

TEAM ID :593214



EMPLOYEE PERFORMANCE PREDICTION USING ML

Project Report

Team-593214

Topic - Machine Learning Approach For Employee Performance Prediction

Team Members-

Madhav a nair

Aditi kotecha

Sreenath S

INDEX

- 1. INTRODUCTION**
 - 1.1 Project Overview
 - 1.2 Purpose
- 2. LITERATURE SURVEY**
 - 2.1 Existing problem
 - 2.2 References
 - 2.3 Problem Statement Definition
- 3. IDEATION & PROPOSED SOLUTION**
 - 3.1 Empathy Map Canvas
 - 3.2 Ideation & Brainstorming
- 4. REQUIREMENT ANALYSIS**
 - 4.1 Functional requirement
 - 4.2 Non-Functional requirements
- 5. PROJECT DESIGN**
 - 5.1 Data Flow Diagrams & User Stories
 - 5.2 Solution Architecture
- 6. PROJECT PLANNING & SCHEDULING**
 - 6.1 Technical Architecture
 - 6.2 Sprint Planning & Estimation
 - 6.3 Sprint Delivery Schedule
- 7. CODING & SOLUTIONING (Explain the features added in the project along with code)**
 - 7.1 Feature 1
 - 7.2 Feature 2
 - 7.3 Database Schema (if Applicable)
- 8. PERFORMANCE TESTING**
 - 8.1 Performace Metrics
- 9. RESULTS**
 - 9.1 Output Screenshots
- 10. ADVANTAGES & DISADVANTAGES**
- 11. CONCLUSION**
- 12. FUTURE SCOPE**
- 13. APPENDIX**
 - Source Code
 - GitHub & Project Demo Link

1. INTRODUCTION

1.1 Project Overview

In the modern workplace, understanding and predicting employee performance is crucial for organizational success. Employee performance prediction using machine learning techniques has the potential to revolutionize HR processes, talent management, and decision-making. This project aims to develop a predictive model that can effectively forecast employee performance, allowing organizations to make informed decisions and optimize resource allocation. The project involves the following steps:-

1. **Data Collection:** Gather comprehensive and relevant data pertaining to employees, including but not limited to personal information, job roles, past performance metrics, feedback, training history, and any other relevant variables.
2. **Data Preprocessing:** Clean, preprocess, and standardize the collected data to ensure consistency, accuracy, and the removal of any irrelevant or redundant features.
3. **Feature Engineering:** Identify and create new features that could enhance the predictive power of the model. This may involve extracting patterns, generating new variables, or transforming existing ones.
4. **Model Selection:** Explore a variety of machine learning algorithms suitable for the prediction of employee performance. Compare and select the most suitable models, such as regression, classification, or ensemble methods, based on their performance metrics.
5. **Model Development:** Build, train, and fine-tune the selected machine learning models using the prepared dataset. Experiment with different hyperparameters and model architectures to enhance predictive accuracy and generalization.
6. **Model Evaluation:** Assess the performance of developed models using appropriate evaluation metrics. Perform cross-validation and test the models on unseen data to ensure robustness and generalizability.
7. **Deployment and Integration:** Integrate the developed model into the existing HR systems or develop a user-friendly interface for HR professionals to input data and obtain predictions. Ensure scalability, security, and ease of use in the deployment process.
8. **Performance Improvement:** Continuously monitor and refine the model's performance by gathering new data, retraining the model, and incorporating feedback to enhance predictive accuracy and relevance.

1.2 Purpose

The purpose of implementing Machine Learning for Employee Performance Prediction is multi-fold, aiming to revolutionize and enhance the management of human resources within organizations. **Informed Decision Making:** By utilizing historical data and various performance indicators, machine learning models can predict future employee performance. These insights empower HR professionals and managers to make informed decisions regarding promotions, training needs, and resource allocation.

Optimizing Talent Management: Predictive models can assist in identifying high-potential employees, areas requiring skill development, and potential performance issues. This optimizes talent management processes and ensures resources are directed where they are most needed.

Proactive Performance Management: Anticipating potential issues or identifying areas for improvement enables HR departments to take proactive measures. They can provide support and interventions to mitigate risks of underperformance before they significantly impact the organization.

Objective Evaluation: Machine learning models can reduce subjectivity and bias in performance evaluations. They provide a more objective assessment, enhancing fairness and consistency in performance appraisal processes.

Efficiency and Time-Saving: Automating the prediction of employee performance through machine learning can save considerable time and effort. This allows HR professionals to focus on strategic initiatives and decision-making, rather than spending extensive time on evaluations.

Continuous Improvement: Machine learning models can continually learn and adapt to new data, leading to ongoing improvements in the accuracy and relevance of predictions over time.

Talent Retention and Succession Planning: Understanding performance patterns can help identify and retain valuable talent within the organization. It also aids in succession planning by grooming future leaders based on their performance and potential.

Enhanced Employee Engagement and Motivation: Identifying factors that contribute to high performance can aid in developing strategies to improve employee engagement and motivation. Recognizing and rewarding top performers based on predictive insights can further motivate the workforce.

Strategic Planning and Competitive Edge: Having a robust employee performance prediction model allows organizations to plan strategically. This can lead to better talent acquisition, improved productivity, and a more agile response to market demands, providing a competitive advantage.

Ethical and Fair Evaluation: Leveraging machine learning models ensures fair and unbiased evaluation systems, promoting a more ethical and inclusive work environment.

In essence, the primary purpose of using machine learning for employee performance prediction is to enhance HR practices and talent management, enabling organizations to make data-driven decisions that improve efficiency, fairness, and overall performance within the workforce.

2. LITERATURE SURVEY

2.1 Existing problem

Employee performance prediction is a complex task plagued by several challenges. The multifaceted nature of employee performance, which encompasses diverse and subjective factors like skills, motivation, and work environment, poses a considerable hurdle. Data quality and availability contribute significantly to the problem, as incomplete or inconsistent historical performance data may compromise the accuracy of predictions. Human behavior's unpredictability, along with biases in evaluation metrics and interconnected variables, further complicates the creation of accurate predictive models. Moreover, changes in job roles, ethical concerns regarding fairness and legal compliance, and the employees' perception and acceptance of these models add layers of complexity. Contextual nuances and the frequency of performance evaluations also impact the precision of predictions. Overcoming these challenges necessitates a comprehensive approach that combines quantitative data with qualitative insights, integrating both analytics and human expertise while ensuring continuous

refinement and ethical considerations in the development of these predictive models.

2.2 References

2.4 Problem Statement Definition

The problem statement in using machine learning for employee performance prediction involves developing an accurate and reliable predictive model that can effectively forecast and evaluate employee performance within an organization. This includes addressing various challenges such as the multi-dimensional nature of performance metrics, the quality and availability of historical data, subjective evaluations, biases in assessment, the dynamic and unpredictable aspects of human behavior, and legal or ethical concerns related to fairness and privacy.

The goal is to create a predictive model that overcomes these challenges, leveraging historical data while accounting for the complexities of human performance in a fair and unbiased manner. This model should be capable of adapting to changes in job roles and responsibilities while providing accurate predictions on an ongoing basis, incorporating both quantitative and qualitative factors to deliver insights that are valuable for HR decision-making processes. Ultimately, the aim is to develop a machine learning-based solution that assists in making data-driven decisions related to talent management, resource allocation, and performance improvement strategies within the organization while ensuring fairness, reliability, and compliance with legal and ethical standards.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

User Persona: In the center of the canvas, we identify the primary persona we want to understand, such as HR professionals, team managers, data scientists, or employees themselves.

What They Say: In this section, note down direct quotes or statements made by the persona regarding employee performance prediction. These can be comments or feedback on the existing processes or tools.

What They Do: Document the observable actions and behaviors of the persona related to employee performance prediction. This might include conducting performance reviews, collecting data, or making decisions based on performance data.

What They Think & Feel: Explore the persona's thoughts, emotions, and concerns. What are their motivations, goals, and worries when it comes to predicting employee performance through machine learning?

Pain Points: Identify the challenges and frustrations your persona faces. This may include data quality issues, time constraints, or difficulties in interpreting machine learning results.

Gains: Note the positive outcomes and benefits your persona seeks. This can encompass improved decision-making, enhanced employee productivity, and better organizational performance.

