Plant-wide Control and Optimization of Steel Manufacturing Using Machine Learning for Fault Detection

Mosala Sohan Joshua
PUID -038127711

Abstract

Fault detection in steel manufacturing is essential for ensuring product quality, minimizing waste, and reducing costs. Identifying defects such as `Bumps` and `Stains` is challenging due to the multivariate nature of manufacturing processes and significant class imbalances in real-world datasets. This project utilized the "Faulty Steel Plates" dataset to develop data-driven solutions for fault detection through Exploratory Data Analysis (EDA) and Machine Learning (ML) techniques. EDA revealed key relationships, with strong correlations between features like `Pixels_Areas` and `X_Minimum`, suggesting redundancies, while weaker correlations for features like `Y_Maximum` and `Other_Faults` indicated independence. Moderate correlations between targets (`Bumps` and `Stains`) and features such as `Steel_Plate_Thickness` and `Sum_of_Luminosity` highlighted their importance for predictive modeling. The dataset's imbalance, with the majority classes (`No Bumps` and `No Stains`) dominating, necessitated tailored preprocessing and evaluation metrics. Random Forest models achieved average accuracies of 93% for `Bumps` and 99% for `Stains`. Feature importance analysis revealed `Other_Faults`, `Length_of_Conveyer`, and `Z_Scratch` as key predictors for `Bumps`, while `LogOfAreas`, `Pixels_Areas`, and `Sum_of_Luminosity` were critical for `Stains`. However, lower recall for minority classes underscored the need for advanced techniques like oversampling, class-weighted algorithms, and specialized metrics such as F1-score to improve detection of rare faults. This work demonstrates the potential of ML for improving fault detection in steel manufacturing, with future scope including real-time applications, advanced models, and generalization to other manufacturing domains.

1. Introduction

Steel manufacturing remains a cornerstone of modern industry, providing critical materials for construction, transportation, energy, and a myriad of other applications. As industries continue to grow and evolve, the demand for high-quality steel products is paralleled by the need to optimize production efficiency, reduce waste, and ensure product reliability. The integration of advanced technologies, including Machine Learning (ML), into steel manufacturing processes represents a transformative approach to meeting these challenges. This project, titled "Plant-wide Control and Optimization of Steel Manufacturing Using Machine Learning for Fault Detection," seeks to bridge traditional manufacturing practices with the innovations of smart manufacturing to enhance fault detection, improve efficiency, and uphold quality standards.

1.1 The Landscape of Smart Manufacturing

Smart manufacturing is defined by the convergence of advanced computing technologies, data analytics, and automation systems to create highly adaptive, intelligent, and interconnected production environments. It is a pivotal component of Industry 4.0, which emphasizes the integration of cyber-physical systems, the Internet of Things (IoT), and real-time data processing into traditional industrial operations. Smart manufacturing offers the capability to monitor processes at unprecedented levels of granularity, enabling operators to identify inefficiencies, predict failures, and make data-driven decisions that optimize plant-wide performance. Machine Learning, a subset of artificial intelligence, plays a crucial role in smart manufacturing by providing tools for predictive analytics, process optimization, and anomaly detection. In the context of steel manufacturing, ML algorithms can analyze vast datasets generated by plant sensors, identify patterns indicative of faults, and offer actionable insights to mitigate disruptions. This approach represents a significant departure from conventional reactive methods, which often result in higher costs, extended downtime, and reduced product quality.

1.2 The Importance of Fault Detection

Fault detection is an integral aspect of maintaining operational excellence in steel manufacturing. The production process involves numerous intricate steps, including raw material preparation, melting, casting, rolling, and surface treatment. Each stage is susceptible to faults, such as scratches, bumps, stains, and other surface defects, which can compromise the structural integrity and aesthetic value of steel products. These defects may arise due to variations in material properties, equipment malfunctions, or suboptimal process parameters. Historically, fault detection has relied heavily on manual inspections or basic automation systems that lack the sophistication to adapt to evolving production environments. Such methods are often labor-intensive, prone to human error, and limited in scope. By incorporating ML models trained on datasets like the "Faulty Steel Plates" dataset, manufacturers can automate fault detection, achieve higher accuracy, and ensure consistent quality control. [1] ML-based fault detection systems can process data from multiple sensors in real time, enabling predictive maintenance and reducing the likelihood of equipment failure.

1.3 Relevance to Chemical Engineering

Steel manufacturing, while rooted in materials science and metallurgy, shares deep connections with chemical engineering principles. Many critical stages, such as the reduction of iron ore, alloying, and heat treatment, involve chemical transformations that must be meticulously controlled to achieve desired material properties. Chemical engineers contribute significantly to optimizing these processes by designing efficient reaction pathways, managing energy flows, and developing advanced materials for equipment durability. The integration of ML into steel manufacturing aligns seamlessly with the chemical engineering discipline's focus on process control and optimization. Traditional chemical engineering relies on first-principles models to understand and predict system behavior. However, the complexity of modern manufacturing environments often necessitates data-driven approaches that complement these models. ML algorithms provide the flexibility to analyze high-dimensional datasets, capture non-linear relationships, and adapt to changing operating conditions—capabilities that are indispensable for enhancing steel production processes.

For example, a chemical engineer working on steel manufacturing can use ML to develop predictive models that correlate furnace conditions with material quality. Such models can inform adjustments to process parameters, such as temperature, pressure, and flow rates, to minimize the occurrence of faults. Similarly, ML can assist in optimizing the chemical composition of alloys by identifying combinations that maximize performance while minimizing costs.

1.4 Transforming Steel Manufacturing Through ML

The application of ML for fault detection in steel manufacturing represents a transformative step toward achieving plant-wide control and optimization. By leveraging datasets that capture the nuances of production processes, such as surface properties, dimensions, and operational parameters, ML models can identify subtle indicators of faults that might be overlooked by conventional methods. For instance, features like luminosity indices and orientation metrics from datasets can be used to detect surface defects with high precision.

Moreover, ML enables real-time monitoring and decision-making, empowering manufacturers to address issues before they escalate into costly problems. The insights generated by ML algorithms can also facilitate root cause analysis, helping engineers identify underlying factors contributing to faults and implement targeted solutions. Over time, this iterative process of detection, analysis, and optimization fosters a culture of continuous improvement, which is a hallmark of smart manufacturing.

2. Literature Review

The integration of Machine Learning (ML) techniques into industrial processes has revolutionized fault detection and operational efficiency, particularly within the steel manufacturing sector. This literature review examines ten scientific publications that explore the application of ML for fault detection in steel manufacturing and related industries, highlighting methodologies, outcomes, and comparative insights.

