**Employee Productivity Prediction using Machine Learning**

**Author:** Abhijeet Kushwaha

**Date:** August 10, 2025

**Version:** 1.0

**1. Abstract**

This project presents a comprehensive system designed to predict the productivity of garment workers using machine learning. By analyzing a range of factors—such as team performance, targeted goals, and financial incentives—we developed a predictive model to forecast worker output. This model is deployed as a user-friendly web application using the Flask framework, allowing managers to input worker-specific data and receive an instant productivity prediction. The primary goal is to provide a data-driven tool that can aid in talent management, resource allocation, and workforce optimization strategies, ultimately enhancing organizational efficiency.

**2. Project Objective**

The main objectives of this project are:

* To perform a thorough exploratory data analysis (EDA) on the garment worker productivity dataset to uncover key insights and relationships.
* To preprocess and clean the data to make it suitable for machine learning.
* To build, train, and evaluate several regression models to find the most accurate predictor of employee productivity.
* To save the best-performing model for future use.
* To develop a web application with a clean and intuitive user interface where users can input data and receive predictions from the trained model.

**3. Dataset Information**

* **Dataset Used:** garments\_worker\_productivity.csv
* **Description:** This dataset contains various attributes related to the productivity of garment factory workers.
* **Key Features (Input Variables):** quarter, department, day, team, targeted\_productivity, smv, over\_time, incentive, idle\_time, idle\_men, no\_of\_style\_change, no\_of\_workers, month.
* **Target Variable (What we predict):** actual\_productivity.

**4. Technology Stack**

* **Language:** Python 3.x
* **Data Manipulation & Analysis:** Pandas, NumPy
* **Data Visualization:** Matplotlib, Seaborn
* **Machine Learning:** Scikit-learn, XGBoost
* **Web Framework:** Flask
* **Development Environment:** Jupyter Notebook, Visual Studio Code

**5. Machine Learning Workflow**

The entire machine learning process was conducted in the Training files/Employee\_Prediction.ipynb notebook.

**5.1. Data Loading and Initial Analysis**

The dataset was loaded into a Pandas DataFrame. Initial analysis involved using data.head() to view the first few rows and data.describe() to get a statistical summary. A correlation heatmap was generated to visualize the initial relationships between numerical features.

**5.2. Data Preprocessing and Cleaning**

1. **Handling Missing Values:** We checked for null values using data.isnull().sum(). The wip column was found to have a significant number of missing values and was dropped from the dataset.
2. **Date Feature Engineering:** The date column was converted from a string to a proper datetime object. The month was then extracted as a separate feature, and the original date column was dropped.
3. **Cleaning Categorical Data:** The department column had inconsistent values ('finishing' and 'finishing '). These were merged into a single 'finishing' category to ensure data consistency.
4. **Categorical Encoding:** The text-based categorical columns (quarter, department, day) were converted into numerical representations using a custom MultiColumnLabelEncoder class, making them suitable for the machine learning models.

**5.3. Model Building and Training**

The dataset was split into features (X) and the target variable (y). It was then further divided into an 80% training set and a 20% testing set. Three different regression models were trained:

1. **Linear Regression:** A baseline model to establish initial performance.
2. **Random Forest Regressor:** An ensemble model known for its robustness and accuracy. We used 200 trees (n\_estimators=200) with a max\_depth of 5.
3. **XGBoost Regressor:** A powerful gradient boosting model, often providing state-of-the-art performance. It was configured with n\_estimators=200, max\_depth=5, and a learning\_rate=0.1.

**5.4. Model Evaluation and Selection**

Each model was evaluated on the unseen test data using three key metrics:

* **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values.
* **Mean Absolute Error (MAE):** Measures the average absolute difference.
* **R2 Score:** Represents the proportion of variance in the target variable that is predictable from the input features.

The **XGBoost Regressor** consistently provided the lowest error (MSE, MAE) and the highest R2 Score, indicating it was the most accurate and reliable model. Therefore, it was selected as the final model for this project.

**5.5. Saving the Model**

The trained XGBoost model was saved to a file named **gwp.pkl** using Python's pickle library. This allows us to load and use the model in our web application without needing to retrain it.

**6. Web Application Development**

The web application was built using Flask.

**6.1. Project Structure**

The application code is organized within the Flask/ directory:

Flask/

│

├── app.py # The main backend script

├── gwp.pkl # The saved machine learning model

└── templates/

├── base.html # The main layout template

├── home.html # The landing page

├── about.html # The project description page

├── predict.html # The user input form

└── result.html # The page to display the prediction

**6.2. Backend (app.py)**

The app.py script handles the application's logic:

* It initializes a Flask app.
* It loads the gwp.pkl model into memory when the application starts.
* It defines routes (@app.route(...)) to serve the different HTML pages.
* The /predict route handles both GET requests (to display the form) and POST requests (to process submitted data).
* When form data is submitted via POST, the script retrieves all input values, converts them into a NumPy array in the correct format, and feeds it to the loaded model to get a prediction.
* Finally, it renders the result.html page, passing the final prediction text to it.

**7. Application Screenshots**

Here are screenshots of the final web application in action.

1. **Home Page**
2. **About Page**
3. **Prediction Form**

**8. How to Run the Project**

To run this project on a local machine, follow these steps:

1. **Prerequisites:** Ensure you have Python 3 and pip installed.
2. **Clone the Project:** Download or clone the project folder to your local machine.
3. **Set up a Virtual Environment (Recommended):**
4. # Navigate to the main project directory
5. cd Employee\_Performance\_Project
6. # Create a virtual environment
7. python -m venv venv
8. # Activate it
9. # On Windows:
10. .\venv\Scripts\activate
11. # On macOS/Linux:
12. source venv/bin/activate
13. **Install Dependencies:** Install all the required Python libraries.
14. pip install flask numpy pandas scikit-learn xgboost
15. **Run the Flask Application:**
16. # Navigate into the Flask directory
17. cd Flask
18. # Run the application script
19. python app.py
20. **Access the Application:** Open your web browser and go to the URL provided in the terminal, which is usually **http://127.0.0.1:5000**.

**9. Conclusion**

This project successfully demonstrates a complete, end-to-end machine learning pipeline. We have taken a raw dataset, cleaned and processed it, trained and evaluated multiple models, and deployed the best one into a functional and user-friendly web application. The final tool provides a practical solution for predicting employee productivity, offering valuable insights that can help organizations make smarter, data-informed decisions.