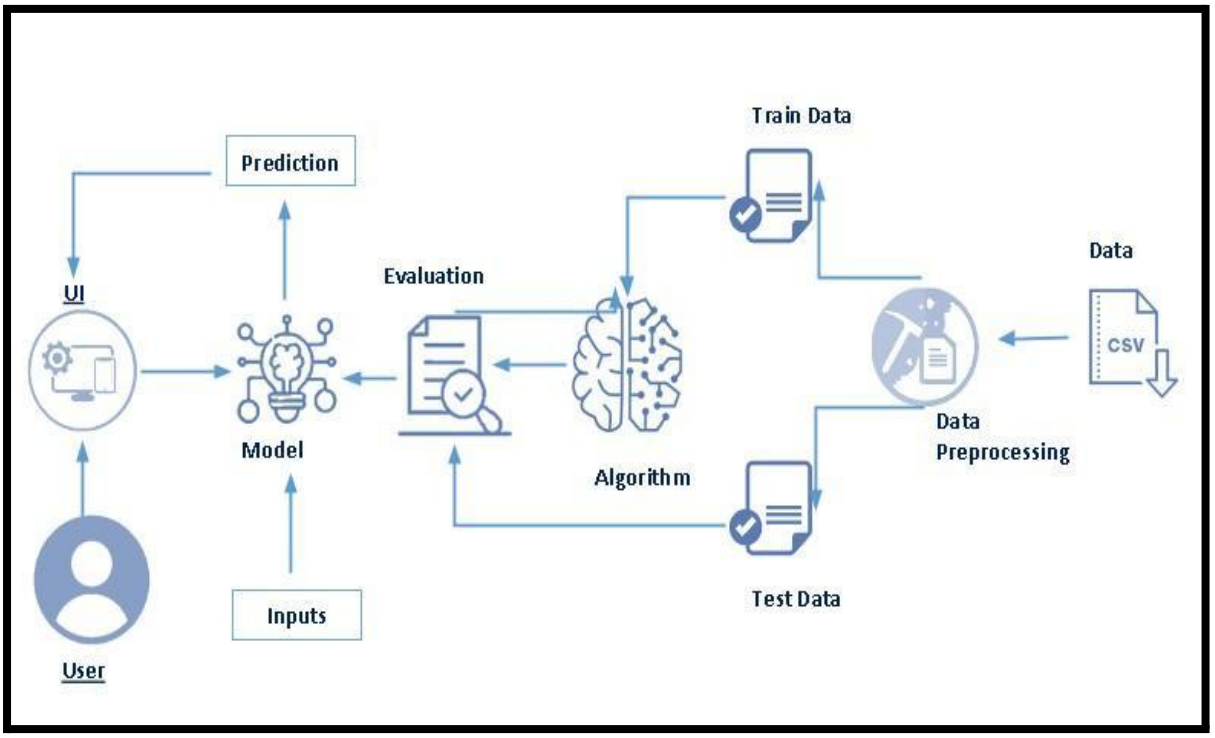
Employee Performance Prediction

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.

Technical Architecture:



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:**
  - 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.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **ML Concepts**
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
  - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
  - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
  - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
  - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
  - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

