SMARTBRIDGE
Let's Bridge the Gap

Smart Internz

h

EMPLOYEE PERFORMANCE PREDICTION USING MACHINE LEARNING

EMPLOYEE PERFORMANCE PREDICTION USING MACHINE LEARNING PROJECT REPORT

Executive Summary :

Employee Performance Prediction is an advanced machine learning project designed to predict and evaluate employee efficiency based on various HR and productivity-related parameters. The system utilizes supervised learning algorithms to analyze employee data and provide accurate performance predictions along with actionable HR recommendations. By leveraging these insights, the model helps organizations make data-driven decisions for promotions, appraisals, and workforce optimization, ultimately enhancing overall productivity and decision-making accuracy.

Project Scenarios

Scenario 1 : HR Decision support :

An HR manager can input employee metrics such as attendance, experience, and training hours into the system.

The system predicts the employee's expected performance score, helping HR make objective promotion and appraisal decisions.

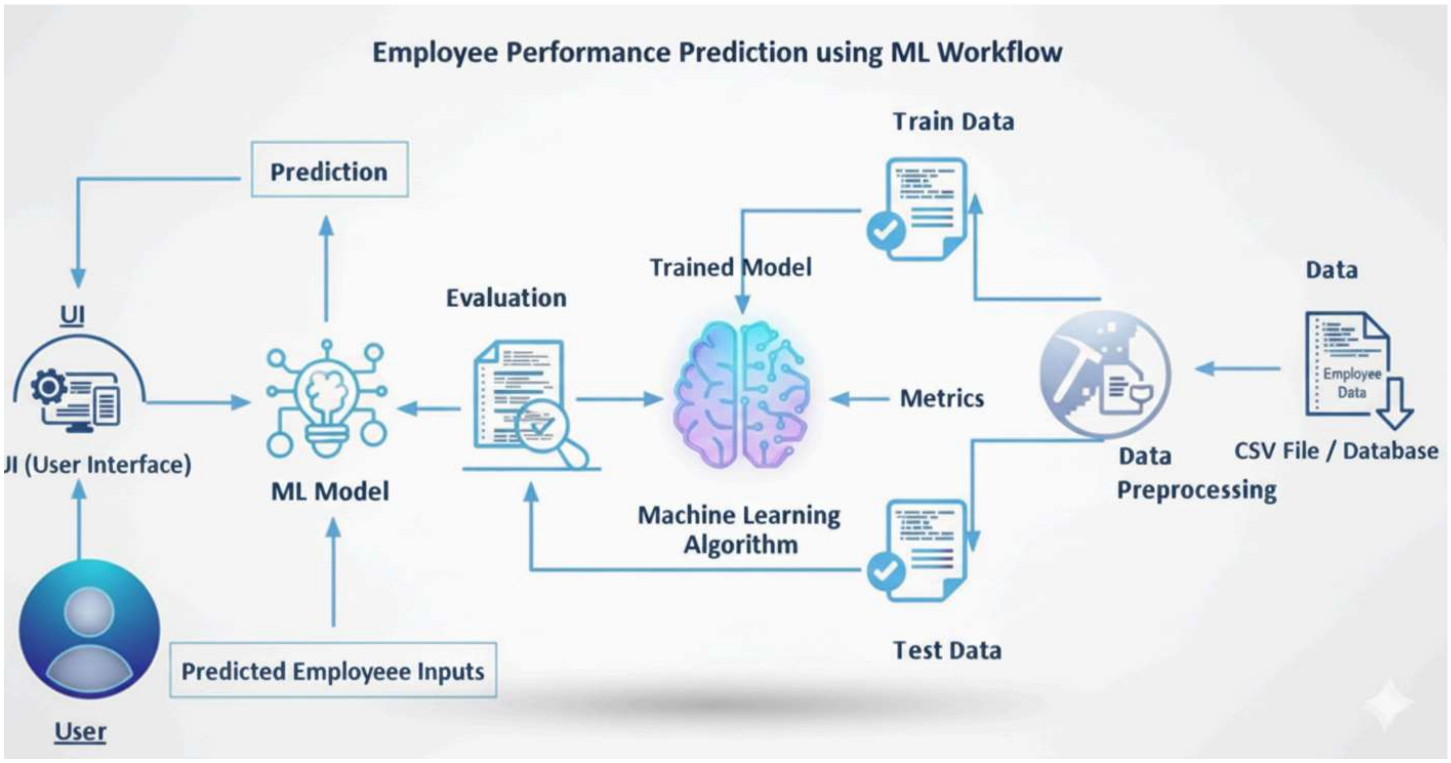
Scenario 2: Performance Improvement Planning:

The model identifies employees with lower predicted performance, enabling HR teams to design personalized training and mentoring programs to improve productivity.

Scenario 3: Workforce Optimization:

Managers can use the prediction system to allocate projects and resources efficiently based on predicted employee performance levels.

Technical Architecture :



Project Flow:

- *User interacts with the UI to enter the input*
- *Entered input is analyzed by the model which is integrated*
- *Once the model analyses the input the prediction is showcased on the UI*

Project Activities:

- ***Data Collection & Preparation***
  - *Collect the data*
  - *Data Preparation*
- ***Exploratory Data Analysis***
  - *Descriptive statistical analysis*
  - *Visual Analysis*

- ***Model Building***
    - *Training the model in multiple algorithms*
    - *Model selection and evaluation*
- ***Model Deployment***
    - *Save the best model*
    - *Integrate with Web Framework*

Prerequisites

Software Requirements:

- ***Anaconda Navigator and Visual Studio***
- ***Python packages:***
    - *numpy*
    - *pandas*
    - *scikit-learn*
    - *matplotlib*
    - *scipy*
    - *pickle*
    - xgboost
    - *seaborn*
    - *Flask*

Prior Knowledge Required:

- ***ML Concepts***
    - *Supervised learning*
    - *Unsupervised learning*
    - *Linear regression*
    - Random Forest
    - Xgboost

- ***Flask Basics***

Milestone 1: Data Collection & Preparation

Data collection is fundamental to machine learning, providing the raw material for training algorithms and making predictions.for employee performance prediction we use different aspects to make perfect decisions about the employee performance.

Activity 1: Dataset Collection

*The **Employee Performance Prediction** project utilizes a dataset containing information about the garment manufacturing process and the productivity of working teams. The data provides a comprehensive view of operational, resource, and time-based factors. - link*

The dataset includes the following features:

Time and Contextual Information

- ***date**: Date of the record (MM-DD-YYYY format).*
- ***day**: Day of the week (temporal context).*
- ***quarter**: A portion of the month (month divided into four quarters).*
- ***department**: The associated production department (e.g., Sewing, Finishing).*
- ***team**: The associated team number.*

Operational and Goal Metrics

- ***targeted_productivity**: The productivity goal set by the authority for each team on a daily basis.*
- ***smv (Standard Minute Value)**: The allocated time for a specific task or operation.*
- ***wip (Work in Progress)**: The number of unfinished items (includes missing values).*
- ***no_of_style_change**: The number of changes in the style of a particular product.*