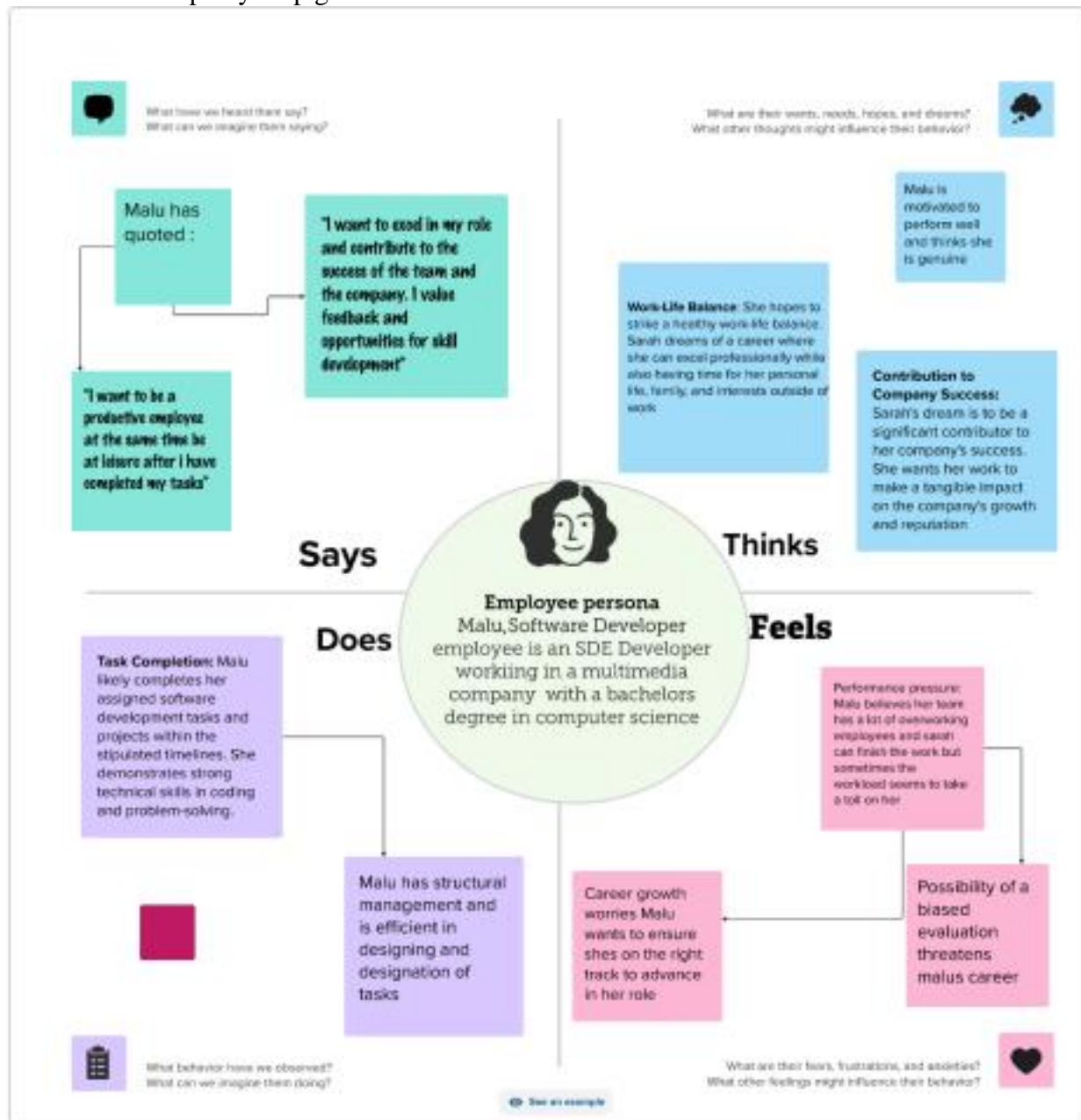
Influences: List external factors that influence your persona's decision-making, such as company culture, management expectations, or industry trends.

Context & Environment: Consider the environment in which your persona operates. Is it a fast-paced startup, a large corporation, or a government agency? These contextual factors can significantly impact their approach to employee performance prediction.

Needs & Goals: Summarize the key needs and goals of your persona in relation to machine learning for employee performance prediction. This might include the need for user-friendly tools, accurate data, or actionable insights.

Quotes & Visuals: Add any relevant images, illustrations, or additional quotes that further emphasize the persona's emotions, challenges, and aspiration

Below is the empathy map given





Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare

🕒 1 hour to collaborate

👤 2-8 people recommended



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

Team gathering

All team members of the current project are active for the session they are:

- 1.Madhav A Nair
2. S.sreenath
- 3.Aditi Koelcha

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session. Our goal is to idealise the problem statement and construct solutions via a session

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#)



1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

PROBLEM

THIS PROBLEM IS TWOFOLD: SUBJECTIVITY IN PERFORMANCE EVALUATIONS AND A LACK OF TRANSPARENCY IN THE PROCESS. THIS CAN RESULT IN EMPLOYEES FEELING UNDERVALUED, OVERLOOKED, OR UNFAIRLY TREATED. ADDITIONALLY, IT HINDERS THE ORGANIZATION'S ABILITY TO IDENTIFY HIGH-POTENTIAL EMPLOYEES AND PROVIDE THEM WITH THE RIGHT DEVELOPMENT OPPORTUNITIES.....



Key rules of brainstorming

To run an smooth and productive session

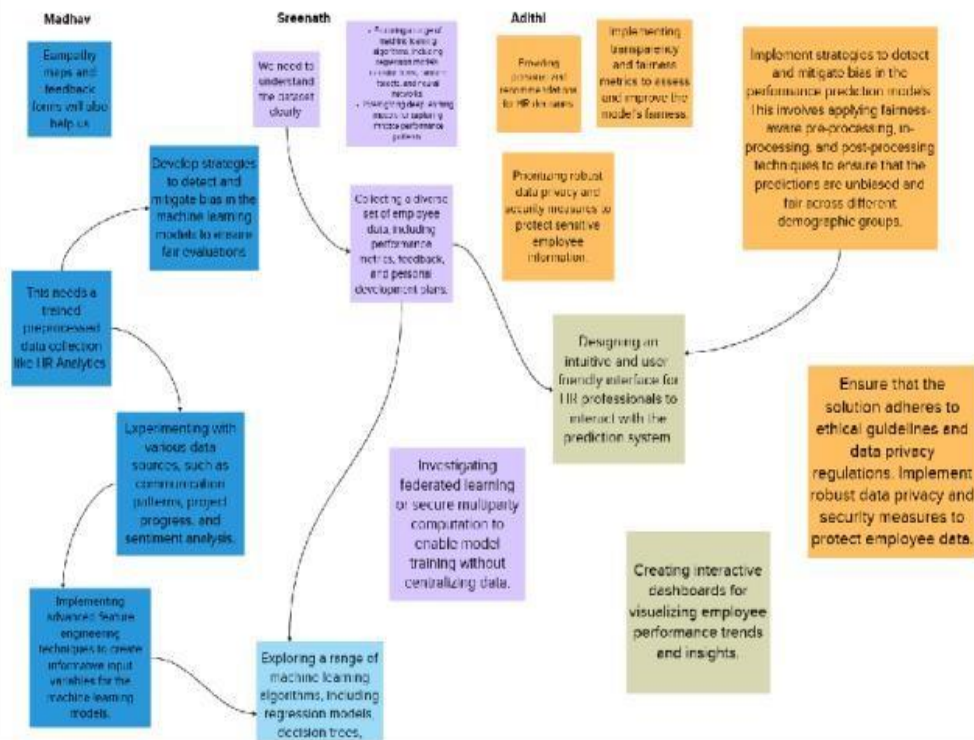
- 🗣️ Stay in topic.
- 💡 Encourage wild ideas.
- ⏸️ Defer judgment.
- 👂 Listen to others.
- 🗣️ Go for volume.
- 👁️ If possible, be visual.

2 Brainstorm

BrainStorming map was done in a talking session with the three members of the project to create some ideas and understand the idea prioritization we have given each colour on the idea based of the individual who produced it , mixed colours are for ideas that developed form the original idea

TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

⌚ 20 minutes

TIP

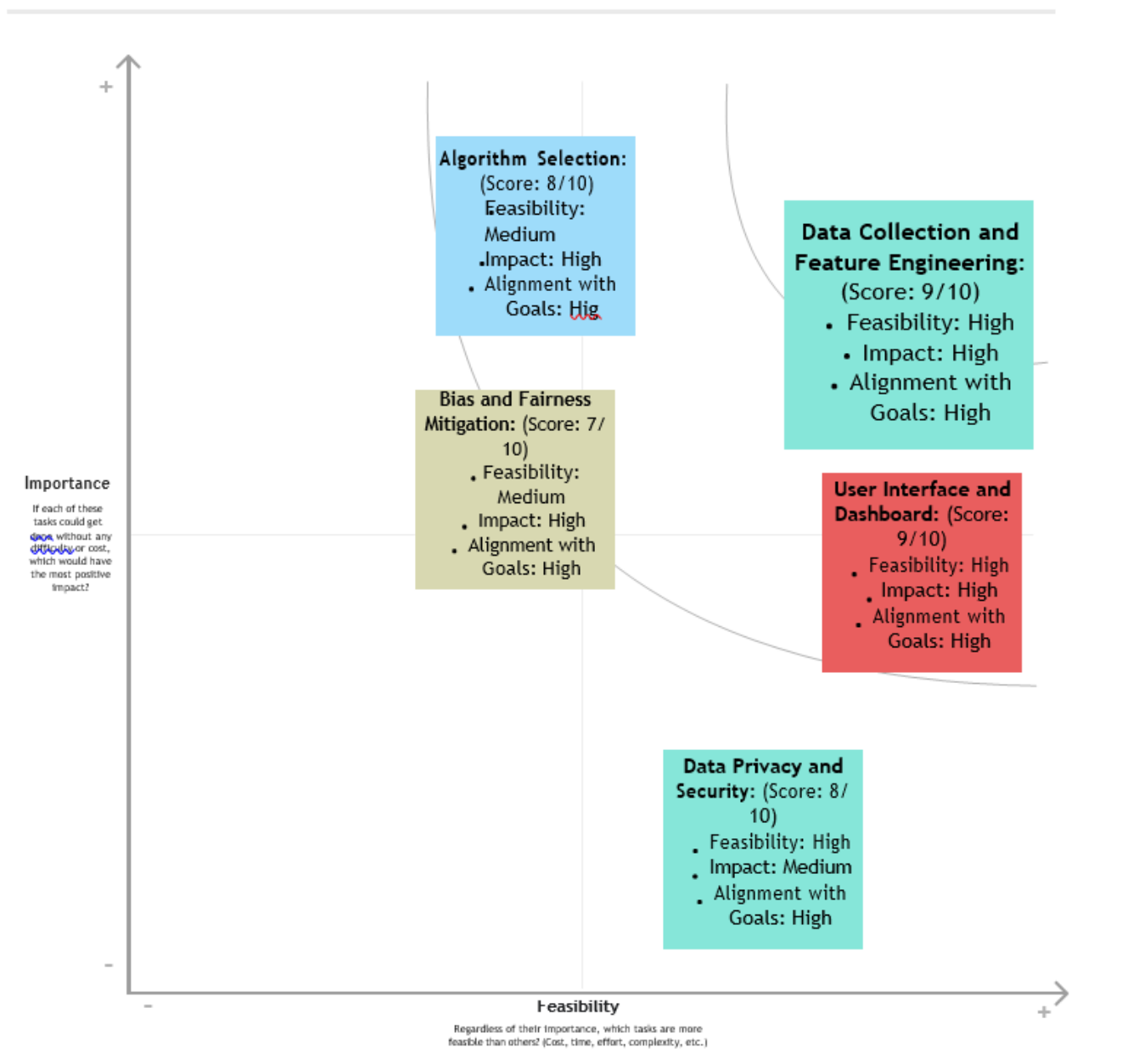
Add custom made tags to sticky notes to make it easier to first browse, organize, and categorize important ideas as they arise within your map.

Data Collection and Quality Improvement: Collect and integrate a diverse range of data sources, including performance metrics, feedback, peer reviews, and personal development plans. Implement data quality checks to ensure accuracy.

Feature Engineering: Utilize advanced feature engineering techniques to create informative input variables, capturing various aspects of employee performance, including productivity, teamwork, innovation, and leadership.

Algorithm Selection: Investigate machine learning algorithms that are known for their fairness and transparency, such as interpretable regression models or fairness-aware models. Utilize a combination of algorithms for robust predictions.

Bias and Fairness Mitigation: Implement strategies to detect and mitigate bias in the performance prediction models. This involves applying fairness-aware pre-processing, in-processing, and post-processing techniques to ensure that the predictions are unbiased and fair across different demographic groups.



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

functional requirements specify the specific features and capabilities that the system or model should possess to effectively predict and evaluate employee performance. These requirements typically involve the functionalities and behaviors that the machine learning system must exhibit to meet the objectives of predicting employee performance accurately and efficiently. Here are some functional requirements for such a system:

1. **Data Collection and Integration:** The system should be capable of gathering, integrating, and processing various types of data, including employee demographics, historical performance metrics, training records, feedback, and other relevant information, from multiple sources.
2. **Data Preprocessing and Cleaning:** It should have functionalities for data preprocessing, ensuring data quality, handling missing values, outliers, and standardizing the data for further analysis.
3. **Feature Engineering and Selection:** The system should be able to derive new features from the available data or select the most relevant features that contribute to accurate performance predictions.
4. **Model Training and Development:** Capabilities for training and developing machine learning models using various algorithms like regression, classification, or ensemble methods to predict employee performance accurately.
5. **Model Evaluation and Validation:** Functionality to evaluate model performance using appropriate metrics such as accuracy, precision, recall, and F1-score. This should include cross-validation techniques to ensure the model's robustness and generalization.
6. **Real-time Prediction and Integration:** Capability to integrate the trained model into existing HR systems or provide a user-friendly interface that allows users to input new data for real-time performance predictions.
7. **Scalability and Adaptability:** The system should be scalable, capable of handling a large volume of data, and adaptable to changes in employee roles, responsibilities, or the addition of new data for continuous improvement.
8. **Interpretability and Explainability:** Ideally, the system should provide insights into why a certain prediction was made, ensuring transparency and interpretability to end-users, particularly HR professionals.
9. **Continuous Improvement and Maintenance:** Functionality for continuously updating and maintaining the model by incorporating new data and insights to enhance predictive accuracy and relevance over time.
10. **Ethical and Legal Compliance:** Ensure that the model operates in compliance with ethical standards, respects privacy, avoids biases, and is fair in its predictions.

4.2 Non-Functional requirements

Non-functional requirements in machine learning for employee performance prediction define the qualities, constraints, and attributes that the

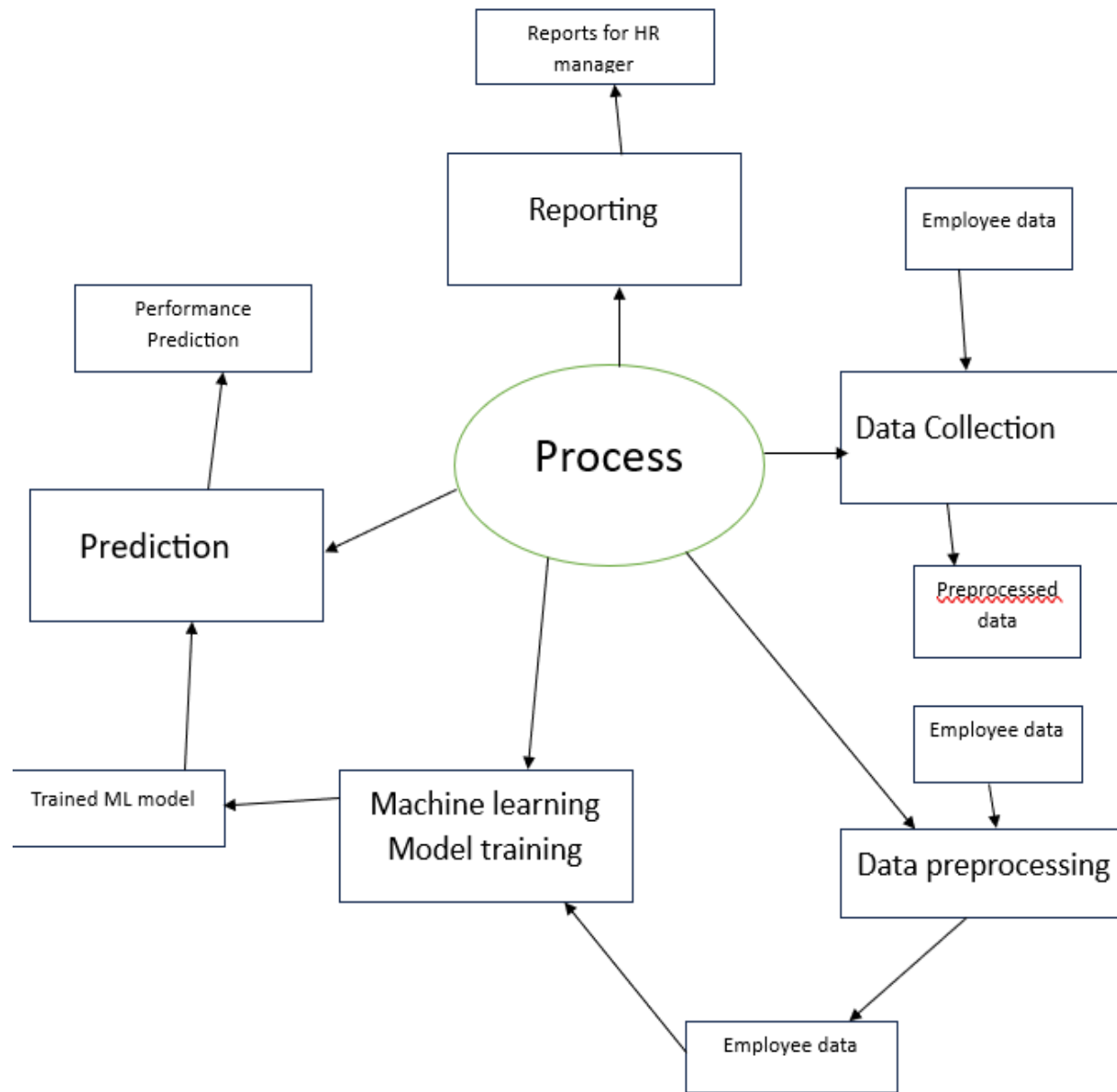
system or model should possess, beyond specific functionalities. These requirements focus on aspects related to the performance, usability, security, and scalability of the system. Here are some non-functional requirements for this system:

1. Performance: The system should provide timely and accurate predictions, with a high level of accuracy and minimal latency in delivering results.
2. Scalability: It should be capable of handling a growing volume of data and an increasing number of users, maintaining performance as the system scales.
3. Interpretability and Explainability: The model should be transparent and provide explanations for the predictions made, ensuring it can be understood and accepted by HR professionals and other stakeholders.
4. Reliability: The system should be dependable and consistently provide accurate predictions, demonstrating a low margin of error and high reliability.
5. Security: Ensuring the security and privacy of employee data is crucial. The system should comply with security protocols and regulations to protect sensitive employee information.
6. Usability and User Experience: The system should have a user-friendly interface, allowing HR professionals to interact with and interpret the model's outputs easily. This includes clear visualizations and an intuitive interface.
7. Robustness and Adaptability: The model should be robust and able to adapt to changing trends and new data. It should perform well even in the presence of outliers or noisy data.
8. Compliance and Ethics: It should adhere to legal regulations and ethical standards, avoiding discrimination, bias, or unethical use of data in its predictions.
9. Maintainability and Upgradability: The system should be easy to maintain, update, and upgrade. This includes incorporating new data, retraining the model, and implementing improvements efficiently.
10. Resource Efficiency: It should be resource-efficient, utilizing computational resources effectively to ensure optimal performance without excessive resource consumption.
11. Compatibility and Integration: The system should be compatible with various data sources, APIs, and HR systems, allowing seamless integration and exchange of information.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priorit y	Releas e
-----------	-------------------------------------	-------------------------	----------------------	---------------------	--------------	-------------

HR Manager	Employee Performance Prediction using ML	USN-1	As an HR manager, I want to upload employee performance data easily, so I can use the machine learning model for predictions.	<ul style="list-style-type: none"> There should be an option to upload a CSV file containing employee performance data. The system should validate the format of the uploaded data. The uploaded data should be stored securely for processing. 	High	1.0
---------------	---	-------	---	--	------	-----

		USN-2	As an HR manager, I want to specify the attributes and metrics to be used for performance prediction, so I can customize the model according to our organization's needs	<ul style="list-style-type: none"> • There should be a user-friendly interface for selecting the attributes and metrics. • The selected attributes should be used in the machine learning model for prediction. 	Medium	1.0
--	--	-------	--	---	--------	-----

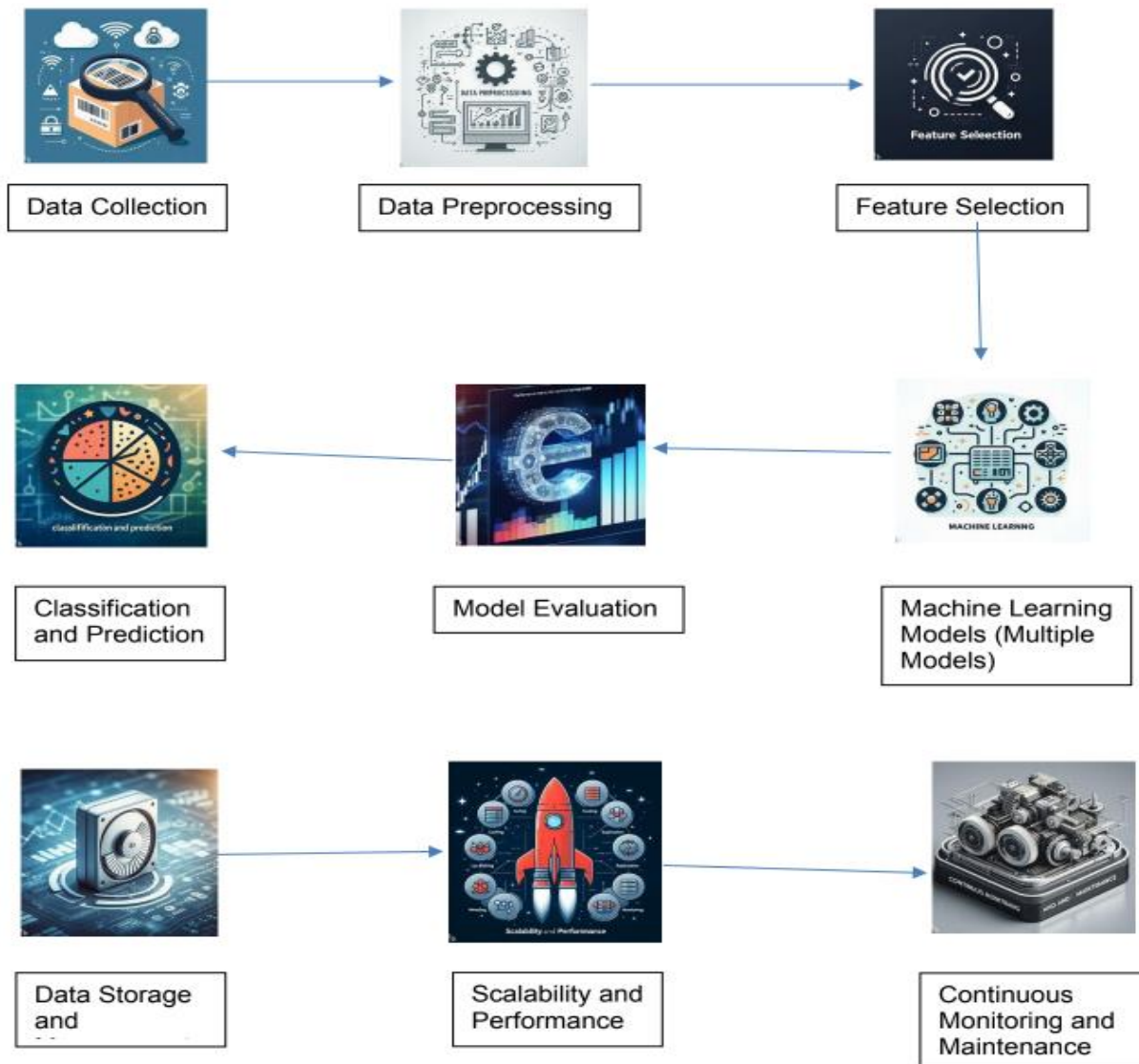
Data Analyst		USN-3	As a data analyst, I want to preprocess the employee performance data, including handling missing values and outliers, to ensure data quality and accuracy	<ul style="list-style-type: none"> • Data preprocessing should include handling missing values, outliers, and data normalization. • The preprocessed data should be stored in a designated location. 	High	1.0
---------------------	--	-------	--	--	------	-----

Data Scientist		USN-4	<p>As a data scientist, I want to train the machine learning model using the preprocessed data, so I can develop a prediction model.</p>	<ul style="list-style-type: none"> • The system should have tools and libraries for training machine learning models. • The model should be trained using the preprocessed data. 	High	1.0
-----------------------	--	-------	--	--	------	-----

		USN-5	As a data scientist, I want to evaluate the model's performance using various metrics like accuracy, precision, recall, and F1 score, to ensure its reliability.	<ul style="list-style-type: none"> • The system should provide options to evaluate the model's performance using standard metrics. • Evaluation results should be displayed for analysis. 	Medium	1.0
System Administrator		USN-6	As a system administrator, I want to ensure data privacy and security by implementing proper access controls to protect employee data.	<ul style="list-style-type: none"> • Access to employee data should be controlled based on user roles. • Data should be encrypted and stored securely. 	High	1.0

		USN -7	As a system administrator, I want to schedule regular model retraining to keep predictions up to date.	<ul style="list-style-type: none">• The system should allow scheduling retraining at specified intervals.• Retraining should use the most recent data.	Medium	1.1
--	--	--------	--	---	--------	-----

5.2 Solution Architecture



6.Project Planning And Scheduling

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

6.1 Technical Architecture

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection and Preprocessing	USN-1	As a data scientist, I need to collect and preprocess employee data, including individual and domain-specific characteristics, education, socioeconomic status, and psychological factors, to prepare it for analysis.	5	High	Sreenath Madhav Aditi
		USN-2	As a data scientist, I need to clean and normalize the collected data to ensure data quality and consistency for accurate analysis.	3	Medium	Sreenath
Sprint-2	Model Development and Training	USN-3	As a machine learning engineer, I want to develop and train supervised learning models, including Support Vector Machines, Random Forest, Naive Bayes, Neural Networks, and Logistic Regression, using the preprocessed data.	8	High	Madhav

		USN-4	As a machine learning engineer, I need to fine-tune the model hyperparameters for optimal performance.	2	Medium	Aditi
		USN-5	As a machine learning engineer, I want to implement the 10-fold validation technique to assess model correctness and robustness.	5	High	Madhav
Sprint-2	Evaluation and Reporting	USN-6	As a data analyst, I want to evaluate the performance of the models and generate insights into employee performance and commitment based on the model results.	5	High	Sreenath
		USN-7	As a data analyst, I want to create reports and	3	Medium	Aditi

			visualizations to communicate the findings effectively to stakeholders.			

