data['year'] = data['date'].dt.year

# Perform one-hot encoding for categorical features
data = pd.get_dummies(data, columns=['quarter', 'department'], drop_first=True)

# Define your features (X) and target variable (y)
X = data.drop(['actual_productivity', 'date'], axis=1)
y = data['actual_productivity']

# Split the dataset into a training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Random Forest regression model
model_rf = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model to the training data
model_rf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model_rf.predict(X_test)
```

Activity 3: Xgboost model

XGBoost has been initialized with the name model_xgb. Then predictions are taken from x_test given to a variable named pred3. After that Mean absolute error, mean squared error & r2_scrores are obtained.

Now let's see the performance of all the models and save the best model

```
import pandas as pd
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load your dataset
data = pd.read_csv("/kaggle/input/productivity-prediction-of-garment-employees/garments_worker_productivity.csv")

# Convert the 'date' column to datetime format
data['date'] = pd.to_datetime(data['date'])

# Extract day, month, and year from the 'date' column
data['day'] = data['date'].dt.day
data['month'] = data['date'].dt.month
data['year'] = data['date'].dt.year

# Perform one-hot encoding for categorical features
data = pd.get_dummies(data, columns=['quarter', 'department'], drop_first=True)

# Define your features (X) and target variable (y)
X = data.drop(['actual_productivity', 'date'], axis=1)
y = data['actual_productivity']

# Split the dataset into a training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an XGBoost regression model
model = xgb.XGBRegressor()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)
```

Activity 4: Compare the model

For comparing the above three models MSE, MAE & r2_scroes are used.

Linear Regression:

```
# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the evaluation metrics
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.013858478059003965
Mean Absolute Error: 0.07293121276796696
R-squared: 0.4780719129548352
```

Random Forest:

```
# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the evaluation metrics
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared:", r2)
```

XGBoost:

```
# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the evaluation metrics
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.013858478059003965
Mean Absolute Error: 0.07293121276796696
R-squared: 0.4780719129548352
```

After calling the function, the results of models are displayed as output. From the three model xgboost is performing well.

Activity 5: Evaluating performance of the model and saving the model

From sklearn, metrics r2_score is used to evaluate the score of the model. On the parameters, we have given y_test & pred3. Our model is performing well. So, we are saving the model by pickle.dump().

```
# Print the evaluation metrics
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared:", r2)

# Check if R-squared is above the threshold
if r2 > threshold:
    print("Employee is highly productive ")
else:
    print("Employee is low productive")
```

