Machine Learning Approach For Employee Performance Prediction With IBM

Prepared by

Amit Kumar Thakur

CONTENT

INTRODUCTION :

•Overview:-

In this project we are going to analyze 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.

•Purpose:-

The purpose of this project is to predict the performance of employees.

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

PROJECT DESCRIPTION :

The Machine Learning Approach for Employee Performance Prediction with a comprehensive system designed to analyze various data points related to employees' work performance and use machine learning algorithms, leveraging ML technology stack, to predict and evaluate their future performance. By incorporating factors such as past performance metrics, training data, feedback, and external factors, the system aims to provide insights that can aid in talent management, resource allocation, and workforce optimization strategies.

LITERATURE SURVEY :

- **Existing problem:**

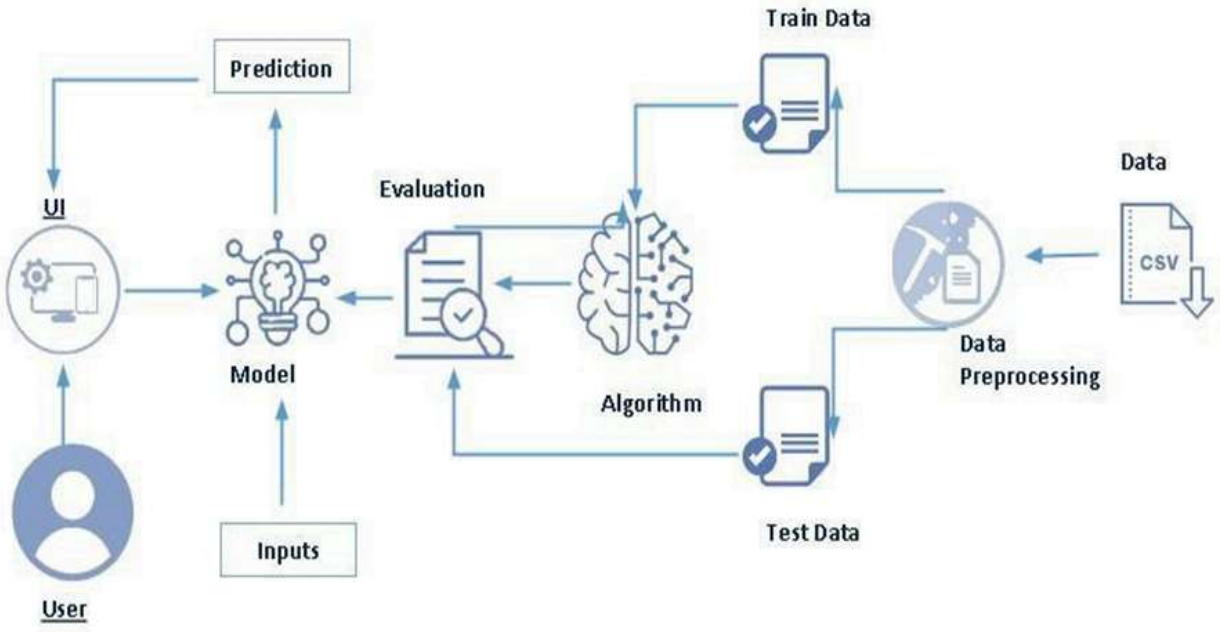
On previous system employee performance is calculated using paper works by evaluating the performance of the employee by hand

- **Proposed solution:**

As an alternative to the existing problem this project is made to automate the performance of the employee.

THEORETICAL ANALYSIS:

- **Block Diagram20**



- **Hardware Minimum Requirement:**
  - **CPU :** PENTIUM III Processor
  - **Memory :** 128 MB • **Cache : 512KB**
  - **Floppy Disk :** 1.44MB • **Hard Disk : 4.3GB**
  - **Display :** 15” Monitor
  - **Key Board:** Standard 108 keys Enhanced Keyboard
  - **Mouse :** MS Serial Mouse

