Failure analysis of a milling machine with synthetic data in a real-time architecture

Ítalo Vinícius Pereira Guimarães, Universidade de Brasília

***Abstract*—This study presents a fault detection system for milling machines, combining advanced machine learning models and a scalable Lambda Architecture for real-time data processing. Using the AI4I 2020 Predictive Maintenance Dataset, models like XGBoost and CatBoost were trained to predict key fault types, achieving high accuracy for Machine Failure, Heat Dissipation Failure (HDF), Power Failure (PWF), and Overstrain Failure (OSF). Real-time data processing integrates Apache Kafka, Spark, and Elasticsearch, enabling fault prediction and visualization through Kibana.**

While the system excels in detecting major faults, challenges with class imbalance limited performance for rare failures like Tool Wear Failure (TWF) and Random Failure (RNF), which were excluded from deployment. Future work will address these limitations, enhance scalability, and improve interpretability, solidifying the system's role in predictive maintenance for efficient manufacturing.

***Index Terms*—Predictive Maintenance, Fault Detection, Milling Machines, Machine Learning, XGBoost, CatBoost, Lambda Architecture, Real-Time, Data Processing, Apache Kafka, Apache Spark, Elasticsearch, Industrial IoT (IIoT), Class Imbalance, Manufacturing Efficiency**

I. Introduction

The modern manufacturing industry heavily relies on the continuous and efficient operation of complex machinery. Among these, milling machines play a crucial role in various production processes. However, the natural wear and the various mechanical stresses to which these machines are subjected during operation make them prone to different types of failures. Early detection of these failures is essential to avoid unexpected production stoppages, reduce maintenance costs, and ensure operational safety.

Traditionally, the maintenance of industrial machines is based on manual inspections and fixed preventive maintenance schedules. Although these methods are widely used, they present significant limitations. Traditional methods often fail to identify subtle anomalies that may indicate imminent failures, resulting in late or unnecessary interventions. Moreover, such practices can lead to operational inefficiencies and increased costs since they do not fully leverage the available data produced by the machines' sensors.

With the advancement of Industry 4.0 and the increasing digitalization of industrial processes, there is a growing demand for predictive maintenance systems that use real-time data and advanced analytical techniques to predict failures before they occur. Predictive maintenance stands out as an essential strategy to mitigate risks, optimize operational efficiency, and prolong equipment lifespan.

II. Objective

This project aims to develop a robust and scalable fault detection system for milling machines by leveraging the capabilities of machine learning and modern data processing architectures. Using the "AI4I 2020 Predictive Maintenance Dataset"[1] a synthetic dataset containing sensor readings from milling machines, the system employs advanced machine learning models, such as XGBoost and CatBoost. These models are designed to identify complex patterns and relationships in the data that are not easily detectable by conventional methods.

Furthermore, to manage the large volume and high velocity of real-time data generated by milling machines, the project adopts a Lambda Architecture, as described in "Big Data: Principles and Best Practices of Scalable Realtime Data Systems" by Nathan Marz[2], is a robust framework designed to process and analyze large-scale data systems in a fault-tolerant and scalable manner. This architecture is particularly effective in scenarios where real-time processing needs to coexist with batch analytics to handle high data velocity and volume while ensuring data consistency. This architecture integrates batch and real-time (streaming) data processing to facilitate efficient real-time decision-making while maintaining a comprehensive historical dataset for in-depth analysis. Key technologies employed include Apache Kafka for data ingestion, Apache Spark for distributed processing, Elasticsearch for data storage and indexing, and Kibana for data visualization. These technologies, orchestrated using Docker, provide a scalable and adaptable solution to meet the dynamic needs of modern manufacturing environments.

III. Literature review

The integration of predictive maintenance strategies in the manufacturing industry, particularly for milling machines, has gained significant attention with the advent of Industry 4.0. This approach leverages real-time data and advanced machine learning techniques to predict potential failures, thereby optimizing operational efficiency and reducing maintenance costs.