```
Mean Squared Error: 0.021224067124298793
Mean Absolute Error: 0.10725581350934929
R-squared: 0.2006743665256705
Employee is low productive
```

Milestone 5: 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 server side script

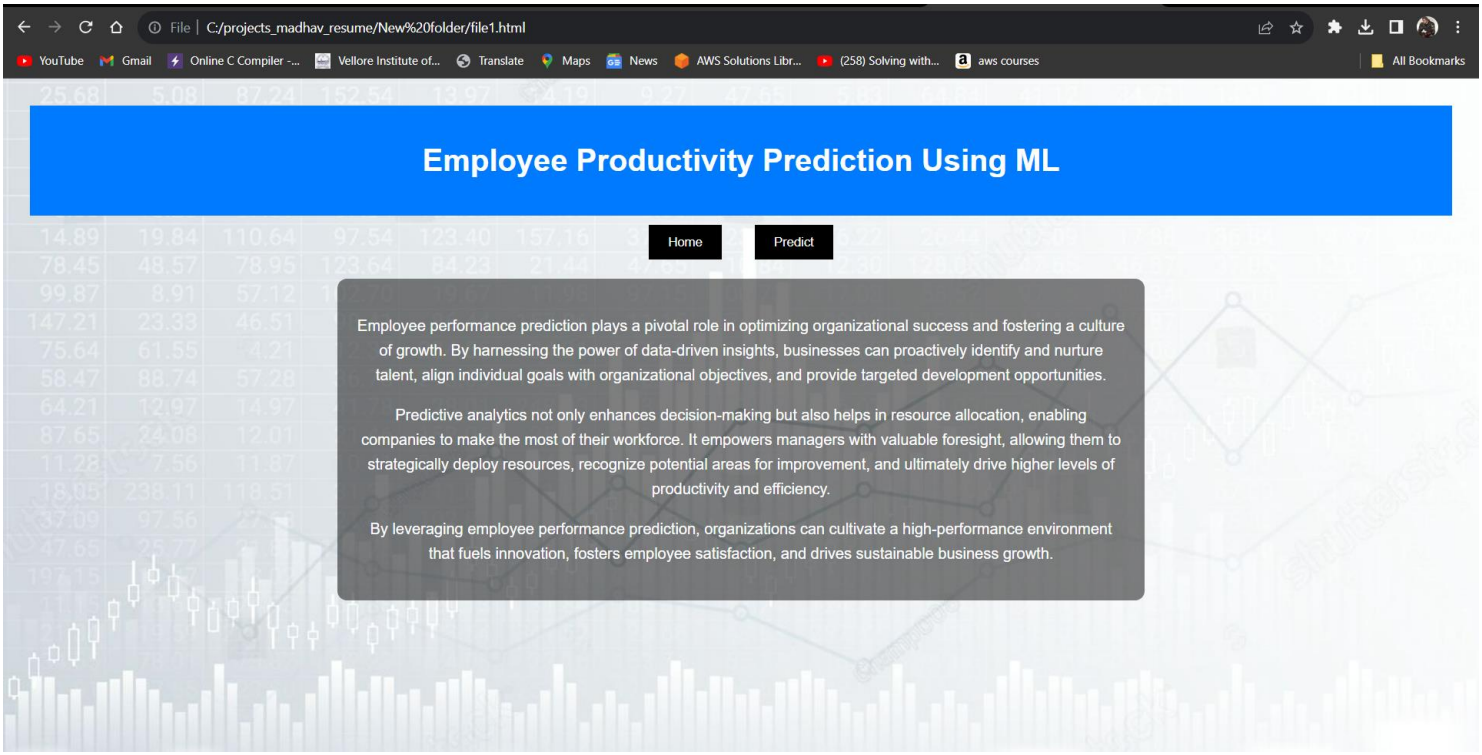
Activity1: Building Html Pages:

For this project create three HTML files namely

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

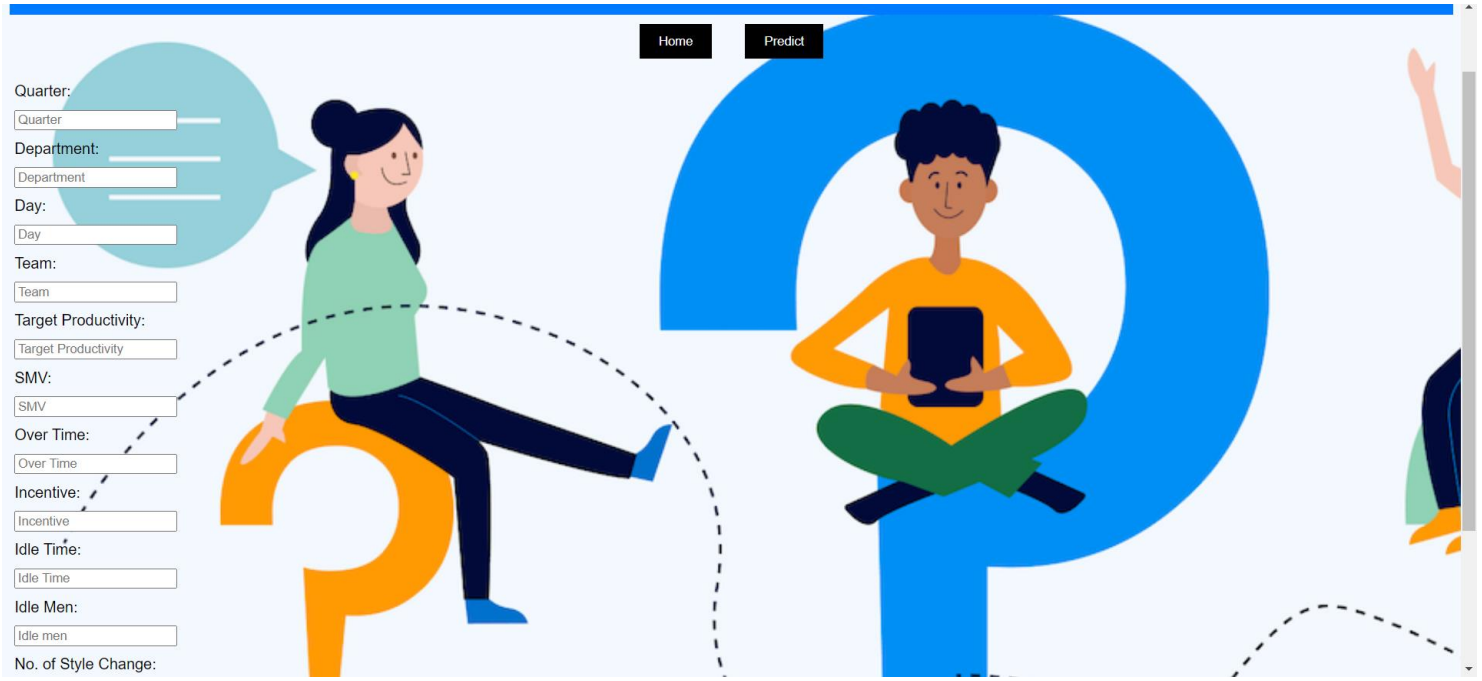
and save them in templates folder.

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



Now when you click on predict button from top right corner you will get redirected to predict.html

Lets look how our predict.html file looks like:



Now when you click on submit button from left bottom corner you will get redirected to submit.html

Lets look how our submit.html file looks like:

Activity 2: Build Python code:

Import the libraries

