Project report

Project title:

Machine Learning Approach for Employee Performance Prediction

Team members:

G. Venkata Dinesh (TL)

J. Hareesh

C. Baba fakruddin

Team id: **738306**

INDEX

1. Introduction

1.1. Project overviews

Predicting employee performance using machine learning (ML) involves leveraging data about employees, such as their demographics, job history, performance reviews, and possibly external factors like market conditions or company policies. Here's an overview of how such a project might be structured:

- Data Collection: The first step is gathering relevant data. This could include employee demographics (age, gender, education level), job-related data (position, department, salary), performance metrics (such as ratings from performance reviews, sales figures, project completion rates), and any other relevant factors that might influence performance.
- Data Preprocessing: Once the data is collected, it needs to be pre-processed. This involves cleaning the data (handling missing values, removing duplicates), transforming variables (converting categorical variables into numerical ones through techniques like one-hot encoding), and scaling features if necessary to ensure all variables contribute equally to the analysis.
- Feature Selection/Engineering: Feature selection involves choosing the most relevant variables that contribute to predicting employee performance. Feature engineering may also be done to create new features from existing ones that could better represent the problem. Techniques like PCA (Principal Component Analysis) or feature importance analysis can help in this stage.
- Model Selection: Various machine learning algorithms can be applied to predict employee performance, such as linear regression, decision trees, random forests, gradient boosting machines, or neural networks. The choice of algorithm depends on factors like the nature of the data, interpretability of the model, and the desired level of accuracy.
- Model Training: The selected model is trained on the preprocessed data. This involves feeding the algorithm with historical data and letting it learn the patterns in the data to make predictions about future employee performance.
- Model Evaluation: After training the model, it needs to be evaluated to assess its performance. This is typically done using metrics like accuracy, precision, recall, F1-score, or area under the ROC curve (AUC),

depending on the nature of the problem (e.g., classification or regression).

- Hyperparameter Tuning: Many machine learning algorithms have hyperparameters that need to be tuned to optimize model performance. Techniques like grid search or random search can be used to find the best combination of hyperparameters.
- Model Deployment: Once the model is trained and evaluated satisfactorily, it can be deployed into production. This involves integrating the model into existing systems or workflows so that it can make real-time predictions about employee performance.
- Monitoring and Maintenance: After deployment, the model needs to be monitored to ensure that it continues to perform well over time. This may involve periodically retraining the model with new data or updating it to account for changes in the business environment.
- Ethical Considerations: Throughout the project, it's important to consider ethical implications, such as fairness, privacy, and bias. Steps should be taken to ensure that the model is fair and unbiased and that employee privacy is respected.

Overall, predicting employee performance using machine learning is a complex but potentially valuable endeavor that can help organizations make more informed decisions about hiring, promotion, and talent management.

1.2. 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.

2. Project initialization and planning

2.1. Define problem statement

The project aims to develop a machine learning model to accurately predict the performance levels of employees within an organization based on a diverse set of factors, including individual attributes, organizational characteristics, and psychological traits. The primary goal is to create a predictive tool that assists in identifying high-performing employees, understanding the factors influencing performance, and informing strategic decision-making in human resource management.

2.2. Project proposal (proposal solution)

This project proposal gives the perfect understanding of the employees performance this will help to increase the productivity of the employees and know the efficiency of employees.

Project Overview	
Objective	• 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.
Scope	We can able to get the real-world hands-on experience on ML project.
Problem Statement	
Description	The project aims to develop a predictive model that accurately forecasts employee performance levels within an organization.
Impact	It will impact the company's productivity in a positive way.
Proposed Solution	
Approach	We use basic ML approach to complete the project
Key Features	Predicting employees performance using prior work.

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**

Hardware		
Computing Resources	CPU/GPU specifications, number of cores	2-intel i5 laptops
Memory	RAM specifications	16 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, NumPy
Development Environment	IDE, version control	Google colab, Git
Data		
Data	Source, size, format	Kaggle dataset, 16 kb, csv format

2.3. Initial project planning

- User interacts with the UI to enter the input.
- Entered input is analysed 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 analysing 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

3. Data collection and preprocessing phase

3.1. Data Collection plan and raw data source identified

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan Template

Section	Description
Project Overview	Predicting the employees performance by taking their pervious data.
Data Collection Plan	We get the Dataset from the kaggle.

Raw Data Sources Identified	List the raw data sources with relevant details (as a short description).

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
garment_workers_productivity	We get the dataset form Kaggle.	https://www.kaggle.com/datasets/utkarshsarbahi/productivity-prediction-of-garment-employees	Excel	16 kb	Public (Available to everyone)

3.2. Data quality report

This dataset captures employee performance metrics derived from their previous work experiences across various industries and roles. It includes attributes such as tenure, performance ratings, productivity metrics, and feedback from supervisors.

Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset	No issue with the data quality	Moderate	No Resolution Plan needed

3.3. Data exploration and preprocessing

Cleanse and preprocess the collected data to handle missing values, outliers, and encode categorical variables. Ensure data quality and consistency for accurate model training.

Section	Description
Data Overview	The Dataset has 15 features and 1187 observations.
Univariate Analysis	Exploration of individual variables.
Bivariate Analysis	Relationships between two variables (correlation, scatter plots).
Multivariate Analysis	Patterns and relationships involving multiple variables.
Outliers and Anomalies	Address outliers and anomalies by implementing robust data cleaning techniques, selecting resilient machine learning algorithms, and utilizing ensemble methods to ensure the accuracy and reliability of the predictive model for employee performance.
Data Preprocessing Code Screenshots	
Loading Data	`#import dataset to the pandas dataframe` `data=pd.read_csv('/content/garments_worker_productivity.csv')`
Handling Missing Data	`#printing first 5 rows` `data.head()`

	`` #handling date and department column data['date']=pd.to_datetime(data['date']) data['month'] = data['date'].dt.month data.drop('date', axis=1, inplace=True) ```
Data Transformation	``` from sklearn.preprocessing import StandardScaler scaler = StandardScaler() x_train_scaled = scaler.fit_transform(x_train) x_test_scaled = scaler.transform(x_test) x_train_scaled=x_train x_test_scaled=x_test ```
Feature Engineering	``` data['quarter']=Encoder.fit_transform(data['quarter']) data['department']=Encoder.fit_transform(data['department']) data['day']=Encoder.fit_transform(data['day']) ```
Save Processed Data	``` #checking the rows and columns of dataset data.shape #checking some info about the datset data.info() ```

4. Model development phase

4.1. Feature selection report

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	Description	Selected (Yes/No)	Reasoning
Date	Date specifies the employees work	No	It does not gives any useful information to the prediction data
quarter	It gives info of which quarter worker work the most	No	It just specifies the time line of the employee so it does not gives any specific data.
department	Specifies the kind of work the worker does	Yes	We convert it from objects to integers.
day	It gives the day employee worked on	Yes	It helps to predict the employees performance more specifically.
team	It specifies in which team he worked in	Yes	It helps to predict the employees performance more specifically.
Targeted_ productivity	It specifies target productiviy	Yes	It is in the form integer so it does not effect the outcome
smv	It is the terminology of Garment worker	Yes	It helps to predict the employees performance more specifically.
Wip	It is the terminology of Garment worker	Yes	It helps to predict the employees performance more specifically.
Over_time	It gives in of the workers overtime	Yes	This help to increase the productivity.

Idle_time	It gives the idle time of the worker	Yes	This helps to find the idle time of the workers.
No_of_style_change	It specifies the no of styles the worker can able to make.	Yes	It helps to predict the employees performance more specifically.
incentive	It specifies the incentives given to the workers.	Yes	It helps to predict the employees performance more specifically.
idle_men	It specifies the idle men awork.	Yes	It helps to predict the employees performance more specifically.
No_of_worker	It gives the info of no of workers in the team	Yes	It is used to predict the productivity of the worker.
Actual_productivity	It specifies the actual productivity of the workers	yes	The ultimate outcome the model will be based on this

4.2. Model selection report

Based on the provided metrics for the three models (Linear Regression, Random Forest Regressor, and XGBoost Regressor), we can make the following observations:

1)Linear Regression:
Moderate Mean Squared Error (MSE) values for both training and testing data.
Relatively low R-squared (R2) scores, indicating weaker fit to the data.
Consistent Mean Absolute Error (MAE) values.

2)Random Forest Regressor:
Lowest Mean Squared Error (MSE) on testing data among the three models, indicating better prediction accuracy.

High R-squared (R2) scores on both training and testing data, suggesting a good fit to the data and capturing more variance.
Consistent Mean Absolute Error (MAE) values.

3)XGBoost Regressor:
Moderate Mean Squared Error (MSE) values on both training and testing data.
Lower R-squared (R2) scores compared to Random Forest Regressor, indicating slightly weaker performance in capturing variance.
Consistent Mean Absolute Error (MAE) values.