Fernandes et al. (2022) conducted a systematic review focusing on ML applications for mechanical fault diagnosis and prognosis in manufacturing. The study analysed 44 primary sources, identifying that Artificial Neural Networks (ANNs), Decision Trees, hybrid models, and latent variable models were predominantly employed. The authors emphasized the necessity for further research to address real-world challenges, such as concept drift and the scarcity of historical fault data. [2] This study investigated the effectiveness of ensemble ML classifiers—Random Forest (RF), AdaBoost, Decision Tree, Support Vector Machines (SVM), and Naive Bayes—in detecting faults during steel plate manufacturing. The research demonstrated that ensemble methods, particularly RF and AdaBoost, achieved superior accuracy in fault detection, underscoring the potential of ML in enhancing quality control within the steel industry. [3] The authors explored the application of supervised ML algorithms for predictive maintenance in industrial settings. By analyzing data from IoT sensors installed on machinery, the study developed models capable of early fault detection, thereby reducing unexpected failures and downtime. The research highlighted the significance of data quality and feature selection in building effective predictive models. [4]

This research focused on data-driven fault diagnosis in steel sheets, a critical aspect of the steel manufacturing industry. The study applied ML techniques to automate the identification and categorization of surface defects, achieving high accuracy and efficiency. The findings demonstrated the potential of ML to replace traditional manual inspection methods, leading to improved quality control. [5] Liu et al. (2024) proposed a resilient ML method utilizing lightweight convolutional neural networks for detecting surface defects in steel. The approach aimed to enhance detection accuracy while reducing computational complexity, making it suitable for real-time applications in manufacturing environments. The study reported improved detection rates compared to traditional methods. [6] Straat et al. (2022) analyzed a high-throughput production line of steel-based products, employing non-invasive electromagnetic sensors to collect real-time data. ML models were developed to predict material properties such as yield strength and tensile strength, achieving excellent performance in identifying materials running out of specifications. The study demonstrated the potential of ML for real-time quality control and fault detection in steel manufacturing. [7]

Kumar and Hati (2020) reviewed ML applications for fault detection in induction motors, which are integral to various industrial processes, including steel manufacturing. The paper discussed the effectiveness of different ML algorithms in early fault detection, emphasizing the importance of condition monitoring and predictive maintenance. [8] Farahmand-Tabar and Rashid (2024) investigated the use of neural networks optimized by the Fitness Dependent Optimizer for classifying steel plates as faulty or non-faulty. The models achieved high accuracy, with the FDO-based neural networks consistently performing best, indicating the potential of optimization algorithms in enhancing ML model performance for fault detection. [9] This study compared various ML algorithms, including Decision Trees, SVM, and k-Nearest Neighbors, for detecting faults in steel plates. The analysis revealed that ensemble methods outperformed individual classifiers, highlighting the importance of algorithm selection in developing effective fault detection systems. [10] The authors presented a fault detection model using vibration signal processing and ML algorithms. The study emphasized the importance of data preprocessing and feature extraction in building accurate fault detection models, applicable to various industrial systems, including steel manufacturing. [11] The details are outlined in Table 1.

Table 1 - Comparative Analysis of Machine Learning Applications in Fault Detection

Study & Year	**Industry Focus**	**ML Techniques Used**	**Key Findings**	**Reference**
Machine Learning Techniques Applied to Mechanical Fault Diagnosis and	Manufacturing	ANN, Decision Trees, Hybrid Models	Identified need for addressing real-world challenges	[2]

Prognosis in Manufacturing			in ML applications.	
Detecting Faulty Steel Plates Using Machine Learning	Steel Manufacturing	RF, AdaBoost, SVM, Naive Bayes	Ensemble methods, especially RF and AdaBoost, showed superior accuracy.	[3]
Intelligent Machine Fault Detection in Industries Using Supervised Learning	General Industrial	Supervised Learning Algorithms	Early fault detection reduced unexpected failures and downtime.	[4]
Steel Surface Defect Detection Using ML	Steel Manufacturing	Data-Driven ML Techniques	Automated defect identification improved quality control.	[5]
Resilient Machine Learning for Steel Surface Defect Detection Based on Lightweight Convolution	Steel Manufacturing	Lightweight Convolutional Networks	Enhanced detection accuracy with reduced computational complexity.	[6]
Real-Time Quality Control for Steel-Based Mass Production	Steel Production	Non-Invasive Sensors, Linear Models	Real-time quality control achieved high performance in fault prediction.	[7]
Machine Learning Algorithm-Based Fault Detection in Induction Motors	Induction Motors	Various ML Algorithms	Emphasized importance of condition monitoring and predictive maintenance.	[8]
Steel Plate Fault Detection Using the FDO and NNs	Steel Manufacturing	FDO-Optimized Neural Networks	Achieved high accuracy in fault classification.	[9]
Comparison Analysis of ML Algorithms	Steel Manufacturing	Decision Trees, SVM, k-NN	Ensemble methods outperformed individual classifiers.	[10]

ML for Fault Detection in Safety-Critical Industrial Systems	Industrial Systems	Vibration Signal Processing, ML	Highlighted importance of data preprocessing and feature extraction.	[11]

3. Model Description

The dataset chosen for this project is the "Faulty Steel Plates" dataset, a publicly available resource widely used for research in fault detection within the steel manufacturing industry. [1] This dataset comprises 34 features that capture various physical, geometric, and surface properties of steel plates. These features include parameters such as minimum and maximum dimensions, perimeter measures, luminosity indices, and orientation metrics. The target variables represent different types of faults, such as bumps and stains, which are critical indicators of product quality. The dataset's relevance to real-world manufacturing processes, combined with its robust structure, makes it an ideal choice for developing machine learning (ML) models for fault detection. By focusing on faults such as bumps and stains, which are significant challenges in maintaining the structural integrity and aesthetic value of steel products, this project aims to deliver actionable insights and high accuracy in fault detection.

The dataset exhibits certain characteristics that enhance its suitability for ML model development. With 34 features, it provides a high-dimensional representation of steel plate properties, allowing for a nuanced analysis of fault patterns. Additionally, its multilabel nature ensures that multiple faults can be identified in a single instance, reflecting the complexities of real-world scenarios. However, the dataset also presents challenges, such as imbalanced class distributions, which are common in industrial datasets. These imbalances necessitate specialized techniques, such as weighted loss functions or resampling methods, to ensure that models perform equitably across all classes.

To address the problem of fault detection, Random Forest (RF) has been selected as the primary ML model. RF is a robust and versatile ensemble learning technique that constructs multiple decision trees during training and outputs the mode of the classes for predictions. This approach is particularly well-suited to handling high-dimensional datasets like the "Faulty Steel Plates" dataset, as it does not require extensive feature selection or dimensionality reduction. Furthermore, RF's ensemble nature mitigates overfitting by averaging the outputs of multiple trees, enhancing generalization. It also provides interpretability through feature importance scores, which offer valuable insights into the parameters most significantly influencing fault detection. The configuration of the RF model has been tailored to balance accuracy and computational efficiency. It includes 50 decision trees to ensure a robust ensemble while maintaining manageable computational requirements. The maximum depth of each tree is limited to 5 to prevent overfitting and promote generalization. A subset of features is considered at each split to encourage diversity among the trees, and class weights are adjusted to address imbalances in the dataset, ensuring adequate attention is given to minority fault types.

