Exploratory Data Analysis and Evaluation of Machine Learning Algorithms on the Steel Plates Faults Dataset

Nikita Chernysh
Faculty of Electrical Engineering and Informatics,
Technical University of Košice
nikita.chernysh@tuke.sk

Abstract

This report presents a complete pipeline for data exploration, preprocessing, model selection, training, and evaluation using the Steel Plates Faults dataset from the UCI Machine Learning Repository. Five classification algorithms were trained and compared: Support Vector Machine (SVM), Random Forest, Gradient Boosting, K-Nearest Neighbors (KNN), and Logistic Regression. The models were evaluated based on accuracy, precision, recall, and F1-score using both hold-out testing and 5-fold cross-validation. The Gradient Boosting model demonstrated the best performance with a cross-validation accuracy of 0.7622 and balanced metrics across all classes.

Index Terms

Exploratory Data Analysis, Classification, Machine Learning, Gradient Boosting, Steel Plates Faults, Cross-Validation.

I. Introduction

FAULT detection in steel plates is a critical quality control task in manufacturing, where accurate classification of defects ensures structural reliability and minimizes production costs. The purpose of this study is to explore the dataset, preprocess it appropriately, and benchmark several supervised learning algorithms to identify the most suitable model for this classification problem.

The *Steel Plates Faults* dataset includes 27 numerical features describing geometric and luminosity-based properties of steel plates, along with 7 categorical target classes representing distinct types of surface faults.

November 7, 2025

II. Dataset Overview

The dataset consists of 1,941 samples and 27 features. The target variable represents seven types of steel faults: *Pastry*, *Z_Scratch*, *K_Scratch*, *Stains*, *Dirtiness*, *Bumps*, and *Other_Faults*. The class distribution is highly imbalanced, with the majority class (*Other_Faults*) accounting for approximately 34.7% of all observations.

TABLE I
Target Class Distribution

Class	Count	Proportion (%)
Other_Faults	673	34.67
Bumps	402	20.71
K_Scratch	391	20.14
Z_Scratch	190	9.79
Pastry	158	8.14
Stains	72	3.71
Dirtiness	55	2.83

III. Exploratory Data Analysis (EDA)

The dataset contained no missing values. Correlation analysis revealed several highly correlated features (e.g., *X_Minimum* and *X_Maximum*, correlation = 0.99; *Y_Minimum* and *Y_Maximum*, correlation = 1.00). One-hot encoded variables *TypeOfSteel_A300* and *TypeOfSteel_A400* were found to be perfectly negatively correlated (-1.00).

High correlation among several geometrical and luminosity-related features indicates redundancy, motivating feature selection to reduce multicollinearity and model complexity.