Conclusion:
Based on the provided metrics, the Random Forest Regressor appears to be the best-performing model. It demonstrates the lowest Mean Squared Error (MSE) on the testing data, indicating superior prediction accuracy. Additionally, it exhibits high R-squared (R2) scores on both training and testing data, suggesting a robust fit to the data and capturing more variance compared to the other models. Therefore, for this specific task, the Random Forest Regressor is recommended for further exploration and deployment.

Model Selection Report:

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Linear regression model	Moderate Mean Squared Error (MSE) values for both training and testing data. Relatively low R-squared (R2) scores, indicating	We used every hyper parameter which is used in the data set.	mean squared error in training: 0.021829740434257082 mean squared error in testing: 0.021321517772632737 r2_score in training data: 0.3038198342280549 r2_score in test: 0.1970042499190925

	weaker fit to the data. Consistent Mean Absolute Error (MAE) values.		mean_absolute_error in training data : 0.10769706277175743 mean_absolute_error in testing data: 0.10729554202727433
Random forest model	Lowest Mean Squared Error (MSE) on testing data among the three models, indicating better prediction accuracy. High R-squared (R2) scores on both training and testing data, suggesting a good fit to the data and capturing more variance. Consistent Mean Absolute Error (MAE) values.	We used every hyper parameter which is used in the data set.	mean squared error in training: 0.0022752182381708293 mean squared error in testing: 0.011925308844873023 r2_score in training: 0.9274401903683915 r2_score in test data: 0.5508775490117057 mean_absolute_error in training data: 0.10769706277175743 mean_absolute_error in testing: 0.10729554202727433

Xgboost	Moderate Mean Squared Error (MSE) values on both training and testing data. Lower R-squared (R2) scores compared to Random Forest Regressor, indicating slightly weaker performance in capturing variance. Consistent Mean Absolute Error (MAE) values.	We used every hyper parameter which is used in the data set.	mean squared error in training: 0.0036951760704128597 mean squared error in testing data: 0.012631486322544742 r2_score in training data 0.8821558003859931 r2_score in test data: 0.5242819980091831 mean_absolute_error in training data: 0.10769706277175743 mean_absolute_error in testing: 0.10729554202727433

4.3. Initial model training code, model validation and evaluation report

Regression model is the best fit for the employee performance prediction model.

Initial Model Training Code:

```
#splitting data into train test split
#import train_test_split dependency
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=

x.shape,x_train.shape,x_test.shape
```

Model Validation and Evaluation Report:

<table>
<tr><th>Model</th><th>Regression Report</th><th>Accuracy</th><th>Confusion Matrix</th></tr>
<tr>
<td>Linear regression model</td>
<td>#model building
#importing linear regression dependency
from sklearn.linear_model import LinearRegression
linear=LinearRegression()
linear.fit(x_train,y_train)
#linear model mean squared error
score_train=linear.predict(x_train)
mse_train=mean_squared_error(y_train,score_train)
print("mean squared error in training data in linear regression is:",mse_train)

score_test=linear.predict(x_test)
mse_test=mean_squared_error(y_test,score_test)
print("mean squared error in testing data in linear regression is:",mse_test)

#linear model r2_score
score_train=linear.predict(x_train)
(variable) mse_train: Float
mse_train=r2_score(y_train,score_train)
print("r2_score in training data in linear regression is:",mse_train)

score_test=linear.predict(x_test)
mse_test=r2_score(y_test,score_test)
print("r2_score in test data in linear regression is:",mse_test)

#linear model mean_absolute_error
score_train=linear.predict(x_train)
mse_train=mean_absolute_error(y_train,score_train)
print("mean_absolute_error in training data in linear regression is:",mse_train)

score_test=linear.predict(x_test)
mse_test=mean_absolute_error(y_test,score_test)
print("mean_absolute_error in testing data in linear regression is:",mse_test)</td>
<td>0.303</td>
<td><table>
<tr><th></th><th>Actual Value</th><th>predicted_value</th></tr>
<tr><td>921</td><td>0.268214</td><td>0.432858</td></tr>
<tr><td>321</td><td>0.800359</td><td>0.799398</td></tr>
<tr><td>101</td><td>0.681061</td><td>0.671121</td></tr>
<tr><td>920</td><td>0.325000</td><td>0.591028</td></tr>
<tr><td>58</td><td>0.667604</td><td>0.593638</td></tr>
<tr><td>790</td><td>0.800980</td><td>0.735931</td></tr>
<tr><td>948</td><td>0.768847</td><td>0.549655</td></tr>
<tr><td>969</td><td>0.768847</td><td>0.526311</td></tr>
<tr><td>410</td><td>0.650417</td><td>0.631047</td></tr>
<tr><td>1079</td><td>0.750396</td><td>0.750391</td></tr>
</table></td>
</tr>
</table>

Random forest model

```
#Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
RandomForest = RandomForestRegressor()
RandomForest.fit(x_train, y_train)

#Random Forest Regressor mean squared error
score_train=RandomForest.predict(x_train)
mse_train=mean_squared_error(y_train,score_train)
print("mean squared error in training data in Random Forest Regressor is:",mse_train)

score_test=RandomForest.predict(x_test)
mse_test=mean_squared_error(y_test,score_test)
print("mean squared error in testing data in Random Forest Regressor is:",mse_test)

#Random Forest Regressor r2_score
score_train=RandomForest.predict(x_train)
mse_train=r2_score(y_train,score_train)
print("r2_score in training data in Random Forest Regressor is:",mse_train)

score_test=RandomForest.predict(x_test)
mse_test=r2_score(y_test,score_test)
print("r2_score in test data in Random Forest Regressor is:",mse_test)

#Random Forest Regressor mean_absolute_error
score_train=linear.predict(x_train)
mse_train=mean_absolute_error(y_train,score_train)
print("mean_absolute_error in training data in Random Forest Regressor is:",mse_train)

score_test=linear.predict(x_test)
mse_test=mean_absolute_error(y_test,score_test)
print("mean_absolute_error in testing data in Random Forest Regressor is:",mse_test)
```

0.92

	Actual Value	predicted_value
921	0.268214	0.432858
321	0.800359	0.799398
101	0.681061	0.671121
920	0.325000	0.591028
58	0.667604	0.593638
790	0.800980	0.735931
948	0.768847	0.549655
969	0.768847	0.526311
410	0.650417	0.631047
1079	0.750396	0.750391

Xgboost model

```
#Xgboost regression
import xgboost as xgb
model_xgb=xgb.XGBRegressor(n_estimators=200,max_depth=5,learning_rate=0.1)
model_xgb.fit(x_train,y_train)


#Xgboost mean squared error
score_train=model_xgb.predict(x_train)
mse_train=mean_squared_error(y_train,score_train)
print("mean squared error in training data in Xgboost regression is:",mse_train)

score_test=model_xgb.predict(x_test)
mse_test=mean_squared_error(y_test,score_test)
print("mean squared error in testing data in Xgboost regressionr is:",mse_test)

#Xgboost Regressor r2_score
score_train=model_xgb.predict(x_train)
mse_train=r2_score(y_train,score_train)
print("r2_score in training data in Xgboost regression is:",mse_train)

score_test=model_xgb.predict(x_test)
mse_test=r2_score(y_test,score_test)
print("r2_score in test data in Random Xgboost regressionr is:",mse_test)

#Xgboost regression mean_absolute_error
score_train=linear.predict(x_train)
mse_train=mean_absolute_error(y_train,score_train)
print("mean_absolute_error in training data in Xgboost regression is:",mse_train)

score_test=linear.predict(x_test)
mse_test=mean_absolute_error(y_test,score_test)
print("mean_absolute_error in testing data in Xgboost regression is:",mse_test)
```

0.88

	Actual Value	predicted_value
921	0.268214	0.432858
321	0.800359	0.799398
101	0.681061	0.671121
920	0.325000	0.591028
58	0.667604	0.593638
790	0.800980	0.735931
948	0.768847	0.549655
969	0.768847	0.526311
410	0.650417	0.631047
1079	0.750396	0.750391

5. Model optimization and tuning phase

5.1. Hyperparameters tuning documentation

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model		Tuned Hyperparameters
Linear regression model		Quarter Department day
Random forest model		Quarter Department day
Xgboost model		Quarter Department day

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
Model 1	team targeted_productivity smv wip over_time	Quarter Department day

	incentive idle_time idle_men no_of_style_change no_of_workers actual_productivity	
Model 2	team targeted_productivity smv wip over_time incentive idle_time idle_men no_of_style_change no_of_workers actual_productivity	Quarter Department day

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random forest model	Based on the provided metrics, the Random Forest Regressor appears to be the best-performing model. It demonstrates the lowest Mean Squared Error (MSE) on the testing data, indicating superior prediction accuracy. Additionally, it exhibits high R-squared (R2) scores on both training and testing data, suggesting a robust fit to the data and capturing more variance compared to the other models. Therefore, for this specific task, the Random Forest Regressor is recommended for further exploration and deployment.