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

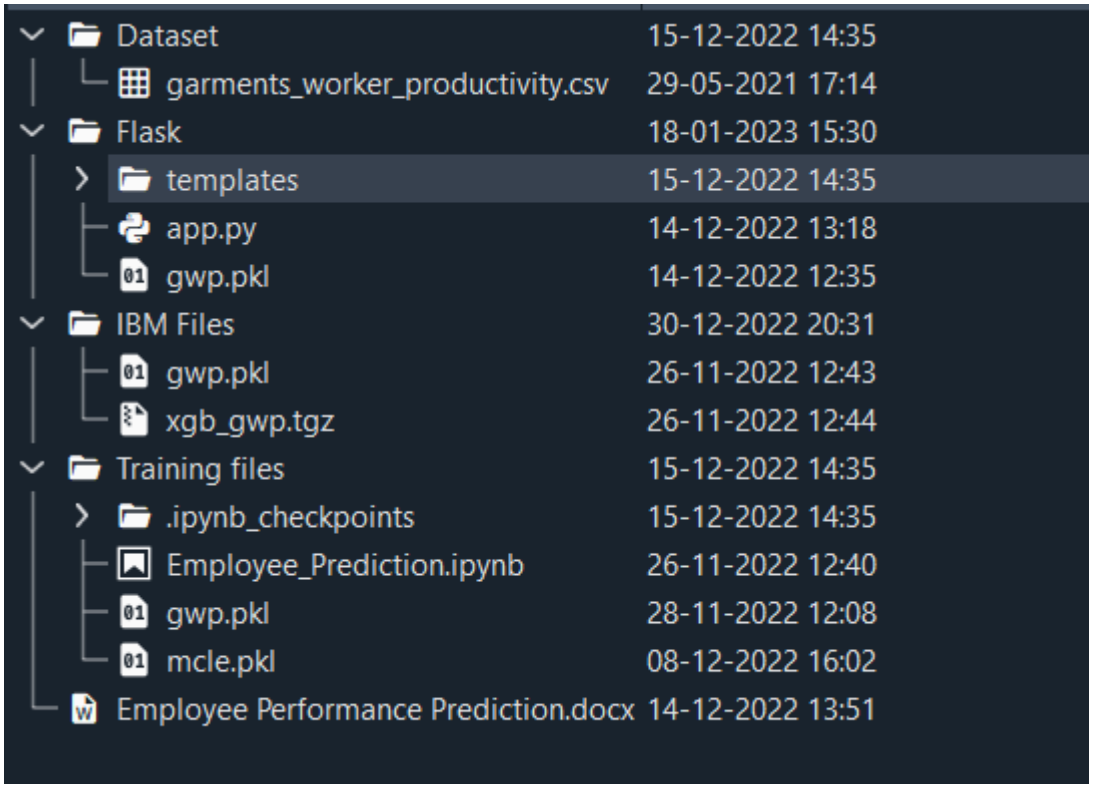
To accomplish this, we have to complete all the activities listed below,

- 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:

Create the Project folder which contains files as shown below



- 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

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.

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 matplotlib.pyplot as plt
import MultiColumnLabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
import pickle
```

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("/content/garments_worker_productivity.csv")
```

```
data.head()
```

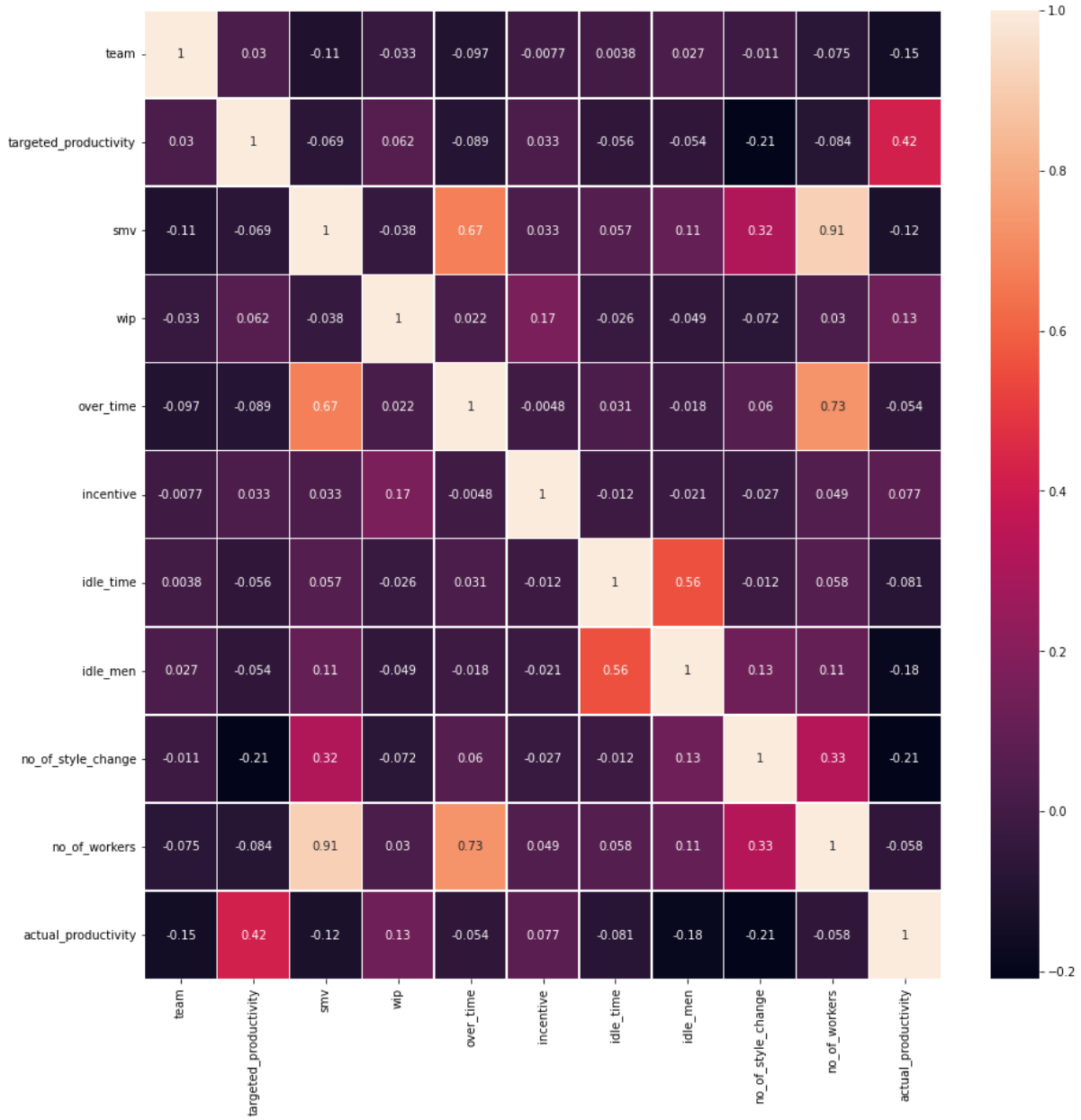
	date	quarter	department	day	team	targeted_productivity	smv
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41
4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90