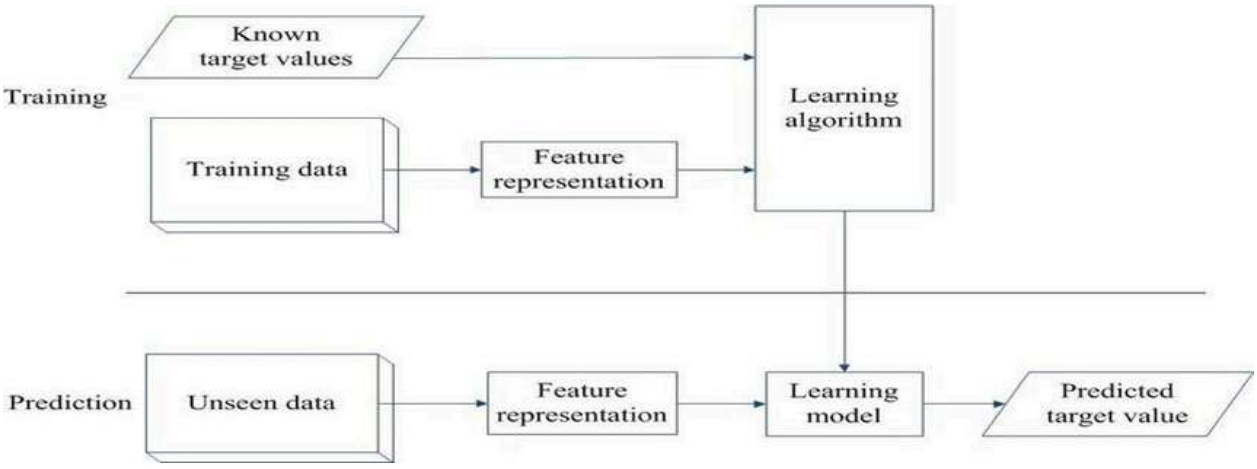
- **Software Minimum Requirement:**
  - **Operating System :** Windows XP, 7, 8 or above
  - **Front Tool :** PHP
  - **Back End Tool :** HTML

#EXPERIMENTAL INVESTIGATIONS :

Based on my analysis since the project is used with Supervised learning techniques namely Support Vector Machines, Random Forest, Naive Bayes, Neural Networks and Logistic Regression. The performance of the employee is analyzed based on the number of days the employee works ,target productivity acquired, over time they worked, how many team members etc and the most accurate result is found out

FLOW CHART :

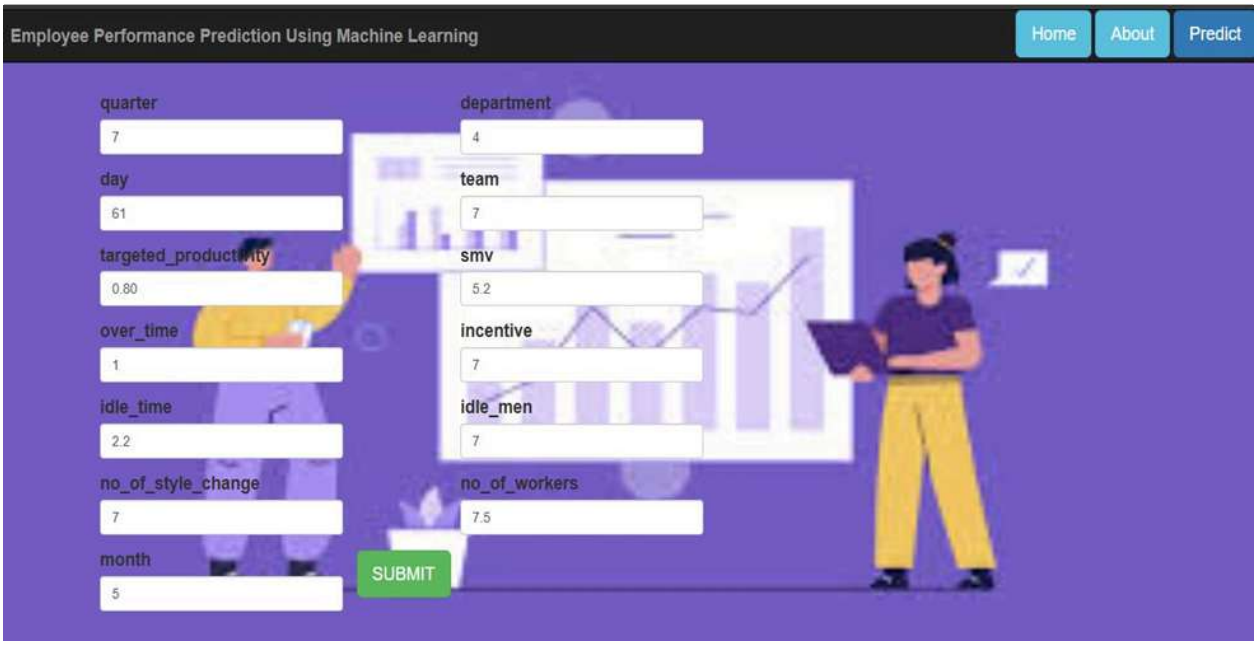
1. User interacts with the UI to enter the input.
2. Entered input is analyzed by the model which is integrated.
3. Once model analyzes the input the prediction is showcased on the UI



RESULT :