5.2. Performance metrics comparison report

Based on the provided metrics for the three models (Linear Regression, Random Forest Regressor, and XGBoost Regressor), we can make the following observations:

1)Linear Regression:
Moderate Mean Squared Error (MSE) values for both training and testing data.
Relatively low R-squared (R2) scores, indicating weaker fit to the data.
Consistent Mean Absolute Error (MAE) values.

2)Random Forest Regressor:
Lowest Mean Squared Error (MSE) on testing data among the three models, indicating better prediction accuracy.
High R-squared (R2) scores on both training and testing data, suggesting a good fit to the data and capturing more variance.

Consistent Mean Absolute Error (MAE) values.

3)XGBoost Regressor:
Moderate Mean Squared Error (MSE) values on both training and testing data.
Lower R-squared (R2) scores compared to Random Forest Regressor, indicating slightly weaker performance in capturing variance.
Consistent Mean Absolute Error (MAE) values.

Conclusion:
Based on the provided metrics, the Random Forest Regressor appears to be the best-performing model. It demonstrates the lowest Mean Squared Error (MSE) on the testing data, indicating superior prediction accuracy. Additionally, it exhibits high R-squared (R2) scores on both training and testing data, suggesting a robust fit to the data and capturing more variance compared to the other models. Therefore, for this specific task, the Random Forest Regressor is recommended for further exploration and deployment

5.3. Final model selection justification

Based on the provided metrics for the three models (Linear Regression, Random Forest Regressor, and XGBoost Regressor), we can make the following observations:

1)Linear Regression:
Moderate Mean Squared Error (MSE) values for both training and testing data.
Relatively low R-squared (R2) scores, indicating weaker fit to the data.
Consistent Mean Absolute Error (MAE) values.

2)Random Forest Regressor:
Lowest Mean Squared Error (MSE) on testing data among the three models, indicating better prediction accuracy.
High R-squared (R2) scores on both training and testing data, suggesting a good fit to the data and capturing more variance.
Consistent Mean Absolute Error (MAE) values.

3)XGBoost Regressor:
Moderate Mean Squared Error (MSE) values on both training and testing data.
Lower R-squared (R2) scores compared to Random Forest Regressor, indicating slightly weaker performance in capturing variance.
Consistent Mean Absolute Error (MAE) values.

Conclusion:
Based on the provided metrics, the Random Forest Regressor appears to be the best-performing model. It demonstrates the lowest Mean Squared Error (MSE) on the testing data, indicating superior prediction accuracy. Additionally, it exhibits high R-squared (R2) scores on both training and testing data, suggesting a robust fit to the data and capturing more variance compared to the other models. Therefore, for this specific task, the Random Forest Regressor is recommended for further exploration and deployment.

6. Result

6.1. Output Screenshots

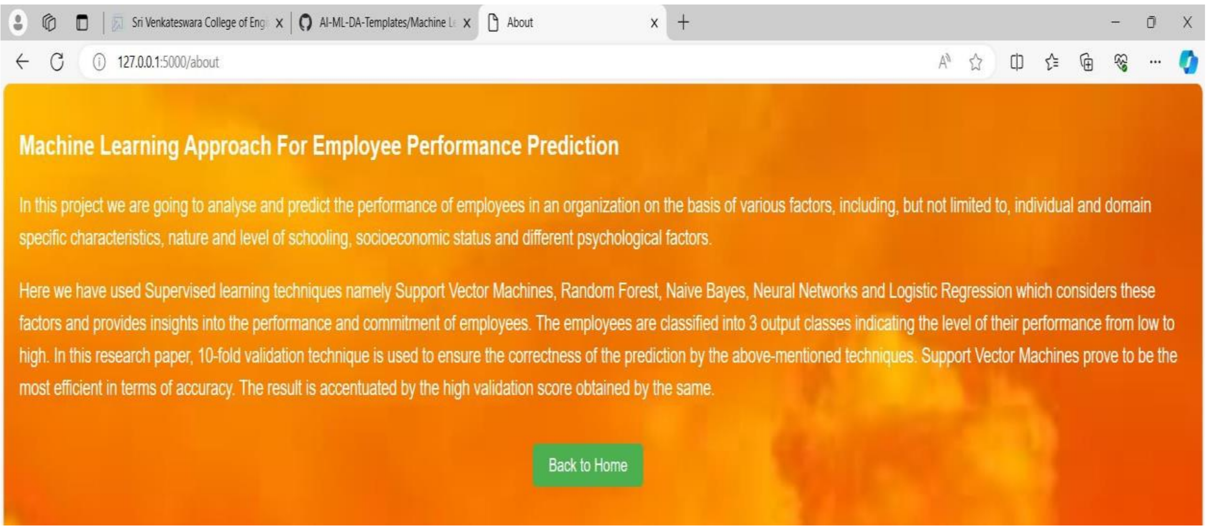
Home Page:

7.

- When we click on the “Predict” button which is on the top right of my web page it willredirects to the another page where we can give inputs to our model.
- When we click on “About” button which is on the top right of my web page it will redirects tothe another page where we find some details about my web page.

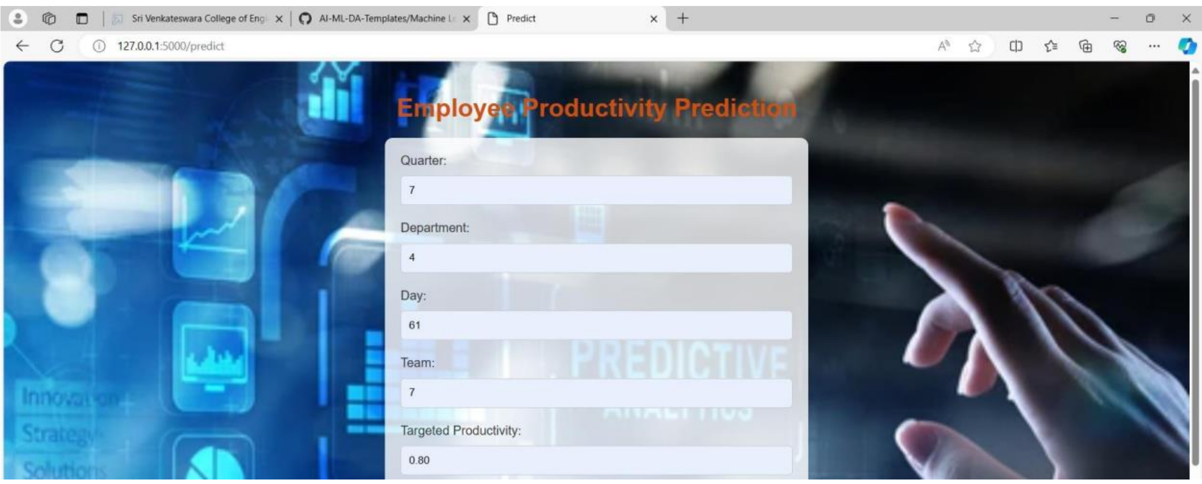
About page:

