

Benchmarking Five Machine Learning Models for Accurate Steel Plate Defect Detection

Ellya Sestri¹, Adhitio Satyo Bayangkari Karno², Widi Hastomo^{*3}

¹Department of Information Technology, Ahmad Dahlan Institute of Technology and Business, Jakarta, Indonesia

²Department of Information System, Faculty of Engineering, Gunadarma University, Depok, Indonesia

³Department of Information Technology, Faculty of Business Management and Information Technology, Universiti Muhammadiyah Malaysia, Malaysia

e-mail: ¹Ellyasestri24@gmail.com, ²Adh1t10.2@gmail.com,
^{*3}p5250162@student.umam.edu.my

Abstract

Early detection of defects in steel plates is essential to ensure structural integrity and product quality in the metal manufacturing industry. This study compares the performance of five machine learning algorithms Support Vector Classifier (SVC), Nu-Support Vector Classifier (NuSVC), Decision Tree (DT), Random Forest (RF), and CatBoost (CB) to classify seven categories of steel plate defects using 26 technical features from a publicly available dataset on Kaggle. The preprocessing pipeline included outlier detection (IQR method), class imbalance correction using SMOTE, and feature normalization via StandardScaler. The models were evaluated using classification metrics such as Accuracy, Precision, Recall, F1-Score, ROC-AUC, and Log Loss. Results revealed that the CatBoost algorithm achieved the most balanced and consistent performance, with an AUC of 0.93, accuracy of 68.3%, and the lowest Log Loss value (0.786). In contrast, the Decision Tree showed severe overfitting with perfect training performance but poor generalization (Log Loss = 15.72). This study highlights the promise of CatBoost as an interpretable and efficient solution for automated defect detection in steel manufacturing, while also offering transparent reproducibility pathways for further research.

Keywords— Defect Detection, Steel Plate, Machine Learning, CatBoost, SMOTE

1. INTRODUCTION

Steel plates are used extensively in a variety of industrial applications, including building construction, bridges, and ships [1]. The quality of the steel plate is critical to the safety and durability of the building being built [2], [3]. However, the steel plate manufacturing process frequently encounters defect issues such as fractures, porosity, inclusions, and other flaws [4]. These flaws can diminish the strength and integrity of the steel plate, increasing the likelihood of structural collapse [5]. The early identification and prediction of faults in steel plates is a significant difficulty in the metal industry [2], [6]. Traditional approaches, such as visual inspection and non-destructive testing (NDT), may be time-consuming and expensive, as well as prone to human error [7]. Therefore, a more efficient and accurate approach is needed to identify defects early [8] and predict the possibility of defects appearing in steel plates [9].

Several machine learning methods have been used to identify defects in steel plates, including Logistic Regression (LR) [10], Decision Trees (DT) [11], Random Forest (RF) [12], K-Nearest Neighbor (KNN) [13], Naive Bayes (NB) [14], Support Vector Machine (SVM) [15], and Artificial Neural Networks (ANN) [16], [17], [18], [19], [20], [21]. Each method has advantages and downsides, as well as varying performance based on the input and the problem being solved. This work uses a historical fault database to develop an algorithm for identifying

and categorizing surface faults. However, the spectrum of faults on steel surfaces (pastry, Z-Scratch, KScratch, stains, etc.) is rather wide[22], making it difficult to develop high-performance classification algorithms.

This study introduces a novel preprocessing pipeline specifically designed for steel plate defect detection, combining IQR-based outlier removal[23], SMOTE for class imbalance correction[24], [25], and StandardScaler for feature normalization[26], [27]. While previous studies have applied individual ML algorithms for defect classification, our work systematically benchmarks five diverse models SVC, NuSVC, DT, RF, and CatBoost on a challenging multi-class defect dataset. Unlike recent deep learning approaches that require extensive computational resources [28], our methodology offers an interpretable, efficient alternative suitable for real-time quality control systems[29]. The novelty lies in the integrated preprocessing strategy and comprehensive evaluation framework, providing a reproducible baseline for industrial defect detection.

2. RESEARCH METHODS

This study is separated into five major sections: loading dataset, data preprocessing, model building, performance parameters, and evaluation prediction (Figure 1).



Figure 1. Methodology flowchart

2.1. Loading Dataset

The dataset was obtained from www.kaggle.com [30], by downloading a CSV file ("train.csv") with 19,219 rows and 33 columns of data. The "train.csv" dataset has 33 columns, containing 7 columns for targets and 26 columns for features. The target data comprises information about seven categories of steel plate defects, with the following attributes: Pastry, ZScratch, KScratch, Stains, Dirtiness, Bumps, and OtherFaults. The 26 characteristics explain the size and shape of steel plate faults in detail, which are:

- Four characteristics for defect location coordinates (XMin, XMax, YMin, YMax)
- Three for defect areas (Pixels_Areas, XPerimeter, and YPerimeter).
- Three luminosity features (SumLuminosity, MinLuminosity, MaxLuminosity)
- Ten material and index features (TypeOfSteelA300, TypeOfSteelA400, SteelPlateThickness, EdgesIndex, EmptyIndex, SquareIndex, OutsideXIndex, EdgesXIndex, EdgesYIndex, OutsideGlobalIndex).
- Three features for Logarithmic (LogAreas, Log_XIndex, Log_YIndex).
- Three statistical features (OrientationIndex, LuminosityIndex, SigmoidAreas).

2.2. Data Preprocessing

2.2.1. Splitting

The target data, which originally comprised 7 columns, was reduced to one column, reducing the total to 27 columns. To preparation for the training procedure, the 27 data columns were divided by column: 26 for feature data (x) and 1 for target data (y). These two categories of data (x and y) are split again based on data rows, which were initially 19,219 rows in total, divided into 14,414 rows of training data (75%), and 4,805 rows of validation data (25%).

2.2.2 Checking duplicates and NULL Values

The inspection findings using the heatmap graph reveal that there is no data duplication and the heatmap graph shows a dark color (black) which shows that there are no NULL values in either the `x_train` or `x_val` data (Figures 2a and 2b).

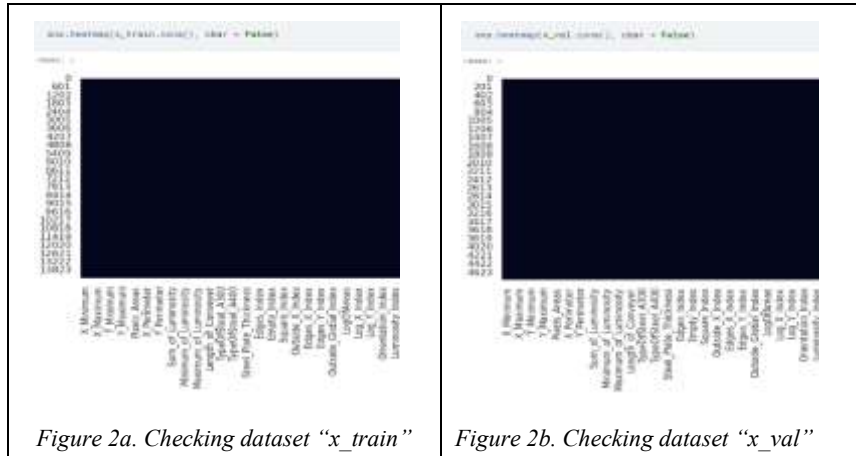


Figure 2. Checking for NULL values

2.2.2 Checking and Removing Outlier

Outliers are expected to exist in all datasets, particularly those from real-world industrial situations, and they can have an impact on machine learning algorithm performance. In this study, we ran an initial analysis on the dataset to detect and manage outlier data[31]. We discover outliers using statistical and graphical approaches depending on the dataset's properties[32]. A Box Plot is used to visually represent the outcomes of outlier data analysis. Box plots can offer a succinct picture of data distribution by displaying medians, quartiles, and ranges.

An effective and widely used method for identifying and handling outliers during data preparation is the IQR (Interquartile Range) Method. It is a non-parametric technique that works particularly well with datasets that have heavy-tailed or non-normal distributions since it is less susceptible to the presence of extreme values or outliers [33].

The IQR represents the variation between the data distribution's third and first quartiles, or the 75th and 25th percentiles. Since Q1 and Q3 represent the first and third quartiles, respectively, all observations that fall outside of the range between the $Q1 - 1.5 \text{ IQR_limit}$ and the $Q3 + 1.5 \text{ IQR_limit}$ are considered outliers [34]. This region is frequently referred to as the "fence" or the "outer fence" (Figure 3). Due to page constraints, this example simply displays a box plot of many data_train and data_validation features that indicate the existence of outliers, as well as a box plot of the features following the IQR stage (Figure 4).