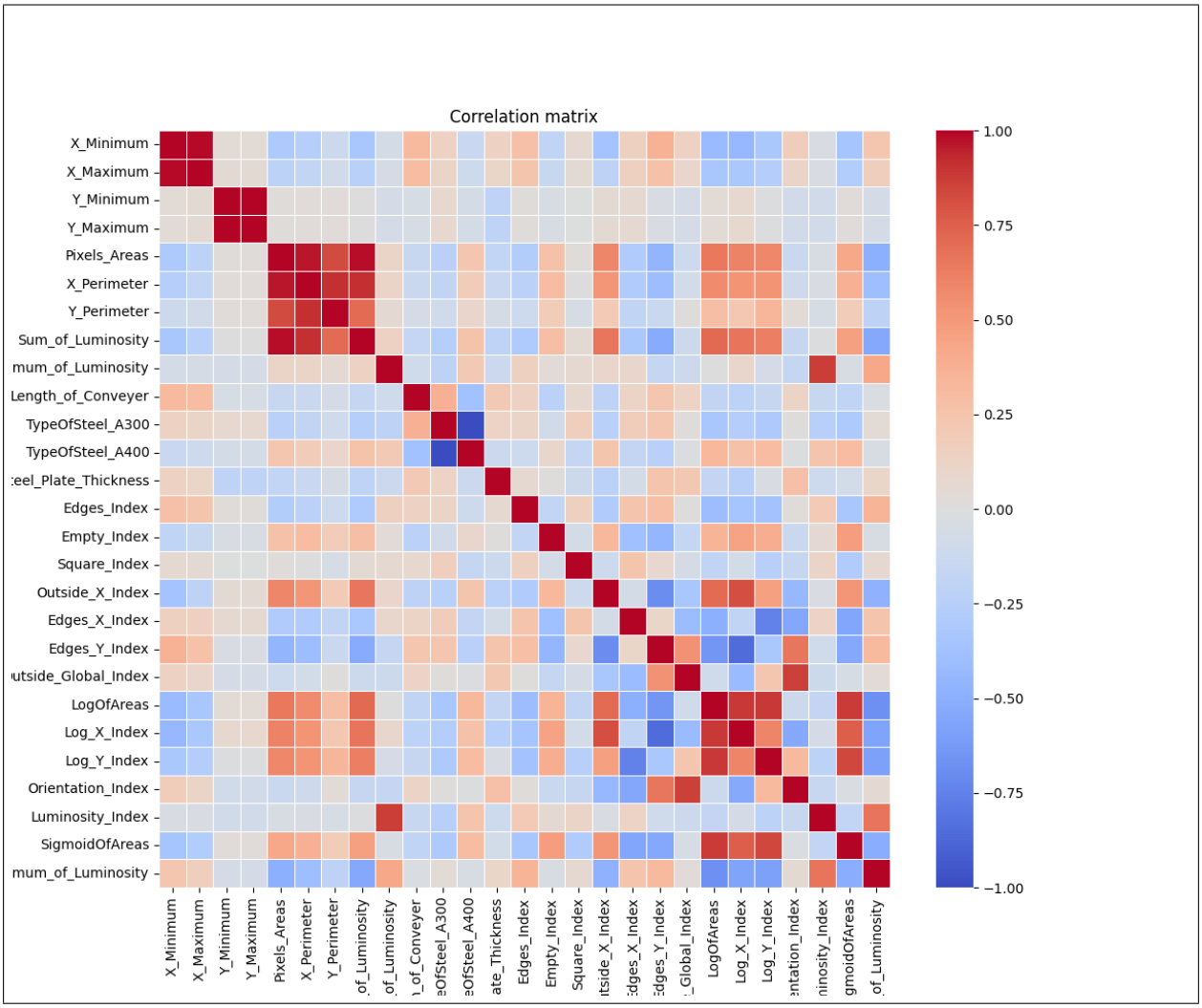

Fig. 1. Correlation Analysis Heatmap. Highly correlated features were dropped before modeling.

IV. Data Preprocessing

Preprocessing included the following steps:

1) Removal of highly correlated features ($|r| > 0.95$).
2) Standardization of continuous features using z-score scaling.
3) Stratified split into training and testing subsets (80/20 split, random seed = 5296).
4) Conversion of multi-output labels to categorical format using `idxmax`.
5) Computation of class weights to mitigate imbalance effects.

V. Model Selection and Rationale

Five algorithms were selected based on their suitability for multi-class imbalanced datasets:

- **Support Vector Machine (SVM)** – robust against non-linear boundaries, effective for high-dimensional data.
- **Random Forest (RF)** – ensemble model providing good generalization and interpretability.
- **Gradient Boosting (GB)** – sequential ensemble learner minimizing bias and variance.
- **K-Nearest Neighbors (KNN)** – non-parametric baseline model sensitive to data distribution.
- **Logistic Regression (LR)** – interpretable linear baseline model with regularization.

VI. Experimental Setup

Each model was implemented in `scikit-learn` with the following parameters:

All models were evaluated on both test data and via 5-fold cross-validation. Metrics used: accuracy, precision, recall, and F1-score.

TABLE II
MODEL CONFIGURATION SUMMARY

Algorithm	Key Parameters
SVM	RBF kernel, C=1.0, class_weight=balanced
Random Forest	200 estimators, max_depth=10, class_weight=balanced
Gradient Boosting	200 estimators, learning_rate=0.1, max_depth=10
KNN	n_neighbors=5
Logistic Regression	max_iter=1000, class_weight=balanced

VII. RESULTS AND DISCUSSION

A. Baseline Test Performance

The Gradient Boosting model achieved the best overall test accuracy (0.74), followed by Random Forest (0.71). Logistic Regression and KNN performed moderately, while SVM exhibited high variance between classes.

