Author: Lynn Achieng Oloo
Github link: https://github.com/Lynn-rose/Garment-Worker-Productivity-Prediction-/tree/main

Garment Worker Productivity Prediction

The Garment Worker Productivity Prediction System is a web-based tool designed to predict the productivity of garment workers based on various input parameters. It employs a machine learning model hosted on a FastAPI backend to process user inputs and return predictions.

The project was done according to the CRISP-DM process as follows:

1. Business Understanding

The garment manufacturing industry is highly competitive and operates on tight margins. Worker productivity is a critical determinant of operational efficiency, cost management, and profitability. Variability in productivity due to factors such as skill levels, work conditions, and task complexities can disrupt production schedules and inflate costs. Predicting worker productivity helps businesses streamline operations, meet deadlines, and achieve higher output quality.

Problem Statement

Garment manufacturing is a labor-intensive industry where worker productivity directly impacts overall efficiency and profitability. Variations in worker performance due to environmental conditions, skill levels, and workload distribution create challenges in maintaining consistent production targets. Predicting productivity levels helps managers make informed decisions to optimize processes and address potential bottlenecks proactively. The following is the main objective of the garment worker productivity prediction:

Objective

The Garment Worker Productivity Prediction project aims to accurately forecast the productivity of garment factory workers based on various workplace, environmental, and operational factors. This prediction is crucial for optimizing resource allocation, improving workflow efficiency, and maintaining high-quality production standards while reducing operational costs.

Proposed Solution

The proposed solution is a predictive analytics system that leverages machine learning models to forecast garment worker productivity. This system will analyze a combination of operational,

environmental, and worker-related factors to provide actionable insights, helping factory managers optimize workflows and resource allocation.

2. Data Understanding

The dataset used for this project is the **Garment Worker Productivity Dataset**. It includes 1,197 records and 14 features, covering different attributes related to the garment production process, such as targeted productivity, overtime, and the actual productivity achieved by each team. Each row represents a record for a team on a particular day, and the target variable (actual productivity) indicates the team's performance.

Key Features:

- **date**: Date of the record.
- **quarter**: The quarter of the year (e.g., Q1, Q2).
- **department**: Department of workers (e.g., sewing, finishing).
- **team**: The number representing the team.
- **targeted_productivity**: The target productivity (between 0 and 1).
- **smv**: Standard Minute Value (time required to complete the task).
- **wip**: Work In Progress (missing values present).
- **over_time**: Overtime in minutes.
- **incentive**: Bonus paid to workers.
- **idle_time**: Time during which no work was done.
- **idle_men**: Number of idle workers.
- **no_of_style_change**: Number of style changes in production.
- **no_of_workers**: Number of workers in the team.
- **actual_productivity**: Target variable representing the productivity achieved (between 0 and 1).

3. Data Preparation

Preprocessing the dataset is essential to ensure that the data is accurate, complete, and in a format suitable for machine learning models. This section describes each preprocessing step in detail, including handling missing values, detecting and handling outliers, encoding categorical variables, and standardizing numerical data. The following steps were taken in the process:

1. Handling Missing Values

Missing values were identified in columns like `wip`

We recall that there are two general strategies for dealing with missing values:

- Fill in missing values (either using another value from the column, e.g. the mean or mode, or using some other value like "Unknown")
- Drop rows with missing values

The whole dataset has 1197 rows and the highest column having missing values is `wip` which has 506 missing values which is about 42.27% of our data. we see that the null values have a very high impact on our data set therefore we decide to fill in missing values with a placeholder i.e the median because the median is less sensitive to outliers and helps maintain consistency.

2. Detect and handle Outliers

Outliers were identified in columns like 'idle_time', 'incentive', and 'actual_productivity' using the interquartile range (IQR) method. Values falling outside the 1.5*IQR range were considered outliers and were either removed or capped.

3. Encoding Categorical Variables

Categorical variables like `quarter`, `department`, and `day` were identified.

We use one-hot encoding for nominal categories (e.g., `department`).