Resource, Effort, and Interruption Factors

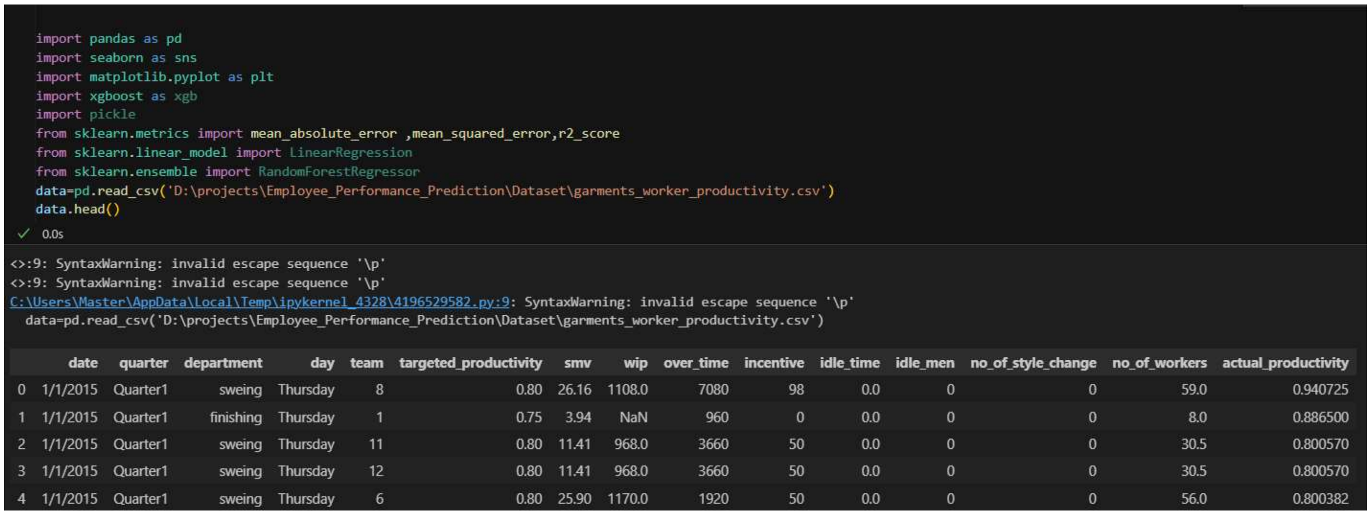
- ***no_of_workers**: The number of workers in each team.*
- ***over_time**: The amount of overtime worked by each team in minutes.*
- ***incentive**: The amount of financial incentive provided to motivate action.*
- ***idle_time**: The amount of time (in minutes) when production was interrupted.*
- ***idle_men**: The number of workers who were idle due to production interruption.*

Target Variable

- ***actual_productivity****: The actual percentage of productivity delivered by the workers, ranging from 0 to 1 (the value your model predicts).*

Activity 1: Data Preparation :

Data Loading and InitialExploration:



```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import xgboost as xgb
import pickle
from sklearn.metrics import mean_absolute_error ,mean_squared_error,r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
data=pd.read_csv('D:\projects\Employee_Performance_Prediction\Dataset\garments_worker_productivity.csv')
data.head()
```

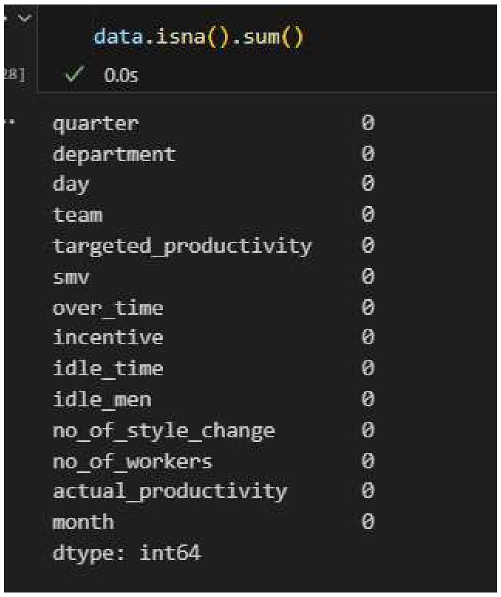
```
✓ 0.0s
```

```
<>:9: SyntaxWarning: invalid escape sequence '\p'
<>:9: SyntaxWarning: invalid escape sequence '\p'
C:\Users\Master\AppData\Local\Temp\ipykernel_4328\4196529582.py:9: SyntaxWarning: invalid escape sequence '\p'
  data=pd.read_csv('D:\projects\Employee_Performance_Prediction\Dataset\garments_worker_productivity.csv')
```

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59.0	0.940725
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8.0	0.886500
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0	0.800382

.Data Cleaning Steps:

- ***Handling Missing Values:****The dataset showed no missing values after inspection*

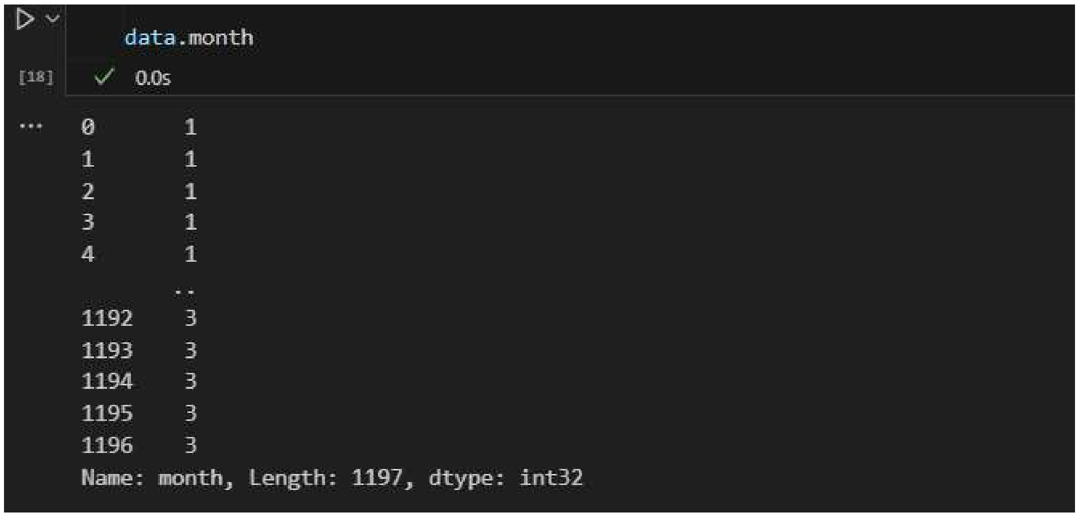


```
data.isna().sum()
```

```
✓ 0.0s
```

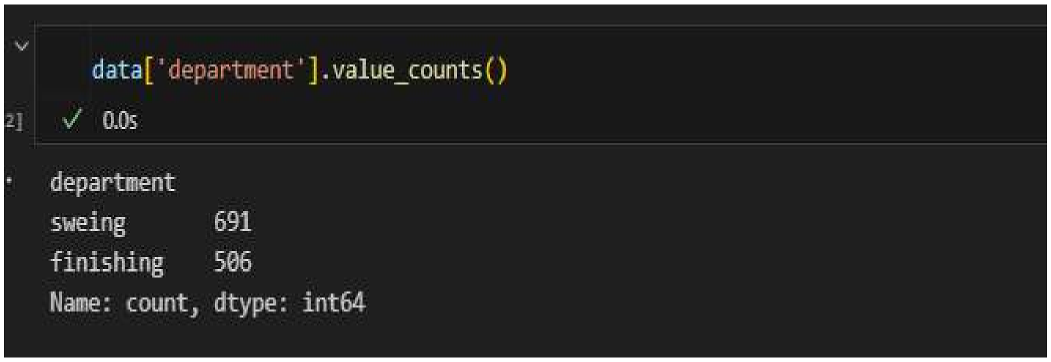
```
quarter                  0
department               0
day                      0
team                     0
targeted_productivity    0
smv                      0
over_time                0
incentive                0
idle_time                0
idle_men                 0
no_of_style_change       0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

- ***Handling Date : After converting date column to month***



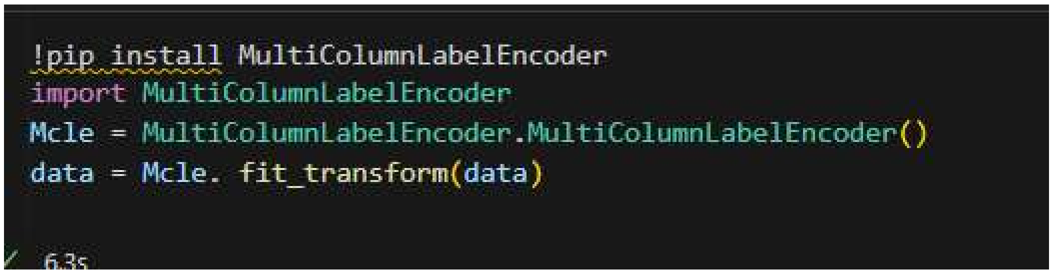
```
data.month
[18]  ✓ 0.0s

...  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
```

- ***Handling department column :** after reducing mistakes*