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
```

```
corrMatrix = data.corr()
fig, ax = plt.subplots(figsize=(15,15)) # Sample figsize in inches
sns.heatmap(corrMatrix, annot=True, linewidths=.5, ax=ax)
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.

```
data.describe()
```

	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.

```
data.shape
```

```
(1197, 15)
```

```
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
```

- 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.

```
data.isnull().sum()
```

```
quarter                    0
department                 0
day                        0
team                       0
targeted_productivity      0
smv                        0
wip                      506
over_time                  0
incentive                  0
idle_time                  0
idle_men                   0
no_of_style_change         0
no_of_workers              0
actual_productivity        0
month                      0
dtype: int64
```

```
data.drop(['wip'],axis=1,inplace=True)
```

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']=data['date'].dt.month
data.drop(['date'],axis=1, inplace=True)
```

```
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: int64
```

- 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.

```
data['department'].value_counts()
```

```
sweing        691
finishing     257
finishing     249
Name: department, dtype: int64
```

```
# Finishing department is split into 2, we will merge them into 1
data['department'] = data['department'].apply(lambda x: 'finishing' if x.replace(" ","") == 'finishing' else 'sweing' )
```

```
data['department'].value_counts()
```

```
sweing       691
finishing    506
Name: department, dtype: int64
```

Activity 3: 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 MultiColoumnLabelEncoder.

- In our project, categorical features are quarter, department, day. With MultiColoumnLabelEncoder encoding is done.

```
import MultiColumnLabelEncoder
Mcle = MultiColumnLabelEncoder.MultiColumnLabelEncoder()
data = Mcle.fit_transform(data)
```

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.

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

```
X=x.to_numpy()
```

```
X
```

```
array([[ 0. ,  0. ,  0. , ...,  0. , 59. ,  1. ],
       [ 0. ,  1. ,  0. , ...,  0. ,  8. ,  1. ],
       [ 0. ,  0. ,  0. , ...,  0. , 30.5,  1. ],
       ...,
       [ 1. ,  1. ,  5. , ...,  0. ,  8. ,  3. ],
       [ 1. ,  1. ,  5. , ...,  0. , 15. ,  3. ],
       [ 1. ,  1. ,  5. , ...,  0. ,  6. ,  3. ]])
```

```
# Splitting the data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y,train_size=0.8,random_state=0)
```

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.