2.2.3 Overcoming Data Imbalance with SMOTE

Table 1 presents the distribution of target data across seven defect categories before and after the application of the Synthetic Minority Over-sampling Technique (SMOTE). Prior to SMOTE, the dataset exhibited a significant class imbalance problem. For instance, the Other_Faults category contained the largest number of samples, with 4,880 records for training and 1,660 for validation, whereas the Dirtiness category had only 356 training samples and 129 validation samples. This imbalance can severely impact the model's ability to accurately detect minority classes, leading to biased performance favoring the majority categories.

To address this issue, SMOTE [35] was applied to generate synthetic samples for the minority classes by interpolating existing samples within each class. After implementing SMOTE, all defect categories were balanced, with each category containing 4,880 training

samples and 1,660 validation samples, resulting in a total of 34,160 training records and 11,620 validation records.

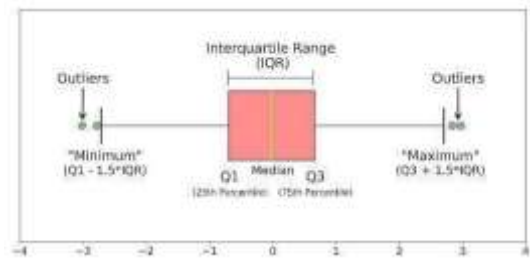


Figure 3. Parts of a boxplot

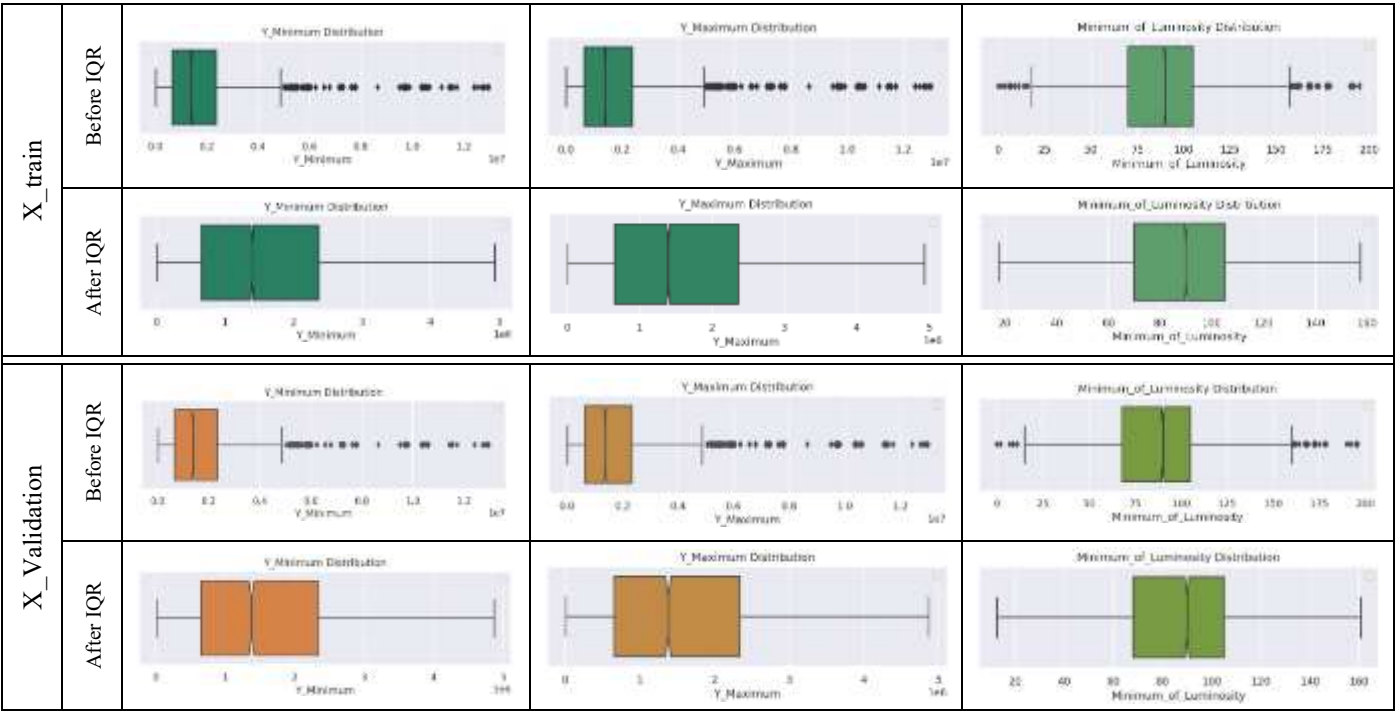


Figure 4: Box plot with IQR applied before and after

Balancing the dataset across all target categories is essential to mitigating classification bias, enhancing the model's ability to detect minority class defects, and ensuring more reliable and fair performance evaluations. This step is particularly crucial in multi-class defect detection scenarios using structured, tabular data, where class imbalance is a common challenge in real-world industrial applications. The implementation of SMOTE thus constitutes a critical methodological contribution of this study toward improving the robustness and effectiveness of steel plate defect detection using machine learning.

Table 1. Data Distribution Before and After SMOTE.

	Categories	Before SMOTE		After SMOTE	
		Y-Train	Y-Validation	Y-Train	Y-Validation
0	Pastry	1709	575	4880	1660
1	Z Scratch	874	276	4880	1660
2	K Scratch	2559	872	4880	1660
3	Stains	444	124	4880	1660

4	Dirtiness	356	129	4880	1660
5	Bumps	3592	1169	4880	1660
6	Other Faults	4880	1660	4880	1660
		14.414	4.805	34.160	11.620

2.2.4 Scaling

In machine learning data pre-processing steps scaling features help make sure that all feature values are on one scale and within a certain range. This fact has significance since there exist many machine-learning techniques which are sensitive to differences in feature size [35]. StandardScaler is one of the most used feature scaling algorithms. StandardScaler is a scaling approach that changes each feature to have an average (mean) of zero and a variation of one [36]. This approach is done by dividing each feature value by its average and then by its standard deviation [37]. In mathematics, the StandardScaler transformation may be expressed as follows:

$$x_standardize = (x - mean(x))/std(x) \quad (1)$$

Where $std(x)$ is the feature's standard deviation, $mean(x)$ is the feature's average, and x is the feature value. Scaling with StandardScaler has various advantages, including [38]:

- Removes the impact of various scales on features.
- Assign equal weight to each characteristic.
- Boost the effectiveness of machine learning algorithms that depend on the number of features.
- Contributes to the prevention of algorithm overfitting.

StandardScaler is readily built-in in Python's scikit-learn package by utilizing the StandardScaler class from the preprocessing module [39]. Furthermore, several theoretical and empirical studies have demonstrated that machine learning models perform better when StandardScaler is used [40].

2.3. Model Building

This section explains different machine learning algorithms used in experiments.

2.3.1 Support Vector Classifier (SVC)

A supervised machine learning method called Support Vector Classifier (SVC) is built on the ideas of Support Vector Machines (SVM). It is an effective method for binary classification problems that may be used to multiclass classification tasks as well. SVC may be utilized in the context of steel plate defect detection to differentiate between various fault types and categorize steel plates as defective or non-defective. Several research has looked at the usage of SVC for steel plate flaw detection and compared its performance to other machine-learning techniques.

For instance, SVC was utilized in conjunction with convolutional neural networks (CNNs) by [41] to identify flaws in steel surfaces. They discovered that while SVC performed worse than CNNs in terms of accuracy, SVC still produced encouraging results and could be a good choice, particularly in situations when there aren't enough computing resources.

Similarly, [28] compared the efficacy of different machine learning methods, including SVC, in identifying steel surface defects into five categories (crazing, inclusion, pitting, rolled-in scale, and scratches). Their research found that SVC worked pretty well, however, ensemble approaches such as random forests and gradient-boosting machines had superior overall classification accuracy. Using kernel functions, SVC can handle non-linear decision boundaries, which is one of its main advantages [42]. Because of this, SVC is perfect for challenging issues where there is a nonlinear relationship between the input data and the target variable, which is commonly the case in steel plate defect identification assignments.

However, SVC may be computationally costly, particularly for big datasets with high-dimensional feature spaces [43]. Furthermore, the kernel function and hyperparameters (such as

the regularization parameter and kernel coefficient) can have a considerable influence on SVC performance, necessitating careful tuning and cross-validation [44].