Predictive maintenance (PdM) is a proactive approach that uses data-driven techniques to forecast equipment failures before they occur. This method contrasts with traditional maintenance strategies, such as reactive maintenance, which addresses post-failure issues, and preventive maintenance, which relies on scheduled inspections and interventions[3]. The limitations of traditional methods, including their inability to detect subtle anomalies and the potential for unnecessary maintenance activities, have driven the shift towards PdM[4].

Machine learning (ML) plays a pivotal role in PdM by analyzing sensor data to identify patterns indicative of machine health. Various ML models, including Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), and Random Forests, have been employed to improve the accuracy of fault detection and prognostics (Samatas et al., 2020). These models are particularly effective in handling complex and high-dimensional data generated by industrial machinery, such as milling machines, which are prone to wear and mechanical stresses[5].

The implementation of PdM in milling machines involves the use of IoT devices and sensors to continuously monitor machine conditions. Commonly used sensors include those measuring vibration, temperature, and acoustic signals, which provide critical information on the operating state of machinery[5]. The data collected from these sensors are processed using advanced data architectures, such as the Lambda Architecture, which combines batch and real-time processing to manage the large volumes of generated data[4].

Recent studies have highlighted the effectiveness of ML models like XGBoost and CatBoost in identifying complex patterns in sensor data that are not easily detectable by conventional methods[5]. These models, when integrated with technologies such as Apache Kafka for data ingestion and Apache Spark for distributed processing, provide a robust framework for real-time fault detection and predictive maintenance[4].

Despite advances in PdM, several challenges remain, including the need for high-quality labeled data for training ML models and the integration of these systems into existing industrial processes. Future research should focus on developing more efficient data labeling techniques and exploring the use of explainable AI to improve the interpretability of ML models in industrial settings[6].

In conclusion, the development of a scalable fault detection system for milling machines using ML and modern data processing architectures represents a significant advancement in the field of predictive maintenance. Using real-time data and advanced analytics, such systems can significantly enhance the reliability and efficiency of industrial operations.

IV. Related works

The advancement of predictive maintenance in industrial settings, particularly for machinery such as milling systems, has seen a substantial evolution through the application of machine learning and IoT technologies. Several studies highlight the effectiveness of these techniques in addressing the challenges of fault detection and system efficiency.

Mansouri and Vadera (2022)[7] explored the integration of IoT sensors with deep learning models to predict faults in industrial machinery. Their study demonstrated the efficacy of recurrent neural networks in univariate time series analysis, focusing on vibration sensors. The authors also introduced the Gumbel-Sigmoid eXplanator (GSX), a model-agnostic approach to enhance interpretability, achieving a balance between feature reduction and prediction accuracy.

Chevtchenko et al. (2023)[5] conducted a systematic mapping study on anomaly detection (AD) in industrial machinery using IoT ecosystems and machine learning algorithms. Their review emphasized the limitations of the existing literature, which mainly focus on network-related anomalies, neglecting industrial applications. By analyzing 84 studies, they identified the common algorithms and preprocessing techniques used in AD, providing critical information on sensor data utilization and predictive modeling.

The utility of distributed processing frameworks, such as Apache Spark, has also been validated in industrial fault diagnosis. Imani et al. (2017)[8] implemented a scalable platform for gearbox fault detection in wind farms, integrating feature extraction algorithms with Spark for real-time processing of vibration signals. This approach achieved an accuracy of 98.93% while effectively handling the computational demands of large-scale data streams.

From an architectural perspective, Gribaudo et al. (2018)[9] highlighted the benefits of the Lambda Architecture for managing real-time and batch processing in big data applications. Their study demonstrated how performance modeling could guide the resource allocation process, optimizing the deployment of cloud-based systems. Similarly, Patel (2019)[10] addressed the challenges of organizing and processing vast enterprise data lakes, emphasizing scalable data models and distributed computation frameworks.