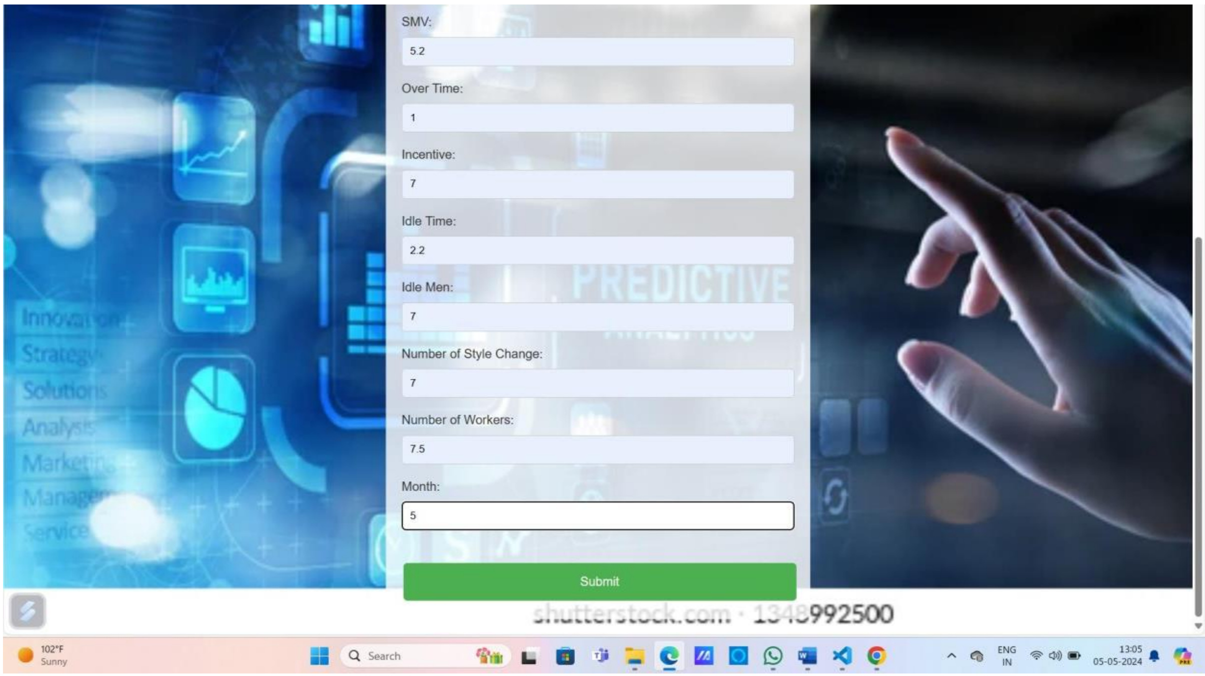
•



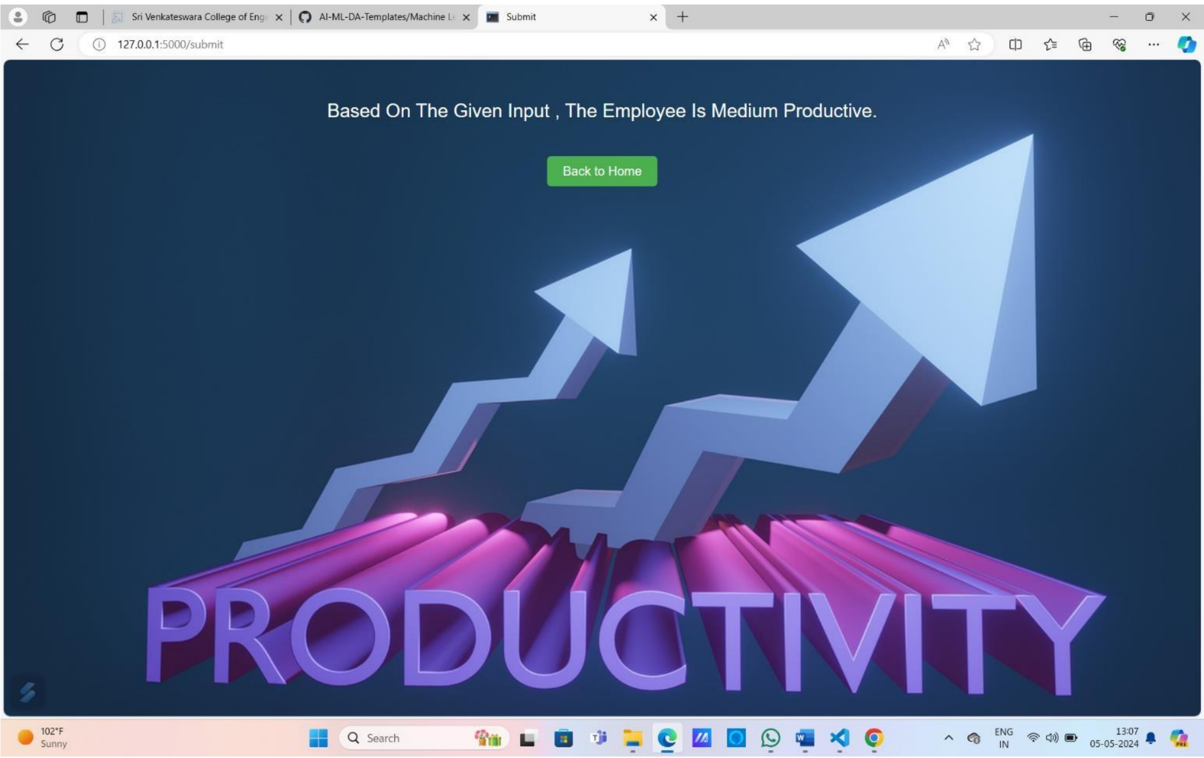
- When we click on “Back to Home” button which is on the bottom of the content of my webpage it will redirects to the home page again.
- When we click on the “Predict” button which is on the top right of home page of my webpage it will redirects to the another page where we can give inputs to our model.

Input 1:

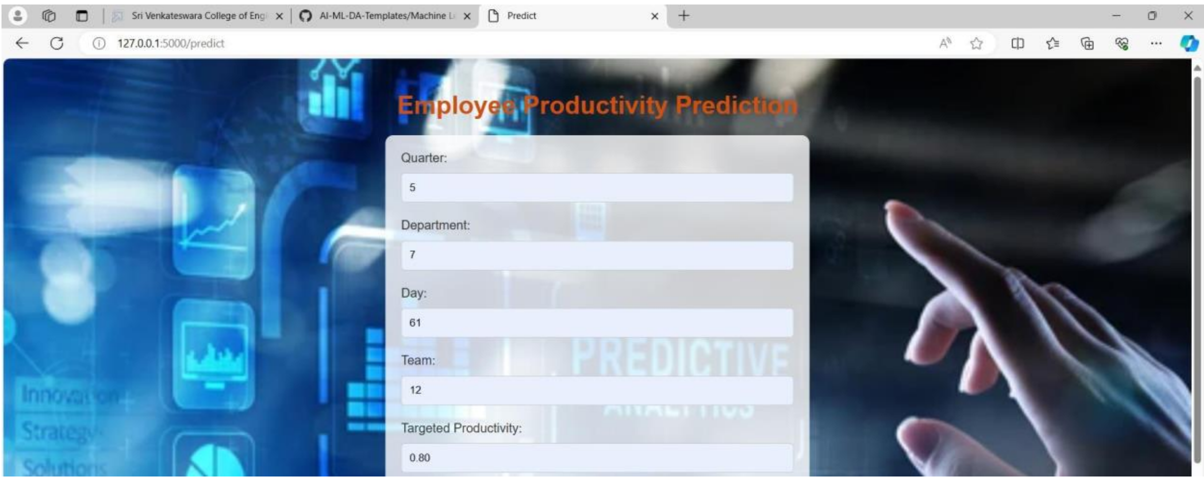


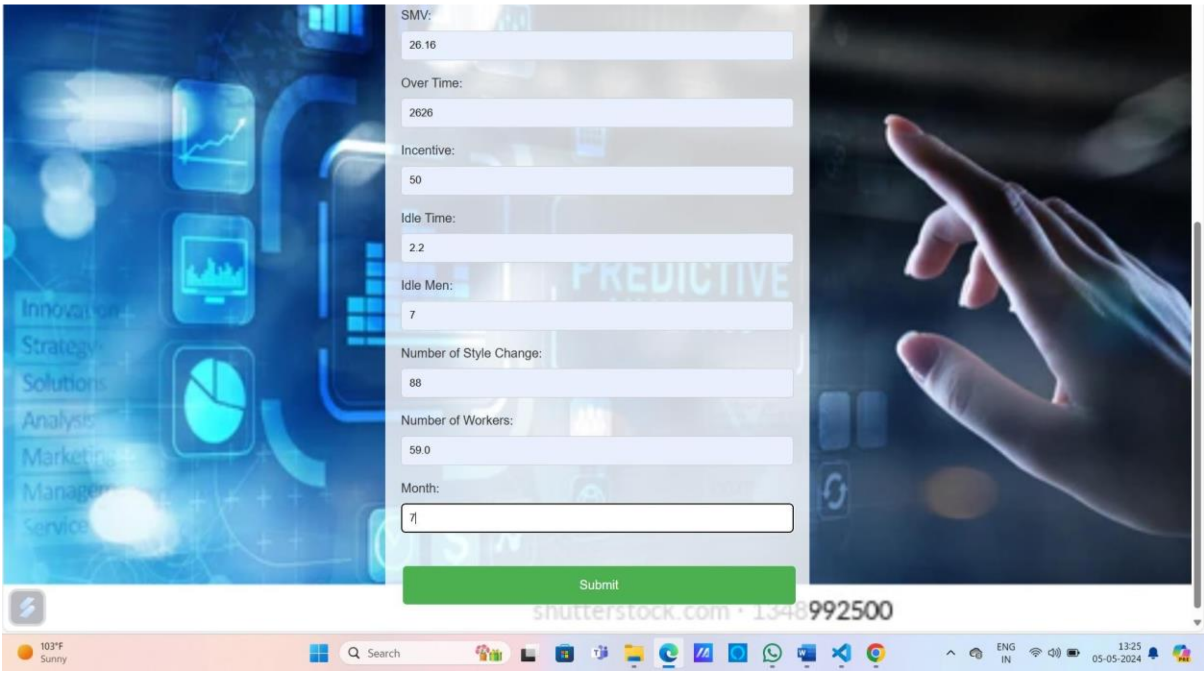

SMV:
5.2
Over Time:
1
Incentive:
7
Idle Time:
2.2
Idle Men:
7
Number of Style Change:
7
Number of Workers:
7.5
Month:
5
Submit
shutterstock.com · 1348992500