4. Feature Scaling

Continuous numerical variables such as `smv`, `over_time`, `incentive`, and `idle_time` were identified. We apply standardization (mean = 0, standard deviation = 1) using `StandardScaler` for consistency in model training.

5. Feature Engineering

Derived new features by extracting meaningful information for columns such as "date," including the identification of the day of the week and the month corresponding to each date within the dataset.

Exploratory Data Analysis

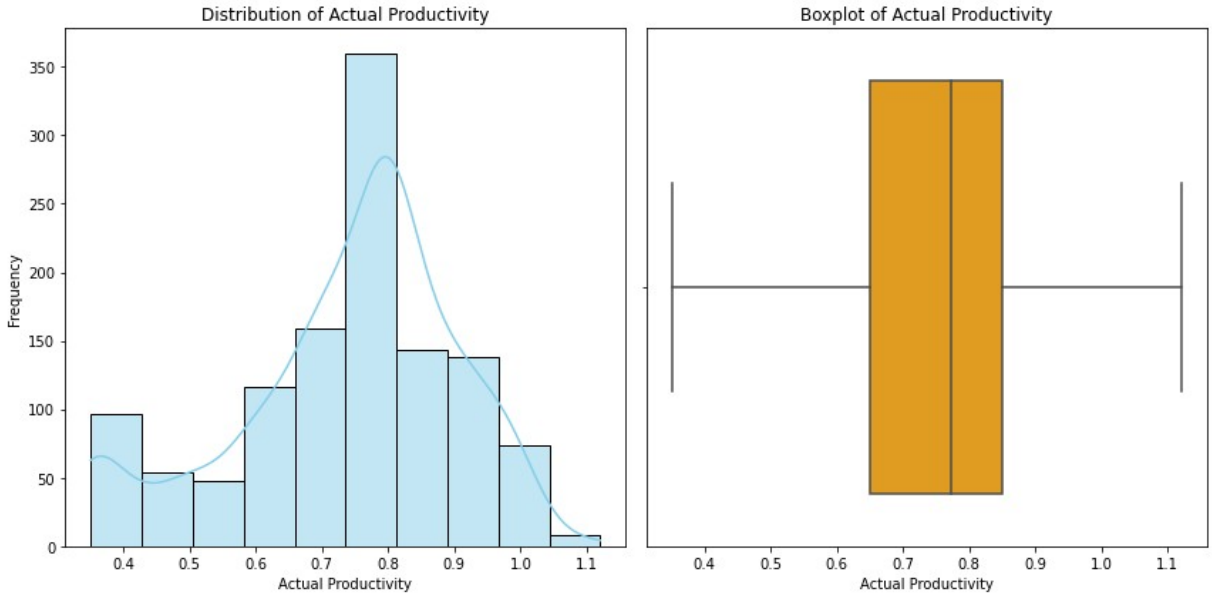
The goal of EDA is to explore the dataset to uncover patterns, relationships, anomalies, and insights that can guide the predictive modeling process. The following steps will be taken:

1. Statistical Summary of Features

Understand the distribution and spread of data. Analyzed key metrics (mean, median, min, max) for numerical features like `smv`, `incentive`, `idle_time`, and `actual_productivity`. Observed potential skewness in distributions.

2. **Analysis of Target Variable**

Understand the distribution of `actual_productivity`. Plotted the distribution of `actual_productivity` using a histogram with KDE. Checked for any skewness or unusual patterns in the target variable**.**