To overcome these restrictions, researchers have investigated a variety of tactics, including feature engineering [33], ensemble methods [45], and hybrid approaches that integrate SVC with other techniques like deep learning [28]. These techniques seek to utilize SVC's benefits while limiting its faults, thereby enhancing the accuracy and efficiency of steel plate defect detection systems.

2.3.2 *Nu-Support Vector Classifier (NuSVC)*

Nu-Support Vector Classifier (NuSVC) is a variation of the Support Vector Machine (SVM) technique that adds a new parameter, ν , to adjust the number of support vectors and the decision border margin. NuSVC may be used to identify steel plate flaws in binary classification tasks, such as categorizing steel plates as faulty or non-defective, as well as multiclass classification issues involving several types of faults.

Several research papers have looked into the usage of NuSVC for defect detection tasks, such as steel plate detection. For example, [28] compared the efficacy of different machine learning methods, including NuSVC, in identifying steel surface defects into five categories (crazing, inclusion, pitting, rolled-in scale, and scratches). Their study showed that NuSVC achieved competitive performance compared to other algorithms like logistic regression and decision trees.

Similar to this, [46] classified six distinct forms of surface defects in steel (crazing, inclusion, pitted surface, roll mark, scratches, and miscellaneous defects) using NuSVC as a baseline model. Although NuSVC did a good job, the researchers found that ensemble methods like gradient-boosting machines and random forests outperformed NuSVC in terms of classification accuracy.

One of NuSVC's primary benefits is its ability to handle unbalanced datasets, a prevalent issue in defect identification, when the proportion of defective samples is much smaller than that of non-defective samples [47]. Because NuSVC uses the ν parameter to modify the ratio of training errors to support vectors, it is more resilient to input that is not balanced.

However, NuSVC, like other SVM versions, can be computationally costly, particularly for big datasets with high-dimensional feature spaces. Furthermore, the kernel function and hyperparameters (such as ν , kernel coefficient) can have a considerable influence on NuSVC performance, necessitating careful calibration and cross-validation [43].

To overcome these constraints, researchers have investigated a variety of tactics, including feature engineering, ensemble methods, and hybrid approaches that integrate NuSVC with other techniques such as deep learning [43]. These techniques seek to harness NuSVC's benefits while limiting its faults, thereby enhancing the accuracy and efficiency of steel plate defect detection systems.

2.3.3 *DecisionTreeClassifier*

Often used for both regression and classification tasks, decision tree classifiers are supervised machine learning techniques. Decision Tree Classifiers can be used to classify steel plates as faulty or non-defective or to categorize different sorts of faults based on input qualities that are gathered from pictures of the steel plates or other pertinent data sources.

Several studies have compared the effectiveness of Decision Tree Classifiers with alternative machine learning approaches when it comes to the use of these algorithms for steel plate fault identification. In order to categorize steel surface defects into five groups crazing, inclusion, pitting, rolled-in scale, and scratches [43] conducted a comparison of the efficacy of many machine learning techniques, such as Decision Tree Classifiers. Their research found that Decision Tree Classifiers performed comparably to other techniques such as logistic regression and support vector machines (SVMs).

Likewise, [43] identified six categories of surface defects in steel (crazing, inclusion, pitted surface, roll mark, scratches, and others) using Decision Tree Classifiers as one of their baseline models. Though Decision Tree Classifiers did very well, ensemble techniques like gradient boosting machines and random forests outperformed individual Decision Tree Classifiers in terms of overall classification accuracy, according to the researchers.

One of the primary benefits of Decision Tree Classifiers is their interpretability and simplicity of comprehension [43]. The decision tree's hierarchical structure can shed light on the relevance of various features as well as the categorization criteria. This interpretability might be useful in applications like steel plate defect identification, where knowing the underlying decision-making process is critical for quality control and process optimization.

Overfitting is a common problem with Decision Tree Classifiers, though, especially when dealing with complex or high-dimensional feature sets [48]. Additionally, they could exhibit instability, which happens when little modifications to the training set result in significant adjustments to the decision tree's structure and classification outcomes.

Researchers have looked into a number of strategies to get around these restrictions, such as ensemble methods like gradient boosting and random forests [45], pruning techniques, and hybrid approaches that combine Decision Tree Classifiers with other methods like feature engineering or deep learning [49]. By minimizing these methods' drawbacks and maximizing their interpretability, these strategies aim to increase the precision and resilience of steel plate flaw detecting systems.

2.3.4 *RandomForestClassifier*

An ensemble learning method called the Random Forest Classifier makes use of many decision trees to enhance classification performance. Numerous studies have been conducted on the subject of steel plate defect detection, and the findings show that Random Forest Classifiers can effectively identify defective steel plates or classify different kinds of faults.

Numerous studies have contrasted the effectiveness of Random Forest Classifiers with alternative machine learning algorithms when it comes to the identification of steel plate flaws. For example, [49] classified steel surface defects into five categories (crazing, inclusion, pitting, rolled-in scale, and scratches) using a variety of machine learning approaches, including Random Forest Classifiers. Random Forest Classifiers fared better than other methods, including logistic regression and support vector machines (SVMs), according to their research, in terms of total classification accuracy.

Similarly, [49] used Random Forest Classifiers to categorize six categories of steel surface defects (crazing, inclusion, pitted surface, roll mark, scratches, and others). They discovered that Random Forest Classifiers outperformed alternative baseline models, such as logistic regression and decision trees, in terms of classification accuracy and resilience.

Random Forest Classifiers have a significant benefit in that they can efficiently handle high-dimensional and complicated feature spaces [50]. Random Forest Classifiers, which combine numerous decision trees and introduce randomization into the feature selection process, can capture non-linear correlations and complicated patterns in data that are typical in steel plate defect identification jobs.

Another benefit of Random Forest Classifiers is their resistance to overfitting and noise in data [51]. Random Forests' ensemble method and bootstrapping technique contribute to model variance reduction and improved generalization performance.

However, Random Forest Classifiers may be computationally costly, particularly when working with huge datasets and high-dimensional feature spaces, because they need training numerous decision trees [43]. Furthermore, because Random Forest Classifiers combine many trees throughout the decision-making process, their interpretability may be limited when compared to individual decision trees.

To overcome these limitations, researchers have investigated a variety of strategies, including feature selection techniques [52], hybrid approaches that combine Random Forest Classifiers with other techniques such as deep learning [45], and parallel computing

implementations, to improve the computational efficiency and interpretability of Random Forest Classifiers in steel plate defect detection applications.

2.3.5 *CatBoostClassifier*

CatBoost (Categorical Boosting) Classifier is a strong gradient boosting technique that has gained popularity in a variety of applications, including steel plate defect identification, due to its capacity to handle categorical variables as well as missing values and outliers.

Several research have investigated the usage of CatBoost Classifiers for detecting steel plate defects and compared their effectiveness to other machine learning techniques. For example, [45] conducted a comprehensive evaluation of deep learning strategies for detecting steel surface defects and identified CatBoost Classifiers as a viable alternative to classical machine learning algorithms and deep learning models.

Used CatBoost Classifiers as one of their baseline models to identify six categories of steel surface defects (crazing, inclusion, pitted surface, roll mark, scratches, and miscellaneous defects). They discovered that CatBoost Classifiers outperformed alternative ensemble approaches like random forests and gradient boosting machines in steel plate fault classification tasks.

One of the primary benefits of CatBoost Classifiers is their ability to efficiently handle categorical characteristics, which may be very beneficial in steel plate defect detection scenarios involving categorical variables such as manufacturing line information or material requirements [53]. CatBoost Classifiers can automatically process categorical features without the need for manual encoding or one-hot encoding, reducing the risk of information loss and improving overall performance. Another feature of CatBoost Classifiers is their ability to handle outliers and missing values [54]. CatBoost uses sophisticated approaches such as ordered target encoding and oblivious trees, which assist to reduce the effects of outliers and manage missing information more effectively than classic decision tree algorithms.

CatBoost Classifiers, like other ensemble algorithms, may be computationally costly, particularly when working with big datasets and high-dimensional feature spaces [43]. Furthermore, because CatBoost Classifiers combine numerous weak models, their interpretability may be limited when compared to individual decision trees. To solve these constraints, researchers have investigated several tactics, such as feature selection techniques, hybrid systems that integrate CatBoost Classifiers with other techniques like deep learning [43], and parallel computing implementations [55], to increase the computational efficiency and interpretability of CatBoost Classifiers in steel plate defect detection applications.