6.2 Sprint Planning & Estimation

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	18	10 Days	12 Oct 2023	22 Oct 2023	16	22 Oct 2023
Sprint-2	12	8 Days	17 Oct 2023	25 Oct 2023	8	27 Oct 2023
Sprint-3	6	6 Days	25 Oct 2023	1 Nov 2023	-	-

6.3 Sprint Delivery Schedule

Velocity:

- Sprint 1: 18 points
- Sprint 2: 12 points
- Sprint 3: 6 points

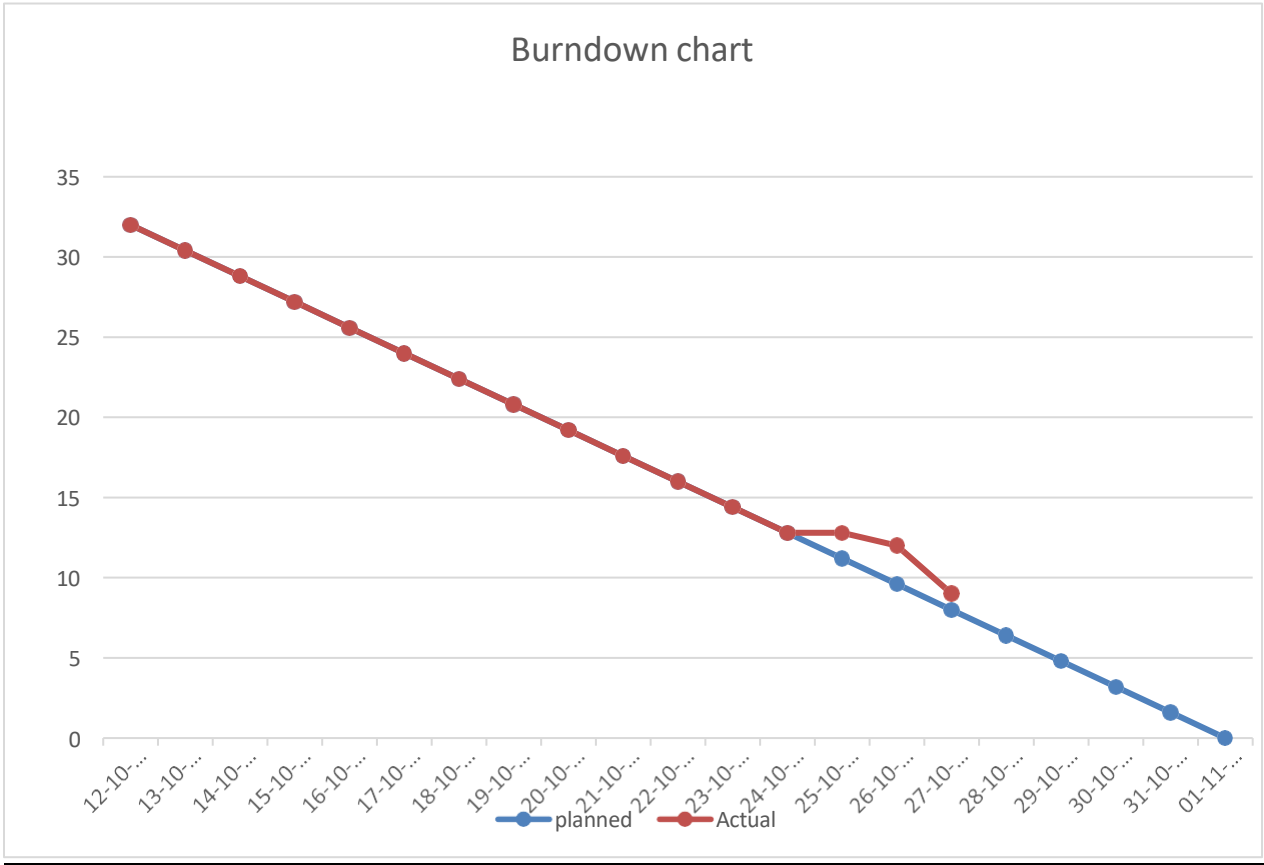
Average Velocity $= (18+12+6)/3$

$= 36/3$

$= 12$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



8. PERFORMANCE TESTING

8.1 Performace Metrics

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.

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

```
3]:
```

	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.

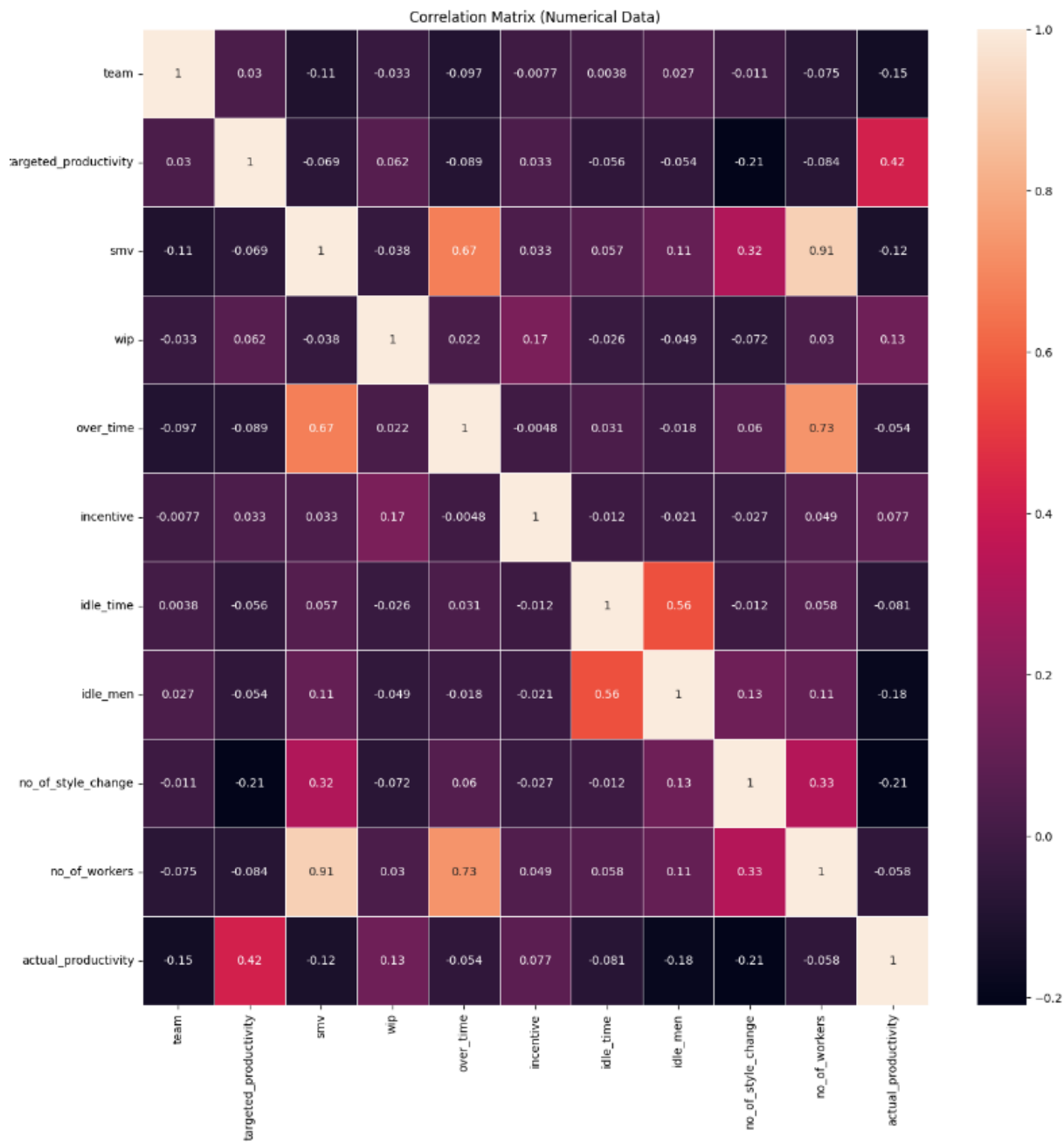
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
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

+ Code

+ Markdown

Milestone 3: Data Pre-processing

As we have understood how the data is let's 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 split 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)
```

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_scores 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_scores` 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_scores` 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_scores` 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:

```
y_pred = model.predict(y_test)