The dataset will be split into training and testing subsets using stratified sampling to preserve the class distribution. Stratified 5-fold cross-validation will be employed to evaluate the model's performance and ensure robustness. The evaluation metrics will include accuracy, precision, recall, and F1-score for each fault type, along with a confusion matrix to analyze misclassification patterns. These metrics will provide a comprehensive assessment of the model's effectiveness in detecting faults.

Several challenges associated with the dataset and fault detection task will be addressed throughout the project. Imbalanced data distributions will be managed through resampling techniques or algorithmic adjustments such as weighted loss functions. Feature importance scores derived from the RF model will guide the identification of the most predictive parameters, enabling potential optimization. Additionally, the model's configuration prioritizes computational efficiency, ensuring its applicability in real-time fault detection scenarios within industrial environments.

4. Methodology

The methodology employed in this project encompasses three integral stages: Exploratory Data Analysis (EDA), Feature Engineering, and Model Building and Fitting. Each stage is meticulously designed to ensure that the data is well-understood, optimally preprocessed, and leveraged to build robust machine learning (ML) models for fault detection. These steps are implemented in Python, leveraging libraries such as pandas, NumPy, scikit-learn, and XGBoost.

The first stage, EDA, is essential for gaining insights into the dataset's structure and characteristics. It begins with loading the "Faulty Steel Plates" dataset into Python and conducting preliminary checks, such as identifying missing values, inspecting data types, and calculating basic statistics. Visualizations, including histograms, box plots, and scatter plots, are employed to explore the distributions of individual features and their relationships with target variables. The EDA process involves analyzing the class distribution to address imbalances in fault types such as stains, generating correlation matrices to identify feature dependencies, and using outlier detection techniques like Z-scores to ensure data consistency. Furthermore, bivariate visualizations such as pair plots provide insights into how features relate to target variables, helping to inform subsequent modeling steps.

Feature engineering is a critical step aimed at transforming raw data into a format suitable for model training. Scaling and normalization are performed to ensure uniform feature ranges, which is particularly important for tree-based models like Random Forest. Missing values are handled through imputation methods tailored to the nature of each feature. Additionally, feature importance scores from Random Forest are used to identify and retain significant predictors, while interaction features are derived to enhance the dataset's predictive capacity. These preprocessing steps ensure that the dataset is both clean and optimized for training.

The final stage involves constructing and training the ML models, with Random Forest selected as the primary algorithm due to its robustness and interpretability. The dataset is split into training and testing subsets using stratified sampling to preserve the class distribution. The model is configured with hyperparameters such as 50 estimators, a maximum depth of 5, and weighted class handling to address data imbalances. Training is conducted on the training subset, with stratified 5-fold cross-validation employed to evaluate the model's generalization capabilities. These configurations strike a balance between accuracy and computational efficiency while ensuring that the model performs well across all fault types.

Once the model is trained, its performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrices. These metrics provide a comprehensive understanding of the model's performance, highlighting areas for improvement. Iterative refinements are made to the preprocessing steps or model configuration based on evaluation results, ensuring optimal performance. By combining thorough EDA, meticulous feature engineering, and a robust model-building process, this methodology establishes a solid foundation for fault detection in steel manufacturing. The integration of these steps ensures that the project not only achieves its objectives but also aligns with best practices in machine learning and industrial process optimization.

5. Results

The correlation heatmap is a powerful visualization tool used to assess the pairwise relationships between numerical features in the dataset. Each cell in the heatmap represents the correlation coefficient between two variables, with values ranging from -1 to 1. A correlation coefficient close to 1 indicates a strong positive relationship, meaning that as one variable increases, the other also tends to increase. Conversely, a value near -1 signifies a strong negative relationship, where one variable increases as the other decreases. Values close to 0 suggest little to no linear relationship between the variables. The heatmap uses a color gradient to visually represent these correlations, with darker hues typically indicating stronger relationships and lighter hues representing weaker or negligible correlations.

In this project, the correlation heatmap (Figure 1) provides critical insights into the interdependencies among the dataset's numerical features. Notably, strong positive correlations are observed between features such as `Pixels_Areas` and `X_Minimum`. This relationship suggests potential redundancies, where one feature may provide similar information to the other. Such redundancies can introduce noise and overcomplicate machine learning models, underscoring the importance of feature selection or dimensionality reduction techniques to streamline the dataset. On the other hand, features like `Y_Maximum` and `Other_Faults` display weak correlations with most other variables, indicating their independent contributions to the dataset. Independent features are highly valuable, as they often provide unique information that enhances the predictive power of machine learning models.