```
data['department'].value_counts()
2]  ✓ 0.0s

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

- ***Inconsistency Corrections:** standardized categorical values*

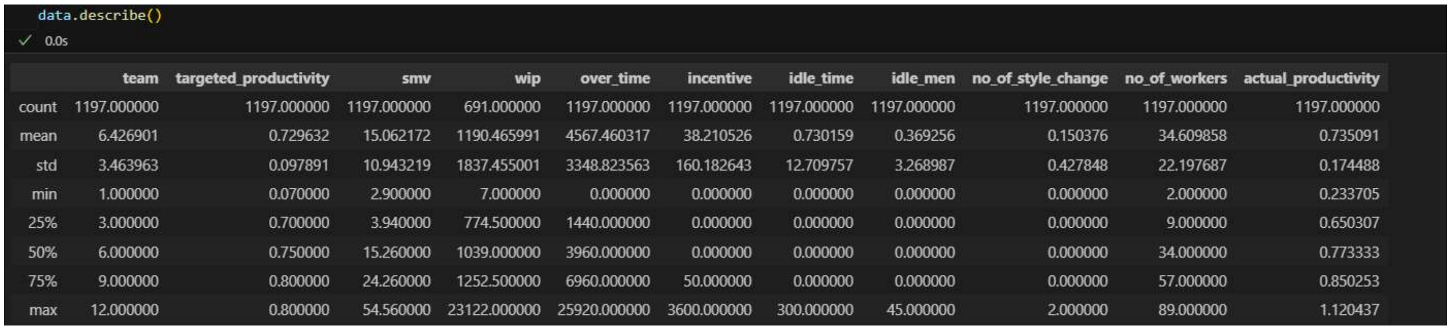


```
!pip install MultiColumnLabelEncoder
import MultiColumnLabelEncoder
Mcle = MultiColumnLabelEncoder.MultiColumnLabelEncoder()
data = Mcle. fit_transform(data)

✓ 6.3s
```

Milestone 2: Exploratory Data Analysis

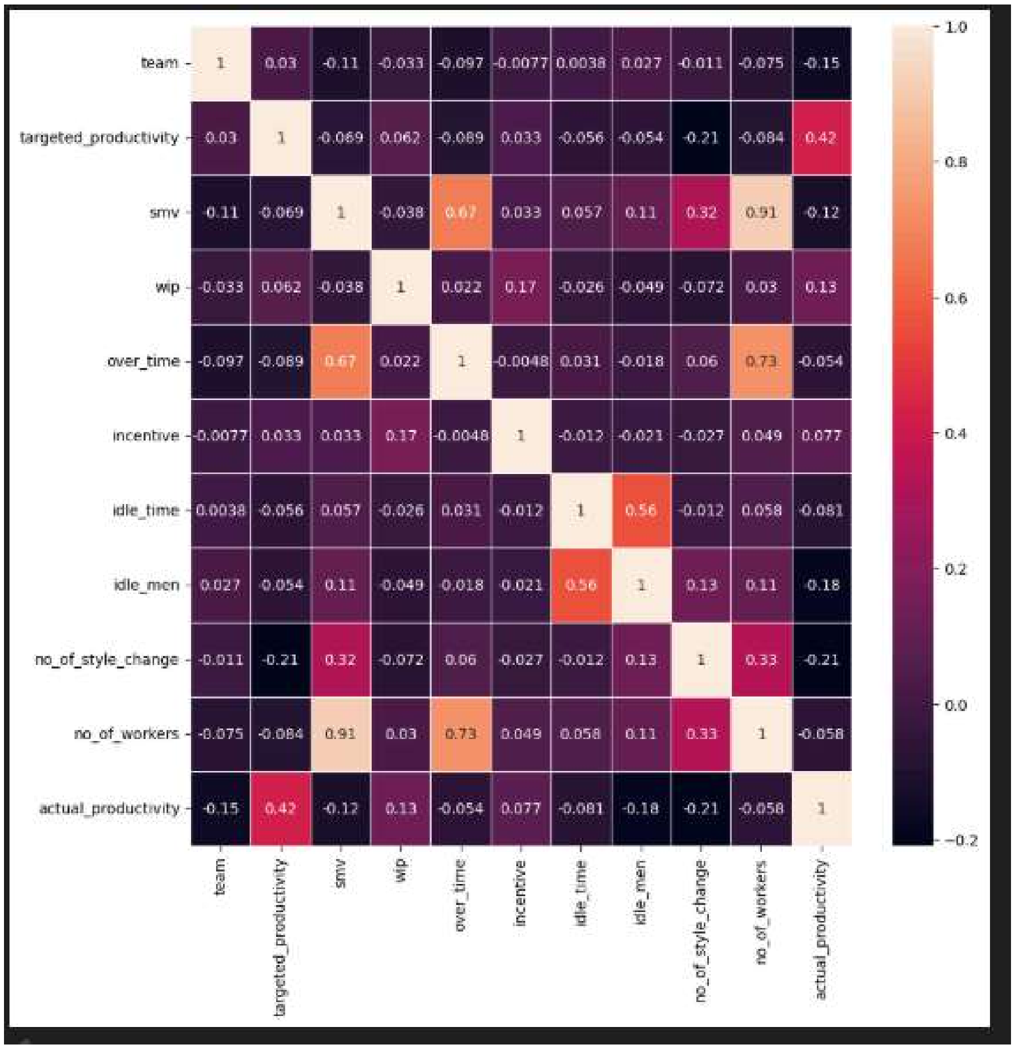
Activity 1: Descriptive Statistics

The describe() functionprovided insights into the distribution of features, revealingbalanced representation across different categories.



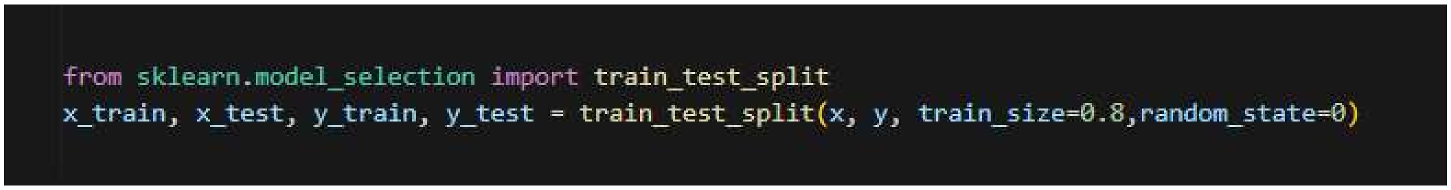
```
data.describe()
✓ 0.0s

             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
```

Activity 2: VisualAnalysis :



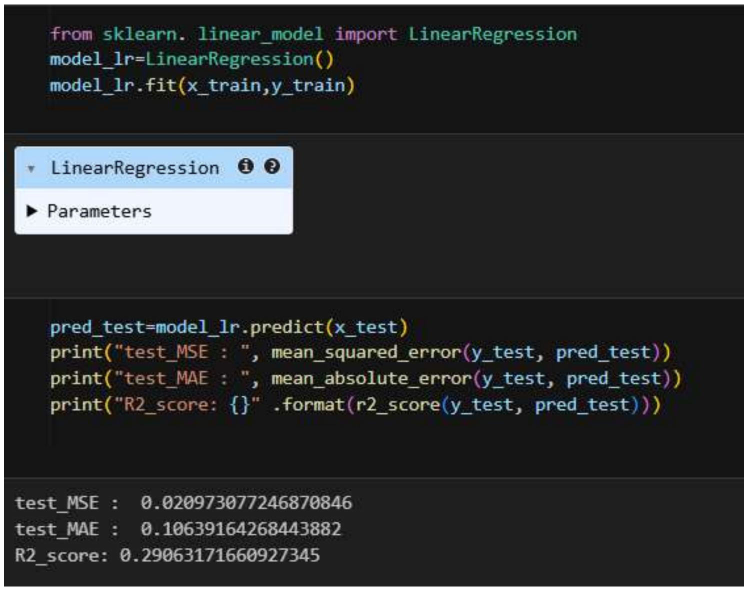
Milestone 3: Model Building & Selection

Activity 1: Data Spliting



```
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)
```