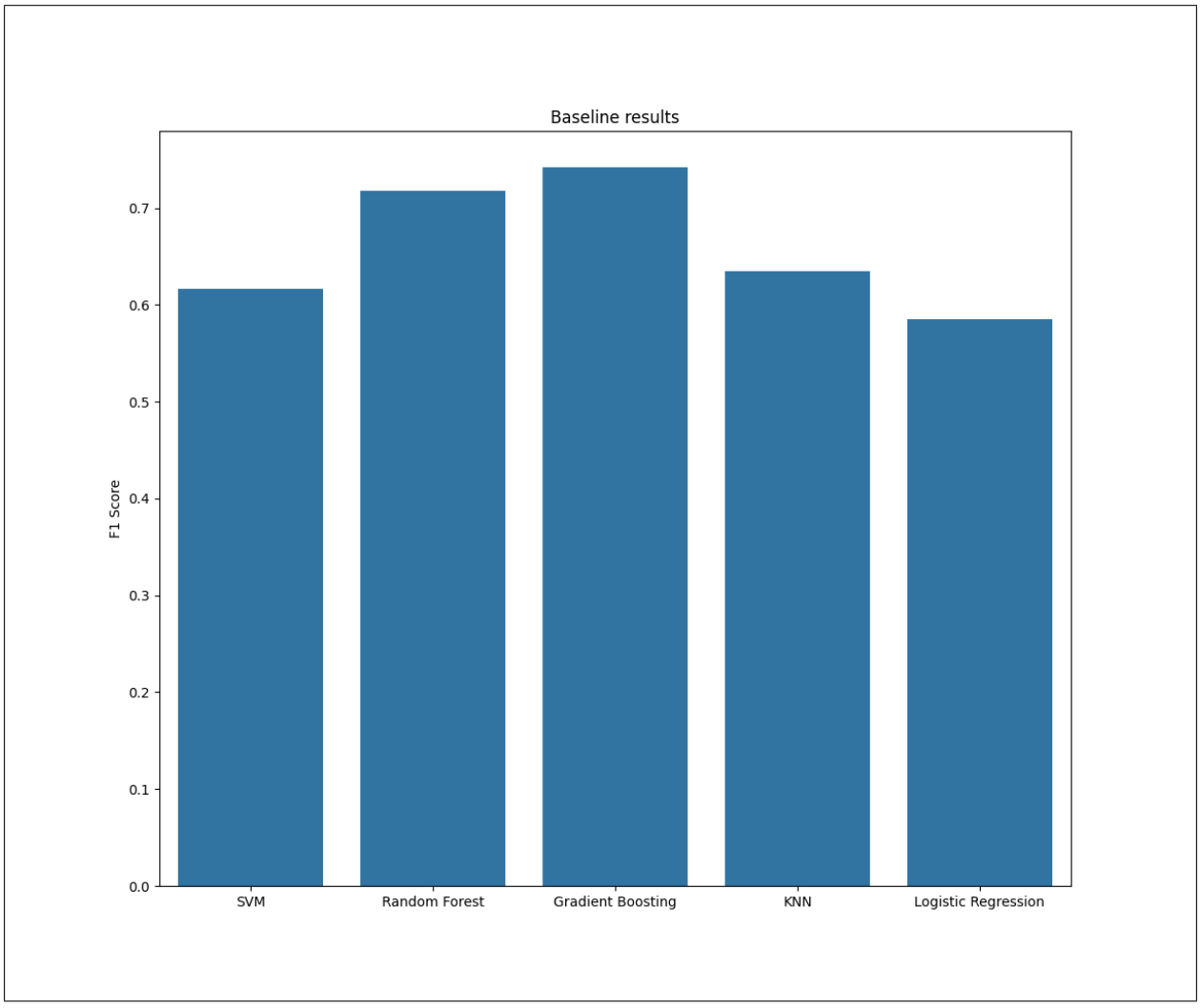

Fig. 2. Baseline F1 score comparison among the five models.

B. Cross-Validation Performance

C. Feature Importance and Error Analysis

Gradient Boosting and Random Forest both identified geometric parameters such as *Pixels_Areas*, *X_Perimeter*, and *Sum_of_Luminosity* as key discriminative features. Misclassifications were primarily observed among visually similar defects (*Bumps* vs *Pastry*, *Z_Scratch* vs *K_Scratch*).