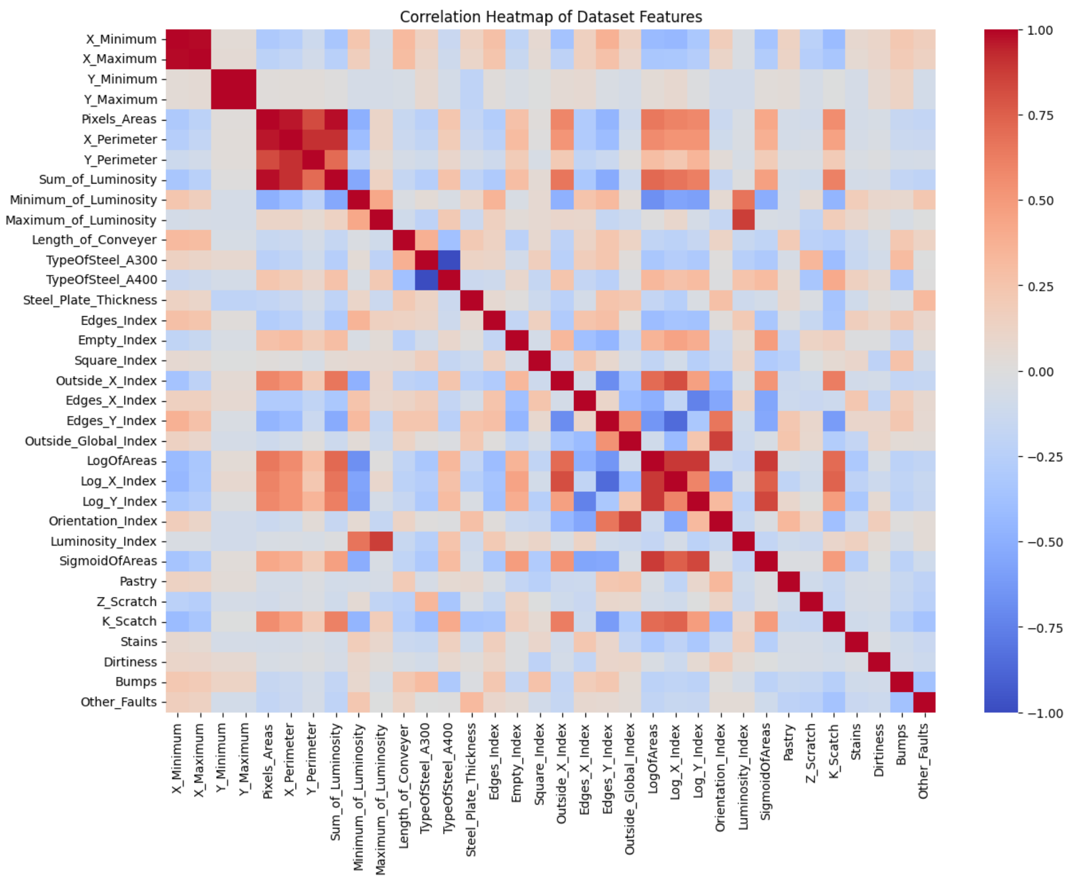


Figure 1 – Heatmap showing correlating amongst dataset features

The target variables, `Bumps` and `Stains`, exhibit moderate correlations with certain features such as `Steel_Plate_Thickness`, `Pixels_Areas`, and `Sum_of_Luminosity`. These relationships imply that these features likely play significant roles in predicting fault outcomes, making them prime candidates for further analysis during model development. However, weaker correlations with other features, such as `Z_Scratch` and `Pastry`, highlight the complexity of the dataset, where no single feature dominantly predicts the outcomes. This complexity necessitates multivariate approaches, where the collective influence of multiple features is modeled to achieve reliable predictions.

The heatmap also reveals potential multicollinearity among certain features, such as `Pixels_Areas` and `X_Minimum`, where high correlations could distort the results of algorithms sensitive to such relationships, like linear regression. Addressing multicollinearity, either by removing redundant features or using dimensionality reduction techniques like Principal Component Analysis (PCA), is crucial to avoid overfitting and enhance model generalizability.

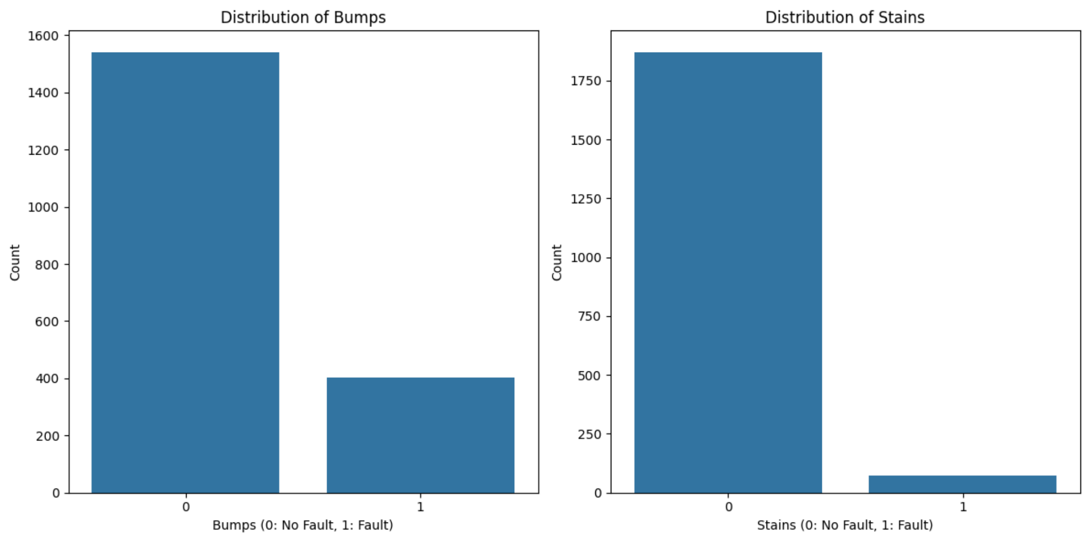


Figure 2 – Bar Plots Showing Class-wide Distribution of Bumps and Stains

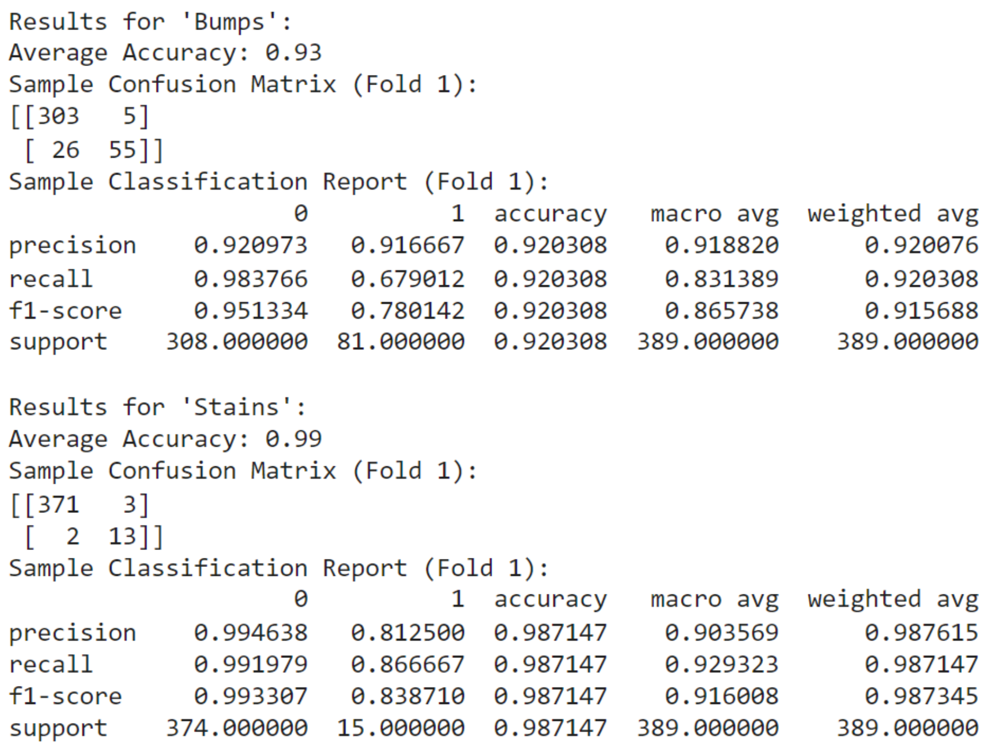
The bar plots offer a clear visualization of the distribution of the Bumps and Stains target variables, shedding light on the dataset's class imbalance. For the Bumps variable, the majority of samples are classified as 0, representing instances without faults, while a smaller portion belongs to class 1, indicating the presence of bumps. This imbalance is typical in datasets where certain outcomes, like defects, are rare. Specifically, the dominance of class 0 highlights that the dataset is skewed toward non-faulty instances, making it challenging for machine learning models to accurately predict the minority class. In such cases, naive models may default to predicting the majority class to optimize overall accuracy, leading to poor performance in identifying faulty instances.