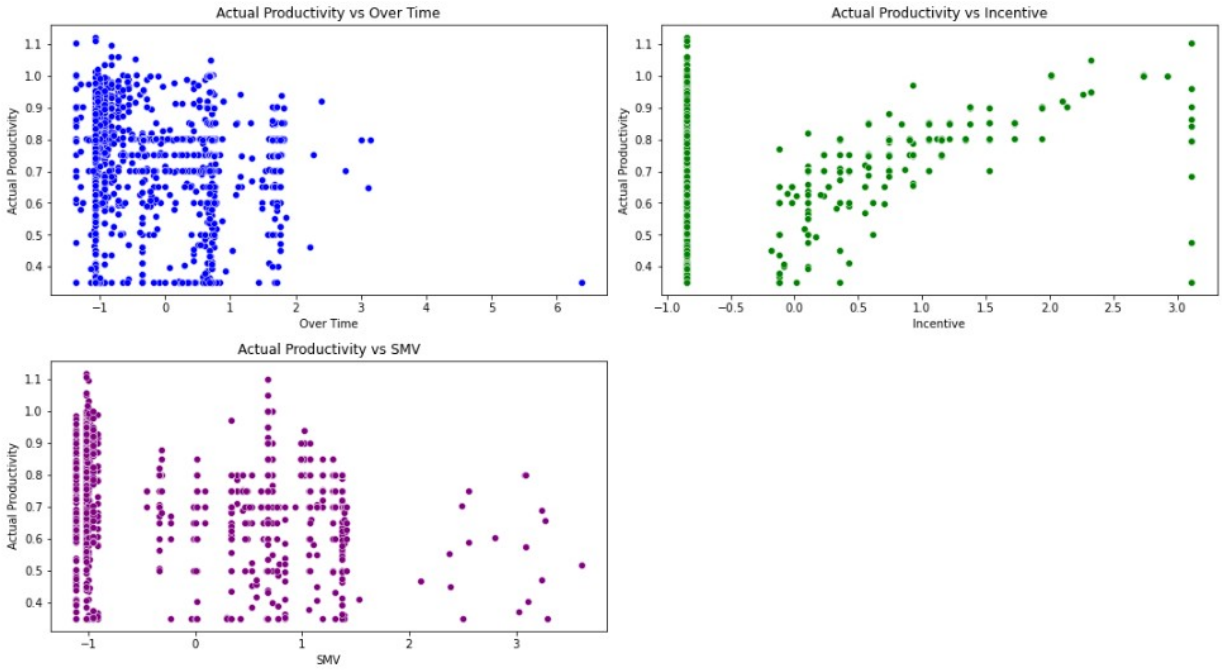


The histogram shows how `actual_productivity` values are distributed (e.g., peaks indicating common values). From this we see that the `actual_productivity` is normally distributed suggesting balanced productivity values.

The boxplot highlights outliers (if any) as points outside the whiskers. From the boxplot we see there are no outliers in the column

3. Feature-Target Relationships

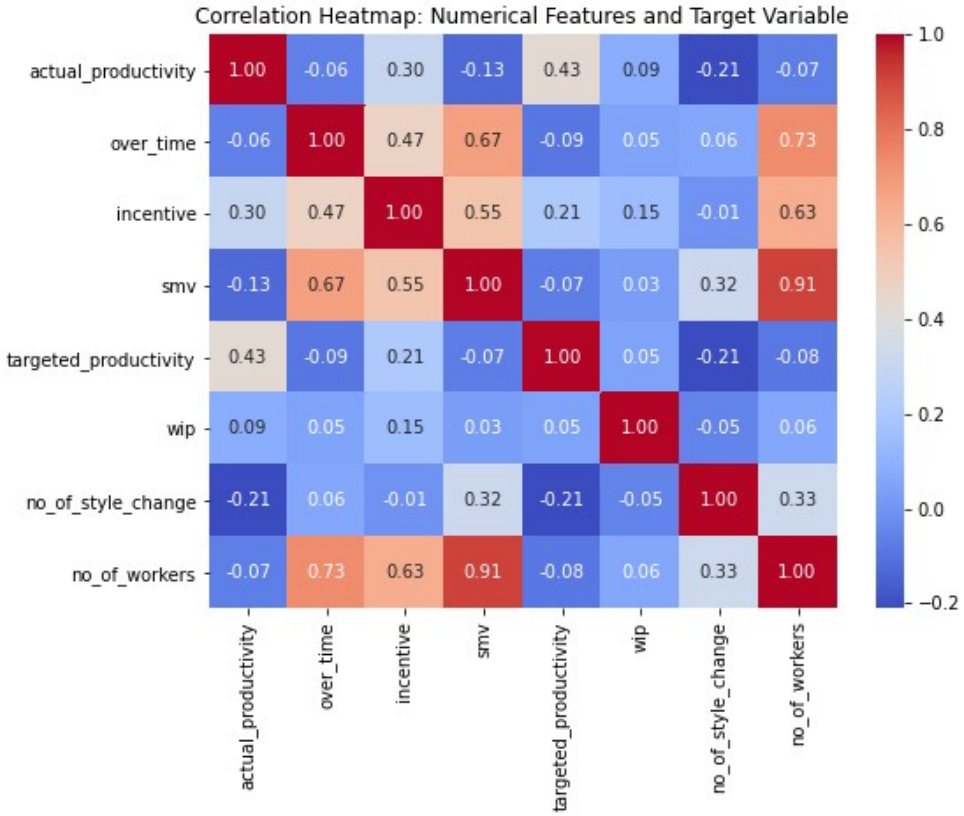
Investigate how features influence `actual_productivity`. Created scatter plots for numerical features (e.g., `smv` vs. `actual_productivity`).



