**Employee Productivity Prediction Using Machine Learning**  
**Author**: Ayush Tripathi  
**Date**: August 10, 2025  
**Version**: 1.0

**1. Abstract**

This project introduces a robust system aimed at predicting the productivity of garment workers by leveraging machine learning techniques. By examining key factors such as team dynamics, goal achievement, and financial incentives, we created a predictive model to estimate worker output. The model is integrated into an intuitive web application, built with the Flask framework, enabling managers to input worker-specific details and receive real-time productivity forecasts. The system is designed to assist in talent management, optimize resource distribution, and improve workforce strategies, ultimately boosting organizational performance.

**2. Project Objectives**

The primary goals of this project include:

* Conducting in-depth exploratory data analysis (EDA) on the garment worker productivity dataset to identify significant patterns and relationships.
* Cleaning and preprocessing the data to ensure it's ready for machine learning applications.
* Developing and evaluating multiple regression models to identify the most accurate predictor for employee productivity.
* Storing the best-performing model for future use.
* Building a user-friendly web application that allows input of worker data and returns productivity predictions.

**3. Dataset Information**

* **Dataset**: garments\_worker\_productivity.csv
* **Description**: This dataset contains various attributes relevant to the productivity of garment factory workers.
* **Key Features (Inputs)**: Quarter, department, day, team, targeted productivity, SMV (Standard Minute Value), overtime, incentives, idle time, idle men, number of style changes, number of workers, month.
* **Target Variable (Prediction)**: Actual productivity.

**4. Technology Stack**

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

**5. Machine Learning Workflow**

The machine learning pipeline was executed in the Training files/Employee\_Prediction.ipynb notebook.

**5.1 Data Loading and Initial Analysis**

The dataset was initially loaded into a Pandas DataFrame. A preliminary analysis was performed using data.head() to inspect the first few rows and data.describe() to generate a statistical summary. We visualized correlations between numerical features through a heatmap.

**5.2 Data Preprocessing and Cleaning**

1. **Handling Missing Data**: We checked for null values using data.isnull().sum(). The wip column contained many missing values and was removed from the dataset.
2. **Date Feature Engineering**: The date column was converted into a datetime object, and the month was extracted as a separate feature. The original date column was then discarded.
3. **Cleaning Categorical Data**: The department column had inconsistencies (e.g., 'finishing' and 'finishing '). These were merged into a single category to standardize the data.
4. **Categorical Encoding**: Text-based categorical features like quarter, department, and day were encoded numerically using a custom MultiColumnLabelEncoder class to make them compatible with machine learning models.

**5.3 Model Development and Training**

The dataset was split into input features (X) and the target variable (y). It was then partitioned into an 80% training set and a 20% test set. Three regression models were developed:

1. **Linear Regression**: A baseline model for comparison.
2. **Random Forest Regressor**: An ensemble method known for its precision and reliability, configured with 200 trees and a maximum depth of 5.
3. **XGBoost Regressor**: A powerful gradient boosting model, tuned with 200 estimators, a max depth of 5, and a learning rate of 0.1.

**5.4 Model Evaluation and Selection**

Each model was assessed on the test set using three evaluation metrics:

* **Mean Squared Error (MSE)**: Measures the squared average difference between predictions and actual values.
* **Mean Absolute Error (MAE)**: Assesses the average absolute difference between predicted and actual values.
* **R² Score**: Represents the proportion of variance in the target variable explained by the model.

The XGBoost model consistently outperformed the others in terms of MSE, MAE, and R², making it the most reliable and accurate model for this task.

**5.5 Model Saving**

The final XGBoost model was saved using Python's pickle library as gwp.pkl, allowing it to be loaded into the web application for real-time predictions without the need for retraining.

**6. Web Application Development**

The web application was built using the Flask framework.

**6.1 Project Structure**

The application’s directory structure is as follows:

Flask/

│

├── app.py # Main backend script

├── gwp.pkl # Saved machine learning model

└── templates/

├── base.html # Main layout template

├── home.html # Landing page

├── about.html # Project description

├── predict.html # User input form

└── result.html # Prediction result page

**6.2 Backend (app.py)**

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

* Initializes the Flask app and loads the gwp.pkl model into memory.
* Defines routes (@app.route(...)) for rendering various HTML pages.
* The /predict route handles both GET and POST requests: displaying the input form and processing the form submission to generate predictions.
* The prediction results are rendered on the result.html page.

**7. Application Screenshots**

* **Home Page**
* **About Page**
* **Prediction Form**

**8. How to Run the Project**

To run this project locally:

1. **Prerequisites**: Ensure Python 3 and pip are installed.
2. **Clone the Project**: Download or clone the project repository to your local machine.
3. **Set up a Virtual Environment** (recommended):
4. # Navigate to the main project directory  
   cd Employee\_Performance\_Project
5. # Create a virtual environment  
   python -m venv venv
6. # Activate it  
     
   # On Windows:  
   .\venv\Scripts\activate  
     
   # On macOS/Linux:  
   source venv/bin/activate
7. **Install Dependencies**:  
   pip install flask numpy pandas scikit-learn xgboost
8. **Run the Flask Application**:  
   # Navigate to the Flask directory  
   cd Flask
9. # Run the application  
   python app.py
10. **Access the Application**: Open a web browser and go to http://127.0.0.1:5000.

**9. Conclusion**

This project successfully demonstrates a full machine learning pipeline. We processed raw data, built and evaluated predictive models, and deployed the most effective model into a web application. The resulting tool provides valuable insights into employee productivity, supporting smarter, data-driven decisions that can optimize organizational performance.