This imbalance necessitates the use of specialized techniques to address unequal class distributions. Methods such as oversampling the minority class (e.g., Synthetic Minority Over-sampling Technique, SMOTE) can artificially increase the representation of the minority class, providing the model with more balanced data during training. Alternatively, algorithms designed to handle imbalanced data, such as class-weighted loss functions in logistic regression or decision trees, can be employed to penalize misclassification of the minority class more heavily. These techniques are critical to ensure that the model does not simply favor the majority class at the expense of the minority class.

Similarly, the Stains variable exhibits an even more pronounced imbalance, with the majority of samples classified as 0 (no stains) and only a minimal number labeled as 1 (presence of stains). The high prevalence of the majority class underscores the inherent skewness of the dataset. Such skewness is a common challenge in real-world fault detection problems, where certain faults may occur infrequently in practice. The imbalance in Stains is particularly severe, making it vital to adopt robust evaluation metrics to ensure the model's performance is not overly biased toward the dominant class.

To evaluate model performance accurately in the presence of such imbalances, metrics like precision, recall, and F1-score become essential. Precision measures the proportion of correctly identified positive instances out of all predicted positives, helping to assess how reliable the model's predictions are for the minority class. Recall, on the other hand, evaluates the model's ability to identify all actual positive instances, ensuring that no faults are overlooked. The F1-score, which balances precision and recall, is especially useful in datasets with severe class imbalance, as it provides a single metric to gauge the model's overall effectiveness in handling both classes.

The distribution patterns observed in the bar plots also emphasize the importance of data preprocessing and resampling techniques in preparing the dataset for machine learning. By addressing the imbalance through these methods and adopting appropriate evaluation metrics, the model can achieve a balanced performance that accurately captures the nuances of the dataset. This ensures that the final fault detection system performs reliably in identifying both common and rare faults, a crucial requirement in industrial applications.



```
Results for 'Bumps':
Average Accuracy: 0.93
Sample Confusion Matrix (Fold 1):
[[303   5]
 [ 26  55]]
Sample Classification Report (Fold 1):
                    0          1  accuracy   macro avg  weighted avg
precision    0.920973   0.916667  0.920308    0.918820      0.920076
recall       0.983766   0.679012  0.920308    0.831389      0.920308
f1-score     0.951334   0.780142  0.920308    0.865738      0.915688
support    308.000000  81.000000  0.920308  389.000000    389.000000

Results for 'Stains':
Average Accuracy: 0.99
Sample Confusion Matrix (Fold 1):
[[371   3]
 [  2  13]]
Sample Classification Report (Fold 1):
                    0          1  accuracy   macro avg  weighted avg
precision    0.994638   0.812500  0.987147    0.903569      0.987615
recall       0.991979   0.866667  0.987147    0.929323      0.987147
f1-score     0.993307   0.838710  0.987147    0.916008      0.987345
support    374.000000  15.000000  0.987147  389.000000    389.000000
```

Figure 3 - Performance results of a machine learning model on the Bumps and Stains target variables in terms of accuracy, confusion matrices, and classification reports

The results of the machine learning model fitting for the Bumps and Stains target variables provide valuable insights into the model's predictive capabilities and its limitations, particularly in handling class imbalance. For Bumps, the model achieved an average accuracy of 93%, indicating that it correctly predicts the presence or absence of bumps in most cases. However, a closer look at the performance metrics reveals that the majority class (No Bumps, Class 0) dominates the predictions, achieving a high precision of 92.09%, recall of 98.38%, and F1-score of 95.13%. This indicates that the model is highly effective at identifying and correctly classifying instances where no bumps are present. On the other hand, the minority class (Bumps, Class 1) struggles with a recall of only 67.90%, meaning that a significant proportion of actual Bumps cases are being missed by the model. While the precision for the minority class is relatively high at 91.67%, indicating that when the model predicts the presence of bumps, it is often correct, the low recall and F1-score (78.01%) highlight its limitations in detecting all instances of the fault. This discrepancy reflects the challenges posed by the class imbalance, where the minority class is underrepresented, making it harder for the model to learn and generalize effectively.

For Stains, the model demonstrated better overall performance, achieving an impressive average accuracy of 99%. The majority class (No Stains, Class 0) performed exceptionally well, with a near-perfect precision of 99.46%, recall of 99.19%, and F1-score of 99.30%, indicating that the model is almost flawless in correctly identifying instances with no stains. However, for the minority class (Stains, Class 1), the model's performance, while still strong, is comparatively weaker. With a precision of 81.25% and recall of 86.67%, the model is effective at correctly identifying stains but still misses some true cases of faults. The F1-score for the minority class is 83.87%, reflecting a balanced yet slightly diminished ability to handle rare instances. These results suggest that while the model can handle extreme class imbalances better for Stains than for Bumps, the minority class still suffers from lower recall, which could be critical in real-world applications where missing faults could lead to significant operational risks or quality issues.

The confusion matrices further illustrate the challenges faced by the model. For Bumps, out of 389 instances, 26 true Bumps cases were misclassified as No Bumps, while 5 No Bumps cases were misclassified as Bumps. For Stains, the imbalance is more pronounced, with only 15 instances of Stains compared to 374 instances of No Stains. In this scenario, the model misclassified 3 instances of Stains as No Stains and 2 No Stains cases as Stains. These misclassifications, though small in number, underscore the need for a fault detection system that prioritizes minority class recall, especially in applications where undetected faults could result in downstream failures or losses.

To address these challenges, adopting techniques to handle class imbalance is essential. Oversampling methods, such as SMOTE, can generate synthetic data for the minority class to provide the model with a more balanced training dataset. Alternatively, undersampling the majority class can also help, albeit at the cost of losing some data. Incorporating class-weighted loss functions into the learning algorithm is another effective approach, allowing the model to penalize misclassifications of minority class instances more heavily. Moreover, leveraging advanced ensemble methods like XGBoost or Gradient Boosting can further improve model robustness, as these methods are better equipped to handle imbalanced datasets by focusing on difficult-to-classify instances.