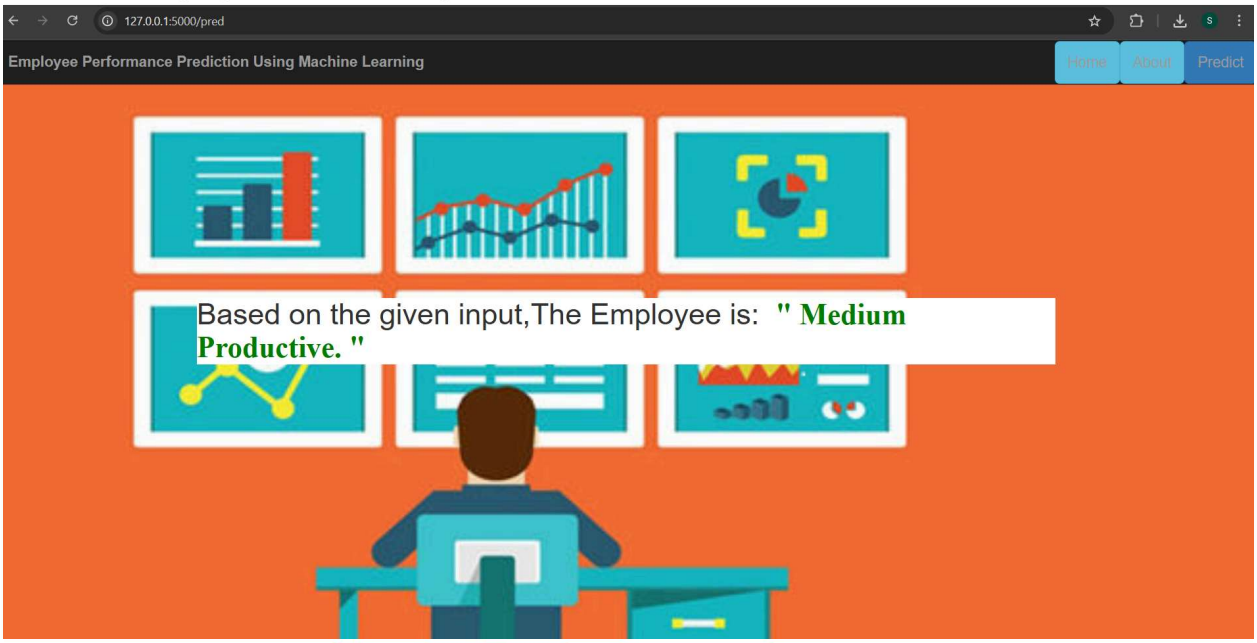
HOMEPAGE :

> INPUT-1



>> OUTPUT-1:

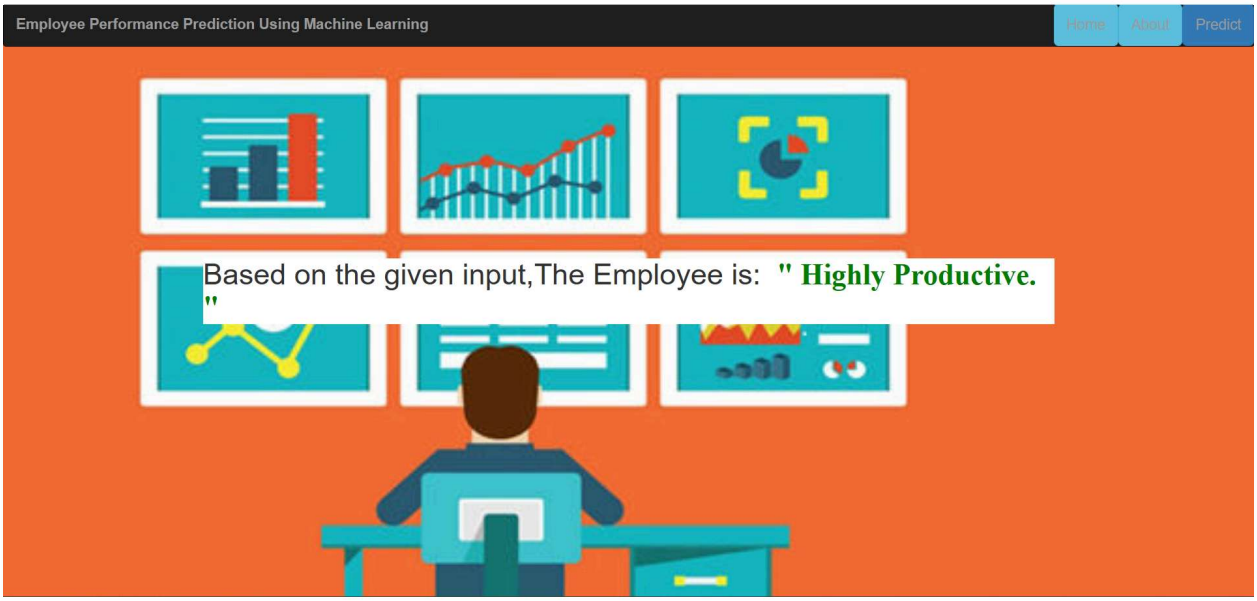
The Employee is medium productive



> INPUT-2:

>> OUTPUT-2:

The employee is Highly productive



 # ADVANTAGES:

1. Provides clarity
2. Enhances efficiency
3. Promotes job satisfaction
4. Increases motivation
5. Enables objective decision-making
6. Helps plan for training needs.