The following models were tested and compared using regression metrics (MSE, MAE, R²):
- Linear Regression:

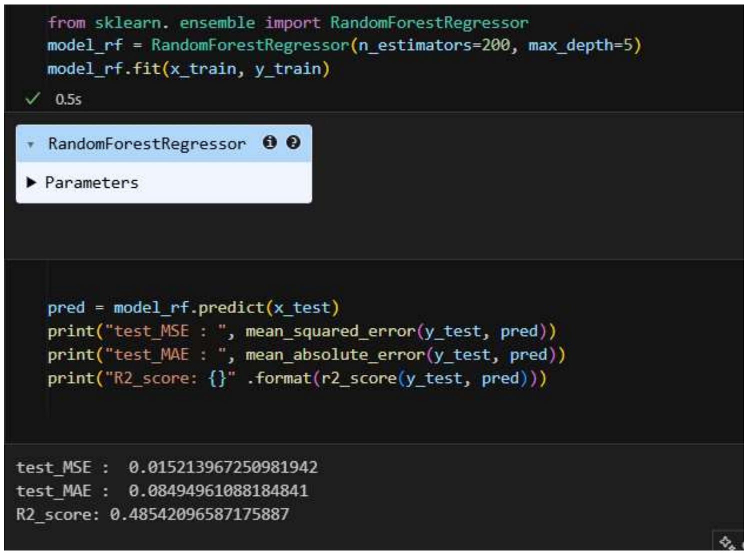


```
from sklearn. linear_model import LinearRegression
model_lr=LinearRegression()
model_lr.fit(x_train,y_train)
```

```
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 :  0.020973077246870846
test_MAE :  0.10639164268443882
R2_score: 0.29063171660927345
```

- Random Forest:

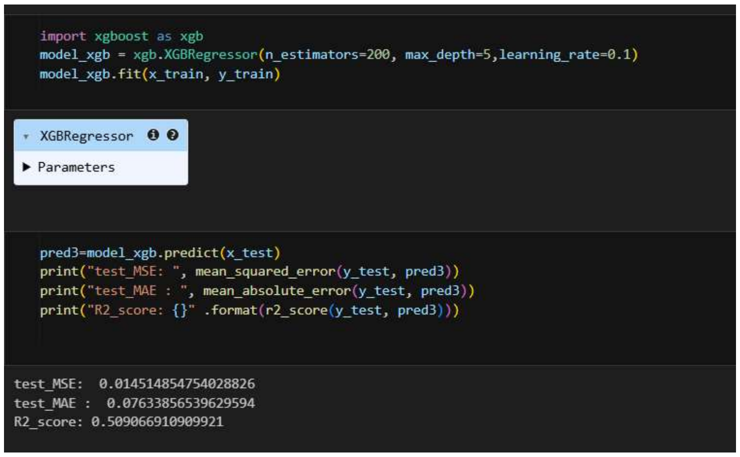


```
from sklearn. ensemble import RandomForestRegressor
model_rf = RandomForestRegressor(n_estimators=200, max_depth=5)
model_rf.fit(x_train, y_train)
```

```
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 :  0.015213967250981942
test_MAE :  0.08494961088184841
R2_score: 0.48542096587175887
```

- XGBoost:



```
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)
```

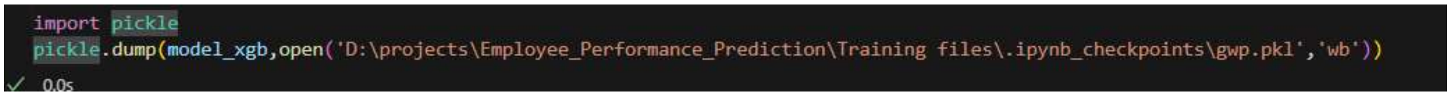
```
pred3=model_xgb.predict(x_test)
print("test_MSE: ", mean_squared_error(y_test, pred3))
print("test_MAE : ", mean_absolute_error(y_test, pred3))
print("R2_score: {}" .format(r2_score(y_test, pred3)))
```

```
test_MSE:  0.014514854754028826
test_MAE :  0.07633856539629594
R2_score: 0.509066910909921
```

The XGBoost model was finalized due to its superior performance and interpretability.

Milestone 4: Model Deployment

Activity 1: Model Persistence



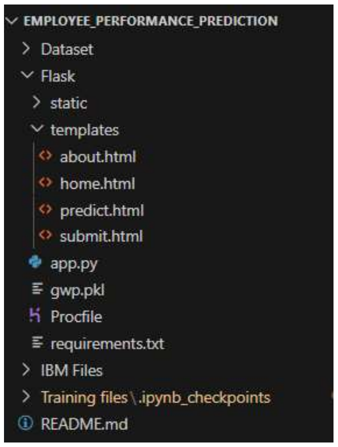
```
import pickle
pickle.dump(model_xgb,open('D:\projects\Employee_Performance_Prediction\Training files\.ipynb_checkpoints\gwp.pkl','wb'))
```

Model Serialization Benefits:

- *Persistent storageof trained parameters*
- *Version controlfor model updates*
- *Deployment-ready format*
- *Cross-platform compatibility*

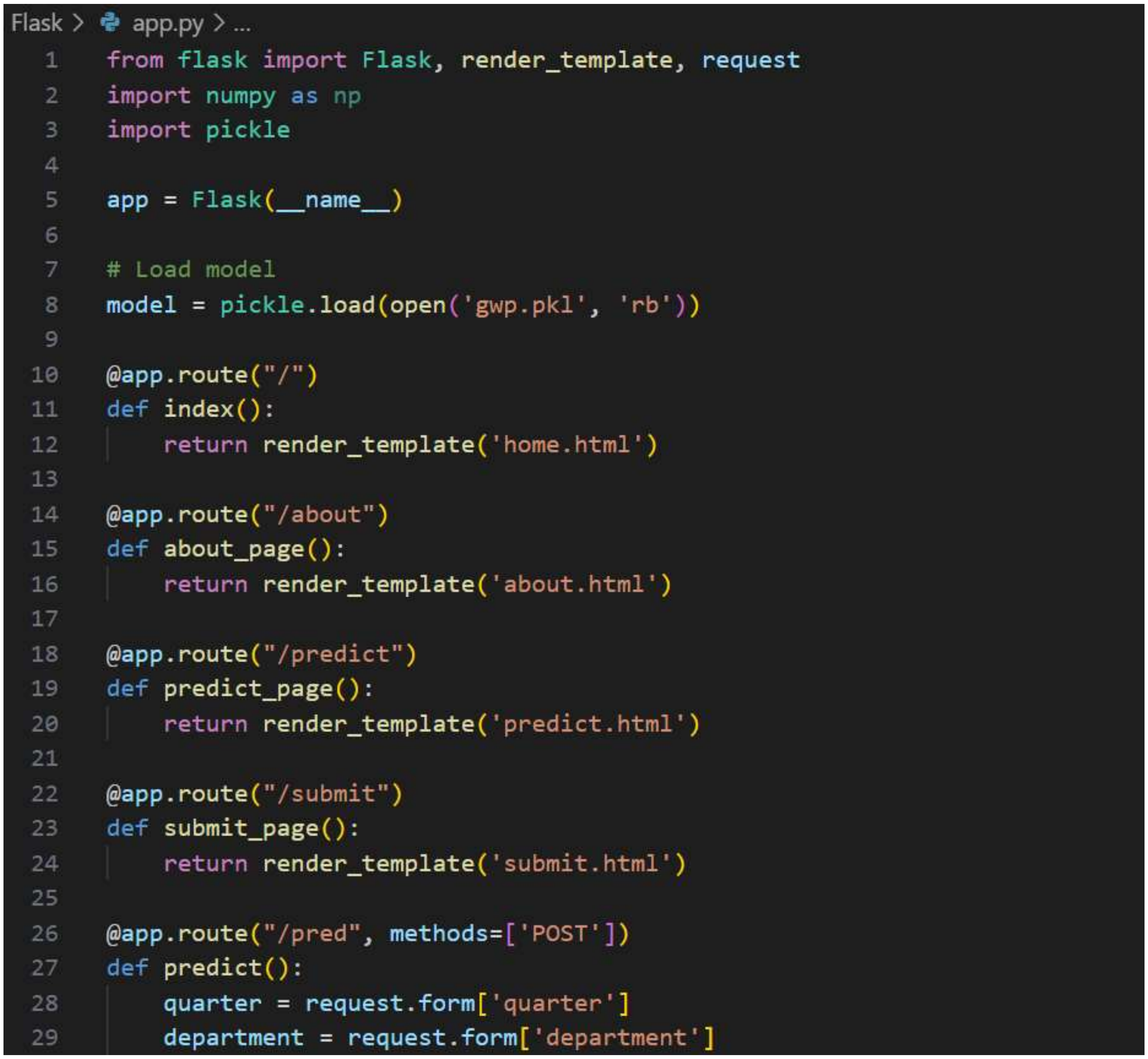
Flask Web Application :