Output 1:

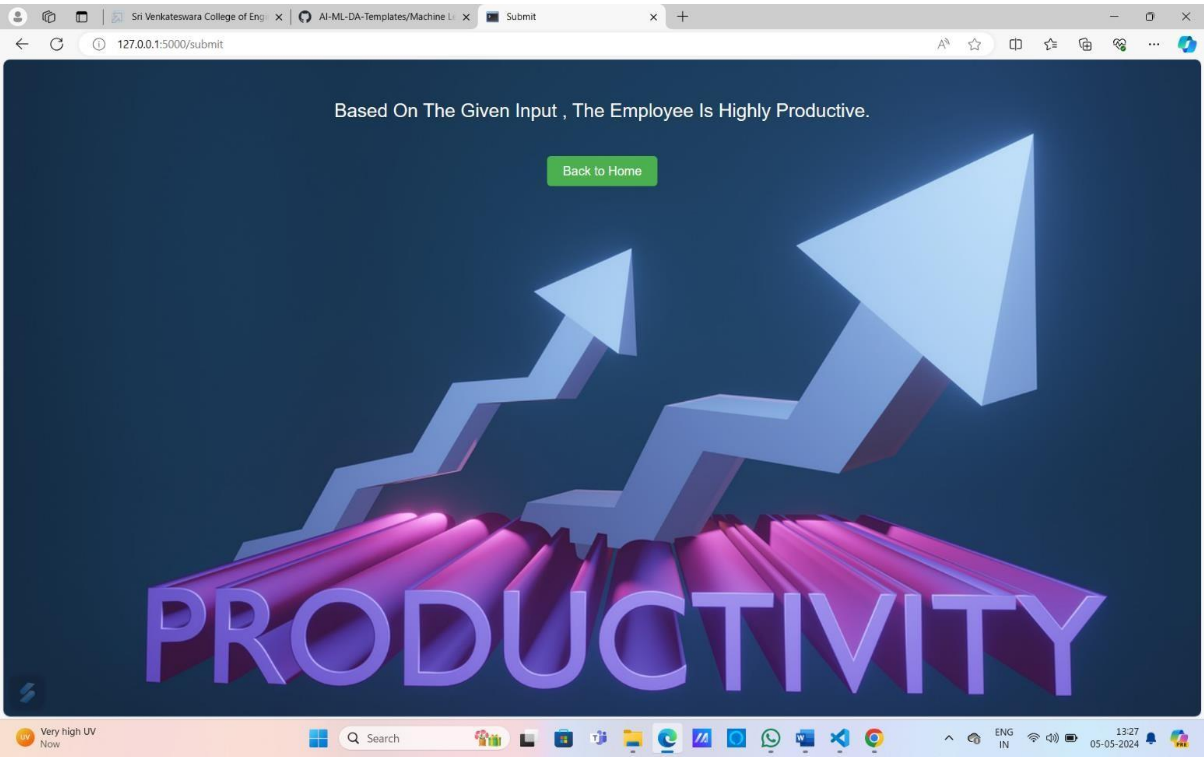

127.0.0.1:5000/submit
Based On The Given Input , The Employee Is Medium Productive.
Back to Home
PRODUCTIVITY

Input 2:





Output 2:



- When we click on "Back to Home" button which is on the bottom of the result of my web
- page it will redirects to the home page again.

7. Advantages and Disadvantages

Advantages

1. **Robustness and Stability**:
    - Random Forest aggregates results from multiple decision trees, reducing the risk of overfitting. This characteristic is particularly useful when predicting complex outcomes like employee performance, where data can be noisy or incomplete.
2. **High Accuracy**:
    - It typically delivers strong performance across a variety of tasks. The ensemble nature of the algorithm allows it to achieve high accuracy, which is desirable for employee performance prediction.
3. **Handles Different Data Types**:
    - Random Forest can work with both numerical and categorical data, providing flexibility in handling the diverse datasets often found in HR or employee-related projects.
4. **Feature Importance**:
    - It provides insights into feature importance, allowing you to understand which factors contribute most to employee

performance. This can be valuable for HR decision-making and strategic planning.

Disadvantages

1. **Lack of Interpretability**:
   - Random Forests are considered "black-box" models, making them difficult to interpret. This can be a significant drawback in HR contexts where explainability is crucial for decision-making and compliance.
2. **Resource-Intensive**:
   - Building a Random Forest requires significant computational resources, especially with large datasets and a high number of trees. This factor might be a limitation for smaller HR departments or projects with limited resources.
3. **Hyperparameter Tuning**:
   - The performance of Random Forest can depend on the correct tuning of hyperparameters, like the number of trees, max depth, and others. This process can be complex and time-consuming.
4. **Potential Overfitting with Large Forests**:
   - Although generally robust against overfitting, if the number of trees is too high or the depth is too great, overfitting could occur.

8. Conclusion

This project 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. The performance is evaluated successfully.

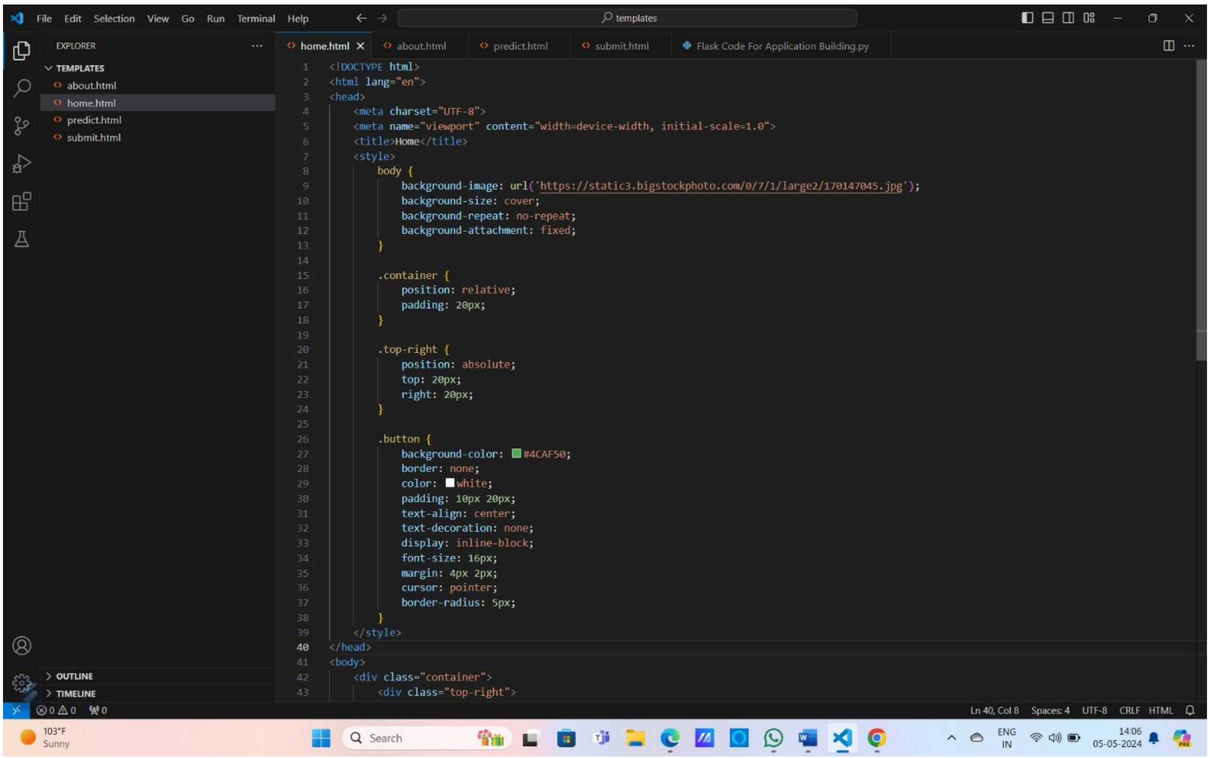
9. Future Scope

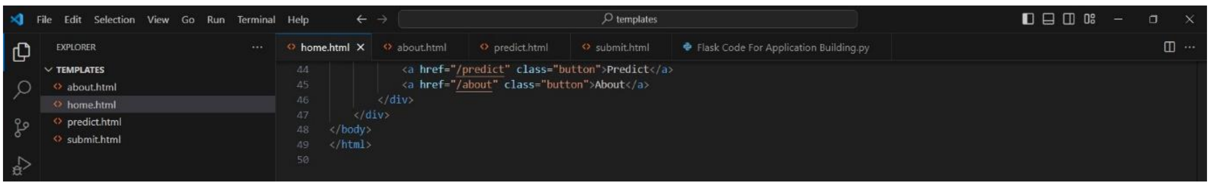
1. Integration with HR Systems
2. Enhanced Data Collection and Analysis
3. Ethical and Bias Considerations
4. Advanced Predictive Analytics
5. Employee Well-being and Retention