DISADVANTAGES:

1. The absence of goal setting and defined milestones
2. Using performance management solely as a measurement tool
3. Establishing trust
4. Untrained managers
5. It's an annual activity

APPLICATIONS:

a. Attendance
b. Time management
c. Training
d. Initiative & innovation

CONCLUSION:

This project analyzes and predicts 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.

APPENDIX

1. app.py

```python
from flask import Flask, render_template, request
import pickle

app = Flask(__name__)

# Load your ML model (make sure model.pkl is in the same folder)
model = pickle.load(open('model.pkl', 'rb'))

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

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

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

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

@app.route("/pred", methods=['POST'])
def prediction():
    # Extract form data
    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']

    # Prepare input for prediction - convert all to correct types
    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)
    ]]

    prediction = model.predict(total)[0]  # Assuming model.predict returns list/array

    # Decide prediction message
    if prediction <= 0.3:
        text = 'Averagely Productive.'
    elif 0.3 < prediction <= 0.8:
        text = 'Medium Productive.'
    else:
        text = 'Highly Productive.'

    # Render submit.html with prediction text
    return render_template('submit.html', prediction_text=text)

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

2. *home.html*

home.html

Flask > templates > home.html > ...

```
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Home</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css" />
  <style>
    body {
      background-image: url("{{ url_for('static', filename='h.png') }}");
      background-size: cover;
      background-color: lightgreen;
    }
  </style>
</head>
<body>
  <nav class="navbar navbar-inverse">
    <div class="container-fluid">
      <div class="navbar-header">
        <strong>
          <a class="navbar-brand" href="{{ url_for('home') }}">
            Employee Performance Prediction Using Machine Learning
          </a>
        </strong>
      </div>
      <ul class="nav navbar-nav navbar-right">
        <li><a href="{{ url_for('home') }}" class="btn btn-info btn-lg">Home</a></li>
        <li><a href="{{ url_for('about') }}" class="btn btn-info btn-lg">About</a></li>
        <li><a href="{{ url_for('predict') }}" class="btn btn-primary btn-lg">Predict</a></li>
      </ul>
    </div>
  </nav>
  <center>
    <!-- Your content here -->
  </center>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</body>
```

3. about.html

about.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <title>About</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css" />
  <style>
    body {
      background: linear-gradient(to right, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)),
      url("{{ url_for('static', filename='h.png') }}");
      background-size: cover;
      background-color: powderblue;
    }
  </style>
</head>

<body>
  <nav class="navbar navbar-inverse">
    <div class="container-fluid">
      <div class="navbar-header">
        <strong><a class="navbar-brand" href="{{ url_for('home') }}">Employee Performance Prediction Using Machine
            Learning</a></strong>
      </div>
      <ul class="nav navbar-nav navbar-right">
        <li><a href="{{ url_for('home') }}" class="btn btn-info btn-lg">Home</a></li>
        <li><a href="{{ url_for('about') }}" class="btn btn-info btn-lg">About</a></li>
        <li><a href="{{ url_for('predict') }}" class="btn btn-primary btn-lg">Predict</a></li>
      </ul>
    </div>
  </nav>

  <div class="container">
    <center>
      <h2
        style="background: linear-gradient(to right, rgba(201, 201, 201, 0.6), rgba(201, 201, 201, 0.6));color:black;margin-top:20%;margin-ri
      <h2
        style="background: linear-gradient(to right, rgba(201, 201, 201, 0.6), rgba(201, 201, 201, 0.6));color:black;margin-top:20%;margin-ri
        <p>
          Any business's success depends on its employees.<br />
          Businesses that realize this are concerned about employee output and productivity.<br />
          Productivity has a compounding effect at different levels in the workplace, meaning that high productivity at
          a lower level or vice versa can cascade.<br />
          Hence, analysis of performance of employees in any organization is the need of the hour.
        </p>
      </h2>
    </center>
  </div>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</body>

</html>
```

4. submit.html

submit.html

Flask > templates > submit.html > html > body > div.container > h1 > span

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Output</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css" />
  <style>
    body {
      background-image: url("{{ url_for('static', filename='sub.png') }}");
      background-size: cover;
      background-color: LightSteelBlue;
    }
    h3.big {
      line-height: 1.8;
    }
  </style>
</head>
<body>

<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <strong>
        <a class="navbar-brand" href="{{ url_for('home') }}">Employee Performance Prediction Using Machine Learning</a>
      </strong>
    </div>
    <ul class="nav navbar-nav navbar-right">
      <li><a href="{{ url_for('home') }}" class="btn btn-info btn-lg">Home</a></li>
      <li><a href="{{ url_for('about') }}" class="btn btn-info btn-lg">About</a></li>
      <li><a href="{{ url_for('predict') }}" class="btn btn-primary btn-lg">Predict</a></li>
    </ul>
  </div>
</nav>

<div class="container"style="width:70%">
  <h1 style="background-color: white; margin-top:23%; width:100%;">
```