Application Architecture



```
EMPLOYEE_PERFORMANCE_PREDICTION
  > Dataset
  v Flask
    > static
    v templates
      about.html
      home.html
      predict.html
      submit.html
    app.py
    gwp.pkl
    Procfile
    requirements.txt
  > IBM Files
  > Training files\.ipynb_checkpoints
  README.md
```

Backend Implementation (app.py)



Flask > app.py > ...

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

app = Flask(__name__)

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

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

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

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

@app.route("/submit")
def submit_page():
    return render_template('submit.html')

@app.route("/pred", methods=['POST'])
def predict():
    quarter = request.form['quarter']
    department = request.form['department']
```

Frontend Implementation

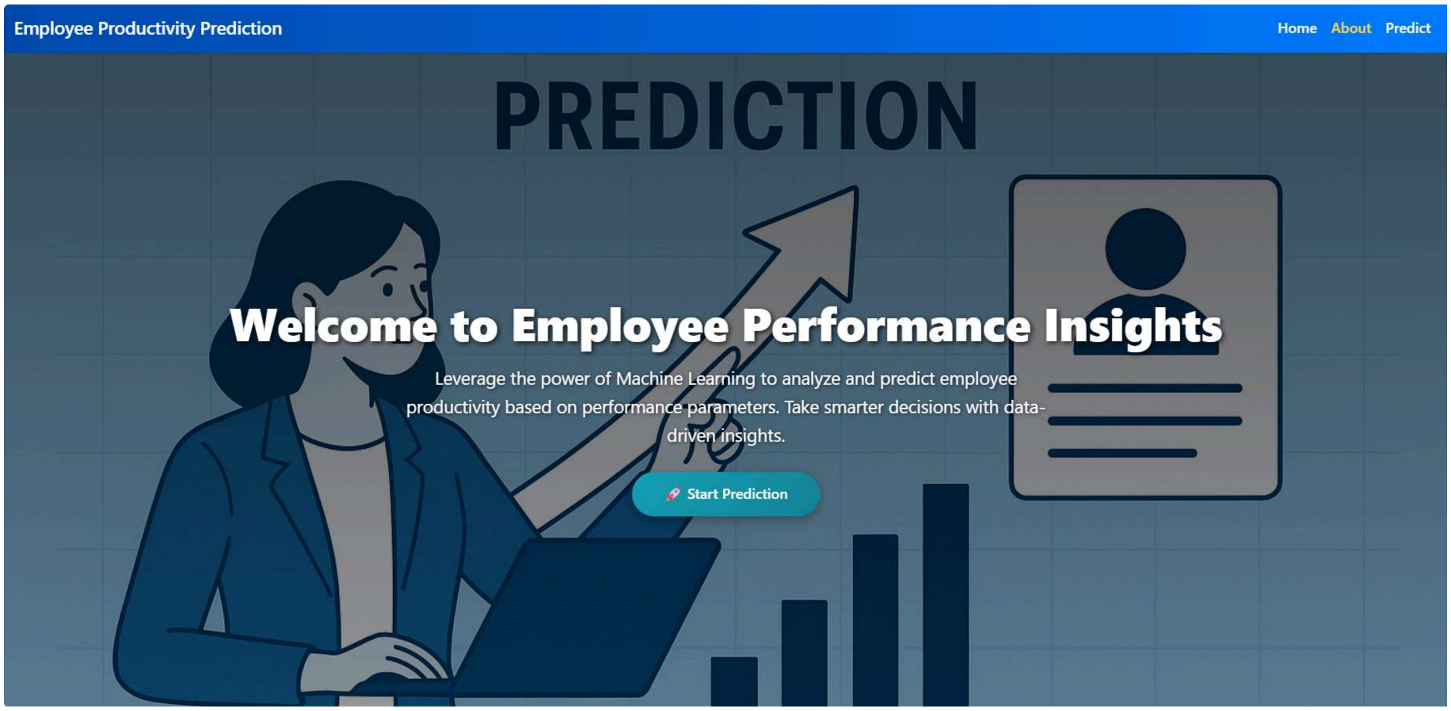
Professional Interface(index.html):

The web application features:

- **Professional Interface:** Clean and modern design for performance analysis
- **Responsive Layout:** Works seamlessly across all devices
- **Real-Time Validation:** Ensures accurate employee data input
- **Predictive Insights:** Color-coded results with productivity recommendations

Application screenshots:

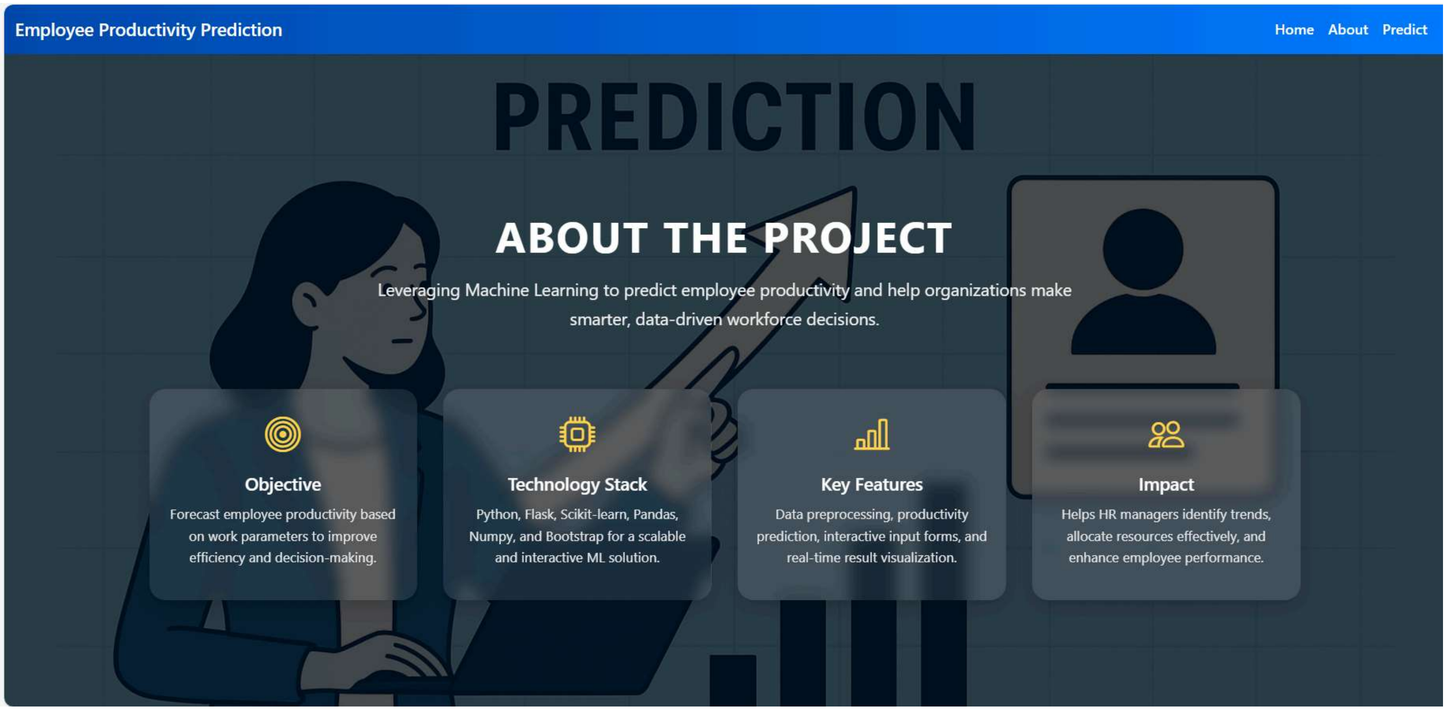
1. ***Home Page Interface***



Welcome to Employee Performance Insights