# 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

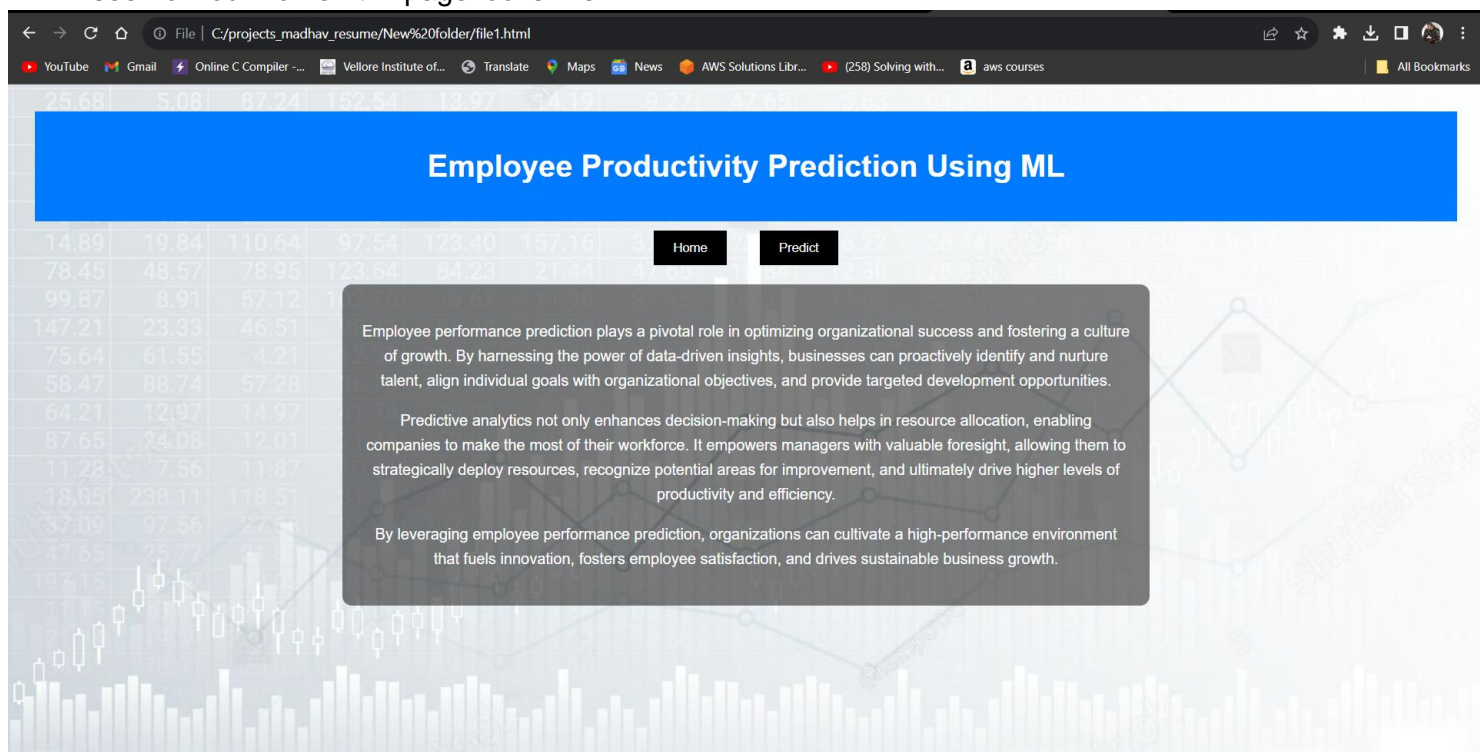
Activity1: Building Html Pages:

For this project create three HTML files namely

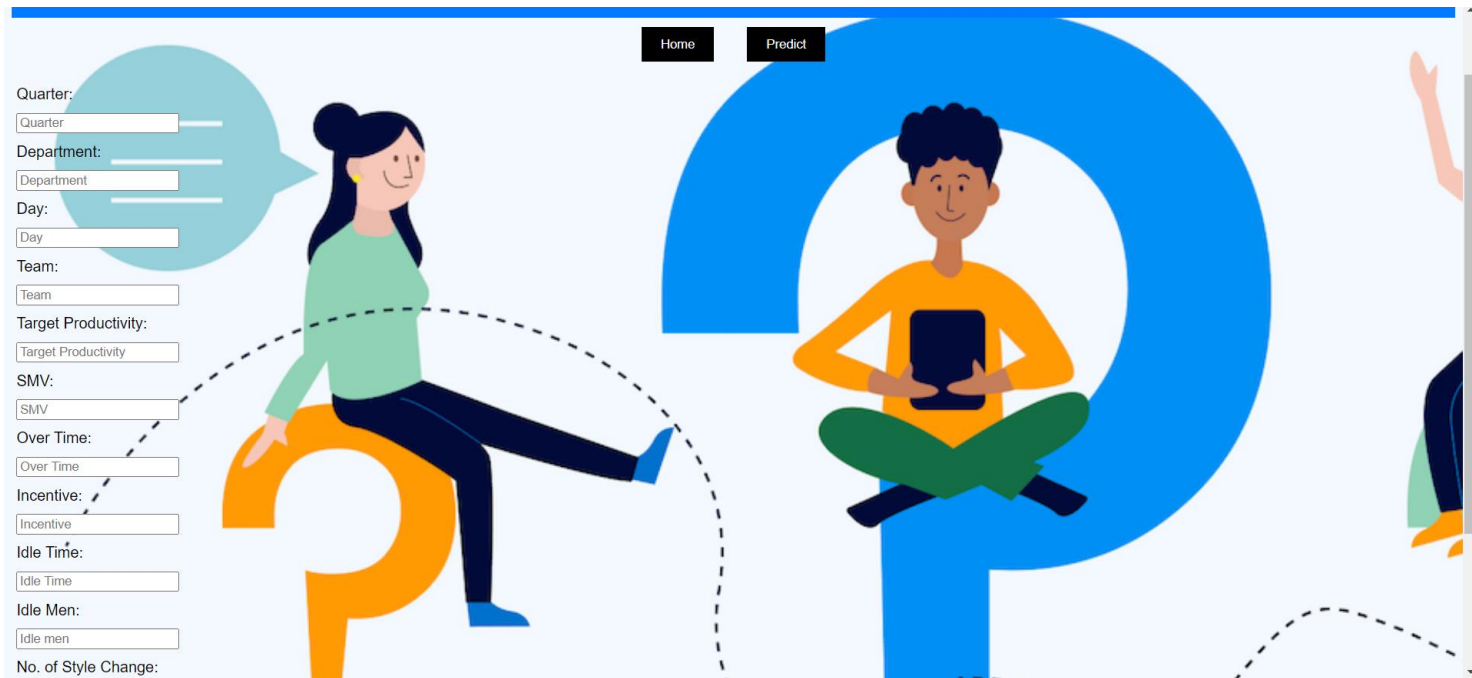
- about.html
- home.html
- predict.html
- submit.html

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:



Quarter:

Department:

Day:

Team:

Target Productivity:

SMV:

Over Time:

Incentive:

Idle Time:

Idle Men:

No. of Style Change:

Home Predict

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

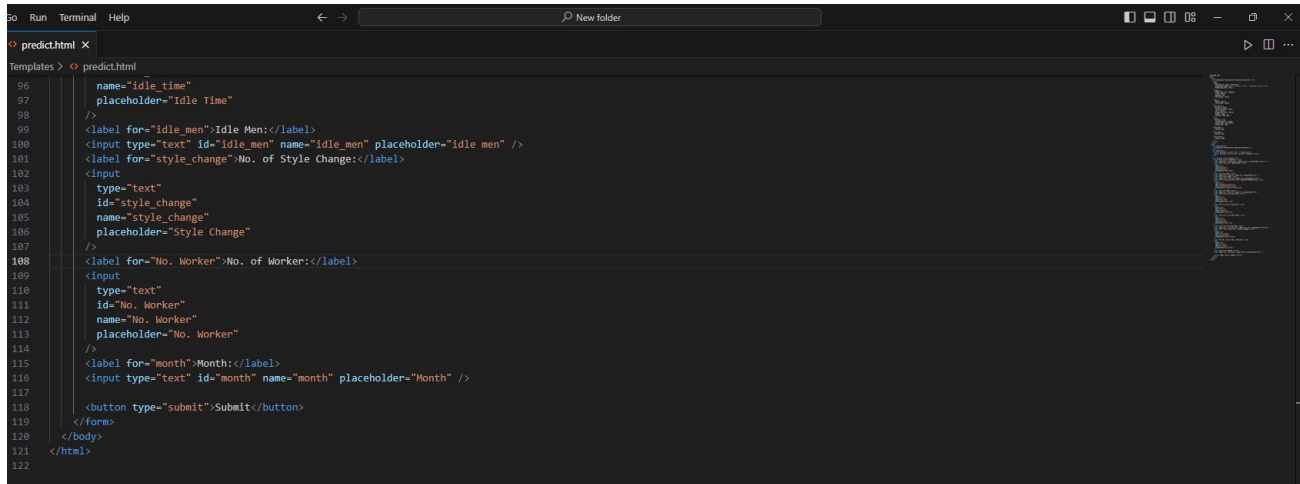
```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

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'))
```

