Predicting Employee Productivity in the Garment Factory Using Machine Learning Techniques.

Francesco Manco
Università degli studi di Bari Aldo Moro
f.manco10@studenti.uniba.it

Abstract

In this work, a classification system was developed to predict employee productivity in the garment industry using machine learning techniques. The dataset used includes variables related to daily production and employees' operational characteristics. After careful pre-processing and feature selection, a comparison is made between various machine learning models to classify employee productivity within the industry into the following two classes: 'Low Productivity' and 'High Productivity'. By optimizing the hyper-parameters and comparing different models, it was possible to identify the model with the best performance. The results highlight the importance of machine learning tool applications to optimize the workforce.

1 Introduzione

The ability to predict employee productivity is a key factor for operational efficiency in manufacturing industries, especially in the garment sector. Effective management of human resources and optimization of production processes can significantly improve business competitiveness. In this context, machine learning techniques can provide useful tools for identifying recurring patterns and predicting future performance. In this paper, the Productivity Prediction of Garment Employees dataset is used to classify employees into two productivity categories: *Low Productivity* and *High Productivity*. To achieve this, advanced pre-processing techniques, feature selection using *Mutual Information*, and classification models were used to identify the model with the best performance. Hyper-parameter optimization was implemented to improve the predictive capabilities of the models used.

2 Dataset

The dataset used in this study comes from the UCI Machine Learning Repository and contains 1197 samples representing working days on different production lines in the garment industry. Each sample contains variables related to daily production and operational parameters, allowing the analysis of working conditions and actual productivity. The main variables included in the dataset are the following:

- **date**: Date in the format MM-DD-YYY.
- **day**: Day of the week.
- **quarter**: A portion of the month. A month is divided into four quarters.
- **department**: Department associated with the instance.
- **team_no**: Team number associated with the instance.
- **no_of_workers**: Number of workers for each team.
- **no_of_style_change**: Number of changes in the style of a particular product.
- **targeted_productivity**: Target productivity set by the authority for each team each day.
- **smv**: *Standard Minute Value*, time allocated for a task.

- **wip**: *Work in Progress*, number of incomplete items in production.
- **over_time**: Amount of overtime performed by each team, in minutes.
- **incentive**: Amount of financial incentive (in BDT) that stimulates a given course of action.
- **idle_time**: Time during which production was interrupted for various reasons.
- **idle_men**: Number of workers idle due to the interruption of production.
- **actual_productivity**:Actual percentage of productivity achieved by workers, ranging from 0 to 1.

The target variable was constructed based on the variable **actual_productivity**, which represents the actual productivity achieved compared to the target set.

2.1 Transition from a Regression Task to a Binary Classification Task

The dataset is initially designed to be able to predict a 'real' value of employee productivity, so it is set up for a regression task. To be able to translate the dataset to the classification task, it was decided to impose a threshold on the value of "actual_productivity", if the threshold is not met, the class "Low Productivity" (0) is assigned, otherwise the class "High Productivity" (1) is assigned. To set the threshold, the distribution of the variable 'target_productivity' was analyzed and the average of this feature was chosen as the evaluation parameter. This decision was made because a company's expectations are, more often than not, that individual employees meet the average productivity of the entire workforce. The threshold was therefore set at 0.72, and the class distribution is defined as follows 1:

- Low Productivity (0): 462.
- High Productivity (1): 735.

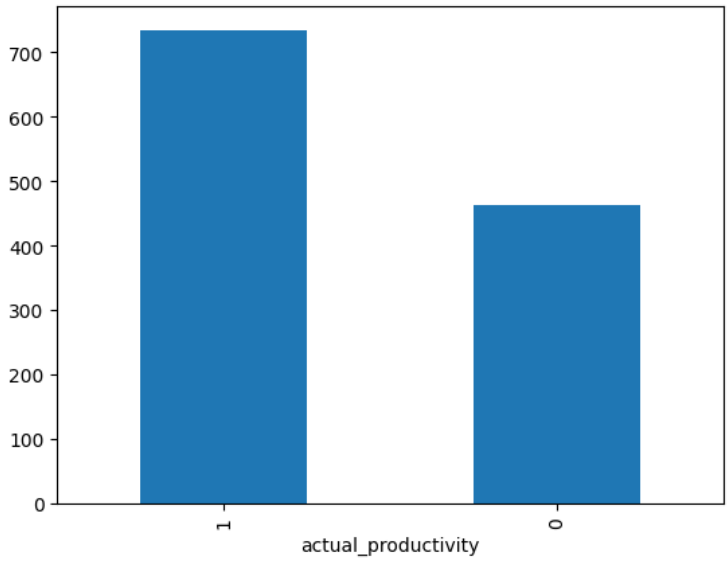

Figure 1: Target class distribution

3 Modelling

This section illustrates the method implemented to classify employee productivity into two distinct categories: Low Productivity and High Productivity. The methodology includes steps of data pre-processing, feature selection, model specification, hyper-parameter optimization and comparative model evaluation.