```
<div class="container"style="width:70%">
  <h1 style="background-color: white; margin-top:23%; width:100%;">
    Based on the given input,The Employee is:  
    <span style="color: green;font-weight:bold;font-family:Times new roman;">" {{ prediction_text }} "</span>
  </h1>
</div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</body>
</html>
```

5.predict.html

predict.html

Flask > templates > predict.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Predict</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <style>
    body {
      background-image: url("{{ url_for('static', filename='Pred.png') }}");
      background-size: cover;
      background-color: CadetBlue;
    }
    h3.big {
      line-height: 1.8;
    }
    .form-section {
      background-color: rgba(255, 255, 255, 0.8);
      padding: 20px;
      border-radius: 10px;
      margin-top: 20px;
    }
  </style>
</head>
<body>

<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <strong><a class="navbar-brand" href="{{ url_for('home') }}">Employee Performance Prediction</a></strong>
    </div>
    <ul class="nav navbar-nav navbar-right">
      <li><a href="{{ url_for('home') }}" class="btn btn-info btn-lg">Home</a></li>
      <li><a href="{{ url_for('about') }}" class="btn btn-info btn-lg">About</a></li>
      <li><a href="{{ url_for('predict') }}" class="btn btn-primary btn-lg">Predict</a></li>
    </ul>
  </div>
</nav>

<div class="container form-section">
  <form action="{{ url_for('prediction') }}" method="post">

    <!-- Form fields (same as your original code) -->
    <!-- Quarter & Department -->
    <div class="form-group row">
      <div class="col-xs-3">
        <label for="f1">Quarter</label>
        <input class="form-control" id="f1" name="quarter" required type="text">
      </div>
      <div class="col-xs-1"></div>
      <div class="col-xs-3">
        <label for="f2">Department</label>
        <input class="form-control" id="f2" name="department" required type="text">
      </div>
    </div>

    <!-- Day & Team -->
    <div class="form-group row">
      <div class="col-xs-3">
        <label for="f3">Day</label>
        <input class="form-control" id="f3" name="day" required type="text">
      </div>
      <div class="col-xs-1"></div>
      <div class="col-xs-3">
        <label for="f4">Team</label>
        <input class="form-control" id="f4" name="team" required type="text">
      </div>
    </div>
```

```
    <!-- Targeted Productivity & SMV -->
    <div class="form-group row">
      <div class="col-xs-3">
        <label for="f5">Targeted Productivity</label>
        <input class="form-control" id="f5" name="targeted_productivity" required type="text">
      </div>
      <div class="col-xs-1"></div>
      <div class="col-xs-3">
        <label for="f6">SMV</label>
        <input class="form-control" id="f6" name="smv" required type="text">
      </div>
    </div>

    <!-- Overtime & Incentive -->
    <div class="form-group row">
      <div class="col-xs-3">
        <label for="f7">Over Time</label>
        <input class="form-control" id="f7" name="over_time" required type="text">
      </div>
      <div class="col-xs-1"></div>
      <div class="col-xs-3">
        <label for="f8">Incentive</label>
        <input class="form-control" id="f8" name="incentive" required type="text">
      </div>
    </div>

    <!-- Idle Time & Idle Men -->
    <div class="form-group row">
      <div class="col-xs-3">
        <label for="f9">Idle Time</label>
        <input class="form-control" id="f9" name="idle_time" required type="text">
      </div>
```

```
      <div class="col-xs-1"></div>
      <div class="col-xs-3">
        <label for="f10">Idle Men</label>
        <input class="form-control" id="f10" name="idle_men" required type="text">
      </div>
    </div>

    <!-- Style Change & Workers -->
    <div class="form-group row">
      <div class="col-xs-3">
        <label for="f11">No. of Style Change</label>
        <input class="form-control" id="f11" name="no_of_style_change" required type="text">
      </div>
      <div class="col-xs-1"></div>
      <div class="col-xs-3">
        <label for="f12">No. of Workers</label>
        <input class="form-control" id="f12" name="no_of_workers" required type="text">
      </div>
    </div>

    <!-- Month -->
    <div class="form-group row">
      <div class="col-xs-3">
        <label for="f13">Month</label>
        <input class="form-control" id="f13" name="month" required type="text">
      </div>
    </div>

    <!-- ✅ Submit button -->
    <div class="form-group">
      <button type="submit" class="btn btn-success btn-lg">SUBMIT</button>
```

```
          <input class="form-control" id="f13" name="month" required type="text">
        </div>
      </div>

      <!-- ✅ Submit button -->
      <div class="form-group">
        <button type="submit" class="btn btn-success btn-lg">SUBMIT</button>
      </div>

    </form>
  </div>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  </body>
  </html>
```

6. model training:

jupyter Employee_performance_prediction Last Checkpoint: 8 days ago

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

###importing library

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import MultiColumnLabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
import xgboost as Xgb
import pickle
```

###Read The Data

[2]:
```
data=pd.read_csv("garments_worker_productivity.csv")
```

[3]:
```
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
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59.0
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8.0
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30.5
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30.5
4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0