HTML page:



```
96     name="idle_time"
97     placeholder="Idle Time"
98   />
99   <label for="idle_men">Idle Men:</label>
100   <input type="text" id="idle_men" name="idle_men" placeholder="idle men" />
101   <label for="style_change">No. of Style Change:</label>
102   <input
103     type="text"
104     id="style_change"
105     name="style_change"
106     placeholder="Style Change"
107   />
108   <label for="No. Worker">No. of Worker:</label>
109   <input
110     type="text"
111     id="No. Worker"
112     name="No. Worker"
113     placeholder="No. Worker"
114   />
115   <label for="month">Month:</label>
116   <input type="text" id="month" name="month" placeholder="Month" />
117
118   <button type="submit">Submit</button>
119 </form>
120 </body>
121 </html>
122
```

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:

```

@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:

```

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/predictions

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 -

9. RESULTS

9.1 Output Screenshots

Flask running status

Input1

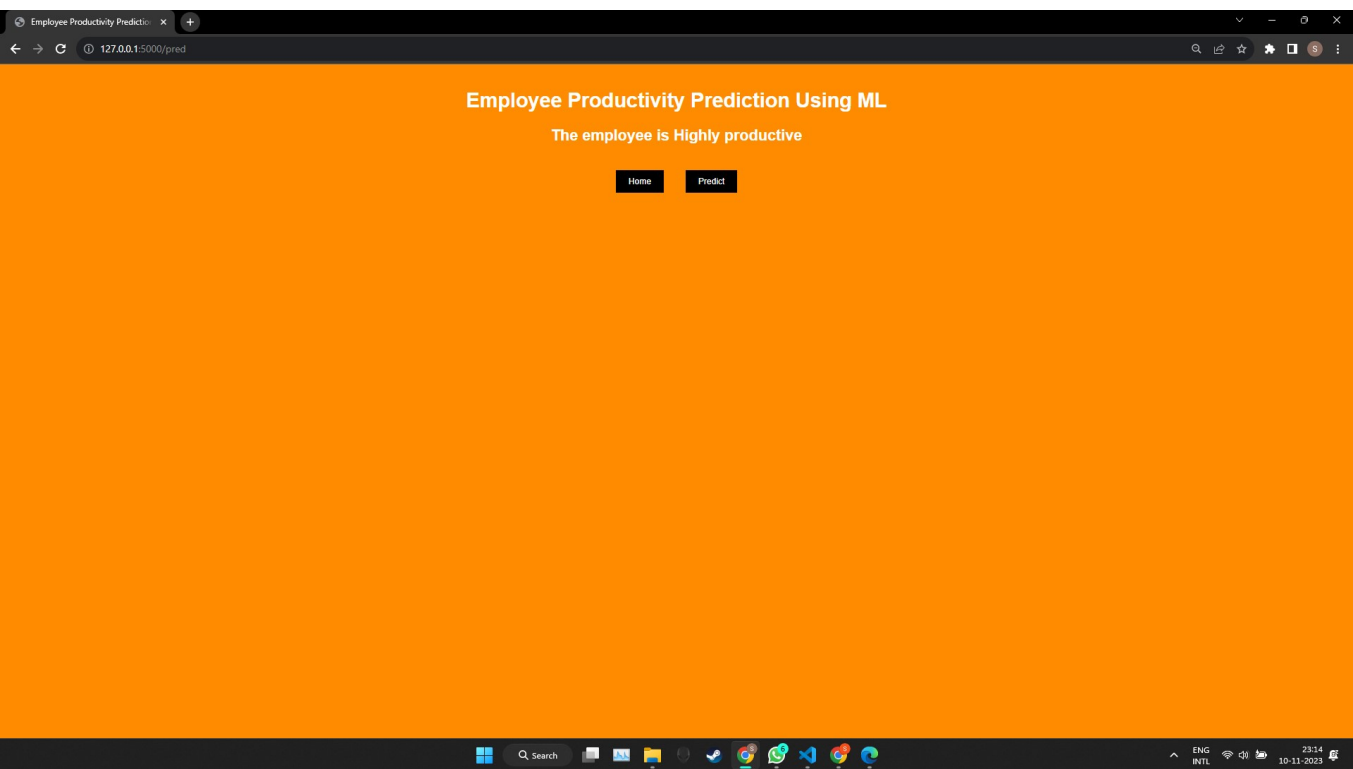
Employee Productivity Prediction Using ML

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

23:13 10-11-2023

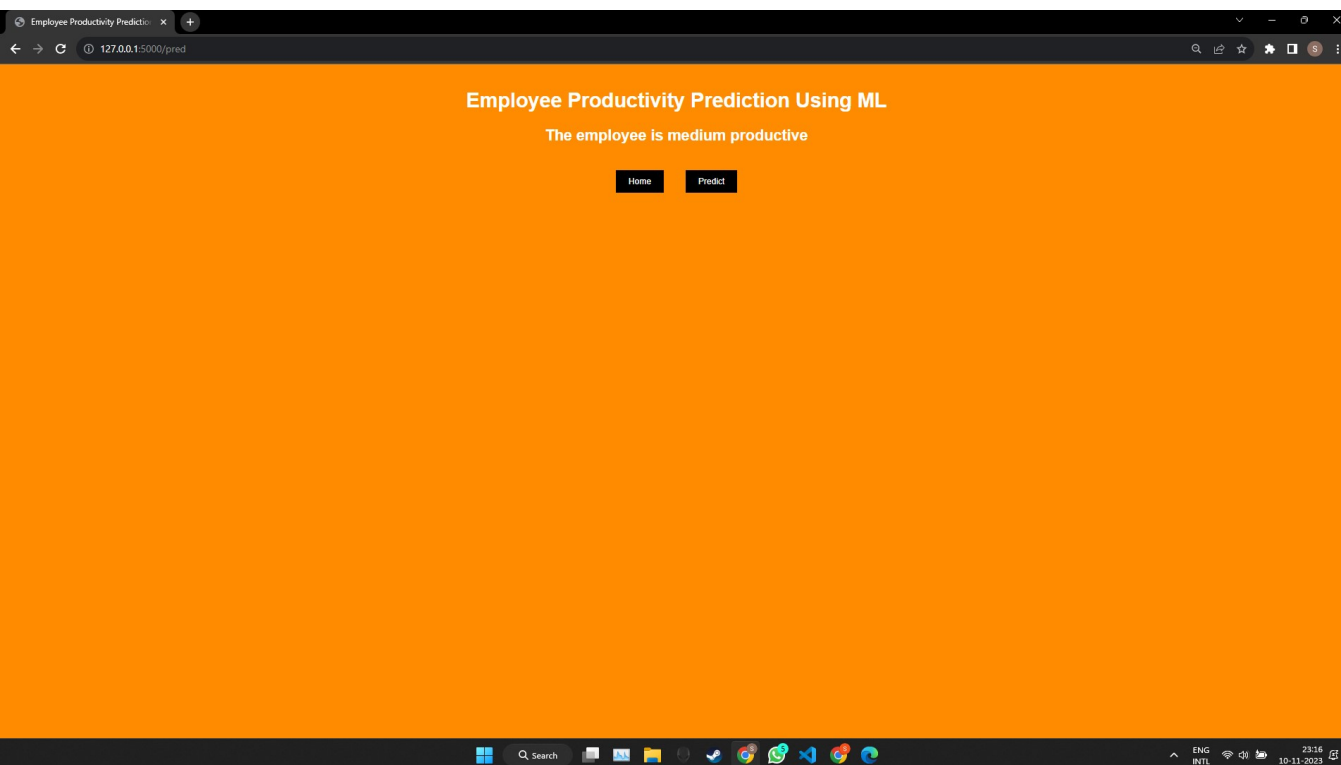
Output1



Input 2

A screenshot of the same web application, but now showing the input form. The browser's address bar shows the URL "127.0.0.1:5000/predict". The application has a blue header with the title "Employee Productivity Prediction Using ML". Below the header, there are two black buttons with white text: "Home" and "Predict". The main content area has a light blue background with three large, stylized question marks in orange, blue, and green. Three cartoon characters are sitting on these question marks: a woman on the orange one, a man on the blue one, and a woman on the green one. On the left side of the page, there is a form with the following fields: "Quarter:" (7), "Department:" (4), "Day:" (61), "Team:" (7), "Targeted Productivity:" (0.80), "SMV:" (5.2), "Over Time:" (1), "Incentive:" (7), "Idle Time:" (2.2), "Idle Men:" (7), "No. of Style Change:" (7), "No. of Worker:" (7.5), and "Month:" (6). A "Submit" button is at the bottom of the form. The browser's taskbar and system tray are visible at the bottom.

Output 2



10. ADVANTAGES & DISADVANTAGES

Advantages of Using Machine Learning for Employee Performance Prediction: *Data-Driven Insights:*

Advantage: ML models can analyze large volumes of historical data to identify patterns and trends, providing valuable insights into factors influencing employee performance.