3.0.1 Pre-Processing

The data pre-processing phase was crucial in preparing the dataset for model training. The following operations were performed systematically:

3.1 Data Cleaning

The dataset underwent comprehensive cleaning procedures, including::

- Unification of the categorical values 'finishing_' and 'finishing' in the department variable to correct distributional anomalies.
- Standardisation of the quarter variable by merging 'Quarter 4' and 'Quarter 5' into a single 'Quarter 4' category, following the description of the feature
- Implementation of imputation by averaging for missing values in the Work-in-Progress (WIP) variable
- Normalisation of data types for variables with inconsistencies
- Removal of redundant time column (dates)
- Application of One-Hot-Encoding to categorical variables quarter, department and day

After this pre-processing phase, the dataset is ready to be used in the experiment.

3.2 Experimentation

The dataset was partitioned according to an 80-20 ratio for the training and evaluation phases. The methodology implemented a nested cross-validation structure, comprising a 5-partition internal cross-validation for hyperparameter optimization and a 10-partition stratified external cross-validation with 10 repetitions, ensuring robust and bias-free performance estimation. The experimental framework evaluated four classification algorithms: K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree and Random Forest.

3.2.1 Tuning of Hyperparameters

Hyperparameters are configuration parameters that are not learnt during the training phase but must be set a priori, and their correct configuration is essential for optimal performance on the specific dataset. In order to identify the best configuration of the hyper-parameters for each model implemented (KNN, Logistic Regression, Decision Tree and Random Forest), a systematic search strategy based on nested cross-validation was adopted. This methodology involves a 10-fold repeated external (outer CV) cross-validation to obtain a robust performance estimate, combined with a 5-fold internal (inner CV) cross-validation for the optimization of the hyper-parameters. The nested cross-validation was chosen to avoid selection bias and obtain an unbiased estimate of model performance.

A specific search space of hyperparameters was defined for each algorithm. For the KNN, the number of neighbours, the distance metric and the neighbour weights were considered. For Logistic Regression, the regularisation parameter, penalty type and solver were evaluated. In the case of the Decision Tree, the split criterion, the split strategy, the maximum depth of the tree and the minimum number of samples required to make a split were optimised. Finally, for the Random Forest, the number of trees, the maximum tree depth, the cost-complexity pruning parameter to control the pruning of the trees were considered. The results of the nested cross validation are summarised in the following table1

Table 1: Model performance with optimised hyperparameters

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
KNN	0.779	0.782	0.779	0.772
Log. Regression	0.768	0.769	0.768	0.766
Decision Tree	0.826	0.832	0.826	0.826
Random Forest	0.853	0.857	0.853	0.853

As we can see from the hyper parameter search phase, the best performing models are the **Decision Tree** and **Random Forest**. After using nested cross validation to search for hyper parameters, a final training was carried out on the entire training set and then the best models were tested on the test set, obtaining the following results.

Table 2: Model performance with optimised hyperparameters on the test set

Model	Accuracy	Precision	Recall	F1-score
KNN	0.77	0.77	0.77	0.76
Log. Regression	0.79	0.79	0.79	0.79
Decision Tree	0.80	0.79	0.80	0.80
Random Forest	0.85	0.85	0.85	0.85