Correlation Analysis

[4]:
```
import seaborn as sns
import matplotlib.pyplot as plt
numeric_data = data.select_dtypes(include='number')
corrMatrix = numeric_data.corr()
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(corrMatrix, annot=True, linewidths=0.1, ax=ax)
plt.show()
```

jupyter Employee_performance_prediction Last Checkpoint: 11 days ago

File Edit View Run Kernel Settings Help

Trusted

Markdown

JupyterLab Python 3 (ipykernel)

Descriptive Analysis

```
[5]: #descriptive analysis
data.describe()
```

[5]:

	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_pr
count	1197.000000	1197.000000	1197.000000	691.000000	1197.000000	1197.000000	1197.000000	1197.000000	1197.000000	1197.000000	11
mean	6.426901	0.729632	15.062172	1190.465991	4567.460317	38.210526	0.730159	0.369256	0.150376	34.609858	
std	3.463963	0.097891	10.943219	1837.455001	3348.823563	160.182643	12.709757	3.268987	0.427848	22.197687	
min	1.000000	0.070000	2.900000	7.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	
25%	3.000000	0.700000	3.940000	774.500000	1440.000000	0.000000	0.000000	0.000000	0.000000	9.000000	
50%	6.000000	0.750000	15.260000	1039.000000	3960.000000	0.000000	0.000000	0.000000	0.000000	34.000000	
75%	9.000000	0.800000	24.260000	1252.500000	6960.000000	50.000000	0.000000	0.000000	0.000000	57.000000	
max	12.000000	0.800000	54.560000	23122.000000	25920.000000	3600.000000	300.000000	45.000000	2.000000	89.000000	

Checking For NULL Value

```
[6]: data.shape
(1197,15)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1197 entries, 0 to 1196
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   date                   1197 non-null   object
```

jupyter Employee_performance_prediction Last Checkpoint: 11 days ago

File Edit View Run Kernel Settings Help

Trusted

Markdown

JupyterLab Python 3 (ipykernel)

```
 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
```

```
[7]: data.isnull().sum()
```

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

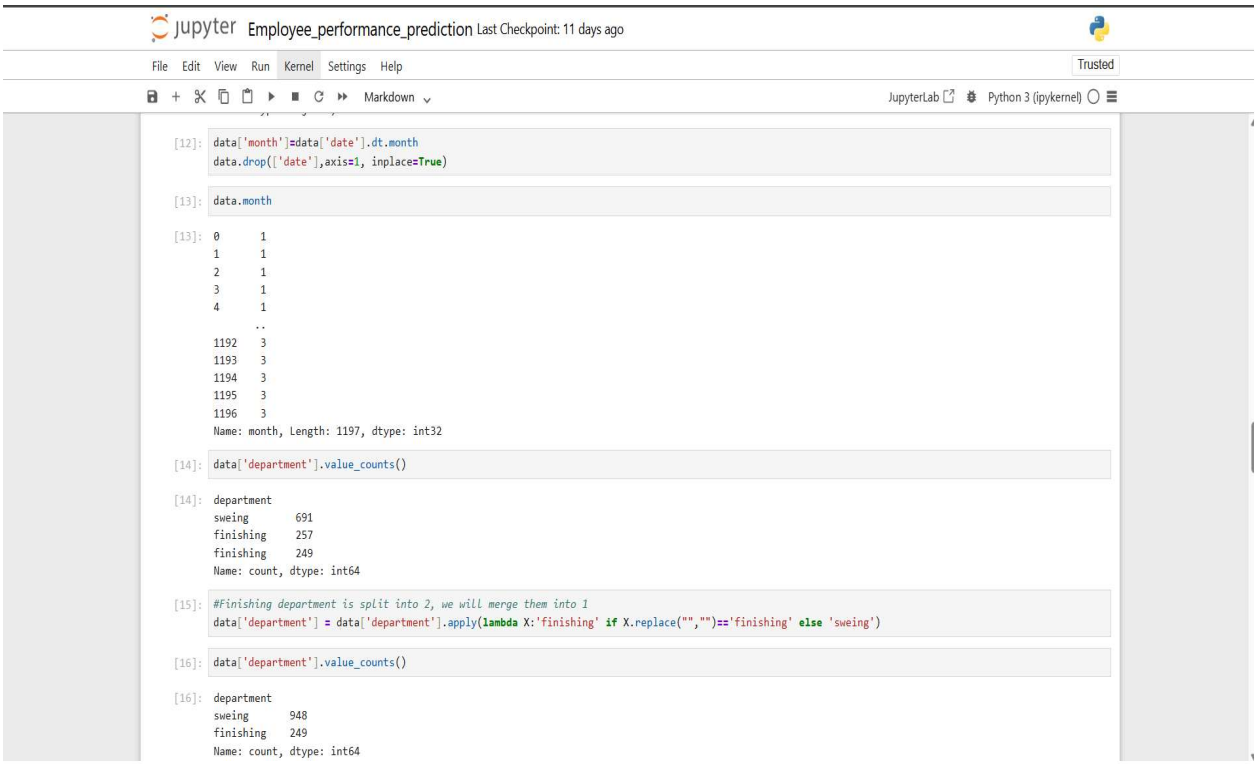
```
[8]: data.drop(['wip'], axis=1, inplace=True, errors='ignore')
```