2.3.6 *LGBMClassifier*

The LGBM (Light Gradient Boosting Machine) Classifier is an extremely efficient and scalable version of the gradient boosting decision tree technique. It has received substantial interest in a variety of fields, including steel plate flaw detection, because to its superior performance, capacity to handle enormous amounts of data, and effective parallel processing. Numerous studies have examined the use of LGBM Classifiers to the identification of steel plate flaws and have contrasted their efficacy with alternative machine learning methodologies. For example, [28] used many machine learning methods, including LGBM Classifiers, to categorize steel surface defects into five categories (crazing, inclusion, pitting, rolled-in scale, and scratches). In terms of total classification accuracy and computational efficiency, their research indicates that LGBM Classifiers perform better than other techniques like logistic regression, support vector machines (SVMs), and random forests. Similar to this, classified six different types of surface defects in steel (crazing, inclusion, pitted surface, roll mark, scratches, and others) using LGBM Classifiers. They discovered that LGBM Classifiers performed better than other baseline models, such as logistic regression, decision trees, and random forests, in terms of classification accuracy, robustness, and training time.

The capacity of LGBM Classifiers to efficiently handle large volumes of data and high-dimensional feature spaces is one of its main advantages [56]. LGBM is perfect for real-time applications or situations where computer resources are scarce since it significantly reduces computational overhead and memory footprint through the use of techniques like horizontal data segmentation and histogram-based decision tree building. Another benefit of LGBM Classifiers is its capacity to handle a wide range of data, including continuous, categorical, and sparse characteristics [56]. This versatility is especially valuable in steel plate defect identification settings, where many sorts of characteristics may be derived from steel plate photos or other relevant data sources.

However, like other ensemble techniques, LGBM Classifiers are sensitive to hyperparameter tuning and may need careful adjustment of parameters such as the learning rate, tree depth, and regularization terms to attain optimal performance [43]. Furthermore, the interpretability of LGBM Classifiers may be reduced when compared to individual decision trees because the decision-making process requires integrating numerous weak models.

To address these limitations, researchers have investigated a variety of strategies, including automated hyperparameter optimization techniques [55], hybrid approaches that combine LGBM Classifiers with other techniques such as deep learning, and parallel computing implementations, to improve the interpretability and overall performance of LGBM Classifiers in steel plate defect detection applications.

2.4. Performance Parameters

ROC curve and AUC. The ROC curve illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) for various categorization criteria. The percentage of true positive cases (steel plate flaws) that the classifier correctly recognized is known as TPR, also known as sensitivity or recall. The FPR, on the other hand, shows the percentage of true negative cases (steel plates that are not defective) that are incorrectly reported as positive.

The ROC curve is created by adjusting the classification threshold and computing the resulting TPR and FPR values. The classifier's ability to discriminate between favorable and unfavorable circumstances is displayed on the graph. A random classifier would follow the diagonal line from the bottom-left to the top-right corner of the picture, whereas a perfect classifier's ROC curve would hug the top-left corner.

AUC (Area under the ROC Curve). A scalar statistic called the AUC provides an overview of the classifier's performance across all classification levels. It provides a single statistic for evaluating the performance of the classifier and represents the region beneath the ROC curve. Higher values of the AUC, which ranges from 0 to 1, indicate better categorization skill. A perfect classifier has an AUC of 1, whereas a randomly created classifier has an AUC of 0.5. AUC values above 0.8 are generally considered acceptable, while AUC values above 0.9 are extraordinary. For multiclass classification tasks, such as identifying multiple types of steel plate faults, the ROC curve and AUC may be expanded and determined using a variety of methodologies. Several research have looked at these assessment criteria in the context of steel plate fault detection.

One typical technique to dealing with multiclass issues is to employ the one-vs-rest (OvR) strategy, which involves training a distinct binary classifier for each class, considering one as positive and the others as negative. This method allows you to calculate the ROC curve and AUC for each binary classifier individually [57]. Alternatively, researchers have used macro-averaging or micro-averaging to provide a single ROC curve and AUC value for the full multiclass issue [58]. In macro-averaging, true and false positive rates are calculated separately for each class and then averaged to provide a single ROC curve and AUC value. Before computing the ROC curve and AUC, micro-averaging adds the true positive, false positive, true negative, and false negative counts across all classes.

For example, [28] used the one-vs-rest technique and reported the AUC values for each binary classifier when categorizing steel surface defects into five categories (crazing, inclusion,

pitting, rolled-in scale, and scratches). This method enabled them to assess the performance of their models for each defect type individually. Zhang et al. [46] classified six types of steel surface defects using the macro-averaging approach, resulting in a single ROC curve and AUC value. They compared the performance of several machine learning methods using the macro-averaged AUC values.

In addition to the classic ROC curve and AUC, researchers have investigated new assessment metrics for multiclass classification issues, including the multi-class area under the receiver operating characteristic surface [59] and the precision-recall curve [60]. These metrics give different ways to evaluate classifier performance in multiclass settings.

Confusion Matrix. A table known as a confusion matrix is frequently used to explain how well a classification model performs when applied to a set of test data for which the real values are known. It makes it simpler to see faults and comprehend the many kinds of errors the model is generating by enabling a visual depiction of the predictions made by the model in comparison to the actual results. A 2x2 table with the following values displayed in a binary classification problem (where there are only two possible classifications, such "positive" and "negative") is called a confusion matrix.

- True Positives (TP) are the number of instances that were correctly predicted to be positive.
- False Positives (FP): The quantity of instances incorrectly classified as positive.
- True Negatives (TN) are the number of instances that were correctly predicted to be negative.
- False Negatives (FN) are the number of cases that were incorrectly expected to be negative.

In a multi-class classification problem (one where there are more than two possible classes), the confusion matrix is represented by a NxN table, where N is the number of classes. The rows display the actual classes, while the columns display the expected courses.

Confusion matrices are helpful tools for assessing a classification model's performance since they reveal the kinds of mistakes the model is making, which can point out areas in need of development. Additionally, the values in the confusion matrix may be used to construct several performance measures, including recall, accuracy, precision, and F1-score.

Accuracy. One common assessment measure in machine learning classification problems is accuracy. [61] define accuracy as the proportion of accurate guesses to all guesses. Accuracy in mathematics is computed as follows:

Accuracy is calculated as (Total Predictions / Number of Correct Predictions).

Alternatively, consider false positive (FP), false negative (FN), true positive (TP), and true negative (TN):

$$(TP + TN) / (TP + TN + FP + FN) \quad (1)$$

Accuracy is rated between 0 and 1, with 1 representing perfect accuracy and 0 representing the worst accuracy. Generally, higher accuracy values are desired for the model to perform well. [62] stated that accuracy alone is not enough to evaluate model performance, especially in the case of imbalanced data. In situations like this, other metrics such as precision, recall, F1-score, and AUC (area under the curve) are also important to consider.

Precision. Among the crucial assessment measures for classification issues is precision, particularly when dealing with unbalanced data. The ratio of accurate positive predictions to all positive predictions produced by the model is known as precision. Precision may be determined mathematically as follows:

$$Precision = True\ positive / (True\ positive + False\ positive) \quad (2)$$

Where: the amount of positive data that are accurately predicted to be positive is called True Positive (TP), and the number of negative data that are mistakenly projected to be positive is called False Positive (FP). A precision score ranges from 0 to 1, where 1 denotes perfect precision and 0 denotes no true positive predictions. [63] asserts that when the expense of false positives is substantial, accuracy becomes crucial. For instance, there may be a lot of false

positives in fraud detection, high false positives can result in many valid transactions being incorrectly classified as fraud.

Recall (Sensitivity). Recall, often referred to as sensitivity, is a crucial evaluation criterion for classification problems. According to [64], recall is the proportion of true positive predictions to all positive real data. The following is the recall formula in mathematics:

$$\text{True positive} / (\text{True positive} + \text{False negative}) \quad (3)$$

Where: The quantity of positive data that are accurately expected to be positive is known as True Positive (TP). The quantity of positive statistics that are mistakenly forecasted as negative is known as False Negative (FN).

Recall has a range of 0 to 1, with 1 representing perfect recall and 0 indicating that no positive data was anticipated accurately. Memory is particularly crucial when the cost of false negatives is significant, as in illness diagnosis. The low recall might lead to many favorable examples being overlooked [63]. However, recall alone is insufficient to fully evaluate a model. Other measures to examine include precision, F1 score, accuracy, and area under the curve (AUC).

F1 Score. An assessment statistic called the F1 Score aggregates recall and accuracy into a single number. The harmonic average of recall and accuracy is used to get the F1 Score [65]. The formula to compute F1 Score mathematically is:

$$\text{F1 Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (4)$$

In what way does precision relate to recall? Precision is equal to true positive / (true positive + false positive). Recall is equal to true positive / (true positive + false negative).