10. Appendix

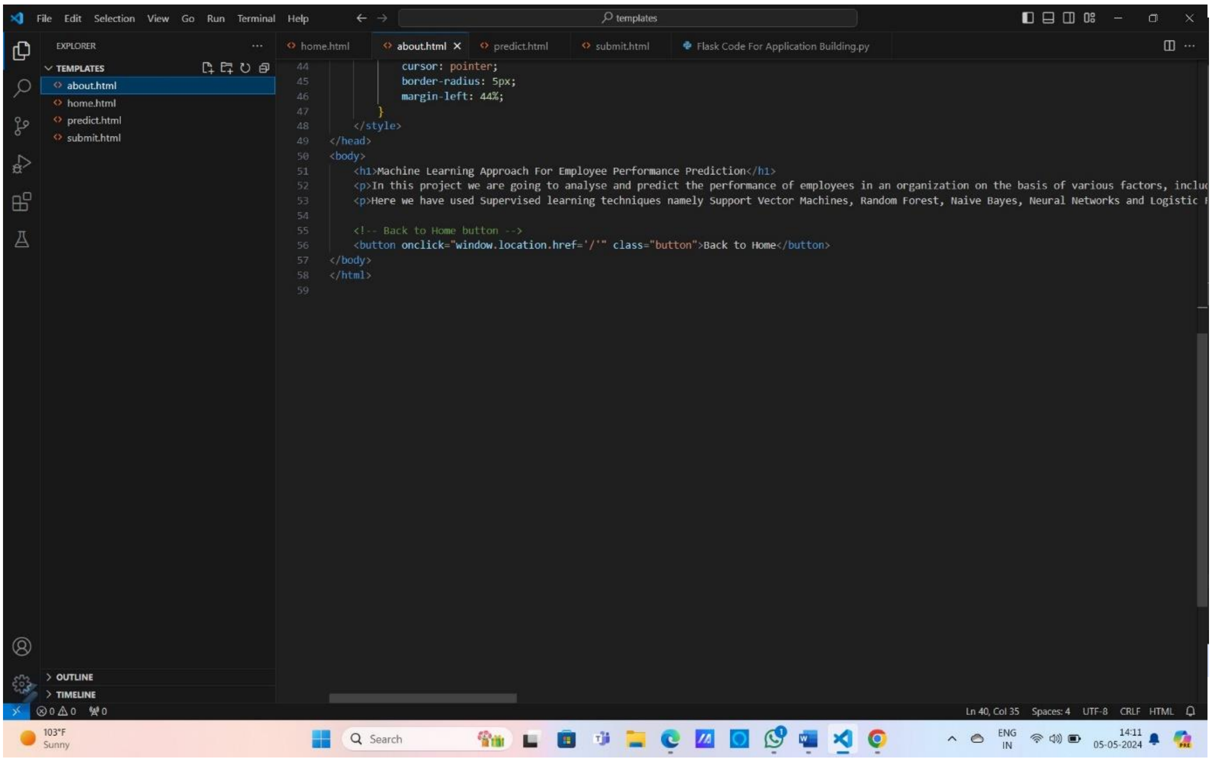
10.1. Source code

home.html



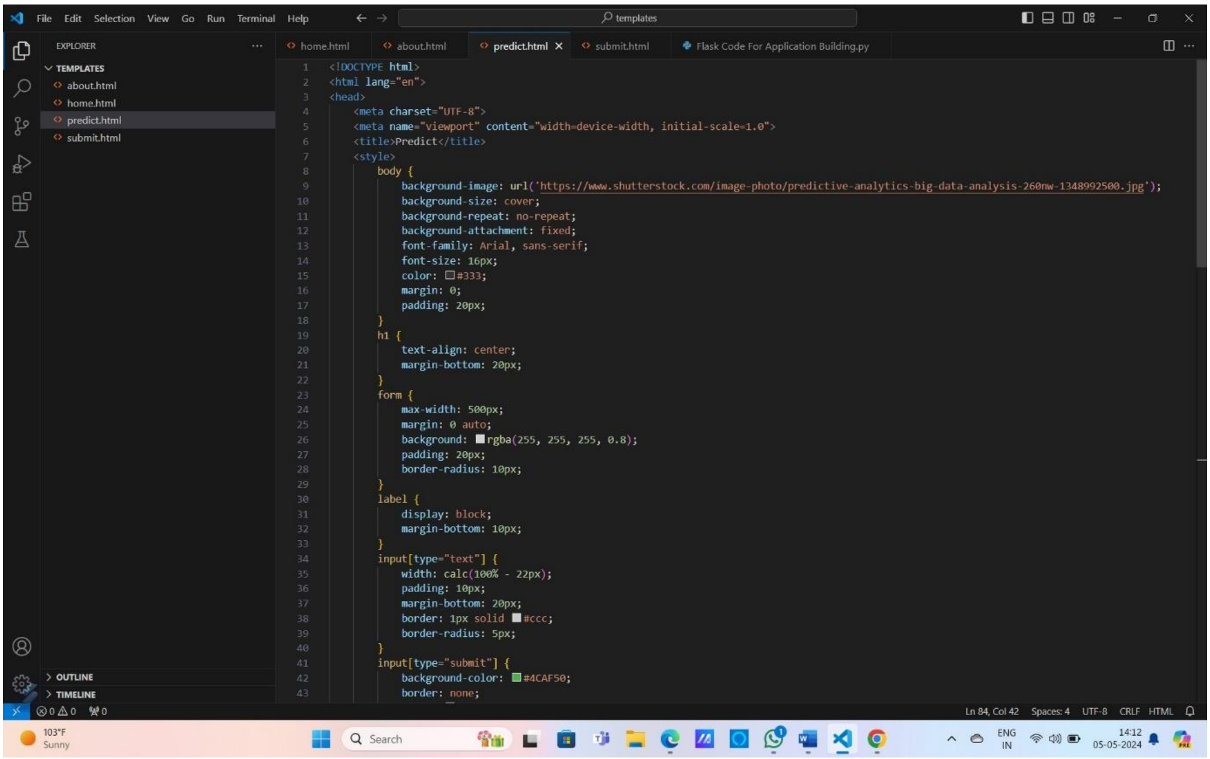


about.html

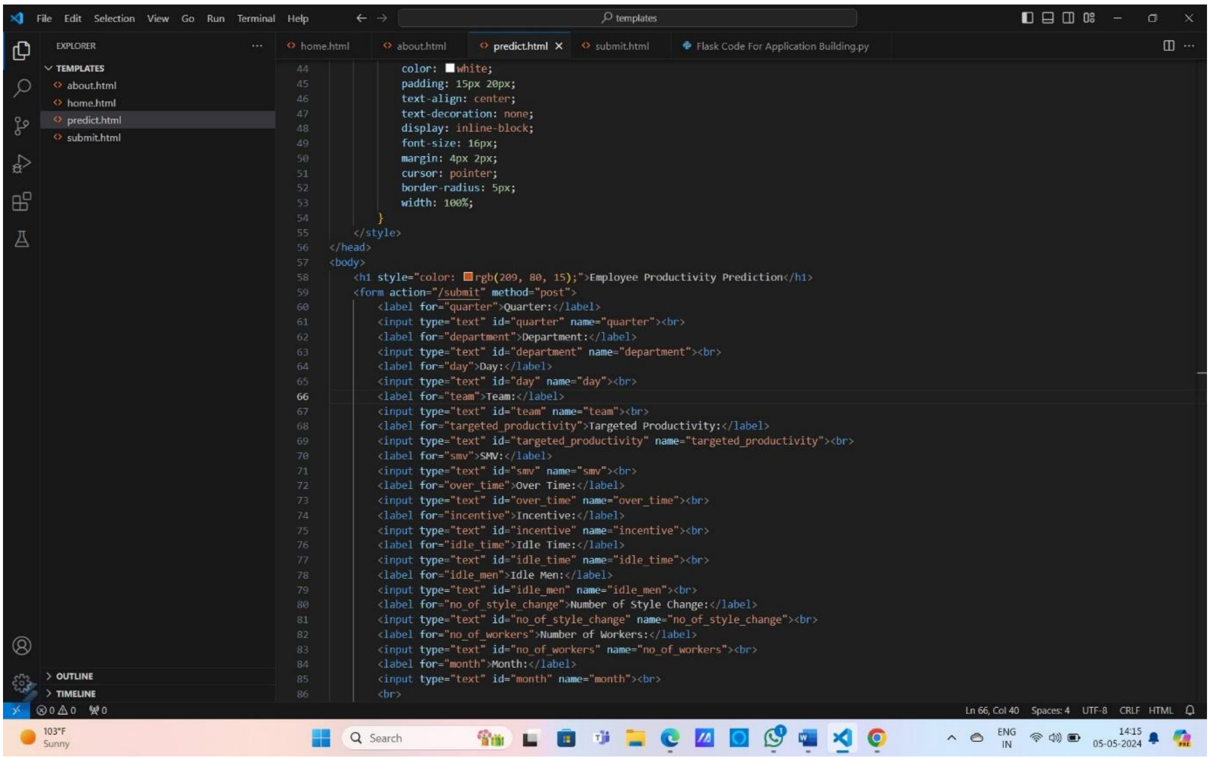


```
            cursor: pointer;
            border-radius: 5px;
            margin-left: 44%;
        }
    </style>
</head>
<body>
    <h1>Machine Learning Approach For Employee Performance Prediction</h1>
    <p>In this project we are going to analyse and predict the performance of employees in an organization on the basis of various factors, incluc
    <p>Here we have used Supervised learning techniques namely Support Vector Machines, Random Forest, Naive Bayes, Neural Networks and Logistic R

    <!-- Back to Home button -->
    <button onclick="window.location.href='/'" class="button">Back to Home</button>
</body>
</html>
```

