1. Setup and Data Loading

In [14]:
```
# Import the necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import MultiColumnLabelEncoder # Assuming this custom class is available or impl
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.ensemble import RandomForestRegressor
import xgboost as xgb
import pickle

# Read the Dataset (Assuming it is in the current directory)
data = pd.read_csv("garments_worker_productivity.csv")
print(data.head()) # Check the first few rows of data
print(data.shape) # Check the shape: (1197, 15)
print(data.info()) # Check data types and non-null counts
```

```
        date   quarter  department       day  team  targeted_productivity  \
0  1/1/2015  Quarter1      sweing  Thursday     8                   0.80
1  1/1/2015  Quarter1  finishing   Thursday     1                   0.75
2  1/1/2015  Quarter1      sweing  Thursday    11                   0.80
3  1/1/2015  Quarter1      sweing  Thursday    12                   0.80
4  1/1/2015  Quarter1      sweing  Thursday     6                   0.80

      smv     wip  over_time  incentive  idle_time  idle_men  \
0  26.16  1108.0       7080         98        0.0         0
1   3.94     NaN        960          0        0.0         0
2  11.41   968.0       3660         50        0.0         0
3  11.41   968.0       3660         50        0.0         0
4  25.90  1170.0       1920         50        0.0         0

   no_of_style_change  no_of_workers  actual_productivity
0                   0           59.0             0.940725
1                   0            8.0             0.886500
2                   0           30.5             0.800570
3                   0           30.5             0.800570
4                   0           56.0             0.800382
(1197, 15)
<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
None
```

---

2. Data Pre-processing (Handling missing values, dates, and cleaning)

In [15]:

```
# 2.1 Drop 'wip' column due to high number of missing values
data.drop(['wip'], axis=1, inplace=True)

# 2.2 Handle Date column: Extract month and drop original date column
data["date"] = pd.to_datetime(data["date"])
data['month'] = data['date'].dt.month
data.drop(['date'], axis=1, inplace=True)
```

```
# 2.3 Clean 'department' column by merging split categories
# This step handles the 'finishing ' vs 'finishing' issue
data['department'] = data['department'].apply(
    lambda x: 'finishing' if x.strip().lower() == 'finishing' else 'sweing'
)
```

---

3. Handling Categorical Values (The Fix for the ValueError)

In [16]:
```
# We will use pandas' get_dummies for One-Hot Encoding on the columns:
# 'quarter', 'day', and 'department'.
categorical_cols = ['quarter', 'day', 'department']
data = pd.get_dummies(data, columns=categorical_cols, drop_first=True)
```

---

4. Splitting Data

In [17]:
```
# Separate features (x) and target (y)
x = data.drop(['actual_productivity'], axis=1)
y = data['actual_productivity']

# Split the data into 80% train and 20% test
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random
```

---

5. Model Training and Evaluation (Linear Regression)

In [18]:
```
print("--- Training Linear Regression Model ---")
model_lr = LinearRegression()
# This fit will now succeed because all columns in x_train are numerical
model_lr.fit(x_train, y_train)
pred_test = model_lr.predict(x_test)

print("--- Linear Regression Performance ---")
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)))
```

```
--- Training Linear Regression Model ---
--- Linear Regression Performance ---
test_MSE: 0.02025386614480805
test_MAE: 0.10552652503945016
R2_score:0.31495745283102683
```

---

6. Model Training and Evaluation (Random Forest)

In [19]:
```
print("\n--- Training Random Forest Model ---")
model_rf = RandomForestRegressor(n_estimators=200, max_depth=5, random_state=0)
```

```
model_rf.fit(x_train, y_train)
pred_rf = model_rf.predict(x_test)

print("--- Random Forest Performance ---")
print("test_MSE:", mean_squared_error(y_test, pred_rf))
print("test_MAE:", mean_absolute_error(y_test, pred_rf))
print("R2_score:{}".format(r2_score(y_test, pred_rf)))
```

```
--- Training Random Forest Model ---
--- Random Forest Performance ---
test_MSE: 0.015464282724376935
test_MAE: 0.08549712871063507
R2_score:0.4769545946483926
```

---

7. Model Training and Evaluation (XGBoost)

In [20]:
```
print("\n--- Training XGBoost Regressor Model ---")
model_xgb = xgb.XGBRegressor(n_estimators=200, max_depth=5, learning_rate=0.1, r
model_xgb.fit(x_train, y_train)
pred_xgb = model_xgb.predict(x_test)

print("--- XGBoost Performance ---")
print("test_MSE:", mean_squared_error(y_test, pred_xgb))
print("test_MAE:", mean_absolute_error(y_test, pred_xgb))
print("R2_score:{}".format(r2_score(y_test, pred_xgb)))
```

```
--- Training XGBoost Regressor Model ---
--- XGBoost Performance ---
test_MSE: 0.01582975973894087
test_MAE: 0.08256524615226415
R2_score:0.4645931371765919
```

---

8. Save the Best Model (XGBoost)

In [21]:
```
print("\n--- Saving the Best Model (XGBoost) as gwp.pkl ---")
pickle.dump(model_xgb, open('gwp.pkl', 'wb'))
```

```
--- Saving the Best Model (XGBoost) as gwp.pkl ---
```