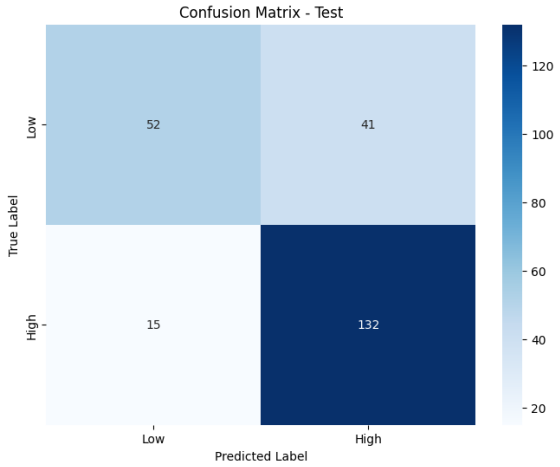

Figure 2: KNN

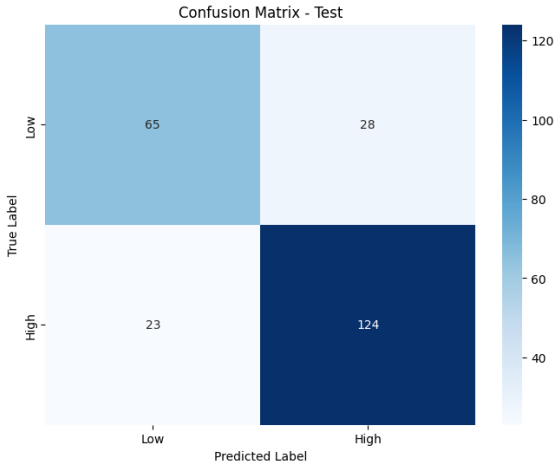

Figure 3: Logistic regression

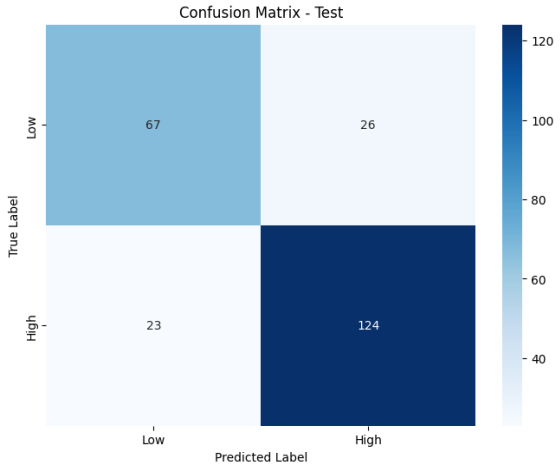

Figure 4: Decision tree

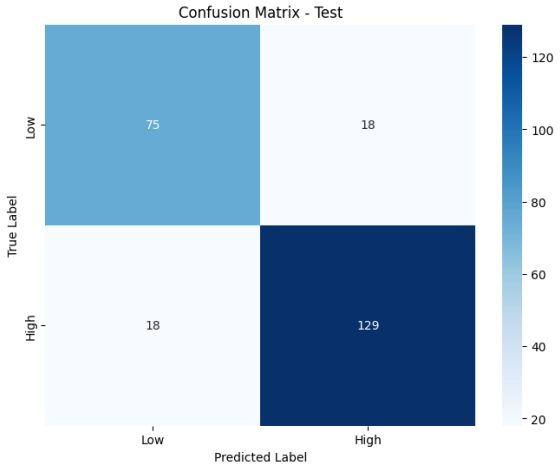

Figure 5: Random Forest

Figure 6: Confusion matrix on test set

Experimental results showed superior performance for the Random Forest and Decision Tree algorithms, with accuracies of 85% and 80% respectively. Logistic Regression and KNN showed lower performance, reaching accuracies of 76% and 77% respectively. The resubstitution error analysis revealed a tendency for overfitting in the KNN algorithm, attributable to the high dimensionality of the features. To mitigate this issue, a feature selection based on Mutual Information was implemented, identifying 'incentive', 'smv' and 'targeted_productivity' as the most informative variables for the classification task, showed in the following graph 7.

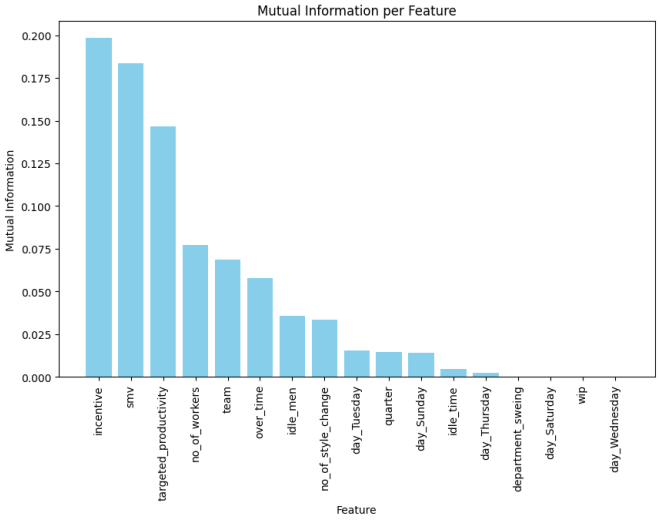

Figure 7: Mutual informtion plot

After feature selection, the KNN model no longer overfitted and increased its performance from an accuracy of 0.77 to 0.78.

4 Comparison of models

In order to compare the models, the ROC curves obtained were analysed and finally a statistical t-paired test was performed. From the analysis of the ROC curves shown in the graph below, it was deduced that, as previously mentioned, the best model is the Random Forest with an AUC of 0.92.

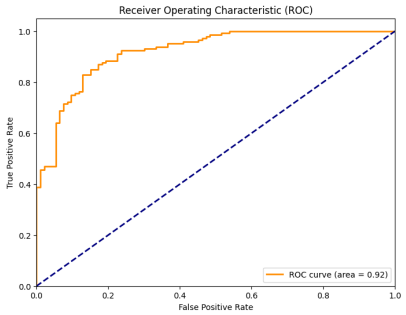

Figure 8: ROC Curve Random forest

Finally, the t-paired student test was carried out in a one-vs-one mode between all models and it turned out that in all comparisons there is a statistically significant difference between the various metrics as all p-values are below the 0.05 value. From this we can deduce that it is correct to choose the Random Forest model as the best model for this task.

5 Conclusione

This study successfully implemented a classification system for predicting employee productivity in the industrial context of the garment industry. The rigorous methodological approach and the systematic optimization of the models led to identifying the Random Forest as the optimal algorithm, with significantly higher performance (accuracy: 85%, AUC: 0.92). These results provide a robust framework for implementing productivity prediction systems in similar industrial settings.