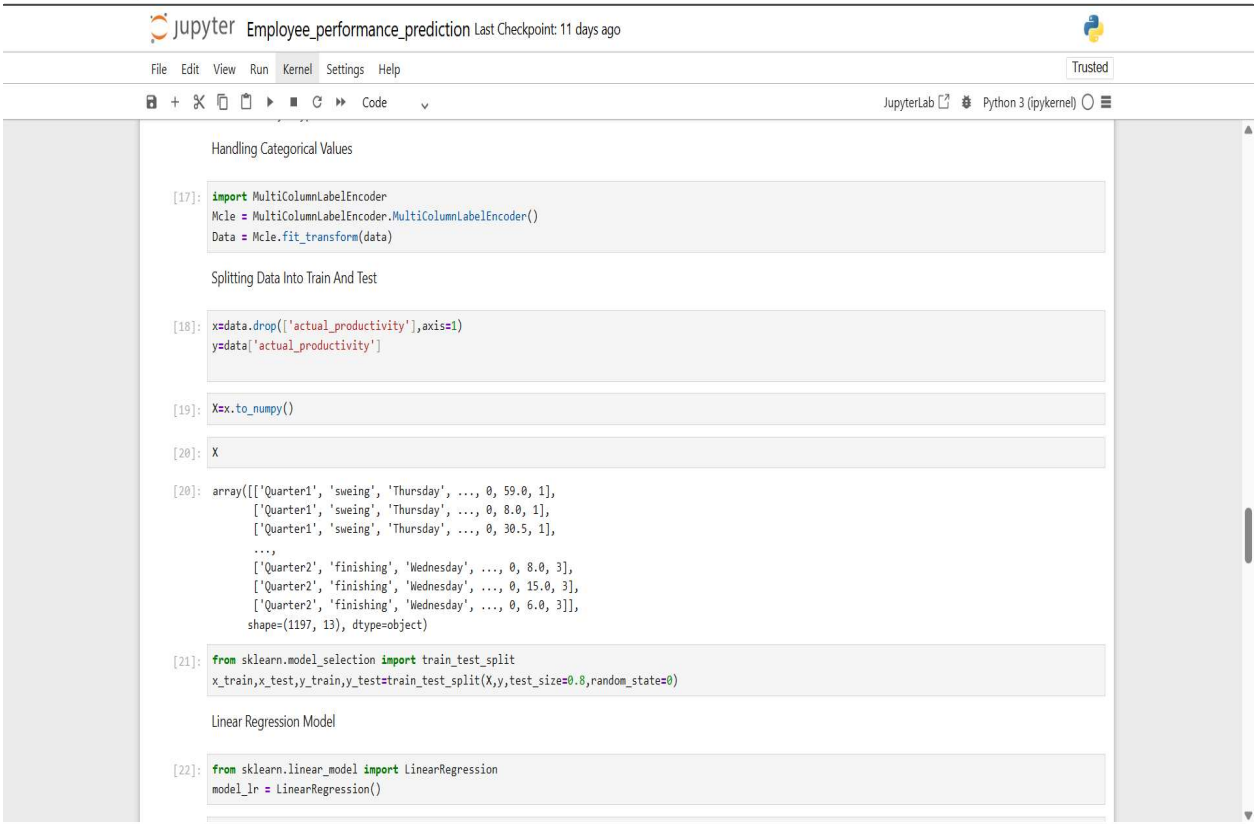
Handling Data & Department Column

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

```
[10]: data.date
```

```
[10]: 0      2015-01-01
1      2015-01-01
2      2015-01-01
3      2015-01-01
```


jupyter Employee_performance_prediction Last Checkpoint: 11 days ago
File Edit View Run Kernel Settings Help
Trusted
Markdown
JupyterLab
Python 3 (ipykernel)
[12]: data['month']=data['date'].dt.month
data.drop(['date'],axis=1, inplace=True)
[13]: data.month
[13]: 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
[14]: data['department'].value_counts()
[14]: department
sweing 691
finishing 257
finishing 249
Name: count, dtype: int64
[15]: #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')
[16]: data['department'].value_counts()
[16]: department
sweing 948
finishing 249
Name: count, dtype: int64