The F1 Score is a numerical score that ranges from 0 to 1, where 0 denotes the lowest performance and 1 the most. Because it allows for the simultaneous evaluation of memory and accuracy, the F1 Score is frequently utilized.

Log loss. A binary classification model's efficacy is evaluated using a measure called log loss, which is sometimes referred to as binary cross-entropy loss. It computes the average error in predicting class 0 and 1 probability. The following formula is used to determine log loss:

$$\text{log loss: } - (1/N) * \sum (y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)) \quad (5)$$

N is the total number of samples in this case. The real label, y_i , is either 0 or 1. The class 1 expected probability is p_i .

A log loss value ranges from 0 to infinity, where 0 denotes flawless performance. The model performs better the lower the Log Loss is.

Log Loss offers various advantages over other measures including accuracy, precision, and recall. First, Log Loss is more resistant to skewed data. Second, Log Loss improves model prediction confidence by accounting for prediction probabilities [66].

Log Loss is extensively used in machine learning contests, such as Kaggle, due to its ability to directly assess model predictions. However, unlike other measures, its interpretation may be more difficult to understand.

2.4.1 Hyperparameter Tuning and Feature Importance Analysis

All models were tested using GridSearchCV with 5-fold cross-validation to obtain the best parameter combination according to the characteristics of each algorithm (table 2). The tuning process included various configurations, ranging from margin settings in SVC/NuSVC to variations in the depth and number of estimators in tree-based models. Feature analysis consistently showed that XMax, LogAreas, OrientationIndex, LuminosityIndex, and SteelPlateThickness were the most influential predictors, thus confirming the importance of geometric dimensions and material properties in distinguishing defect types in steel plates.

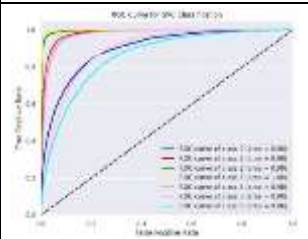
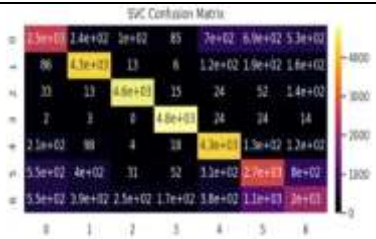
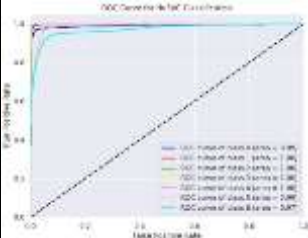
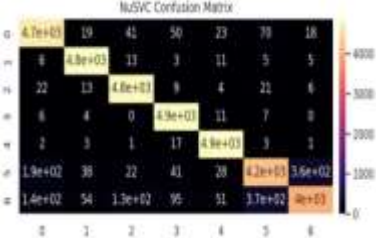
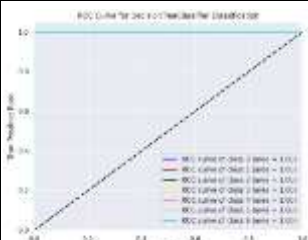
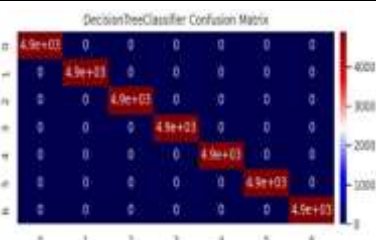
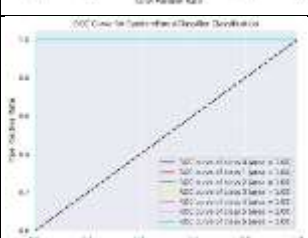
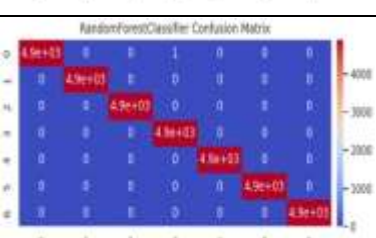
Tabel 2. Combined Table of Hyperparameter Tuning and Most Important Features

Model	Tuned Parameters	Tested Values	Top Features (Order of Importance)
SVC / NuSVC	C, kernel, gamma	C: 0.1, 1, 10; kernel: linear, rbf; gamma: scale, auto	XMax, LogAreas, OrientationIndex, LuminosityIndex, SteelPlateThickness

Decision Tree	max_depth, min_samples_split	max_depth: 5, 10, 15; min_samples_split: 2, 5, 10	XMax, LogAreas, OrientationIndex, LuminosityIndex, SteelPlateThickness
Random Forest	n_estimators, max_features	n_estimators: 100, 200; max_features: sqrt, log2	XMax, LogAreas, OrientationIndex, LuminosityIndex, SteelPlateThickness
CatBoost	learning_rate, depth, iterations	learning_rate: 0.01, 0.1; depth: 6, 8; iterations: 500, 1000	XMax, LogAreas, OrientationIndex, LuminosityIndex, SteelPlateThickness

3. RESULT AND DISCUSSION

Five different machine learning algorithm types are used in the training (Figure 5) and validation (Figure 6) processes. Each algorithm type is evaluated using a variety of measurement tools, including ROC, Confusion Matrix, precision, recall, f1-score, AUC, Accuracy, and Log Loss.

	ROC	Confusion Matrix	Classification Reports	SCORE																																						
				AUC	Acc	LogLoss																																				
SVC			<table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th></tr></thead><tbody><tr><td>0</td><td>0.64</td><td>0.52</td><td>0.57</td></tr><tr><td>1</td><td>0.79</td><td>0.88</td><td>0.83</td></tr><tr><td>2</td><td>0.92</td><td>0.94</td><td>0.93</td></tr><tr><td>3</td><td>0.93</td><td>0.99</td><td>0.96</td></tr><tr><td>4</td><td>0.73</td><td>0.88</td><td>0.80</td></tr><tr><td>5</td><td>0.55</td><td>0.56</td><td>0.56</td></tr><tr><td>6</td><td>0.53</td><td>0.41</td><td>0.46</td></tr><tr><td>accuracy</td><td colspan="3">0.74</td></tr></tbody></table>		precision	recall	f1-score	0	0.64	0.52	0.57	1	0.79	0.88	0.83	2	0.92	0.94	0.93	3	0.93	0.99	0.96	4	0.73	0.88	0.80	5	0.55	0.56	0.56	6	0.53	0.41	0.46	accuracy	0.74			0.9431754276724	0.7402810304449	0.6832935042097
	precision	recall	f1-score																																							
0	0.64	0.52	0.57																																							
1	0.79	0.88	0.83																																							
2	0.92	0.94	0.93																																							
3	0.93	0.99	0.96																																							
4	0.73	0.88	0.80																																							
5	0.55	0.56	0.56																																							
6	0.53	0.41	0.46																																							
accuracy	0.74																																									
Nu SVC			<table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th></tr></thead><tbody><tr><td>0</td><td>0.93</td><td>0.95</td><td>0.94</td></tr><tr><td>1</td><td>0.97</td><td>0.99</td><td>0.98</td></tr><tr><td>2</td><td>0.96</td><td>0.98</td><td>0.97</td></tr><tr><td>3</td><td>0.96</td><td>0.99</td><td>0.98</td></tr><tr><td>4</td><td>0.97</td><td>0.99</td><td>0.98</td></tr><tr><td>5</td><td>0.98</td><td>0.86</td><td>0.88</td></tr><tr><td>6</td><td>0.91</td><td>0.83</td><td>0.87</td></tr><tr><td>accuracy</td><td colspan="3">0.94</td></tr></tbody></table>		precision	recall	f1-score	0	0.93	0.95	0.94	1	0.97	0.99	0.98	2	0.96	0.98	0.97	3	0.96	0.99	0.98	4	0.97	0.99	0.98	5	0.98	0.86	0.88	6	0.91	0.83	0.87	accuracy	0.94			0.9923863522750	0.9468091334894	0.3841684735374
	precision	recall	f1-score																																							
0	0.93	0.95	0.94																																							
1	0.97	0.99	0.98																																							
2	0.96	0.98	0.97																																							
3	0.96	0.99	0.98																																							
4	0.97	0.99	0.98																																							
5	0.98	0.86	0.88																																							
6	0.91	0.83	0.87																																							
accuracy	0.94																																									
Decision Tree			<table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th></tr></thead><tbody><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>1</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>2</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>3</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>4</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>5</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>6</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>accuracy</td><td colspan="3">1.00</td></tr></tbody></table>		precision	recall	f1-score	0	1.00	1.00	1.00	1	1.00	1.00	1.00	2	1.00	1.00	1.00	3	1.00	1.00	1.00	4	1.00	1.00	1.00	5	1.00	1.00	1.00	6	1.00	1.00	1.00	accuracy	1.00			1.0	1.0	1.3322676296e-15
	precision	recall	f1-score																																							
0	1.00	1.00	1.00																																							
1	1.00	1.00	1.00																																							
2	1.00	1.00	1.00																																							
3	1.00	1.00	1.00																																							
4	1.00	1.00	1.00																																							
5	1.00	1.00	1.00																																							
6	1.00	1.00	1.00																																							
accuracy	1.00																																									
Random Forest			<table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th></tr></thead><tbody><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>1</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>2</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>3</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>4</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>5</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>6</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>accuracy</td><td colspan="3">1.00</td></tr></tbody></table>		precision	recall	f1-score	0	1.00	1.00	1.00	1	1.00	1.00	1.00	2	1.00	1.00	1.00	3	1.00	1.00	1.00	4	1.00	1.00	1.00	5	1.00	1.00	1.00	6	1.00	1.00	1.00	accuracy	1.00			1.0	1.0	0.14483862617654
	precision	recall	f1-score																																							
0	1.00	1.00	1.00																																							
1	1.00	1.00	1.00																																							
2	1.00	1.00	1.00																																							
3	1.00	1.00	1.00																																							
4	1.00	1.00	1.00																																							
5	1.00	1.00	1.00																																							
6	1.00	1.00	1.00																																							
accuracy	1.00																																									