```
from sklearn.linear_model import LinearRegression
model_lr=LinearRegression()
```

```
pred_test=model_lr.predict(x_test)
print("test_MSE:",mean_squared_error(y_test, pred_test))
print("test_MAE:",mean_absolute_error(y_test, pred_test))
print("R2_score:{}".format(r2_score(y_test, pred_test)))
```

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.

```
from sklearn.ensemble import RandomForestRegressor
model_rf = RandomForestRegressor(n_estimators=200,max_depth=5)
```

```
pred = model_rf.predict(x_test)
print("test_MSE:",mean_squared_error(y_test, pred))
print("test_MAE:",mean_absolute_error(y_test, pred))
print("R2_score:{}".format(r2_score(y_test, pred)))
```

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.

```
import xgboost as xgb
model_xgb = xgb.XGBRegressor(n_estimators=200, max_depth=5, learning_rate=0.1)
```

```
pred3=model_xgb.predict(x_test)
```

```
print("test_MSE:",mean_squared_error(y_test, pred3))
print("test_MAE:",mean_absolute_error(y_test, pred3))
print("R2_score:{}".format(r2_score(y_test, pred3)))
```

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

Activity 4: Compare the model

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

Linear Regression:

```
pred_test=model_lr.predict(x_test)
print("test_MSE:",mean_squared_error(y_test, pred_test))
print("test_MAE:",mean_absolute_error(y_test, pred_test))
print("R2_score:{}".format(r2_score(y_test, pred_test)))
```

```
test_MSE: 0.0209730772468707
test_MAE: 0.10639164268443838
R2_score:0.29063171660927833
```

Random Forest:

```
pred = model_rf.predict(x_test)
print("test_MSE:",mean_squared_error(y_test, pred))
print("test_MAE:",mean_absolute_error(y_test, pred))
print("R2_score:{}".format(r2_score(y_test, pred)))
```

```
test_MSE: 0.015308208588222775
test_MAE: 0.08536234850163728
R2_score:0.4822334595828116
```

XGBoost:

```
pred = model_rf.predict(x_test)
print("test_MSE:",mean_squared_error(y_test, pred))
print("test_MAE:",mean_absolute_error(y_test, pred))
print("R2_score:{}".format(r2_score(y_test, pred)))
```

```
test_MSE: 0.015308208588222775
test_MAE: 0.08536234850163728
R2_score:0.4822334595828116
```

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().

```
pred3=model_xgb.predict(x_test)
```

```
print("test_MSE:",mean_squared_error(y_test, pred3))
print("test_MAE:",mean_absolute_error(y_test, pred3))
print("R2_score:{}".format(r2_score(y_test, pred3)))
```

```
test_MSE: 0.014019234237879834
test_MAE: 0.07674332915561435
R2_score:0.5258301865425556
```

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.

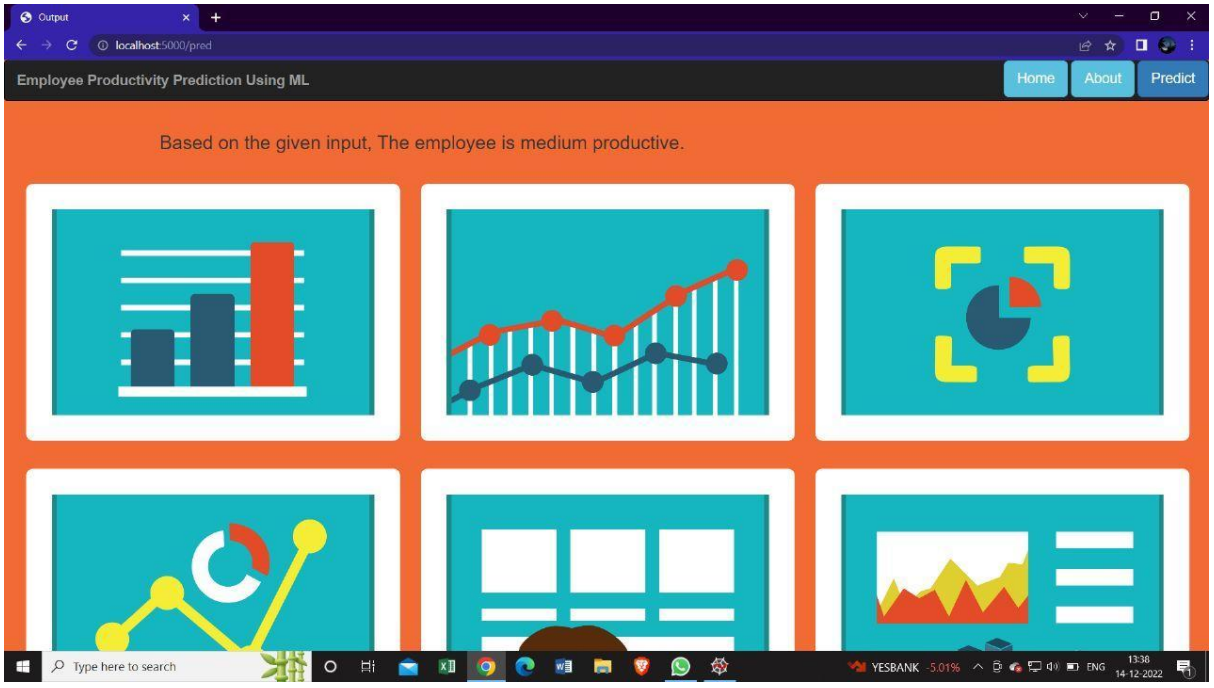
Let's see how our home.html page looks like:

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.