jupyter Employee_performance_prediction Last Checkpoint: 11 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
Handling Categorical Values
[17]: import MultiColumnLabelEncoder
Mcle = MultiColumnLabelEncoder.MultiColumnLabelEncoder()
Data = Mcle.fit_transform(data)
Splitting Data Into Train And Test
[18]: x=data.drop(['actual_productivity'],axis=1)
y=data['actual_productivity']
[19]: X=x.to_numpy()
[20]: X
[20]: array([['Quarter1', 'sweing', 'Thursday', ..., 0, 59.0, 1],
['Quarter1', 'sweing', 'Thursday', ..., 0, 8.0, 1],
['Quarter1', 'sweing', 'Thursday', ..., 0, 30.5, 1],
...,
['Quarter2', 'finishing', 'Wednesday', ..., 0, 8.0, 3],
['Quarter2', 'finishing', 'Wednesday', ..., 0, 15.0, 3],
['Quarter2', 'finishing', 'Wednesday', ..., 0, 6.0, 3]],
shape=(1197, 13), dtype=object)
[21]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.8,random_state=0)
Linear Regression Model
[22]: from sklearn.linear_model import LinearRegression
model_lr = LinearRegression()

Linear Regression Model

[22]:
```
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression()
```

[23]:
```
import pandas as pd

def preprocess_input_data(data_array):
    # Step 1: Define column names
    columns = ['Quarter', 'Department', 'Day', 'Feature1', 'Feature2', 'Feature3', 'Feature4',
               'Feature5', 'Feature6', 'Feature7', 'Feature8', 'Feature9', 'TargetCategory']

    # Step 2: Convert to DataFrame
    df = pd.DataFrame(data_array, columns=columns)

    # Step 3: Convert numeric columns
    numeric_cols = ['Feature1', 'Feature2', 'Feature3', 'Feature4',
                    'Feature5', 'Feature6', 'Feature7', 'Feature8', 'Feature9']
    df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric)

    # Step 4: Encode categorical columns
    categorical_cols = ['Quarter', 'Department', 'Day']
    df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

    return df_encoded
```

[24]:
```
x_train_encoded = preprocess_input_data(x_train)
model_lr.fit(x_train_encoded, y_train)
```

[24]:
```
LinearRegression
LinearRegression()
```

jupyter Employee_performance_prediction Last Checkpoint: 11 days ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[24]:
```
LinearRegression
LinearRegression()
```

[25]:
```
x_test_encoded = preprocess_input_data(x_test)
x_test_encoded = x_test_encoded.reindex(columns=x_train_encoded.columns, fill_value=0)
```

[26]:
```
pred_test=model_lr.predict(x_test_encoded)
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: 0.5610566052960411
test_MAE: 0.1852563678501054
R2_score:-17.508181649759166
```

Random Forest Model

[27]:
```
from sklearn.ensemble import RandomForestRegressor
model_rf = RandomForestRegressor(n_estimators=200,max_depth=5)
```

[28]:
```
model_rf.fit(x_test_encoded,y_test)
```

[28]:
```
RandomForestRegressor
RandomForestRegressor(max_depth=5, n_estimators=200)
```

[29]:
```
pred = model_rf.predict(x_test_encoded)
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: 0.011594019080720594
```

jupyter Employee_performance_prediction Last Checkpoint: 11 days ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

Xgboost Model

[30]:
```
import xgboost as xgb
model_xgb = xgb.XGBRegressor(n_estimators=200, max_depth=5, learning_rate=0.1)
```

[31]:
```
from sklearn.preprocessing import LabelEncoder

for col in x_test_encoded.select_dtypes(include='object').columns:
    le = LabelEncoder()
    x_test_encoded[col] = le.fit_transform(x_test_encoded[col])
```

[32]:
```
model_xgb.fit(x_test_encoded,y_test)
```

[32]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=0.1, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=5,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=200,
             n_jobs=None, num_parallel_tree=None, ...)
```

[33]:
```
pred3 = model_xgb.predict(x_test_encoded)
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: 0.00334252451720476
test_MAE: 0.03372618245067457
```

```
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: 0.011787060827356876
test_MAE: 0.07284284213672713
R2_score:0.6111674635852986
```

[36]:
```
#XGBoot
pred = model_rf.predict(x_test_encoded)
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: 0.011787060827356876
test_MAE: 0.07284284213672713
R2_score:0.6111674635852986
```

Evaluating Performance Of The Model And Saving The Model

[37]:
```
pred3 = model_xgb.predict(x_test_encoded)
```

[38]:
```
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: 0.00334252451720476
test_MAE: 0.03372618245067457
R2_score:0.8897365250685236
```

[40]:
```
pickle.dump(model_xgb, open('model.pkl', 'wb'))
print(" Model successfully saved as 'model.pkl' ...by Amit_Kumar_Thakur")
```

```
Model successfully saved as 'model.pkl' ...by Amit_Kumar_Thakur
```

#Github Link:

https://github.com/AmitKumar1712/Employee_Performance_Prediction.git

#Demo link:

https://youtu.be/GJwmefZejMQ?si=UW2PBzenD5qb5eQZ