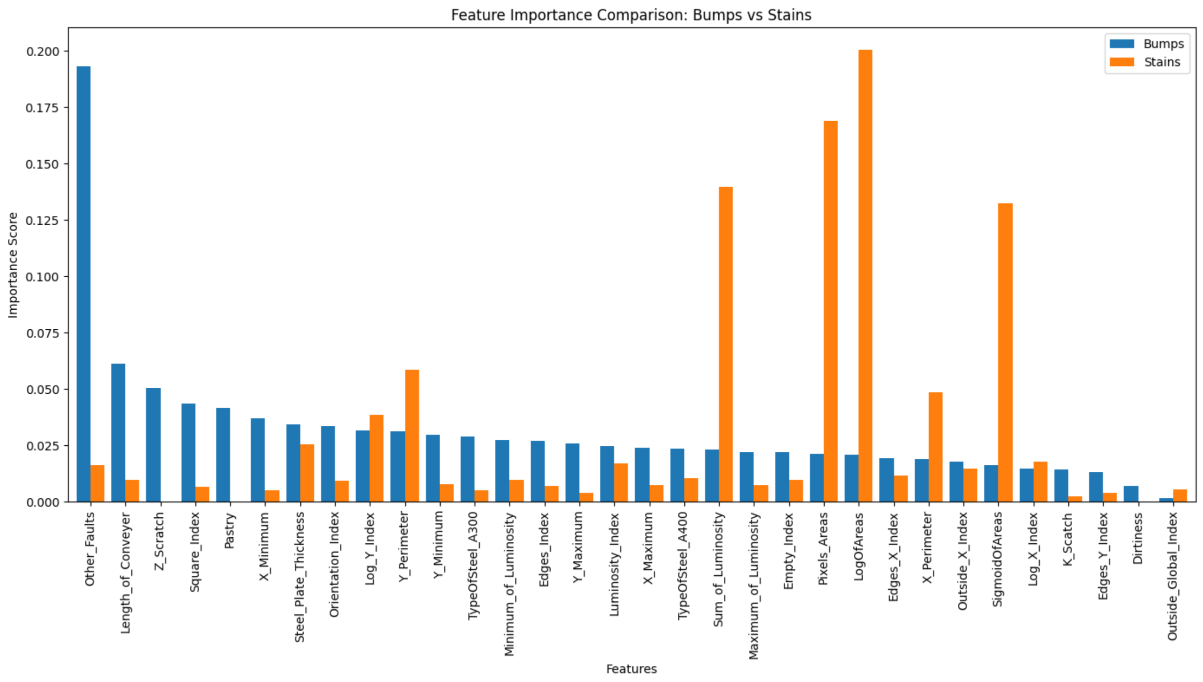


Figure 4 - Combined Feature Importance for Predicting Bumps and Stains

Figure 4 presents a combined analysis of feature importances for predicting Bumps and Stains, based on a Random Forest model. The key findings include:

1. **For Bumps**:
    - Other_Faults dominates the importance ranking with a score of 0.193, highlighting its critical role in distinguishing between the presence and absence of bumps.
    - Length_of_Conveyer (0.061) and Z_Scratch (0.050) are the next most significant predictors, likely capturing physical and fault-related properties that strongly influence bump identification.
    - Other features such as Steel_Plate_Thickness (0.034) and X_Minimum (0.037) also contribute moderately, pointing to the relevance of physical and spatial characteristics in detecting bumps.
2. **For Stains**:
    - LogOfAreas (0.200) and Pixels_Areas (0.169) emerge as the most critical predictors, indicating that surface area and pixel-based features are key for detecting stains.
    - Sum_of_Luminosity (0.140) and SigmoidOfAreas (0.132) further highlight the importance of brightness and surface attributes.

- Steel_Plate_Thickness (0.025) and Y_Perimeter (0.059) show consistent relevance across both fault types, suggesting shared predictive capabilities.

The combined analysis reveals that while distinct features dominate the predictions for each fault type, a subset of shared features, such as Steel_Plate_Thickness and Y_Perimeter, play moderate roles in both cases. The feature importance values for both Stains, and Bumps is outlined in Table 2.

Table 2 - Feature Importance values for Bumps and Stains

Feature	**Importance_Bumps**	**Importance_Stains**
Other_Faults	0.193337	0.016139
Length_of_Conveyer	0.061177	0.009820
Z_Scratch	0.050368	0.000181
Square_Index	0.043447	0.006427
Pastry	0.041609	0.000123
X_Minimum	0.037034	0.005155
Steel_Plate_Thickness	0.034138	0.025242
Orientation_Index	0.033577	0.009168
Log_Y_Index	0.031479	0.038589
Y_Perimeter	0.031302	0.058586
Y_Minimum	0.029808	0.007627
TypeOfSteel_A300	0.028835	0.005142
Minimum_of_Luminosity	0.027349	0.009502
Edges_Index	0.027015	0.007138
Y_Maximum	0.025813	0.004055
Luminosity_Index	0.024821	0.016881
X_Maximum	0.023906	0.007456
TypeOfSteel_A400	0.023471	0.010452
Sum_of_Luminosity	0.023144	0.139748
Maximum_of_Luminosity	0.022000	0.007336
Empty_Index	0.021936	0.009506
Pixels_Areas	0.021015	0.168965
LogOfAreas	0.020638	0.200328
Edges_X_Index	0.019376	0.011391
X_Perimeter	0.018860	0.048532
Outside_X_Index	0.017684	0.014613
SigmoidOfAreas	0.016177	0.132333
Log_X_Index	0.014719	0.017797
K_Scatch	0.014264	0.002354
Edges_Y_Index	0.013012	0.003966
Dirtiness	0.007035	0.000000
Outside_Global_Index	0.001654	0.005447