```
app=Flask(__name__)

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

Render HTML page:

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


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


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


@app.route("/submit")
def home2():
    return render_template('submit.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 html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```python
@app.route("/pred", methods=['POST'])
def predict():
    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['no_of_style_change']
    no_of_workers = request.form['no_of_workers']
    month = request.form['month']
    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)]]
    print(total)
    prediction = model.predict(total)
    print(prediction)
    if prediction <= 0.3:
        text = 'The emmployee is averagely productive.'
    elif prediction >0.3 and prediction <=0.8:
        text = 'The employee is medium productive'
    else:
        text = 'The employee is Highly productive'

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

Here we are routing our app to pred () 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 home.html page earlier.

Main Function:

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

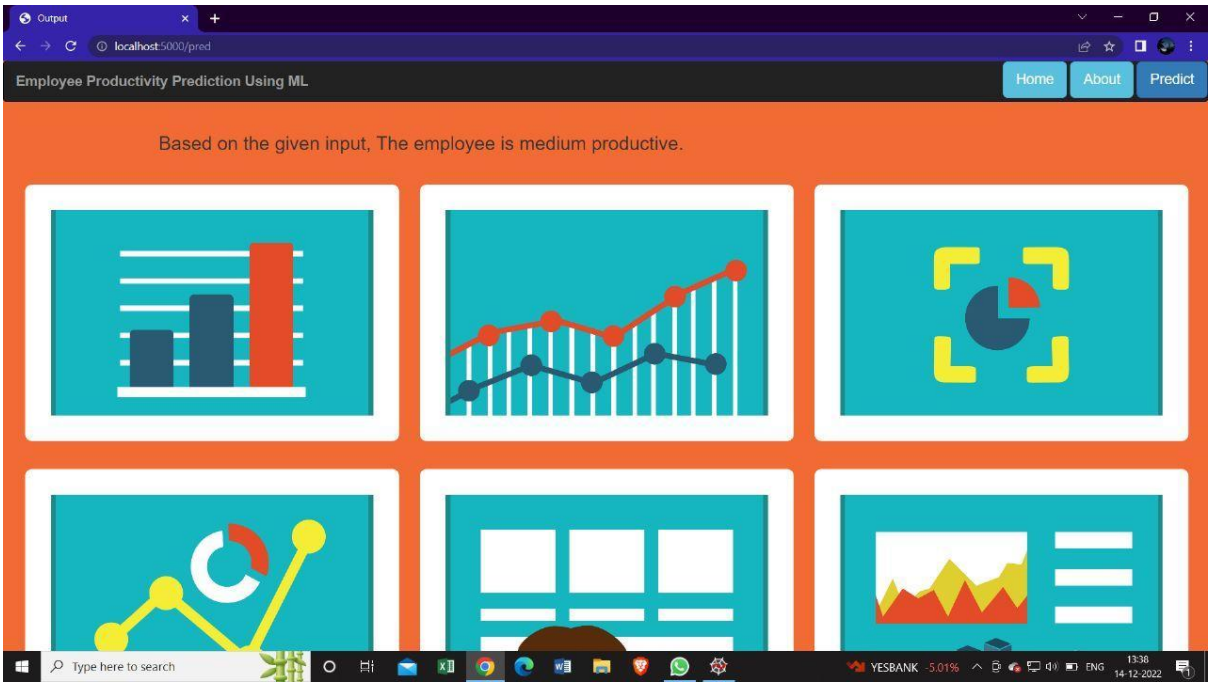
Activity 3: Run the application

- 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.

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a
roduction deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Input 1:

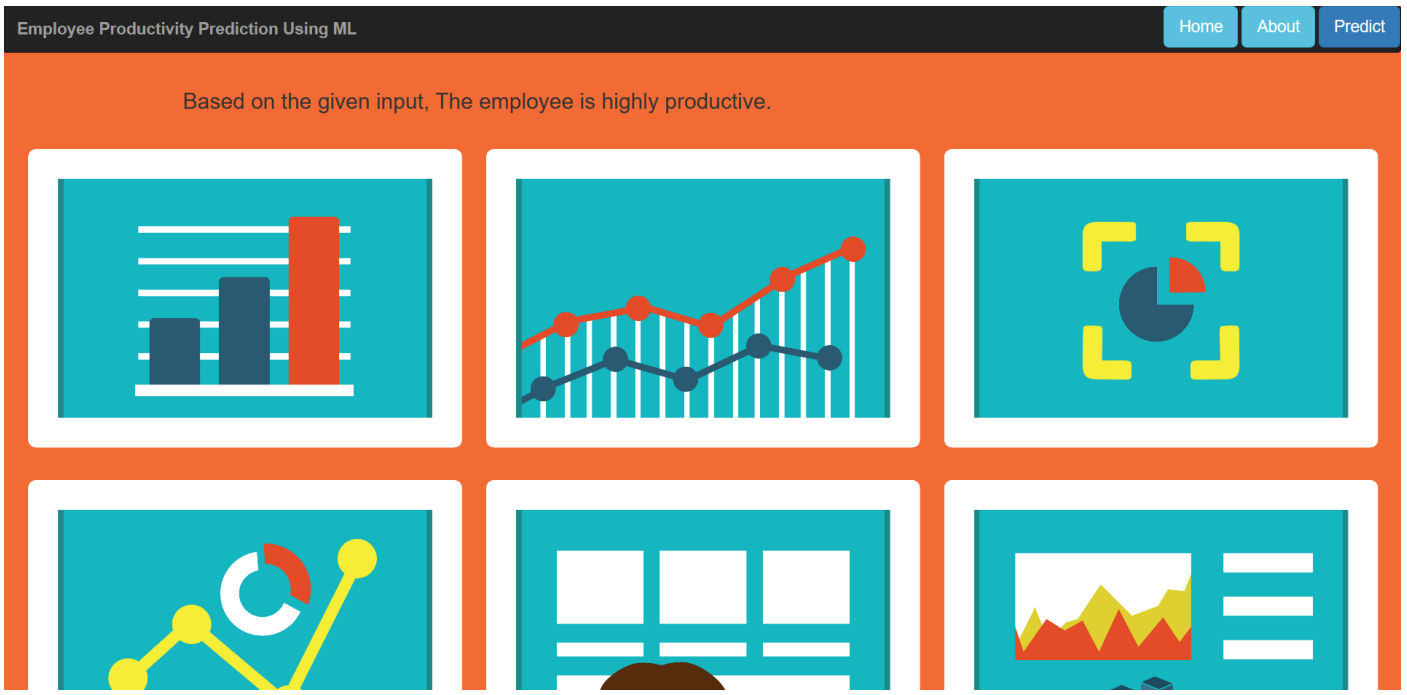
Output 1:



Input 2:


Employee Productivity Prediction Using ML
Home
About
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_workers
59.0
month
7
SUBMIT

Output 2:


Employee Productivity Prediction Using ML
Home
About
Predict
Based on the given input, The employee is highly productive.