Objective Evaluation:

Advantage: ML models can provide a more objective evaluation by relying on quantitative data, reducing the impact of subjective biases that may be present in traditional performance assessments.

Identification of Key Performance Indicators (KPIs):

Advantage: ML algorithms can automatically identify relevant KPIs that contribute most to employee performance, helping organizations focus on critical areas for improvement.

Personalized Development Plans:

Advantage: ML models can generate personalized development plans for employees based on their strengths and weaknesses, fostering targeted skill development.

Proactive Management:

Advantage: ML can enable proactive management by identifying potential performance issues early on, allowing organizations to take corrective actions before they escalate.

Efficiency and Time-Saving:

Advantage: ML-driven automation can streamline the performance evaluation process, saving time for both managers and employees.

Continuous Improvement:

Advantage: ML models can continuously learn and adapt, allowing the performance prediction system to improve over time as more data becomes available.

Disadvantages and Challenges of Using Machine Learning for Employee Performance Prediction:

Data Quality and Bias:

Disadvantage: Poor data quality or biased historical data can lead to inaccurate predictions and reinforce existing biases, potentially resulting in unfair evaluations.

Subjectivity and Complexity of Human Behavior:

Disadvantage: ML models may struggle to capture the complexity of human behavior, especially in roles that require subjective judgment or involve nuanced interpersonal skills.

Privacy Concerns:

Disadvantage: Employee performance prediction often involves analyzing personal data, raising privacy concerns. Striking a balance between data utilization and privacy protection is challenging.

Dynamic Nature of Work:

Disadvantage: Work environments are dynamic, and job roles evolve. ML models may struggle to adapt quickly to changing circumstances, leading to outdated predictions.

Resistance and Trust Issues:

Disadvantage: Employees and managers may resist the implementation of ML-based performance prediction systems due to trust issues, lack of understanding, or fear of job security implications.

Interpretable Models:

Disadvantage: Some ML models, especially complex ones, may lack interpretability, making it difficult for users to understand how predictions are generated.

Ethical and Legal Challenges:

Disadvantage: ML models may inadvertently perpetuate or amplify biases, leading to ethical concerns. Legal compliance, especially regarding fairness and discrimination, can pose challenges.

Overemphasis on Quantitative Metrics:

Disadvantage: Overreliance on quantitative metrics may neglect qualitative aspects of employee performance, such as creativity, innovation, or teamwork.

Employee Morale and Stress:

Disadvantage: Constant monitoring and evaluation through ML models can contribute to

stress and reduced morale among employees, potentially impacting overall performance. In summary, while machine learning offers significant advantages in predicting and managing employee performance, it comes with its set of challenges and considerations, including data quality, bias, privacy concerns, and the dynamic nature of human behavior. Organizations need to carefully navigate these issues to ensure the responsible and effective use of ML in performance prediction.

11. CONCLUSION

In conclusion, leveraging machine learning for employee performance prediction presents a promising avenue for organizations seeking data-driven insights and efficiency in managing workforce productivity. The advantages include data-driven insights, objective evaluation, identification of key performance indicators, personalized development plans, proactive management, efficiency gains, and continuous improvement. However, these benefits come with inherent challenges and disadvantages.

The challenges involve issues of data quality and bias, the complexity of human behavior, privacy concerns, the dynamic nature of work, resistance and trust issues, interpretability of models, ethical and legal challenges, overemphasis on quantitative metrics, and potential impacts on employee morale and stress. Striking a balance between utilizing the power of machine learning and addressing these challenges is crucial for the successful implementation of performance prediction systems.

Organizations must be mindful of ethical considerations, prioritize transparency and fairness, and actively work to mitigate biases in the data and models. Balancing the quantitative and qualitative aspects of performance, respecting privacy, and fostering trust among employees are critical factors in the responsible deployment of machine learning for employee performance prediction.

As technology continues to advance, the collaboration between human judgment and machine insights becomes imperative. While machine learning can offer valuable support in decision-making processes, it should be seen as a complement to, rather than a replacement for, human expertise. The key lies in developing and implementing performance prediction systems that align with organizational values, encourage employee development, and contribute positively to the overall workplace environment.

12. FUTURE SCOPE

The future scope of using machine learning for employee performance prediction holds significant potential for continued advancements and improvements in the realm of human resources and workforce management. Here are some key areas that could shape the future of this field:

1. Advanced Predictive Models:

Future developments may involve more sophisticated machine learning models, including deep learning approaches, to better capture the complexity of human behavior and provide more accurate performance predictions.

2. Integration of Real-Time Data:

The incorporation of real-time data streams, such as continuous feedback, project updates, and social interactions, can enhance the timeliness and relevance of performance predictions, allowing for more dynamic and responsive management.

3. Explainable AI and Model Transparency:

Addressing the challenge of model interpretability is crucial. Future systems may focus on developing more explainable AI models to enhance user understanding and trust in the predictions, facilitating better decision-making.

4. Ethical AI Practices:

There will likely be an increased emphasis on ethical considerations in AI, with a focus on reducing biases, ensuring fairness, and safeguarding employee privacy. Industry standards and regulations may emerge to guide organizations in implementing responsible AI practices.

5. Hybrid Approaches:

The future may see the integration of human expertise with machine predictions, creating hybrid systems that leverage the strengths of both. This collaborative approach can address the limitations of fully automated systems and enhance overall decision quality.

6. Personalized Employee Development:

Machine learning algorithms could evolve to provide more personalized and adaptive employee development plans, tailoring recommendations based on individual strengths, weaknesses, and career aspirations.

7. Predictive Analytics for Employee Engagement:

Beyond performance prediction, future applications may extend to predicting employee engagement levels and identifying factors that contribute to a positive or negative work experience. This can aid in proactively addressing potential retention issues.

8. Continuous Learning Systems:

Future systems may focus on continuous learning, adapting to changes in organizational structures, job roles, and industry dynamics. This adaptability can ensure that predictive models remain relevant and effective over time.

9. Human-Centric Design:

Designing systems with a human-centric approach will be critical. Future developments should prioritize user experience, ensuring that employees, managers, and HR professionals find value in these tools and perceive them as supportive rather than intrusive.

10. International Standards and Regulations:

As the use of AI in HR becomes more prevalent, there may be an establishment of international standards and regulations governing the ethical use of machine learning in employee performance prediction, addressing concerns related to fairness, transparency, and privacy.

In conclusion, the future scope of machine learning in employee performance prediction is promising, with an expected evolution towards more sophisticated models, ethical practices, and a holistic approach that considers the well-being and development of the workforce. As technology continues to advance, organizations that embrace these future trends are likely to gain a competitive edge in talent management and employee satisfaction.

13. APPENDIX

Source Code

Python.ipnyb:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-
python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
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
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all
files under the input directory
#correlation analysis
import seaborn as sns
import matplotlib.pyplot as plt
```



```

#Descriptive analysis
data.describe()

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

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

data[*department'].value_counts)
#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)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets
preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside
of the current session
x=data.drop([ actual_productivity' ],axis=1)
y=data['actual_productivity']
    X=x.to_numpy()
    X

#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)

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
from sklearn.impute import SimpleImputer

# 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)

# Fit the imputer on your training data and transform both the training and testing
data
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Create a Linear Regression model
model = LinearRegression()

# Fit the model to the training data with imputed values
model.fit(X_train_imputed, y_train)

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

# 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)

# Threshold for considering R-squared as high
threshold = 0.5

# 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 ")
elif 0.1 < r2 < threshold:
    print("Employee is low productive")

```

```

else:
    print("Employee is low productive")

#Comparing the values

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)))

```

Html.page

<!DOCTYPE html>

<html>

<head>

<title>Employee performance prediction using ml</title>

</head>

<body>

<form action="/submit" method="POST">

<label for="idle_time">Idle Time:</label>

<input type="text" name="idle_time" placeholder="Time">

<label for="style_change">Style Change:</label>

<input type="text" name="style_change" placeholder="Change">

```
<label for="no_of_workers">Number of Workers:</label>
<input type="text" name="no_of_workers" placeholder="Worker">

<label for="month">Month:</label>
<input type="text" name="month" placeholder="Month">

<button type="submit">Submit</button>

</form>

</body>

</html>
```

Flask .py :

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

app=Flask(__name__)
model = pickle.load(open('xgboost_model.pkl','rb'))

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

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

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

```

@app.route ( "/submit")
def home2():
    return render_template( 'submit.html')

@app.route( "/pred", methods=['POST'])
def predict():
    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]]

        print(total)

```

```

prediction = model.predict(total)
print(prediction)
if prediction <=0.3:
    text = 'The employee 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)
except Exception as e:
    return render_template('submit.html',prediction_text="An error occurred: Invalid
Input")

if __name__ == "__main__":
    app.run(debug=True)

```

14.GitHub & Project Demo Link

Github code: <https://github.com/smartinternz02/SI-GuidedProject-603936-1697647640>

Project demo link:

https://drive.google.com/drive/folders/1N9_Z5ZRpfOwg0noierCfVvscKecqDrIY?usp=sharing