Leverage the power of Machine Learning to analyze and predict employee productivity based on performance parameters. Take smarter decisions with data-driven insights.

2. ***About Page Interface***



ABOUT THE PROJECT

Leveraging Machine Learning to predict employee productivity and help organizations make smarter, data-driven workforce decisions.

Objective

Forecast employee productivity based on work parameters to improve efficiency and decision-making.

Technology Stack

Python, Flask, Scikit-learn, Pandas, Numpy, and Bootstrap for a scalable and interactive ML solution.

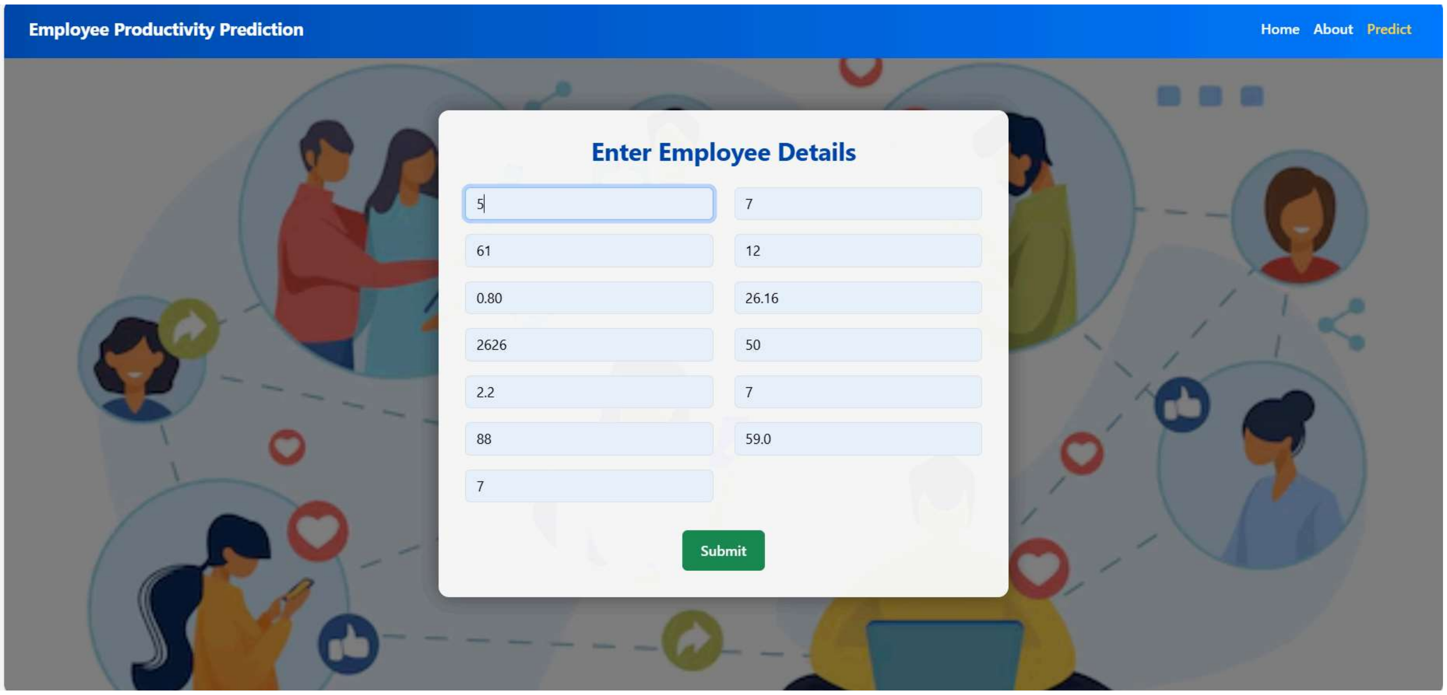
Key Features

Data preprocessing, productivity prediction, interactive input forms, and real-time result visualization.

Impact

Helps HR managers identify trends, allocate resources effectively, and enhance employee performance.

3.Predict Page Interface


Employee Productivity Prediction

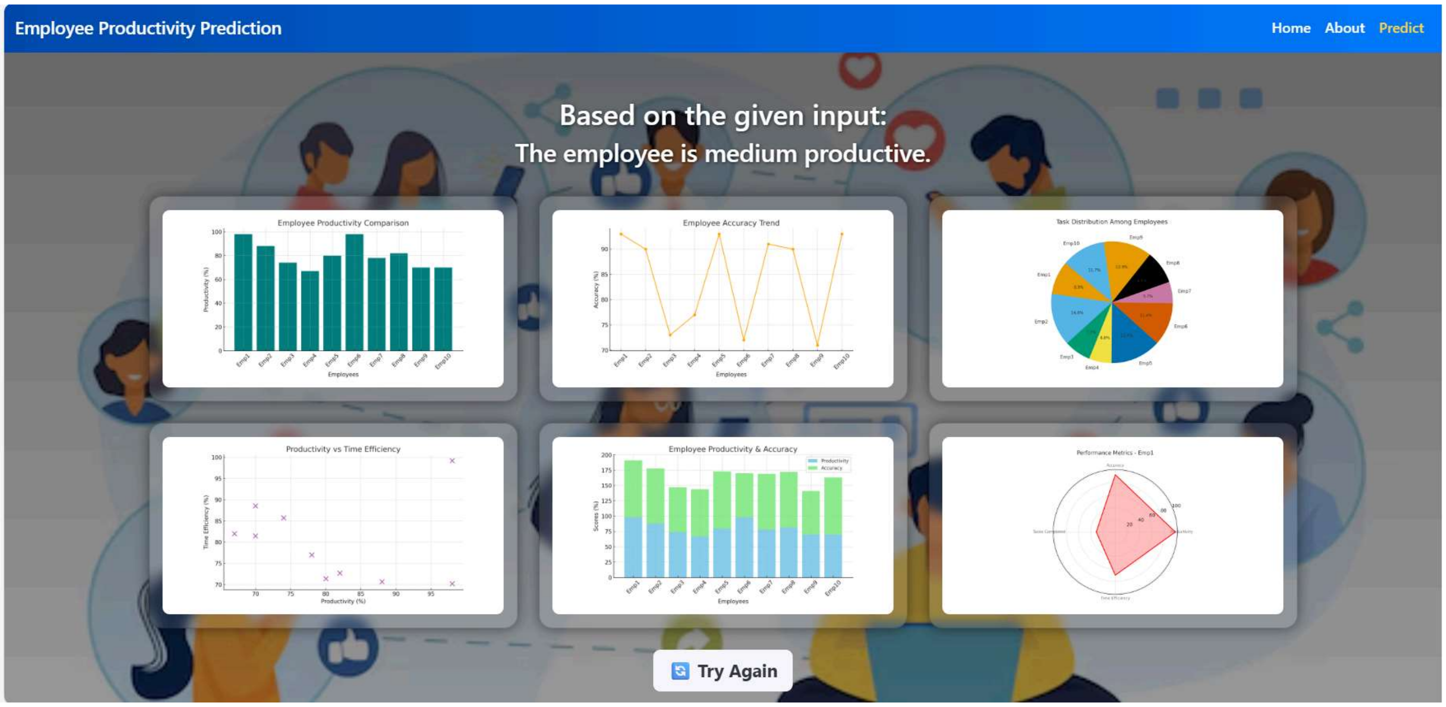
Home About Predict

Enter Employee Details

5	7
61	12
0.80	26.16
2626	50
2.2	7
88	59.0
7	

Submit

4. Submit Page Interface


Employee Productivity Prediction

Home About Predict

Based on the given input:
The employee is medium productive.

Employee Productivity Comparison
Employee Accuracy Trend
Task Distribution Among Employees
Productivity vs Time Efficiency
Employee Productivity & Accuracy
Performance Metrics - Emp1
Try Again

Application Testing and Validation:

To validate the Employee Performance Prediction model, the following two data inputs were used to test its ability to handle different operational scenarios:

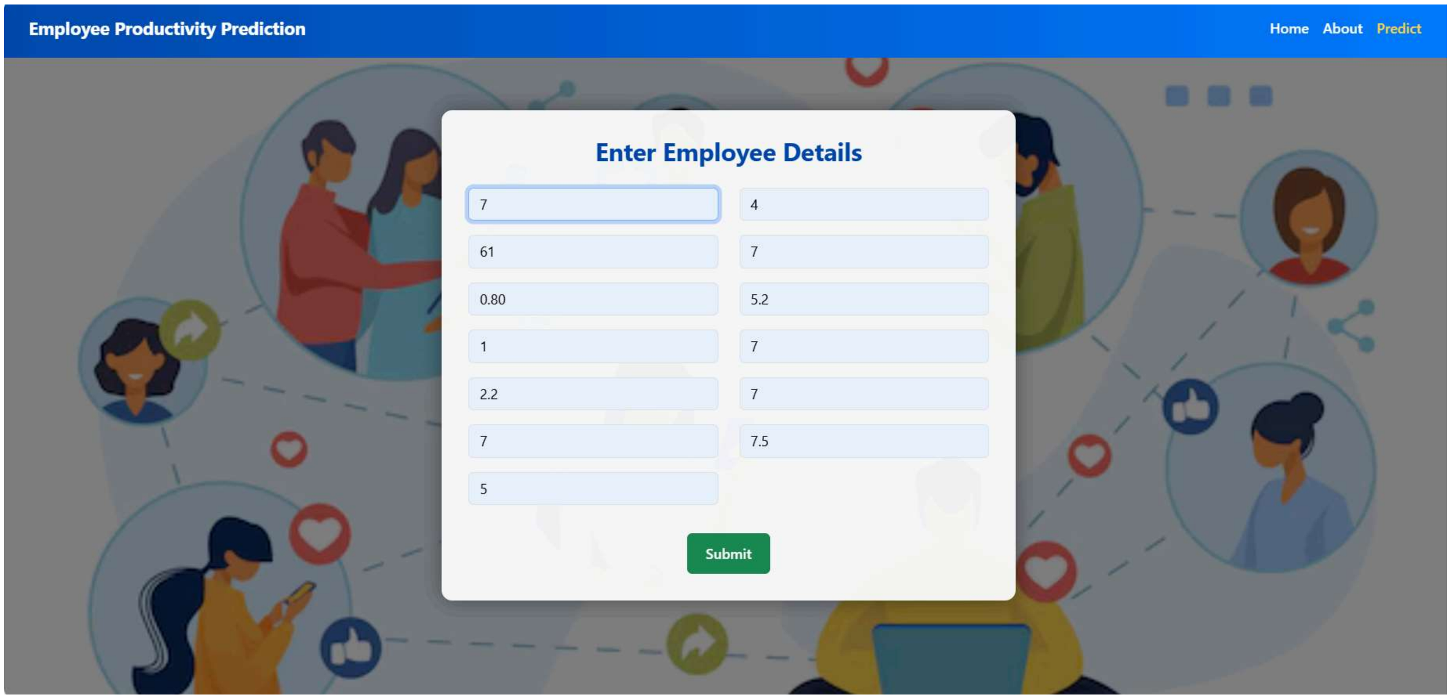
Input - 1:

Feature Name	Value
quarter	7
day	61
targeted_productivity	0.80
over_time	1
idle_time	2.2
no_of_style_change	7
month	5
department	4
team	7
smv	5.2
incentive	7
idle_men	7
no_of_workers	7.5

For this input we get the output as ' **The employee is Highly productive.'**

VISUALIZATIONS:

INPUT-1 :

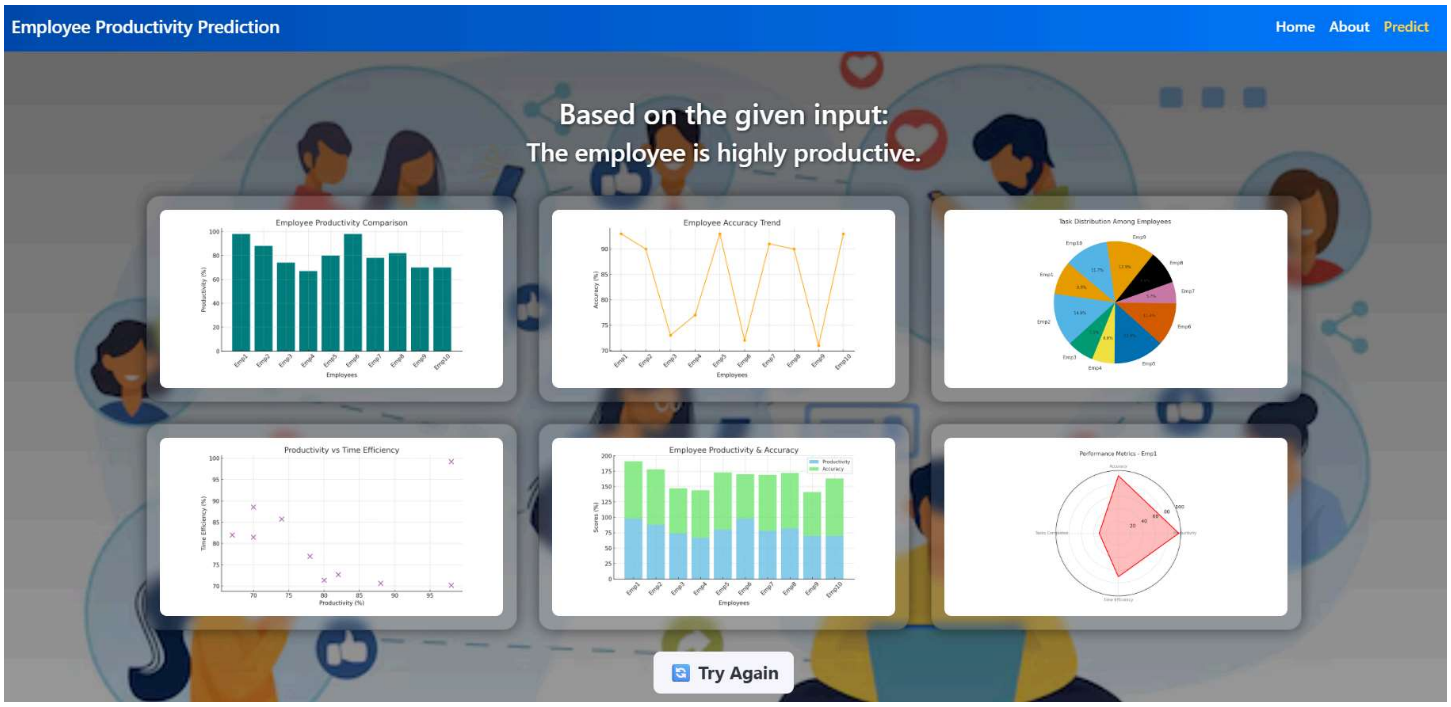


Employee Productivity Prediction

Enter Employee Details

7	4
61	7
0.80	5.2
1	7
2.2	7
7	7.5
5	

OUTPUT-1 :



Employee Productivity Prediction

Based on the given input:
The employee is highly productive.

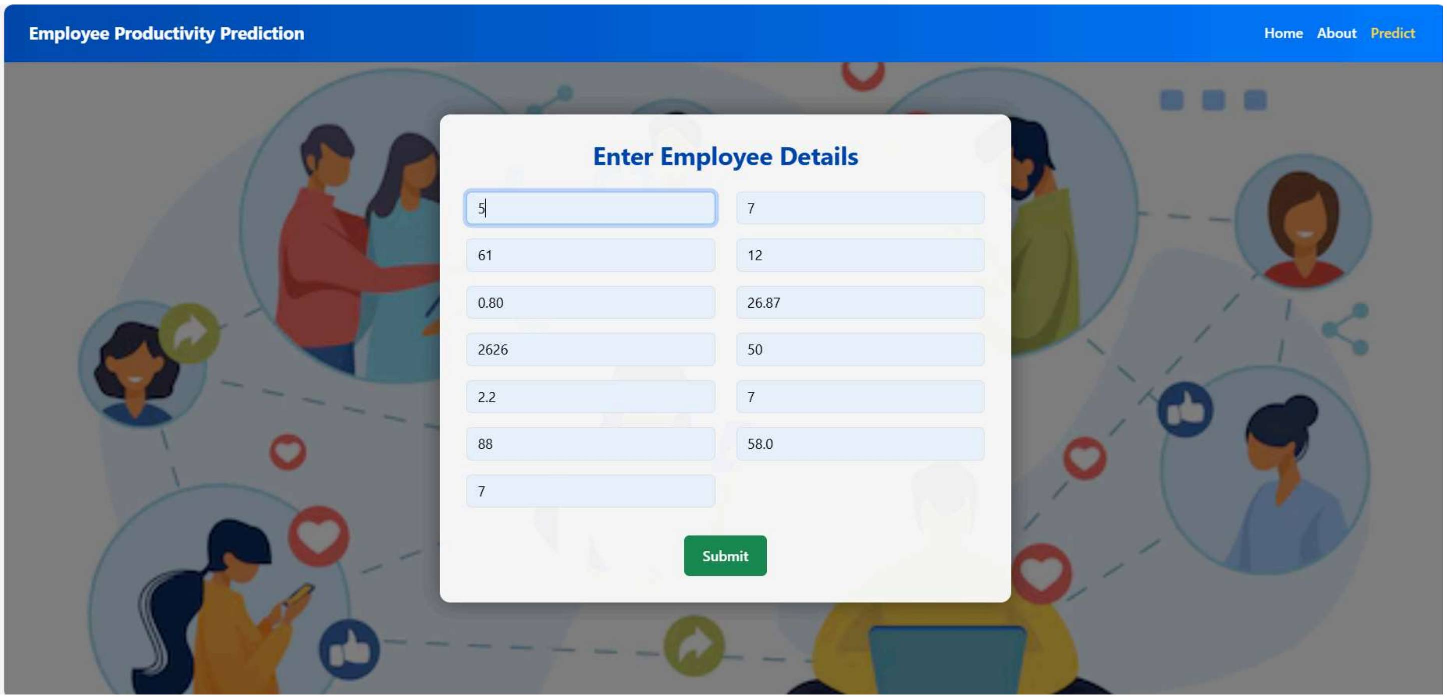
Employee Productivity Comparison
Employee Accuracy Trend
Task Distribution Among Employees
Productivity vs Time Efficiency
Employee Productivity & Accuracy
Performance Metrics - Emp1
Try Again

Input - 2:

Feature Name	Value
quarter	5
day	61
targeted_productivity	0.80
over_time	2626
idle_time	2.2
no_of_style_change	88
month	7
department	7
team	12
smv	26.16
incentive	50
idle_men	7
no_of_workers	58.0

For this input we get the output as " **The employee is Medium productive.'**

Input - 2:



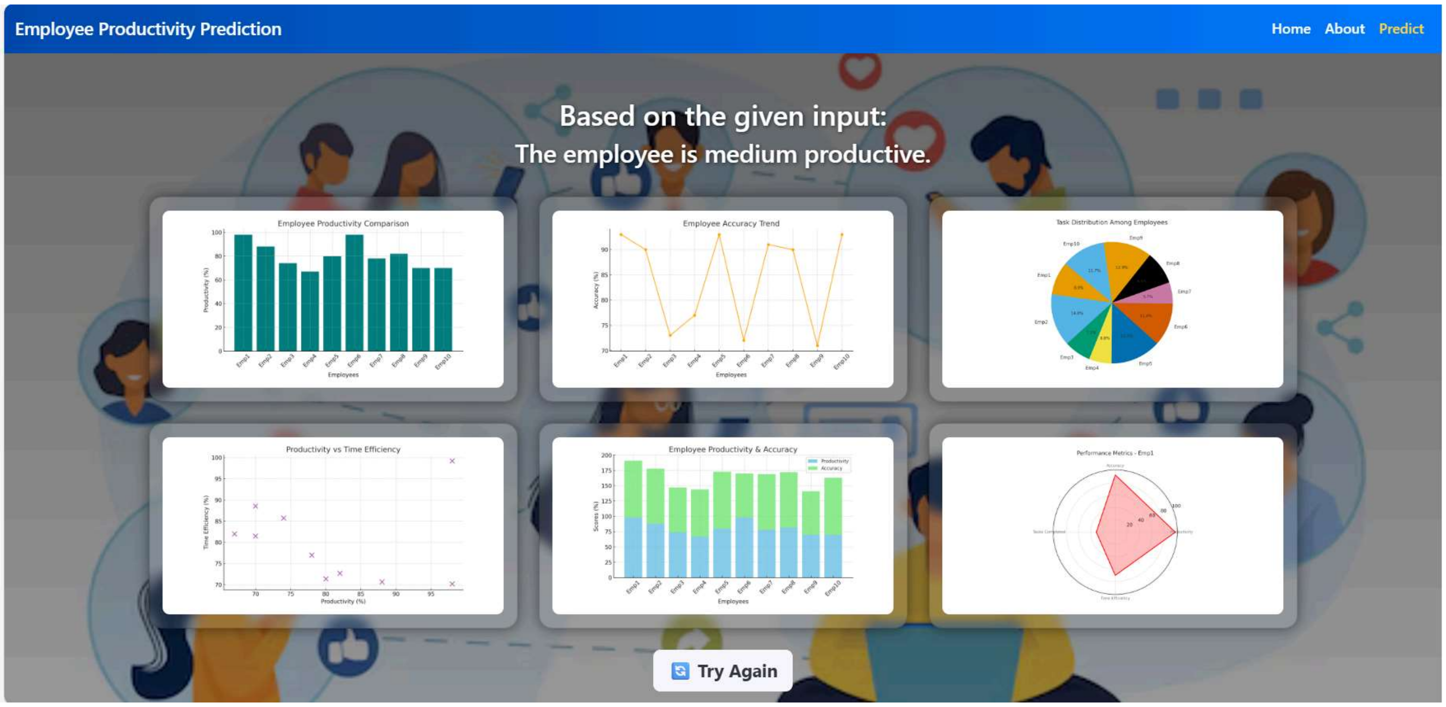
Employee Productivity Prediction

Home About Predict

Enter Employee Details

5	7
61	12
0.80	26.87
2626	50
2.2	7
88	58.0
7	

output - 2:



Employee Productivity Prediction

Home About Predict

Based on the given input:

The employee is medium productive.

Employee Productivity Comparison
Employee Accuracy Trend
Task Distribution Among Employees
Productivity vs Time Efficiency
Employee Productivity & Accuracy
Performance Metrics - Emp1
Try Again

Conclusion

The Employee Performance Prediction System successfully leverages machine learning to analyze key workforce parameters and forecast employee productivity. By integrating predictive analytics with an intuitive web interface, the system enables organizations to identify performance trends, optimize resource allocation, and make data-driven decisions. The project demonstrates how AI can enhance workforce management by providing accurate, actionable insights that improve overall operational efficiency.

Feature Enhancement

Future enhancements to the system can include:

- ***Automated Report Generation:***

  Generate downloadable performance summaries in PDF or Excel formats.

- ***Employee Feedback Integration:***

  Combine predictive data with qualitative feedback for deeper insights.

- ***Advanced Visualization Dashboard:***

  Add interactive trend analysis, heatmaps, and KPI trackers.

- ***Real-Time Data Stream:***

  Integrate with HR systems for continuous monitoring and live performance updates.

- ***Model Optimization:***

  Implement deep learning models or ensemble techniques for improved prediction accuracy.

- ***Role-Based Access Control:***

  Enable secure logins for Admin, Manager, and Employee dashboards.