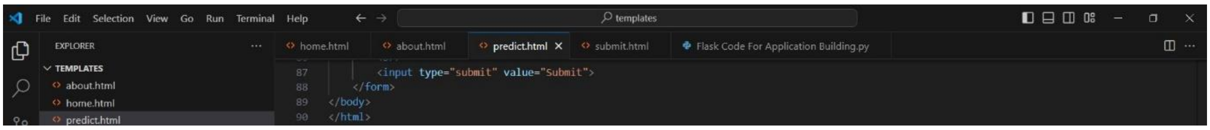
Predict.html



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Predict</title>
    <style>
        body {
            background-image: url('https://www.shutterstock.com/image-photo/predictive-analytics-big-data-analysis-260nw-1348992500.jpg');
            background-size: cover;
            background-repeat: no-repeat;
            background-attachment: fixed;
            font-family: Arial, sans-serif;
            font-size: 16px;
            color: #333;
            margin: 0;
            padding: 20px;
        }
        h1 {
            text-align: center;
            margin-bottom: 20px;
        }
        form {
            max-width: 500px;
            margin: 0 auto;
            background: rgba(255, 255, 255, 0.8);
            padding: 20px;
            border-radius: 10px;
        }
        label {
            display: block;
            margin-bottom: 10px;
        }
        input[type="text"] {
            width: calc(100% - 22px);
            padding: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        input[type="submit"] {
            background-color: #4CAF50;
            border: none;
```

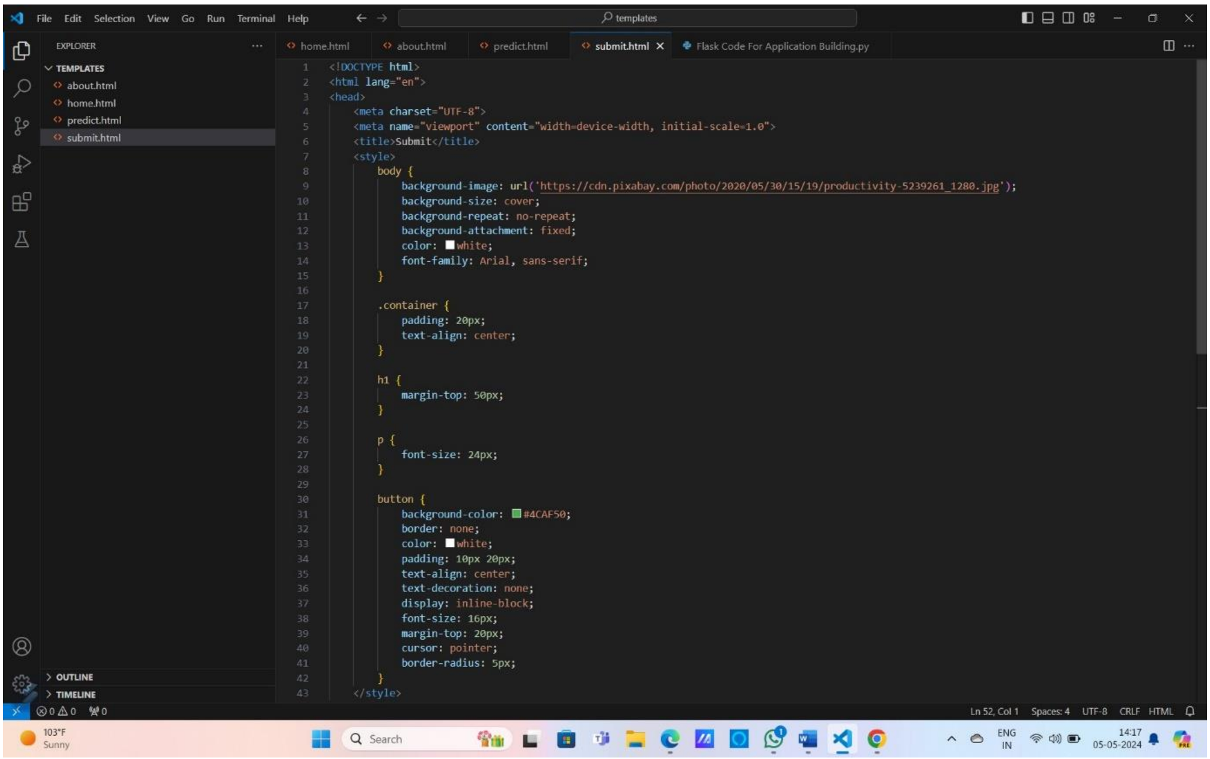


```html
                color: white;
                padding: 15px 20px;
                text-align: center;
                text-decoration: none;
                display: inline-block;
                font-size: 16px;
                margin: 4px 2px;
                cursor: pointer;
                border-radius: 5px;
                width: 100%;
            }
        </style>
    </head>
    <body>
        <h1 style="color: rgb(209, 80, 15);">Employee Productivity Prediction</h1>
        <form action="/submit" method="post">
            <label for="quarter">Quarter:</label>
            <input type="text" id="quarter" name="quarter"><br>
            <label for="department">Department:</label>
            <input type="text" id="department" name="department"><br>
            <label for="day">Day:</label>
            <input type="text" id="day" name="day"><br>
            <label for="team">Team:</label>
            <input type="text" id="team" name="team"><br>
            <label for="targeted_productivity">Targeted Productivity:</label>
            <input type="text" id="targeted_productivity" name="targeted_productivity"><br>
            <label for="smv">SMV:</label>
            <input type="text" id="smv" name="smv"><br>
            <label for="over_time">Over Time:</label>
            <input type="text" id="over_time" name="over_time"><br>
            <label for="incentive">Incentive:</label>
            <input type="text" id="incentive" name="incentive"><br>
            <label for="idle_time">Idle Time:</label>
            <input type="text" id="idle_time" name="idle_time"><br>
            <label for="idle_men">Idle Men:</label>
            <input type="text" id="idle_men" name="idle_men"><br>
            <label for="no_of_style_change">Number of Style Change:</label>
            <input type="text" id="no_of_style_change" name="no_of_style_change"><br>
            <label for="no_of_workers">Number of Workers:</label>
            <input type="text" id="no_of_workers" name="no_of_workers"><br>
            <label for="month">Month:</label>
            <input type="text" id="month" name="month"><br>
            <br>
```

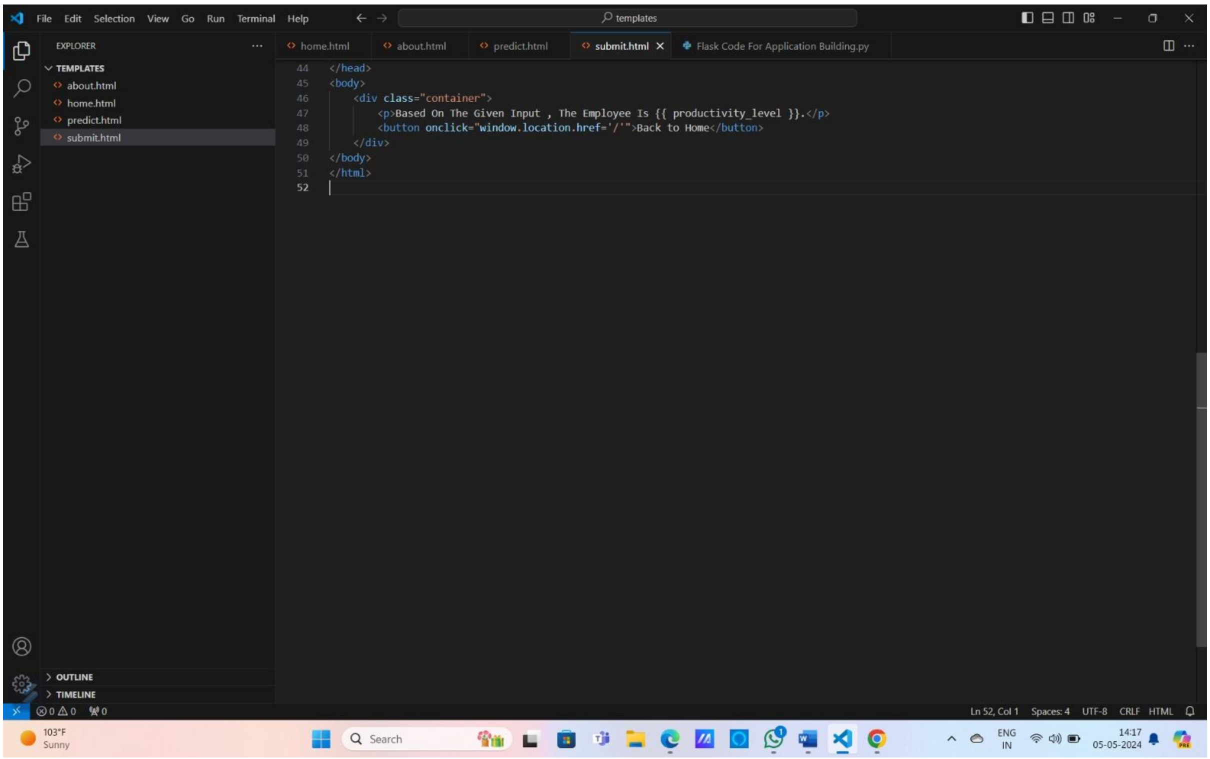