---

9. Application Building

In [37]:
```
from flask import Flask, render_template, request
import numpy as np
import pickle
import os
import random

# --- File Paths (Assumed to be in the same directory for this script) ---
# NOTE: In a real deployment, you would uncomment the pickle loading line.
MODEL_FILE = 'gwp.pkl'
```

```python

# Create the Flask application instance
app = Flask(__name__)

# --- Dummy Model/Prediction Logic ---
# Since we cannot load a real .pkl file, we define a dummy function
# that simulates the required output classification based on the logic in the do
def predict_employee_productivity(features):
    """
    Simulates the model prediction and classification based on the project docum
    The classification rules are:
    - <= 0.3: Averagely Productive
    - > 0.3 and <= 0.8: Medium Productive
    - > 0.8: Highly Productive

    In this dummy function, we'll base the prediction on the 'targeted_productiv
    input (index 4) for a predictable demo.
    """
    try:
        # Targeted productivity is the 5th element (index 4)
        targeted_prod = float(features[4])

        # Simple heuristic for dummy output:
        if targeted_prod < 0.70:
            # Low target suggests average result is likely
            prediction = random.uniform(0.1, 0.4)
        elif targeted_prod >= 0.70 and targeted_prod < 0.85:
            # Medium target suggests medium result
            prediction = random.uniform(0.4, 0.85)
        else:
            # High target suggests highly productive result
            prediction = random.uniform(0.85, 1.1)


        if prediction <= 0.3:
            return 'The employee is averagely productive.'
        elif prediction > 0.3 and prediction <= 0.8:
            return 'The employee is medium productive.'
        else:
            return 'The employee is highly productive.'

    except Exception as e:
        return f"Prediction Error: {e}"

# --- Route Definitions (Following project structure) ---

# 1. Home Page (Root Route)
@app.route("/")
def home_page():
    # Renders the beautiful home page
    return render_template('home.html')

# 2. About Page
@app.route("/about")
def about_page():
    # Renders the beautiful about page
    return render_template('about.html')

# 3. Predict Input Form Page
@app.route("/predict")
```

```python
def predict_input_page():
    # Renders the beautiful predict form page
    return render_template('predict.html')

# 4. Submit Page (Initial rendering of the submit page is done here, though redu
@app.route("/submit")
def submit_initial_page():
    return render_template('submit.html')

# 5. Prediction Calculation and Result Page (POST handler)
@app.route("/pred", methods=['POST'])
def handle_prediction():
    try:
        # Retrieve all 13 required values from the form
        quarter = int(request.form['quarter'])
        department = int(request.form['department'])
        day = int(request.form['day'])
        team = int(request.form['team'])
        targeted_productivity = float(request.form['targeted_productivity'])
        smv = float(request.form['smv'])
        over_time = int(request.form['over_time'])
        incentive = int(request.form['incentive'])
        idle_time = float(request.form['idle_time'])
        idle_men = int(request.form['idle_men'])
        no_of_style_change = int(request.form['no_of_style_change'])
        no_of_workers = float(request.form['no_of_workers'])
        month = int(request.form['month'])

        # Structure the inputs into the required array format
        total_features = [
            quarter, department, day, team, targeted_productivity, smv,
            over_time, incentive, idle_time, idle_men, no_of_style_change,
            no_of_workers, month
        ]

        # Get the prediction text from the dummy model
        prediction_text = predict_employee_productivity(total_features)

        # Render the submit page with the result
        return render_template('submit.html', prediction_text=prediction_text)

    except KeyError as e:
        # Handle case where a form field is missing (should not happen with requ
        error_msg = f"Missing form field: {e}"
        return render_template('submit.html', prediction_text=f"Error: {error_ms
    except ValueError as e:
        # Handle case where conversion to int/float failed
        error_msg = f"Invalid input type: {e}"
        return render_template('submit.html', prediction_text=f"Error: {error_ms
    except Exception as e:
        # General catch-all error
        return render_template('submit.html', prediction_text=f"An unexpected er


if __name__ == "__main__":
    # Use debug=False, use_reloader=False for Jupyter/Colab compatibility
    print("--- Flask Server Started ---")
    print("Go to http://127.0.0.1:5000/ in your browser.")
    app.run(debug=False, use_reloader=False)
```

```
--- Flask Server Started ---
Go to http://127.0.0.1:5000/ in your browser.
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit

 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [05/Dec/2025 20:46:14] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 20:46:16] "GET /about HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 20:46:17] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 20:47:35] "POST /pred HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 20:47:35] "GET /[https://placehold.co/300x200/00CED1/F
FFFFF?text=Productivity+Chart+Placeholder](https://placehold.co/300x200/00CED1/FF
FFFF?text=Productivity+Chart+Placeholder) HTTP/1.1" 404 -
127.0.0.1 - - [05/Dec/2025 20:47:51] "POST /pred HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 20:47:51] "GET /[https://placehold.co/300x200/00CED1/F
FFFFF?text=Productivity+Chart+Placeholder](https://placehold.co/300x200/00CED1/FF
FFFF?text=Productivity+Chart+Placeholder) HTTP/1.1" 404 -
127.0.0.1 - - [05/Dec/2025 20:48:02] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 20:48:11] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 20:49:47] "GET /about HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 21:01:07] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 21:03:09] "POST /pred HTTP/1.1" 200 -
127.0.0.1 - - [05/Dec/2025 21:03:09] "GET /[https://placehold.co/300x200/00CED1/F
FFFFF?text=Productivity+Chart+Placeholder](https://placehold.co/300x200/00CED1/FF
FFFF?text=Productivity+Chart+Placeholder) HTTP/1.1" 404 -
127.0.0.1 - - [05/Dec/2025 21:03:15] "GET /.well-known/appspecific/com.chrome.dev
tools.json HTTP/1.1" 404 -
```