```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

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

HTML page creation :

```
        name="idle_time"
        placeholder="Idle Time"
      />
      <label for="idle_men">Idle Men:</label>
      <input type="text" id="idle_men" name="idle_men" placeholder="idle men" />
      <label for="style_change">No. of Style Change:</label>
      <input
        type="text"
        id="style_change"
        name="style_change"
        placeholder="Style Change"
      />
      <label for="No. Worker">No. of Worker:</label>
      <input
        type="text"
        id="No. Worker"
        name="No. Worker"
        placeholder="No. Worker"
      />
      <label for="month">Month:</label>
      <input type="text" id="month" name="month" placeholder="Month" />

      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```

Flask app source code:

```
ict.html    FlaskCode.py
kCode.py > ...
    try:
        quarter =request.form[ 'quarter']
        department =request.form[ 'department']
        day =request.form[ 'day']
        team =request.form[ 'team']
        targeted_productivity= request.form['targeted_productivity']
        smv=request.form[ 'smv']
        over_time= request.form[ 'Over_time']
        incentive= request.form[ 'Incentive']
        idle_time =request.form[ 'idle_time']
        idle_men =request.form[ 'idle_men']
        no_of_style_change =request.form[ 'style_change']
        no_of_workers=request.form['No. Worker']
        month =request.form[ 'month']
        wip = 1039.0
        missing_feature_2 = 0
        missing_feature_4 = 0
        missing_feature_5 = 0
        missing_feature_6 = 0
        missing_feature_3 = 0
        total= [[int(quarter), int(department), int(day), int(team),
        float (targeted_productivity),float(smv), int(over_time), int(incentive),
        float(idle_time), int (idle_men), int(no_of_style_change), float (no_of_workers), int(month),
        wip, missing_feature_2,
        missing_feature_3, missing_feature_4,
        missing_feature_5, missing_feature_6]]
```

Activity 3: Run the application

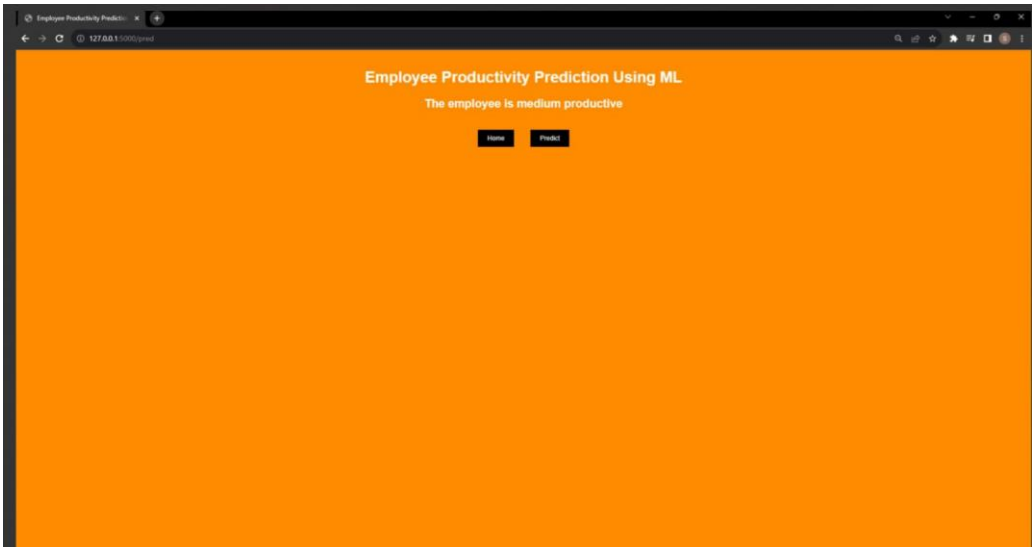
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 138-202-688
127.0.0.1 - - [12/Nov/2023 08:21:19] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Nov/2023 08:21:19] "GET /static/iim.jpg HTTP/1.1" 304 -
127.0.0.1 - - [12/Nov/2023 08:21:19] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [12/Nov/2023 08:21:38] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [12/Nov/2023 08:21:38] "GET /static/iit.png HTTP/1.1" 304 -
127.0.0.1 - - [12/Nov/2023 08:21:45] "POST /pred HTTP/1.1" 200 -
[[7, 4, 61, 7, 0.8, 5.2, 1, 7, 2.2, 7, 7, 7.5, 5, 1039.0, 0, 0, 0, 0, 0]]
[0.42786407]
127.0.0.1 - - [12/Nov/2023 08:23:42] "POST /pred HTTP/1.1" 200 -
```