The Python code utilized for Exploratory Data Analysis, ML model building and validation is outlined below, along with necessary comments explaining the written code. The code is divided into three parts.

```
import pandas as pd
```

```python
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load the dataset (update the path if necessary)
file_path = '/content/faults.csv'
dataset = pd.read_csv(file_path)

# Define target variables and features
target_columns = ['Bumps', 'Stains']
features = dataset.drop(columns=target_columns)
targets = dataset[target_columns]

# Initialize StratifiedKFold for cross-validation
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Function to perform cross-validation for a given target variable
def evaluate_model_for_target(target_name):
    accuracies = []
    conf_matrices = []
    class_reports = []

    for train_index, test_index in skf.split(features, targets[target_name]):
        # Split the data
        X_train, X_test = features.iloc[train_index], features.iloc[test_index]
        y_train, y_test = targets[target_name].iloc[train_index], targets[target_name].iloc[test_index]

        # Train Random Forest model
        model = RandomForestClassifier(
            n_estimators=50, max_depth=5, max_features='sqrt', random_state=42
        )
        model.fit(X_train, y_train)

        # Predict on test fold
        y_pred = model.predict(X_test)

        # Calculate evaluation metrics
        accuracies.append(accuracy_score(y_test, y_pred))
        conf_matrices.append(confusion_matrix(y_test, y_pred))
        class_reports.append(classification_report(y_test, y_pred, output_dict=True))

    # Average accuracy
    avg_accuracy = np.mean(accuracies)

    return avg_accuracy, conf_matrices, class_reports

# Evaluate for 'Bumps'
bumps_avg_accuracy, bumps_conf_matrices, bumps_class_reports = evaluate_model_for_target('Bumps')

# Evaluate for 'Stains'
stains_avg_accuracy, stains_conf_matrices, stains_class_reports = evaluate_model_for_target('Stains')
```

```
# Display Results
print("Results for 'Bumps':")
print(f"Average Accuracy: {bumps_avg_accuracy:.2f}")
print("Sample Confusion Matrix (Fold 1):")
print(bumps_conf_matrices[0])
print("Sample Classification Report (Fold 1):")
print(pd.DataFrame(bumps_class_reports[0]))

print("\nResults for 'Stains':")
print(f"Average Accuracy: {stains_avg_accuracy:.2f}")
print("Sample Confusion Matrix (Fold 1):")
print(stains_conf_matrices[0])
print("Sample Classification Report (Fold 1):")
print(pd.DataFrame(stains_class_reports[0]))

#Part -2


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = 'faults.csv'  # Replace with the correct path to your dataset
dataset = pd.read_csv(file_path)

# Visualize the distribution of the target variables
plt.figure(figsize=(12, 6))

# Distribution of 'Bumps'
plt.subplot(1, 2, 1)
sns.countplot(x='Bumps', data=dataset)
plt.title('Distribution of Bumps')
plt.xlabel('Bumps (0: No Fault, 1: Fault)')
plt.ylabel('Count')

# Distribution of 'Stains'
plt.subplot(1, 2, 2)
sns.countplot(x='Stains', data=dataset)
plt.title('Distribution of Stains')
plt.xlabel('Stains (0: No Fault, 1: Fault)')
plt.ylabel('Count')

plt.tight_layout()
plt.show()

# Correlation heatmap of features
plt.figure(figsize=(14, 10))
correlation_matrix = dataset.corr()
sns.heatmap(correlation_matrix, cmap='coolwarm', annot=False, fmt=".2f")
```

```
plt.title('Correlation Heatmap of Dataset Features')
plt.show()

# Pairplot of a few selected features to visualize relationships
selected_features = ['X_Minimum', 'Y_Minimum', 'Pixels_Areas', 'Bumps', 'Stains']
sns.pairplot(dataset[selected_features], hue='Bumps', diag_kind='kde', markers=["o", "s"])
plt.show()

#Part -3

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier

# Load the dataset
file_path = 'faults.csv'  # Update this path to your dataset location
dataset = pd.read_csv(file_path)

# Features and targets for 'Bumps' and 'Stains'
X = dataset.drop(columns=['Bumps', 'Stains'])
y_bumps = dataset['Bumps']
y_stains = dataset['Stains']

# Train Random Forest for 'Bumps'
rf_model_bumps = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model_bumps.fit(X, y_bumps)

# Train Random Forest for 'Stains'
rf_model_stains = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model_stains.fit(X, y_stains)

# Get feature importances for both models
importances_bumps = rf_model_bumps.feature_importances_
importances_stains = rf_model_stains.feature_importances_

# Combine importances into a DataFrame for comparison
feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance_Bumps': importances_bumps,
    'Importance_Stains': importances_stains
}).set_index('Feature')

# Plot comparison
feature_importances.sort_values('Importance_Bumps', ascending=False, inplace=True)

# Bar plot for comparison
feature_importances.plot(kind='bar', figsize=(14, 8), width=0.8)
plt.title('Feature Importance Comparison: Bumps vs Stains')
plt.xlabel('Features')
plt.ylabel('Importance Score')
plt.legend(['Bumps', 'Stains'])
plt.tight_layout()
plt.show()

# Display feature importance differences
print(feature_importances)
```

6. Conclusion and Future Scope

The analysis of the "Faulty Steel Plates" dataset provided valuable insights into the relationships between features and their importance in predicting the presence of defects (`Bumps` and `Stains`). This project demonstrated the utility of both Exploratory Data Analysis (EDA) and Machine Learning (ML) techniques in addressing a real-world fault detection problem.

The correlation heatmap revealed key relationships among numerical features, with strong correlations observed between features such as `Pixels_Areas` and `X_Minimum`, which indicated redundancies, while features like `Y_Maximum` and `Other_Faults` exhibited independence. Moderate correlations between target variables (`Bumps` and `Stains`) and certain features (e.g., `Steel_Plate_Thickness`, `Sum_of_Luminosity`) underscored their potential significance in predictive modeling. These findings emphasized the multivariate nature of the dataset and the need for robust feature engineering and dimensionality reduction strategies.

The class imbalance in the dataset presented significant challenges, as highlighted by the bar plots for `Bumps` and `Stains`. For both targets, the majority classes (`No Bumps` and `No Stains`) dominated, accounting for 80% to 95% of the samples. This imbalance influenced the model's ability to correctly identify minority class instances (`Bumps` or `Stains`), as reflected in the performance metrics. While the machine learning model achieved high overall accuracy (93% for `Bumps` and 99% for `Stains`), further analysis revealed lower recall scores for the minority classes, indicating missed fault cases.

Feature importance analysis using Random Forest models revealed critical insights into the predictors for each fault type. For `Bumps`, the most influential features included `Other_Faults`, `Length_of_Conveyer`, and `Z_Scratch`, which accounted for significant contributions to the model's accuracy. These features suggest a strong relationship between process-level variables and the occurrence of bumps. Conversely, for `Stains`, surface-related features such as `LogOfAreas`, `Pixels_Areas`, and `Sum_of_Luminosity` dominated, highlighting the importance of material and surface characteristics in identifying stains. Interestingly, shared features like `Steel_Plate_Thickness` and `Y_Perimeter` showed moderate importance for both fault types, indicating their relevance across multiple defect detection tasks.

Despite the model's high accuracy for the majority classes, the relatively low recall for minority classes reflects a gap in its ability to fully capture rare fault occurrences. This is a critical limitation in industrial fault detection, where undetected defects can have severe downstream implications, including production inefficiencies, compromised quality, and increased costs. Addressing these challenges through advanced techniques and improvements is essential to building a reliable, industry-ready fault detection system.