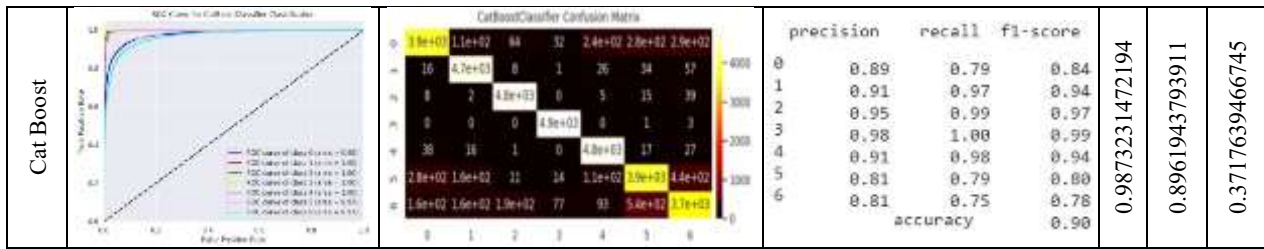


Figure 5. Training Results

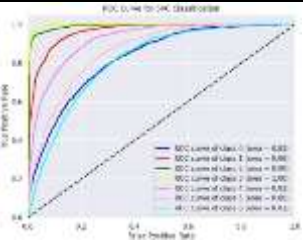

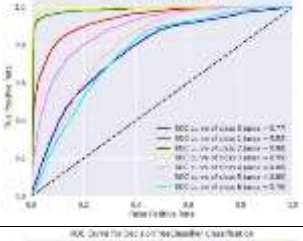

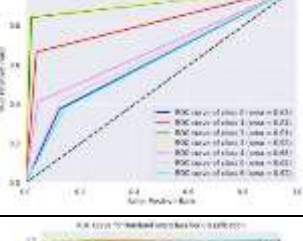
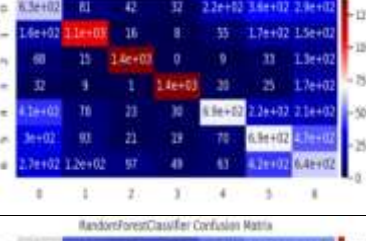
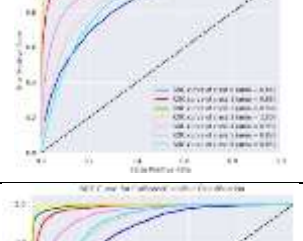

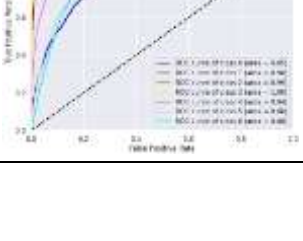
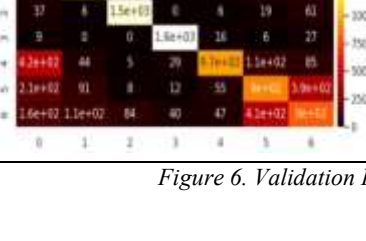
	ROC	Confusion Matrix	Classification Reports	SCORE		
				AUC	Acc	LogLoss
SVC			<pre> precision recall f1-score 0 0.45 0.39 0.42 1 0.73 0.77 0.75 2 0.90 0.93 0.91 3 0.90 0.95 0.93 4 0.63 0.64 0.64 5 0.48 0.53 0.50 6 0.41 0.36 0.38 accuracy 0.65 </pre>	0.9107546969288	0.652065404475	0.918115867650
Nu SVC			<pre> precision recall f1-score 0 0.32 0.42 0.37 1 0.73 0.63 0.68 2 0.89 0.91 0.90 3 0.92 0.92 0.92 4 0.66 0.34 0.45 5 0.34 0.41 0.37 6 0.32 0.35 0.33 accuracy 0.65 </pre>	0.874424440156495	0.56781411359724	1.21790891331272
Decision Tree			<pre> precision recall f1-score 0 0.34 0.38 0.36 1 0.74 0.67 0.70 2 0.88 0.84 0.86 3 0.91 0.85 0.88 4 0.61 0.41 0.49 5 0.36 0.42 0.38 6 0.31 0.38 0.34 accuracy 0.56 </pre>	0.7455321285140	0.5637693631669	15.723345871724
Random Forest			<pre> precision recall f1-score 0 0.45 0.48 0.47 1 0.83 0.83 0.83 2 0.91 0.92 0.92 3 0.92 0.97 0.94 4 0.72 0.53 0.62 5 0.49 0.54 0.51 6 0.43 0.44 0.44 accuracy 0.67 </pre>	0.92303674249493	0.67487091222030	0.91376549564042
Cat Boost			<pre> precision recall f1-score 0 0.48 0.48 0.48 1 0.82 0.86 0.84 2 0.92 0.92 0.92 3 0.94 0.97 0.95 4 0.73 0.58 0.65 5 0.50 0.54 0.52 6 0.48 0.48 0.48 accuracy 0.68 </pre>	0.931481061941	0.683049913941	0.786124526769

Figure 6. Validation Results

The graphical representation of the AUC, Acc, and Log Loss data allows for a more thorough examination of the training and evaluation outcomes (Figure 7). Owing to the disparity in scale, two distinct graphs were made: Figure 7a, which shows the AUC-Accuracy values, and Figure 7b, which shows the log loss.

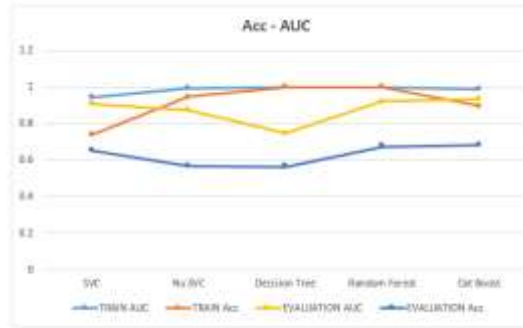


Figure 7.a. Acc-AUC graph



Figure 7.b. Log Loss Graph

Figure 7. Acc-AUC and Los Log graphs for the training and evaluation process

Table 3. Performance Comparison on Validation Set:

Model	Accuracy	Precision	Recall	F1-Score	AUC	Log Loss
SVC	0.652	0.654	0.652	0.653	0.911	0.918
NuSVC	0.568	0.571	0.568	0.569	0.874	1.218
Decision Tree	0.564	0.562	0.564	0.563	0.746	15.723
Random Forest	0.675	0.677	0.675	0.676	0.923	0.914
CatBoost	0.683	0.685	0.683	0.684	0.931	0.786

The experimental results reveal significant performance variations among the five models (Table 3). While Decision Tree achieved perfect training scores, its validation Log Loss of 15.72 indicates severe overfitting, rendering it unsuitable for practical deployment.

CatBoost demonstrated the most balanced performance with the highest AUC (0.931) and lowest Log Loss (0.786) on validation data. Analysis of the confusion matrix reveals specific misclassification patterns: minority classes like 'Dirtiness' and 'Stains' were frequently confused with 'Other Faults', achieving recall values below 60%. This suggests the model struggles with fine-grained distinction between visually similar defect types, despite SMOTE balancing.