Input 1:

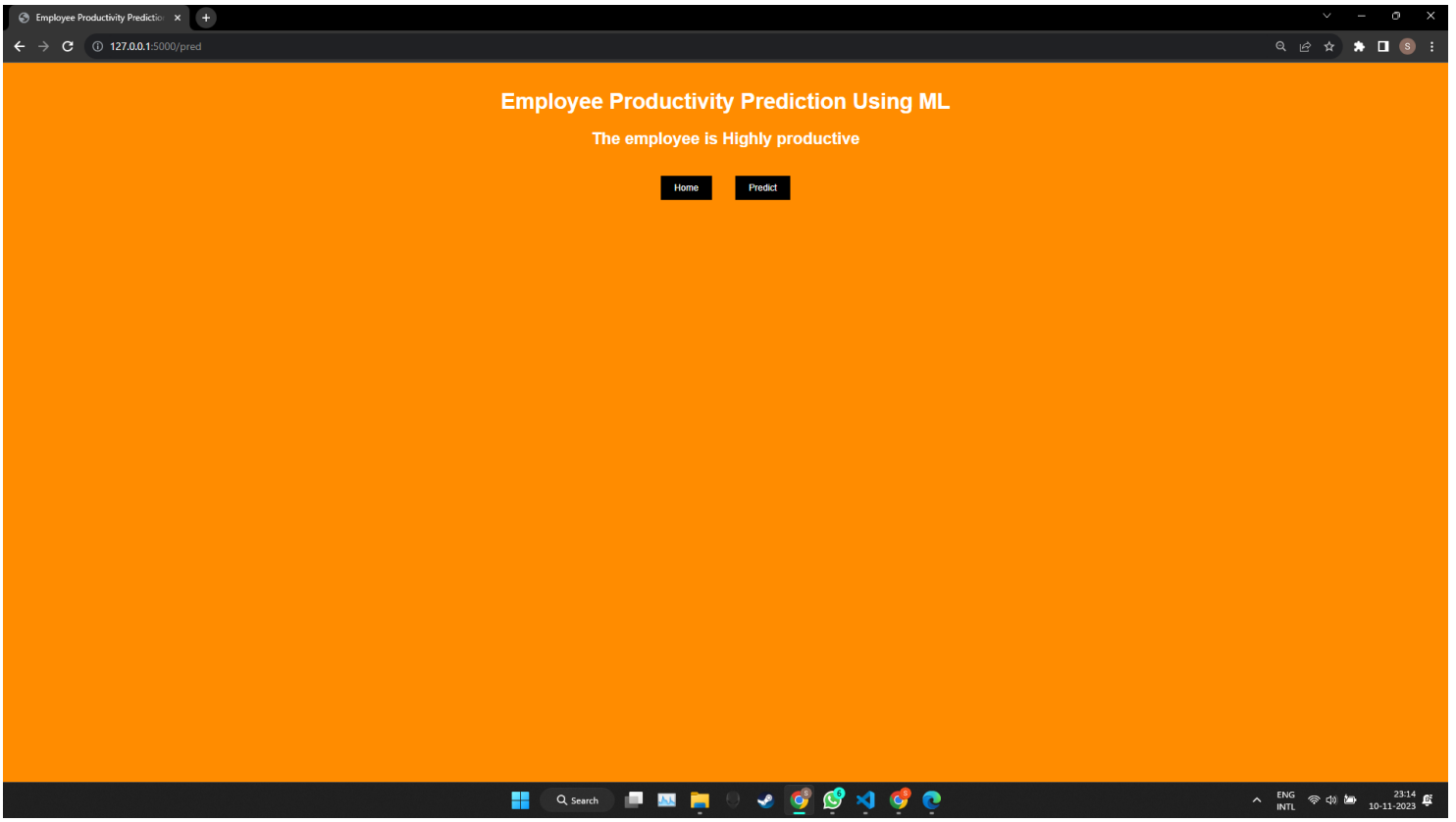
Output 1:



Input 2:


Employee Productivity Prediction Using ML
Home
Predict
Quarter:
5
Department:
7
Day:
61
Team:
12
Targeted Productivity:
0.80
SMV:
26.16
Over Time:
2626
Incentive:
50
Idle Time:
2.2
Idle Men:
7
No. of Style Change:
88
No. of Worker:
59.0
Month:
7
Submit

Output 2:


Employee Productivity Prediction Using ML
The employee is Highly productive
Home
Predict