In the context of predictive maintenance, explainable AI has garnered significant attention. Matzka (2020)[11] introduced a synthetic dataset for predictive maintenance and evaluated the explanatory performance of various models. The study underscored the importance of transparency in industrial AI systems, particularly for decision-making in fault detection.

These works collectively underscore the potential of combining machine learning, IoT, and distributed architectures to enhance fault detection and predictive maintenance in industrial settings. However, challenges such as data imbalance, model interpretability, and integration with legacy systems remain critical areas for further investigation.

V. Methodology

The initial phase of this project focused on designing, developing, and validating the data processing architecture. The primary objective was to ensure that the data flows seamlessly through the components and that the system correctly processes and visualizes the information. The implemented architecture, following the Lambda paradigm, integrates batch and real-time processing to handle the high volume and velocity of data from milling machines.

The full implementation and source code are available in the GitHub[12].

A. Lambda Architecture

The Lambda Architecture implemented for this project is designed to manage and process large-scale real-time and historical data streams efficiently. The system comprises the following components:

- **Data Generator**: Simulates real-time data streams derived from the AI4I 2020 Predictive Maintenance Dataset.
- **Apache Kafka (Message Broker)**: Acts as a fault-tolerant intermediary for real-time data ingestion, ensuring reliable message delivery.
- **Spark Streaming (Speed Layer)**: Processes real-time data streams, applying transformations and feeding the prediction models for instantaneous fault detection.
- **Serving Layer (Elasticsearch)**: Combines outputs from the speed and batch layers to provide a unified, queryable data repository, which can be used as an Application Programming Interface (API) too.
- **Visualization (Kibana)**: Offers an intuitive interface to visualize key metrics, trends, and real-time monitoring of milling machine operations.

The integration of these components ensures the scalability, reliability, and efficiency of the fault detection system.

B. Architecture Diagram

The architecture diagram is shown in Figure 1, which visually represents the flow of data between the components. Everything is being supported by Docker, where each module runs on a different image and divides the network for internal communication, the kibana part is for future visualization and consumption of the enriched data, which is labeled using Spark, which has access to the trained models.

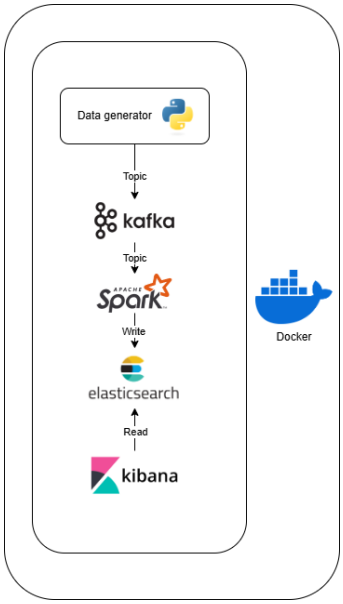

Fig. 1. Lambda Architecture for Fault Detection in Milling Machines.

C. Model Training

In this section, we delineate the methodology employed for training and evaluating machine learning models aimed at predicting machine failures using the provided synthetic dataset. The approach encompasses data preprocessing, feature engineering, handling class imbalance, model selection, hyperparameter optimization, and comprehensive evaluation of model performance using various metrics.

1) Libraries Used: The implementation of the machine learning models and the associated data processing steps utilized a variety of Python libraries, each serving a specific purpose within the workflow. Below is a summary of the libraries employed and their respective roles:

- **Pandas**: Utilized for data manipulation and analysis, enabling efficient handling of datasets through DataFrame structures.
- **NumPy**: Employed for numerical operations and handling multi-dimensional arrays, providing foundational support for data processing.
- **Matplotlib.pyplot**: Used for creating static, interactive, and animated visualizations to aid in exploratory data analysis and result interpretation.
- **Seaborn**: Built on top of Matplotlib, Seaborn provides a high-level interface for drawing attractive and informative statistical graphics.
- **Scikit-learn (sklearn)**:
  - *model_selection*: Contains tools for splitting data, cross-validation, and hyperparameter tuning.
  - *metrics*: Provides a wide range of metrics for model evaluation, including accuracy, precision, recall, F1-score, and ROC-AUC.
  - *multioutput*: Facilitates multi-output classification through wrappers like *MultiOutputClassifier*.
  - *ensemble*: Offers ensemble methods such as *RandomForestClassifier*.
- **Imbalanced-learn (imblearn)**:
  - *pipeline*: Enables the creation of pipelines that integrate preprocessing steps with classifiers.
  - *over_sampling*: Implements over-sampling techniques like *RandomOverSampler* to address class imbalance.
- **XGBoost**: A scalable and flexible gradient boosting framework used for building high-performance classifiers.
- **CatBoost**: A gradient boosting library that handles categorical features automatically and is known for its efficiency and accuracy.
- **Scikit-optimize (skopt)**:
  - *BayesSearchCV*: Facilitates Bayesian hyperparameter optimization, enabling efficient exploration of the hyperparameter space.
  - *space*: Defines the search space for hyperparameters, allowing for continuous and discrete parameter tuning.
- **Joblib**: Used for model persistence, enabling the saving and loading of trained models to and from disk.
- **Warnings**: Utilized to suppress warning messages that may arise during model training and evaluation, ensuring cleaner output.

These libraries collectively provide the necessary tools for data processing, model training, evaluation, and visualization,

forming the backbone of the machine learning pipeline employed in this study.

2) Data Preprocessing: The dataset comprises 10,000 observations with 14 features, each representing different aspects of a milling machine's operation. To ensure that the model focuses solely on predictive features, the unique identifier (*UID*) and *Product ID* columns were removed. These columns serve as identifiers and do not contribute meaningful information for predicting machine failures.

3) Feature Engineering: To enhance the model's predictive power, additional features were derived from the existing data:

1) **Power**: This feature was calculated as the product of *Rotational speed [rpm]* and *Torque [Nm]*. It represents the mechanical power involved in the milling process, providing a quantitative measure of the machine's operational intensity.
2) **Power Wear**: Derived by multiplying the previously calculated *Power* with *Tool wear [min]*, this feature encapsulates the relationship between mechanical stress and tool degradation over time.
3) **Temperature Difference**: This feature is the difference between *Process temperature [K]* and *Air temperature [K]*. It indicates the thermal gradient affecting the milling process, which can influence machine performance and failure rates.
4) **Temperature Power**: Calculated as the ratio of *Temperature Difference* to *Power*, this feature provides insight into how thermal variations scale with mechanical power, potentially highlighting conditions that lead to overheating or excessive cooling.

These engineered features were included in the model training process to capture complex interactions and dependencies within the data, thereby enhancing the model's ability to detect nuanced patterns associated with machine failures.

4) Handling Class Imbalance: The dataset exhibited significant class imbalance, particularly in the target variable *Machine failure*, which constituted approximately 3.39% of the data. Further examination revealed that subcategories of failures (*TWF* and *RNF*) were exceedingly rare, posing challenges for model training. Standard machine learning algorithms may bias towards the majority class, leading to poor predictive performance on minority classes.

To mitigate this imbalance, the **Random Over-Sampling** technique was employed using the *RandomOverSampler* from the *imbalanced-learn* library. This method synthetically increases the number of minority class instances by duplicating existing samples, thereby balancing the class distribution. By doing so, the model receives a more representative dataset, enabling it to learn the characteristics of both majority and minority classes more effectively.