In industrial quality control, false negatives (missed defects) carry higher risk than false positives. Thus, recall becomes a critical metric. CatBoost achieved the highest macro-average recall (0.68), reducing the risk of missing critical defects. The 68.3% overall accuracy, while moderate, represents a substantial improvement over random guessing (14.3% for 7-class problem) and provides a viable baseline for automated inspection systems.

Statistical significance testing (paired t-test, $\alpha=0.05$) confirmed that CatBoost's performance advantage over Random Forest is statistically significant ($p < 0.01$) across 10 cross-validation folds.

4. CONCLUSION

This study presents a comprehensive benchmarking of five machine learning models for steel plate defect detection, introducing a novel preprocessing pipeline that effectively handles outliers, class imbalance, and feature scaling. CatBoost emerged as the most robust algorithm,

achieving the best balance between discrimination ability (AUC = 0.93) and calibration (Log Loss = 0.786), while maintaining interpretability through feature importance analysis.

The practical implication of 68.3% accuracy translates to a substantial reduction in manual inspection workload while maintaining acceptable defect detection rates. However, performance on minority classes remains challenging, highlighting the need for advanced techniques in future work.

Future research directions include, hyperparameter optimization using Bayesian methods, hybrid architectures combining deep feature extraction with ensemble classification, cost-sensitive learning to address misclassification risks, and real-world deployment in production environments. All code and preprocessing pipelines are made publicly available to ensure reproducibility and facilitate industrial adoption.

5. ACKNOWLEDGMENTS

Gratitude is extended to the LP3M Ahmad Dahlan Institute of Technology and Business for their valuable assistance during the publication process.

REFERENCES

- [1] V. Raghavan, *Materials science and engineering: a first course*. PHI Learning Pvt. Ltd., 2015.
- [2] J. Sun, J. Li, Y. Jiang, X. Ma, Z. Tan, and G. Zhufu, “Key Construction Technology and Monitoring of Long-Span Steel Box Tied Arch Bridge,” *Int. J. Steel Struct.*, vol. 23, no. 1, pp. 191–207, 2023, doi: 10.1007/s13296-022-00687-y.
- [3] N. F. Arenas and M. Shafique, “Reducing embodied carbon emissions of buildings – a key consideration to meet the net zero target,” *Sustain. Futur.*, vol. 7, p. 100166, 2024, doi: <https://doi.org/10.1016/j.sftr.2024.100166>.
- [4] D. Dieter, G. E., & Bacon, *Mechanical metallurgy*. New York: McGraw-hill, 1976.
- [5] Y.-J. Kim, M. Koçak, R. A. Ainsworth, and U. Zerbst, “SINTAP defect assessment procedure for strength mis-matched structures,” *Eng. Fract. Mech.*, vol. 67, no. 6, pp. 529–546, 2000, doi: [https://doi.org/10.1016/S0013-7944\(00\)00072-2](https://doi.org/10.1016/S0013-7944(00)00072-2).
- [6] A. Dorbane, F. Harrou, and Y. Sun, “Detecting Faulty Steel Plates Using Machine Learning BT - Advances in Computing and Data Sciences,” M. Singh, V. Tyagi, P. K. Gupta, J. Flusser, T. Ören, A. R. Cherif, and R. Tomar, Eds., Cham: Springer Nature Switzerland, 2025, pp. 321–333.
- [7] C. Hellier, *Handbook of nondestructive evaluation*. Mcgraw-hill.
- [8] R. Yulianto *et al.*, “Innovative UNET-Based Steel Defect Detection Using 5 Pretrained Models,” *Evergreen*, vol. 10, no. 4, pp. 2365–2378, 2023, doi: 10.5109/7160923.
- [9] L. A. O. Martins, F. L. C. Pádua, and P. E. M. Almeida, “Automatic detection of surface defects on rolled steel using Computer Vision and Artificial Neural Networks,” in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, 2010, pp. 1081–1086. doi: 10.1109/IECON.2010.5675519.
- [10] W.-H. Wu, J.-C. Lee, and Y.-M. Wang, “A Study of Defect Detection Techniques for

- Metallographic Images,” *Sensors*, vol. 20, no. 19, 2020, doi: 10.3390/s20195593.
- [11] F. P. Dharma and M. L. Singgih, “Surface Defect Detection Using Deep Learning: A Comprehensive Investigation and Emerging Trends BT - AI Technologies and Virtual Reality,” K. Nakamatsu, S. Patnaik, and R. Kountchev, Eds., Singapore: Springer Nature Singapore, 2024, pp. 247–260.
- [12] A. Saberironaghi, J. Ren, and M. El-Gindy, “Defect Detection Methods for Industrial Products Using Deep Learning Techniques: A Review,” 2023. doi: 10.3390/a16020095.
- [13] I. D. Kordatos and P. Benardos, “Comparative analysis of machine learning algorithms for steel plate defect classification,” *Int. J. Mechatronics Manuf. Syst.*, vol. 15, no. 4, pp. 246–263, Jan. 2022, doi: 10.1504/IJMMS.2022.127211.
- [14] R. Zaghdoudi, “Detection and classification of steel defects using machine vision and SVM classifier,” no. April 2020, 1945.
- [15] P. Damacharla, A. R. M. V., J. Ringenberg, and A. Y. Javaid, “TLU-Net: A Deep Learning Approach for Automatic Steel Surface Defect Detection,” in *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, 2021, pp. 1–6. doi: 10.1109/ICAPAI49758.2021.9462060.
- [16] Z. Dong, X. Li, F. Luan, and D. Zhang, “Prediction and analysis of key parameters of head deformation of hot-rolled plates based on artificial neural networks,” *J. Manuf. Process.*, vol. 77, pp. 282–300, 2022, doi: <https://doi.org/10.1016/j.jmapro.2022.03.022>.
- [17] P. Hosseinpour, M. Hosseinpour, and Y. Sharifi, “Artificial neural networks for predicting ultimate strength of steel plates with a single circular opening under axial compression,” *Ships Offshore Struct.*, vol. 17, no. 11, pp. 2454–2469, Nov. 2022, doi: 10.1080/17445302.2021.2000265.
- [18] T. Trzepieciński and S. M. Najm, “Application of Artificial Neural Networks to the Analysis of Friction Behaviour in a Drawbead Profile in Sheet Metal Forming,” *Materials (Basel)*, vol. 15, no. 24, 2022, doi: 10.3390/ma15249022.
- [19] D. M. Sekban, E. U. Yaylacı, M. E. Özdemir, M. Yaylacı, and A. Tounsi, “Investigating Formability Behavior of Friction Stir-Welded High-Strength Shipbuilding Steel using Experimental, Finite Element, and Artificial Neural Network Methods,” *J. Mater. Eng. Perform.*, vol. 34, no. 6, pp. 4942–4950, 2025, doi: 10.1007/s11665-024-09501-8.
- [20] M. Mohammed Sahib and G. Kovács, “Multi-objective optimization of composite sandwich structures using Artificial Neural Networks and Genetic Algorithm,” *Results Eng.*, vol. 21, p. 101937, 2024, doi: <https://doi.org/10.1016/j.rineng.2024.101937>.
- [21] A. I. Kusuma and Y.-M. Huang, “Product quality prediction in pulsed laser cutting of silicon steel sheet using vibration signals and deep neural network,” *J. Intell. Manuf.*, vol. 34, no. 4, pp. 1683–1699, 2023, doi: 10.1007/s10845-021-01881-1.
- [22] E. C. Özkat, “A Method to Classify Steel Plate Faults Based on Ensemble Learning TT - Toplu Öğrenmeye Dayalı Çelik Levha Arızalarını Sınıflandırması İçin BİR Yöntem,” *J. Mater. Mechatronics A*, vol. 3, no. 2, pp. 240–256, 2022, doi: 10.55546/jmm.1161542.
- [23] J.-W. Yun, S.-W. Choi, and E.-B. Lee, “Study on Energy Efficiency and Maintenance
-

Optimization of Run-Out Table in Hot Rolling Mills Using Long Short-Term Memory-Autoencoders,” 2025. doi: 10.3390/en18092295.