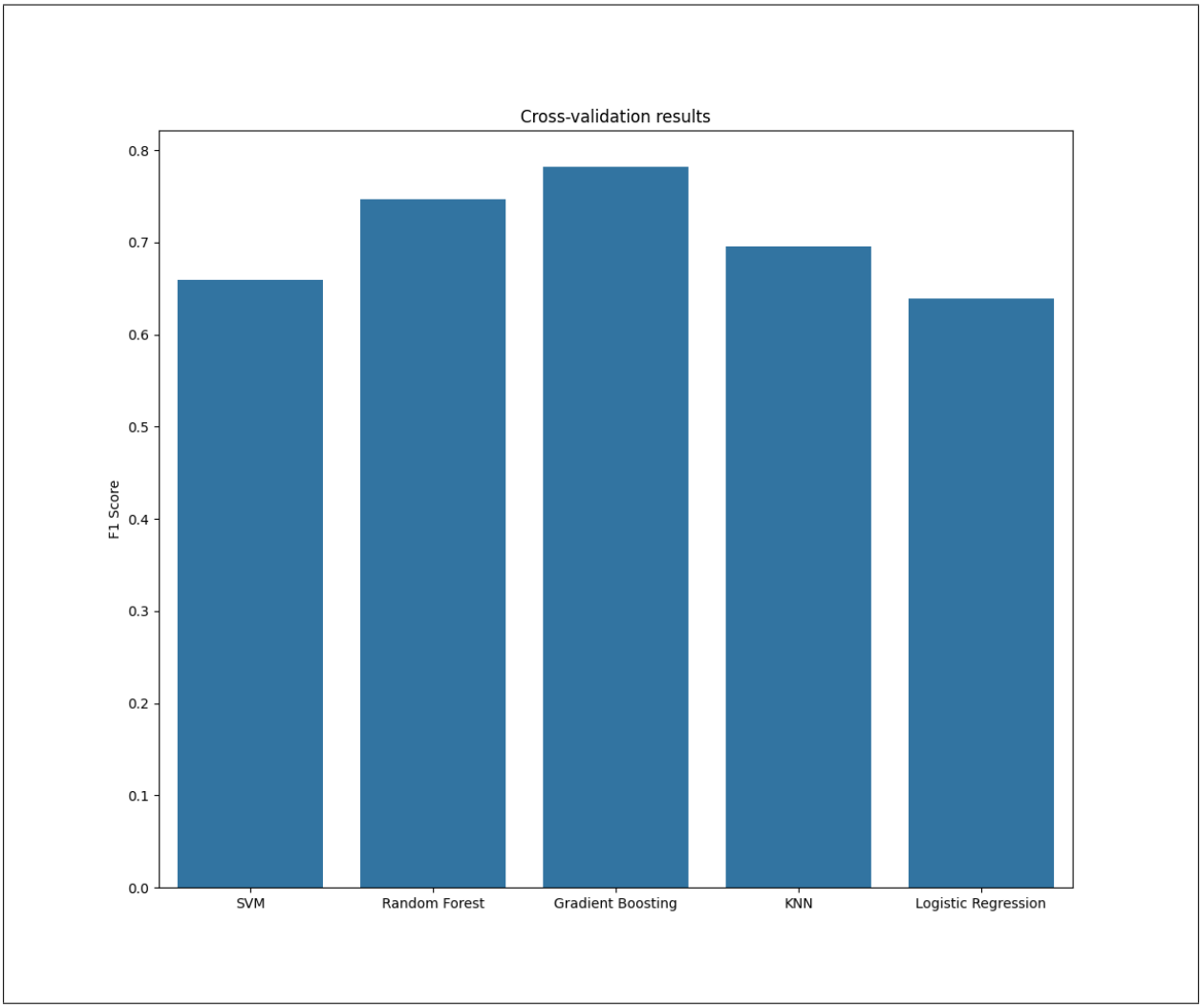

Fig. 3. Cross-Validation F1 score comparison among the five models.

TABLE III
CROSS-VALIDATION RESULTS (5-FOLD AVERAGE)

Model	Accuracy	Precision	Recall	F1-Score
SVM	0.6443	0.7090	0.6443	0.6382
Random Forest	0.7307	0.7397	0.7307	0.7322
Gradient Boosting	**0.7622**	**0.7651**	**0.7622**	**0.7606**
KNN	0.6708	0.6780	0.6708	0.6707
Logistic Regression	0.6095	0.6686	0.6095	0.5982

VIII. CONCLUSION

This study systematically evaluated five supervised learning algorithms for multi-class fault classification in steel plates. Gradient Boosting achieved the best overall results with strong generalization and balanced performance across classes. Random Forest was the second-best, offering slightly lower accuracy but greater interpretability. Future work may explore advanced boosting variants (e.g., XGBoost, LightGBM) and SMOTE-based oversampling for handling data imbalance.

ACKNOWLEDGMENT

The author thanks the Faculty of Electrical Engineering and Informatics at the Technical University of Košice for providing computational resources and guidance for this assignment.

APPENDIX A
SOURCE CODE

Source code with all Jupiter Notebooks, EDA and train results could be found in GitHub Repository. https://github.com/kitnew/steel-faults-ml

REFERENCES

[1] UCI Machine Learning Repository, "Steel Plates Faults Data Set," 2015. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/steel+plates+faults
[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *JMLR*, vol. 12, pp. 2825–2830, 2011.