Clustering at -1: The clustering in the over_time and smv plots suggests that these features are heavily influenced by standardization, with values concentrated near the mean.
Correlation: The strongest apparent relationship is between incentive and actual_productivity, as shown by the upward trend.

4. Correlation Analysis

Identify relationships between numerical variables. Computed the correlation matrix using.Visualized correlations with a heatmap .Identified strongly correlated variables that could impact the model, such as the relationship between `smv` and `actual_productivity`.



- Strong Correlations:

Incentive (0.30) and Targeted Productivity (0.43) show meaningful positive relationships with Actual Productivity. These features should be prioritized in model development.

- Multicollinearity:

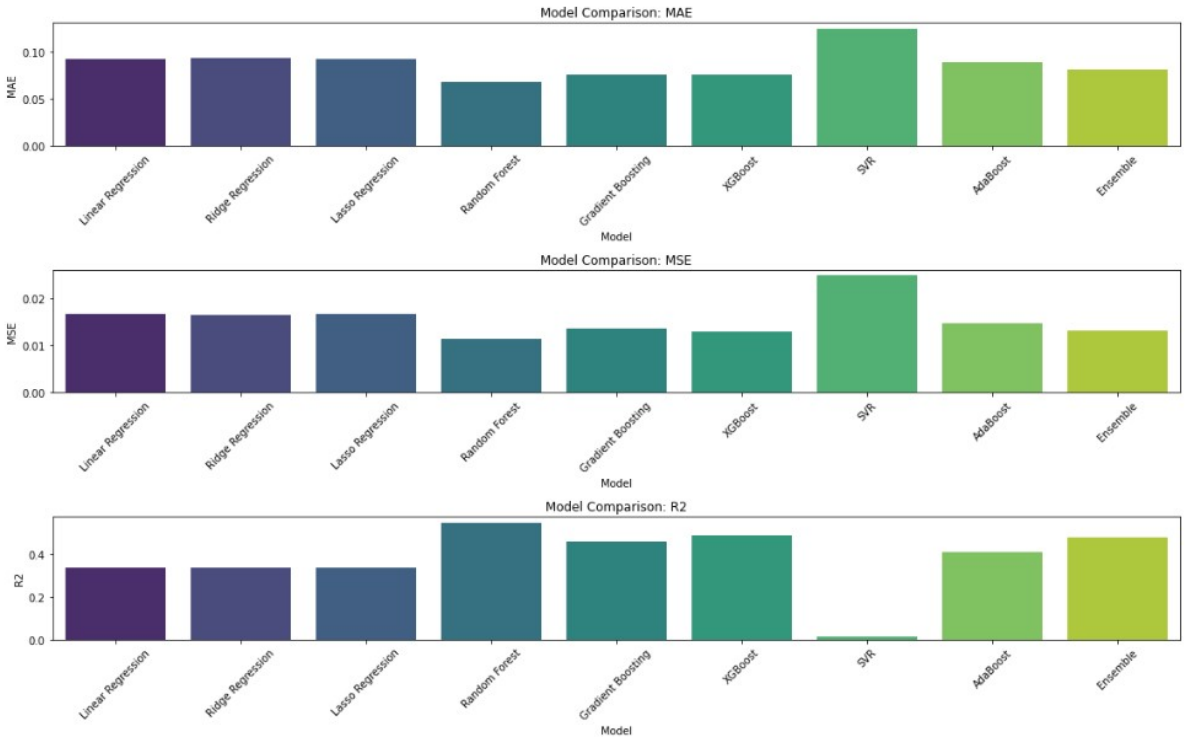
Features like SMV and No of Workers (0.91) and Over Time and SMV (0.67) show strong correlations, which may lead to redundancy. Regularization techniques (e.g., Ridge or Lasso) or feature selection methods should address this.