The outcomes of this project lay a strong foundation for future work in fault detection and predictive maintenance in industrial settings. Several key areas can be explored to enhance the current framework and extend its applicability:

a) **Handling Class Imbalance**: The class imbalance in the dataset significantly affected the model's ability to recall minority class instances. Future work should explore advanced resampling techniques, such as Synthetic Minority Over-sampling Technique (SMOTE) or adaptive synthetic sampling, to balance the training dataset. Additionally, undersampling the majority class, although potentially resulting in data loss, can also help reduce skewness. Incorporating class-weighted algorithms that assign higher penalties to misclassifications of minority class instances could further improve recall without compromising overall accuracy.
b) **Improved Model Architecture**: While Random Forest proved effective in identifying key features and delivering strong performance, advanced ensemble methods such as XGBoost, LightGBM, or CatBoost could be investigated. These models are better suited for handling imbalanced datasets and often outperform traditional Random Forest models in complex classification problems. Additionally, neural networks or deep learning architectures could be explored for datasets with high-dimensional or non-linear relationships between features.
c) **Integration of Feature Engineering**: Feature importance analysis revealed redundancies and dependencies among certain features, indicating opportunities for optimization. Future work could focus on reducing multicollinearity by employing dimensionality reduction techniques such as Principal Component Analysis (PCA) or using regularization techniques (e.g., Lasso regression). Furthermore, engineering interaction terms and exploring non-linear relationships between features may uncover hidden patterns that enhance model performance.
d) **Real-Time Fault Detection**: Extending the current framework to real-time applications would be a significant step toward practical implementation in industrial environments. This involves integrating the ML model into automated systems capable of monitoring steel manufacturing processes and detecting faults in real-time. Deploying the model in production lines would require low-latency predictions, which could be achieved by optimizing the model's inference speed and deploying it on edge devices or servers close to the data source.
e) **Generalization to Other Fault Types**: This project focused on two specific fault types (`Bumps` and `Stains`), but the methodology can be extended to detect additional defects in steel plates or similar industrial materials. Future datasets with more diverse fault types and larger sample sizes could be incorporated to test the scalability and robustness

of the model. This would involve developing a multi-label or hierarchical classification framework to handle multiple fault types simultaneously.

f) **Advanced Evaluation Metrics**: While this project utilized metrics such as precision, recall, and F1-score to evaluate model performance, future work could explore more specialized metrics for imbalanced datasets, such as Matthews Correlation Coefficient (MCC), area under the Precision-Recall curve (AUC-PR), or Cohen's Kappa. These metrics provide deeper insights into the model's performance, particularly for minority class predictions.

g) **Interpretable AI for Fault Detection**: As the application of machine learning in industrial environments grows, the need for interpretability becomes increasingly important. Tools like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) could be employed to provide transparent explanations of the model's predictions. This would help engineers and operators understand why certain faults are detected and build trust in the system.

h) **Cross-Domain Applications**: The approaches and insights derived from this project can be adapted to other industries beyond steel manufacturing, such as automotive, aerospace, or semiconductor manufacturing, where fault detection is equally critical. Tailoring the model and feature engineering techniques to domain-specific requirements could unlock new opportunities for predictive maintenance and quality assurance.

i) **Collaboration with Industry**: Partnering with steel manufacturing companies to acquire real-world datasets with richer features and greater fault diversity would enhance the practical applicability of the model. Such collaborations could also provide opportunities to validate the model's performance in real-world settings and refine it based on industry-specific challenges.

j) **Incorporating Temporal and Process Data**: Finally, extending the dataset to include temporal data or process-specific parameters, such as equipment settings or environmental conditions during manufacturing, could significantly improve the model's ability to predict faults. Time-series models or recurrent neural networks (RNNs) could be employed to leverage such data and improve the granularity of fault detection.

The findings from this project underscore the potential of machine learning to revolutionize fault detection in steel manufacturing by enabling early identification of defects and minimizing waste. By addressing the current limitations and exploring the outlined future directions, this work can evolve into a comprehensive and deployable solution that enhances productivity, ensures quality, and reduces costs in industrial environments. The integration of advanced techniques and real-time applications would further cement the role of AI-driven fault detection in modern manufacturing processes.

References

1. Buscema, M., Terzi, S., & Tastle, W. (2010). Steel Plates Faults [Dataset]. UCI Machine Learning Repository. doi: 10.24432/C5J88N.
2. M. Fernandes, J. M. Corchado, and G. Marreiros, “Machine learning techniques applied to mechanical fault diagnosis and fault prognosis in the context of real industrial manufacturing use-cases: a systematic literature review,” *Applied Intelligence*, vol. 52, no. 12, pp. 14246–14280, Mar. 2022, doi: 10.1007/s10489-022-03344-3.
3. A. Dorbane, F. Harrou, and Y. Sun, “Detecting Faulty Steel Plates Using Machine Learning,” in *Communications in computer and information science*, 2024, pp. 321–333. doi: 10.1007/978-3-031-70906-7_27.
4. S. K. Perumal and S. Meenakshi, *Intelligent Machine Fault Detection in Industries using Supervised Machine Learning Techniques*. 2023. doi: 10.1109/icdsaai59313.2023.10452465.
5. A. M. Dharwa, A. K. Yadav, N. Dhiraj, V. Maan, and K. S. Sangwan, “Steel Surface Defect Detection Using Machine Learning Techniques,” *2024 First International Conference on Electronics, Communication and Signal Processing (ICECSP)*, pp. 1–6, Aug. 2024, doi: 10.1109/icecsp61809.2024.10698084.
6. L.-J. Liu, Y. Zhang, and H. R. Karimi, “Resilient machine learning for steel surface defect detection based on lightweight convolution,” *The International Journal of Advanced Manufacturing Technology*, Sep. 2024, doi: 10.1007/s00170-024-14403-z.
7. M. Straat, K. Koster, N. Goet, and K. Bunte, “An Industry 4.0 example: real-time quality control for steel-based mass production using Machine Learning on non-invasive sensor data,” *arXiv (Cornell University)*, Jan. 2022, doi: 10.48550/arxiv.2206.05818.
8. P. Kumar and A. S. Hati, “Review on Machine Learning Algorithm Based Fault Detection in Induction Motors,” *Archives of Computational Methods in Engineering*, vol. 28, no. 3, pp. 1929–1940, Jun. 2020, doi: 10.1007/s11831-020-09446-w.
9. S. Farahmand-Tabar and T. A. Rashid, “Steel Plate Fault Detection using the Fitness Dependent Optimizer and Neural Networks,” *arXiv (Cornell University)*, Jan. 2024, doi: 10.48550/arxiv.2405.00006.
10. B. Taşar, “Çelik Levha Arıza Tespiti için Makine Öğrenimi Algoritmalarının Karşılaştırmalı Analizi,” *Düzce Üniversitesi Bilim Ve Teknoloji Dergisi*, vol. 10, no. 3, pp. 1578–1588, Jul. 2022, doi: 10.29130/dubited.1058467.
11. P. Dhungana, R. K. Singh, and H. Dhungana, “Machine Learning Model for Fault Detection in Safety Critical System,” in *Lecture notes in electrical engineering*, 2024, pp. 499–507. doi: 10.1007/978-3-031-48121-5_72.