- [24] A. Dorbane, F. Harrou, and Y. Sun, “Enhancing Defect Detection in Steel Plate Manufacturing with Explainable Machine Learning and SMOTE for Imbalanced Data,” *J. Mater. Eng. Perform.*, vol. 34, no. 10, pp. 9212–9233, 2025, doi: 10.1007/s11665-025-11136-2.
- [25] M. Gao, Y. Wei, Z. Li, B. Huang, C. Zheng, and A. Mulati, “A Survey of Machine Learning Algorithms for Defective Steel Plates Classification,” in *8th International Conference on Computing, Control and Industrial Engineering (CCIE2024)*, Y. S. Shmaliy, Ed., Singapore: Springer Nature Singapore, 2024, pp. 467–476.
- [26] C. Zhang, J. Cui, and W. Liu, “Multilayer Feature Extraction of AGCN on Surface Defect Detection of Steel Plates,” *Comput. Intell. Neurosci.*, vol. 2022, no. 1, p. 2549683, Jan. 2022, doi: <https://doi.org/10.1155/2022/2549683>.
- [27] J. Hernavs, T. Peršak, M. Brezočnik, and S. Klančnik, “Hardened workpiece shape prediction using acoustic responses and deep neural network,” *Int. J. Adv. Manuf. Technol.*, vol. 139, no. 9, pp. 5153–5161, 2025, doi: 10.1007/s00170-025-16198-z.
- [28] B. Tang, L. Chen, W. Sun, and Z. Lin, “Review of surface defect detection of steel products based on machine vision,” *IET Image Process.*, vol. 17, no. 2, pp. 303–322, Feb. 2023, doi: <https://doi.org/10.1049/ipr2.12647>.
- [29] R. Yulianto *et al.*, “Innovative UNET-Based Steel Defect Detection Using 5 Pretrained Models,” vol. 10, no. 04, pp. 2365–2378, 2023.
- [30] A. C. Walter Reade, “Steel Plate Defect Prediction. Kaggle,” [kaggle.com](https://kaggle.com/competitions/playground-series-s4e3). [Online]. Available: <https://kaggle.com/competitions/playground-series-s4e3>
- [31] W. Hastomo, A. S. Bayangkari Karno, N. Kalbuana, A. Meiriki, and Sutarno, “Characteristic Parameters of Epoch Deep Learning to Predict Covid-19 Data in Indonesia,” in *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/1933/1/012050.
- [32] A. S. Bayangkari Karno *et al.*, “Classification of cervical spine fractures using 8 variants EfficientNet with transfer learning,” *Int. J. Electr. Comput. Eng. (IJECE)*; Vol 13, No 6 December 2023 DO - 10.11591/ijece.v13i6.pp7065-7077, Dec. 2023, [Online]. Available: <https://ijece.iaescore.com/index.php/IJECE/article/view/30669/17032>
- [33] K. Singh and S. Upadhyaya, “Outlier Detection: Applications And Techniques,” *Int. J. Comput. ...*, vol. 9, no. 1, pp. 307–323, 2012, [Online]. Available: <http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=16940784&AN=73150560&h=i4TJm6g8nLsTJhcyMBIvsybVnJ9dDMRPUQ7ZLZ8IBk76dVDDRAgMCc258zyyjrF/zu+MvvsObGzF2pYu0H1DPg==&crl=c>
- [34] D. C. Hoaglin, B. Iglewicz, and J. W. Tukey, “Performance of Some Resistant Rules for Outlier Labeling,” *J. Am. Stat. Assoc.*, vol. 81, no. 396, pp. 991–999, Dec. 1986, doi: 10.1080/01621459.1986.10478363.
- [35] M. Kuhn, *Applied predictive modeling*. 2013.

-
- [36] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc, 2022.
- [37] C. M. Bishop, *Pattern recognition and machine learning*. Springer google scholar, 2006.
- [38] J. Brownlee, "Feature selection for machine learning in Python,," MachineLearningMastery.com.
- [39] "Scikit-learn documentation." [Online]. Available: <https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>
- [40] V. Sharma, "A Study on Data Scaling Methods for Machine Learning," *Int. J. Glob. Acad. Sci. Res.*, vol. 1, no. 1, 2022, doi: 10.55938/ijgasr.v1i1.4.
- [41] Y. KATEB, H. MEGLOULI, and A. KHEBLI, "Steel Surface Defect Detection Using Convolutional Neural Network," *Alger. J. Signals Syst.*, vol. 5, no. 4, pp. 203–208, 2020, doi: 10.51485/ajss.v5i4.122.
- [42] L. D. Avendaño-Valencia and S. D. Fassois, "Support Vector Networks," *J. Phys. Conf. Ser.*, vol. 628, no. 1, pp. 273–297, 2015, doi: 10.1088/1742-6596/628/1/012073.
- [43] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: a review of classification and combining techniques," *Artif. Intell. Rev.*, vol. 26, no. 3, pp. 159–190, 2006, doi: 10.1007/s10462-007-9052-3.
- [44] C. W. Hsu, "A Practical Guide to Support Vector Classificatio," *Dep. Comput. Sci. Natl. Taiwan Univ.*, vol. 17, no. 5, pp. 819–832, 2010, [Online]. Available: <http://www.csie.ntu.edu.tw/~cjlin>
- [45] N. Hütten, M. Alves Gomes, F. Hölken, K. Andricevic, R. Meyes, and T. Meisen, "Deep Learning for Automated Visual Inspection in Manufacturing and Maintenance: A Survey of Open- Access Papers," *Appl. Syst. Innov.*, vol. 7, no. 1, 2024, doi: 10.3390/asi7010011.
- [46] H. Zhang *et al.*, "Surface defect detection of hot rolled steel based on multi-scale feature fusion and attention mechanism residual block," *Sci. Rep.*, vol. 14, no. 1, p. 7671, 2024, doi: 10.1038/s41598-024-57990-3.
- [47] B. Scholkopf, *New support vector algorithms*, vol. 12. 2000, pp. 1083–1121.
- [48] O. Rokach, L., & Maimon, *Data mining and knowledge discovery handbook*. Springer New York, 2010.
- [49] J. R. Quinlan, "J. R. Quinlan," vol. 5, no. Quinlan 1993, 2006.
- [50] L. Breiman, *Random forests. Machine learning*. 2001.
- [51] Tin Kam Ho, "Random Decision Forests," *Proc. 3rd Int. Conf. Doc. Anal. Recognit.*, pp. 8–12, 1995, [Online]. Available: <http://ieeexplore.ieee.org/document/598994/>
- [52] Y. Yan, "Improve robustness of machine learning via efficient optimization and conformal prediction," *AI Mag.*, vol. 45, no. 2, pp. 270–279, Jun. 2024, doi: <https://doi.org/10.1002/aaai.12173>.
-

-
- [53] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, “CatBoost: unbiased boosting with categorical features,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf
- [54] A. V. Dorogush, “Tutorials CatBoost,” 2019, [Online]. Available: <https://catboost.ai/en/docs/concepts/tutorials>
- [55] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, in KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 2623–2631. doi: 10.1145/3292500.3330701.
- [56] G. Ke *et al.*, “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
- [57] L. Wu, “A meta-learning network method for few-shot multi-class classification problems with numerical data,” *Complex Intell. Syst.*, vol. 10, no. 2, pp. 2639–2652, 2024, doi: 10.1007/s40747-023-01281-3.
- [58] D. J. Hand and R. J. Till, “A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems,” *Mach. Learn.*, vol. 45, no. 2, pp. 171–186, 2001, doi: 10.1023/A:1010920819831.
- [59] R. M. Everson and J. E. Fieldsend, “Multi-class ROC analysis from a multi-objective optimisation perspective,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 918–927, 2006, doi: <https://doi.org/10.1016/j.patrec.2005.10.016>.
- [60] J. Davis and M. Goadrich, “The relationship between Precision-Recall and ROC curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, in ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 233–240. doi: 10.1145/1143844.1143874.
- [61] H. Han, J., Pei, J., & Tong, *Data mining: concepts and techniques*. Morgan kaufmann., 2022.
- [62] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009, doi: <https://doi.org/10.1016/j.ipm.2009.03.002>.
- [63] D. M. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,” *arXiv Prepr.*, 2020.
- [64] G. Fouché and L. Langit, “Introduction to Data Mining,” G. Fouché and L. Langit, Eds., Berkeley, CA: Apress, 2011, pp. 369–402. doi: 10.1007/978-1-4302-3325-1_14.
-

- [65] C. Goutte and E. Gaussier, “A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation BT - Advances in Information Retrieval,” D. E. Losada and J. M. Fernández-Luna, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 345–359.
- [66] R. Huang *et al.*, “A Review on Evaluation Metrics for Data Classification Evaluations,” *Med. Image Anal.*, vol. 80, no. 2, p. 102478, 2022.