4. Modeling

A variety of machine learning models were implemented to predict productivity, each evaluated based on several performance metrics. The models examined included Linear Regression, Random Forest Regressor, Gradient Boosting Regressor, Extreme Gradient Boosting, Support Vector Regressor (SVR), and AdaBoost Regressor. Below is the performance of the modeling:

Model	**MAE**	**MSE**	**RMSE**	**R^2 Score**
Linear Regression	0.0928	0.0166	0.1288	0.3376
Ridge Regression	0.0931	0.0165	0.1286	0.3397
Lasso Regression	0.0928	0.0166	0.1288	0.3383
Random Forest Regressor	0.0674	0.0113	0.1063	0.5490
Gradient Boosting Regressor	0.0760	0.0135	0.1161	0.4620
XGBoost Regressor	0.0752	0.0128	0.1132	0.4886

Support Vector Regressor	0.1247	0.0247	0.1573	0.0122
AdaBoost Regressor	0.0896	0.0150	0.1226	0.4001



1. Mean Absolute Error (MAE)

Observation:

Random Forest, Gradient Boosting, and XGBoost have the lowest MAE, indicating these models produce predictions closest to the actual values on average.

SVR has the highest MAE, meaning it consistently deviates more from the actual values compared to other models.

Ensemble Model also performs well, suggesting combining predictions improves robustness.

2. Mean Squared Error (MSE)

Observation:

Random Forest shows the lowest MSE, followed by Gradient Boosting and XGBoost, indicating they handle large errors better.

SVR has the highest MSE, which highlights its inability to manage larger errors effectively.

Ensemble Model demonstrates relatively low MSE, validating the power of combining models.

3. R^2 Score

Observation:

Random Forest achieves the highest R^2 score, suggesting it explains the variance in the target variable better than others.

Gradient Boosting and XGBoost also perform well in explaining variance, while SVR struggles with the lowest R^2 score.

The Ensemble Model achieves a competitive R^2, showing that leveraging multiple models enhances performance.

We go ahead and do hyperparameter tuning for the best performing models.

Hyperparameter Tuning

We use GridSearchCV to tune hyperparameters of the top models (e.g., Random Forest, Gradient Boosting, and XGBoost).

While XGBoost is a robust model for many scenarios, its performance in this case is weaker than Random Forest and Gradient Boosting. This could be due to insufficient feature interactions or hyperparameter tuning.

Random Forest performs the best overall, with the lowest errors and the highest R^2 score(0.5498), making it the most reliable model for the given task.

Gradient Boosting follows as a decent alternative but is slightly less effective.

XGBoost, while powerful in general, performs the weakest among the three models in this specific scenario.

Based on the findings, we use the Random forest model to build the prediction system.

5. Predictive System Development

Users provide data for features like department, quarter, and specific date through an interactive interface . The system preprocesses the input data to ensure compatibility with the trained model.

Application: The predicted productivity value is displayed to the user in an interpretable format, helping in decision-making or performance monitoring in scenario based conditions.