```html
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

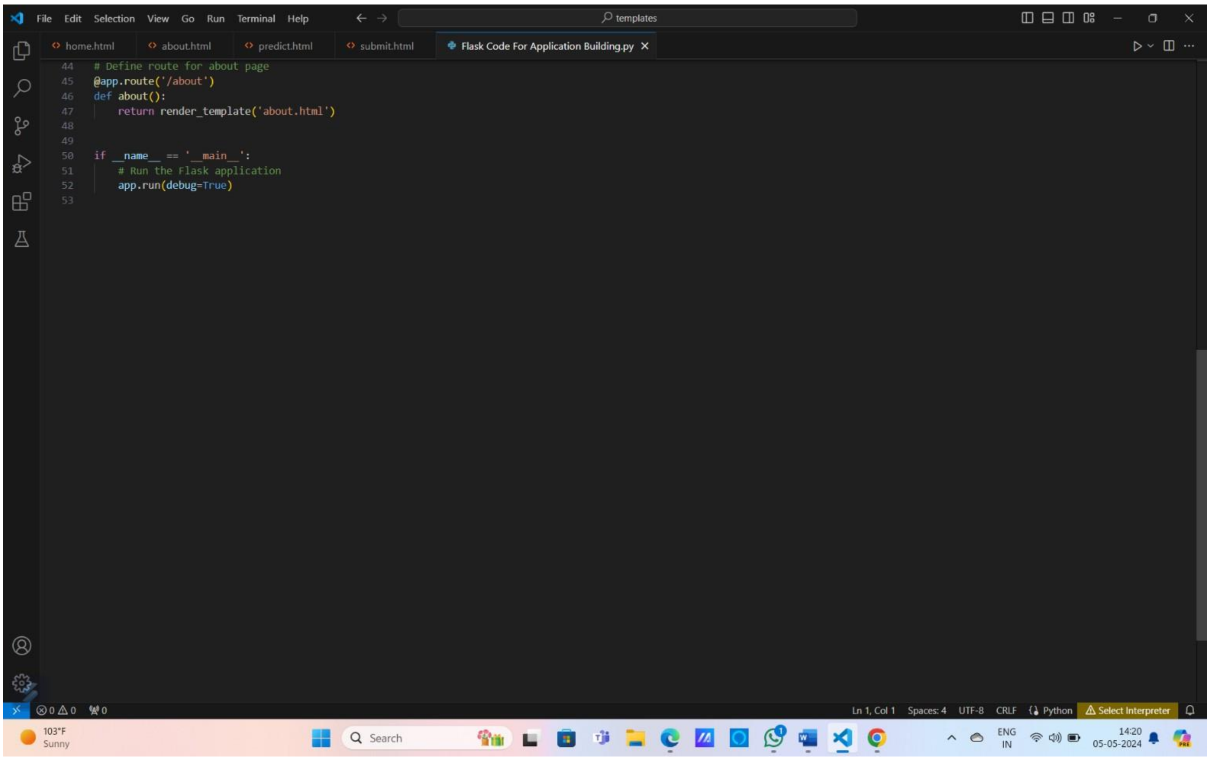
Submit.html

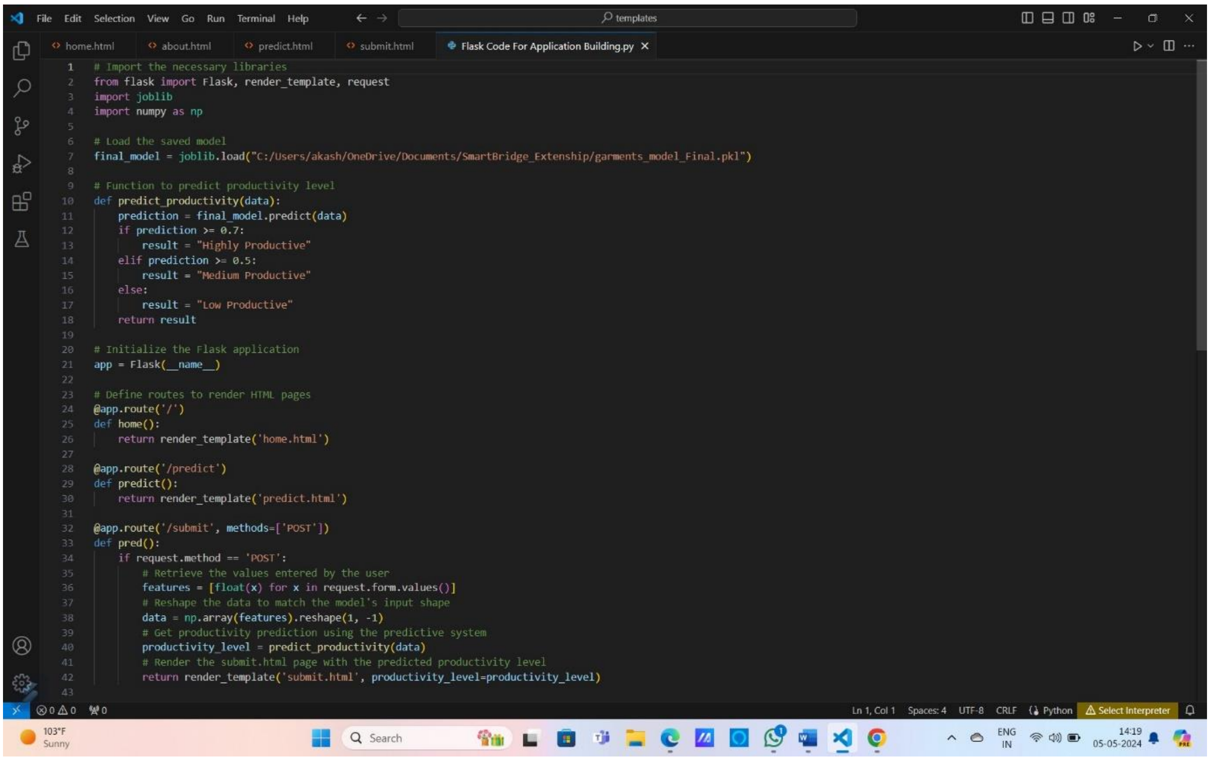


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Submit</title>
    <style>
        body {
            background-image: url('https://cdn.pixabay.com/photo/2020/05/30/15/19/productivity-5239261_1280.jpg');
            background-size: cover;
            background-repeat: no-repeat;
            background-attachment: fixed;
            color: white;
            font-family: Arial, sans-serif;
        }

        .container {
            padding: 20px;
            text-align: center;
        }

        h1 {
            margin-top: 50px;
        }

        p {
            font-size: 24px;
        }

        button {
            background-color: #4CAF50;
            border: none;
            color: white;
            padding: 10px 20px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            margin-top: 20px;
            cursor: pointer;
            border-radius: 5px;
        }
    </style>
```



Flask Code For Application Building.py





10.2. GitHub & Project Link

GitHub Link:

https://github.com/gvdinesh15/Smartbridge_emloyeePerformencePrediction

Project demo Link:

https://drive.google.com/file/d/1alwh6cdnzww3GetH_Sgp8Q4hrE472WSr/view?usp=sharing