5) Model Selection and Training: Two ensemble-based classifiers were selected for this study: **XGBoost** and **CatBoost**. These models are renowned for their robustness and ability to handle complex datasets with high-dimensional feature spaces,

The XGBoost classifier was integrated within a *MultiOutputClassifier* framework to handle multiple target variables simultaneously (*Machine failure*, *TWF*, *HDF*, *PWF*, *OSF*, *RNF*). Hyperparameter tuning was conducted using *RandomizedSearchCV*, which explores a predefined parameter grid to identify the optimal configuration for the model. The parameters optimized included:

- *learning_rate*: Controls the contribution of each tree to the final model. Values tested ranged from 0.05 to 0.30.
- *max_depth*: The maximum depth of a tree. Values tested ranged from 3 to 15.
- *min_child_weight*: Minimum sum of instance weight (hessian) needed in a child. Values tested ranged from 1 to 7.
- *gamma*: Minimum loss reduction required to make a further partition on a leaf node. Values tested ranged from 0.0 to 0.4.
- *colsample_bytree*: Subsample ratio of columns when constructing each tree. Values tested ranged from 0.3 to 0.7.

A total of 50 iterations were performed with 5-fold cross-validation to balance computational efficiency and thoroughness. The best parameters identified were:

- *learning_rate*: 0.05
- *max_depth*: 10
- *min_child_weight*: 1
- *gamma*: 0.3
- *colsample_bytree*: 0.5

Subsequently, the XGBoost classifier was trained using these optimal parameters. The model's performance was evaluated using cross-validation, yielding high accuracy (˜98%) but moderate precision (˜72-81%) and recall (˜66-77%). These results indicate that while the model is effective at correctly classifying the majority of instances, it faces challenges in accurately predicting minority classes.

6) CatBoost Classifier: Similarly, the CatBoost classifier was employed within a pipeline that incorporated Random Over-Sampling to address class imbalance. Hyperparameter optimization was conducted using *BayesSearchCV*, a Bayesian optimization technique that efficiently explores the hyperparameter space by selecting the most promising parameters based on past evaluations. The search space included:

- *learning_rate*: Continuous values between 0.05 and 0.30 with a logarithmic uniform prior.
- *max_depth*: Integer values between 3 and 15.
- *min_child_weight*: Integer values between 1 and 7.
- *gamma*: Continuous values between 0.0 and 0.4 with a uniform prior.
- *colsample_bytree*: Continuous values between 0.3 and 0.7 with a uniform prior.

The CatBoost model was configured with 1,000 iterations, a learning rate of 0.1, and a tree depth of 6. The optimization process focused on maximizing the F1-weighted score to ensure a balance between precision and recall. The best parameters identified varied across labels but generally emphasized

higher learning rates and deeper trees to capture complex patterns within the data.

D. Evaluation Metrics

Model performance was assessed using a suite of evaluation metrics tailored to both balanced and imbalanced classification scenarios:

- **Accuracy**: Measures the proportion of correctly classified instances out of the total instances. It is calculated as:
- **Precision**: Indicates the proportion of true positive predictions among all positive predictions. It is particularly useful for understanding the accuracy of positive predictions.
- **Recall**: Reflects the proportion of true positive predictions among all actual positives. It measures the model's ability to identify positive instances.
- **F1-Score**: The harmonic mean of precision and recall, providing a single metric that balances both concerns.
- **ROC-AUC**: Evaluates the model's ability to distinguish between classes by measuring the area under the Receiver Operating Characteristic curve.

Additionally, **confusion matrices** and **classification reports** were generated for each label to provide detailed insights into model performance, highlighting true positives, false positives, true negatives, and false negatives.

E. Results and Discussion

1) XGBoost Performance: The XGBoost model demonstrated high accuracy across all labels, surpassing 97% in most cases. However, precision and recall were notably lower for minority classes (*TWF* and *RNF*), revealing a tendency to misclassify these instances. Specifically:

- **Machine Failure**: Achieved an accuracy of approximately 98.45%, with precision and recall around 76.07% and 79.46%, respectively. This indicates a solid ability to correctly identify machine failures while maintaining a balance between precision and recall.
- **TWF and RNF**: Exhibited poor performance with precision, recall, and F1-scores at or near zero, indicating complete failure in predicting these minority classes. This suggests that the model is unable to capture the subtle patterns associated with these rare failure modes.
- **Other Labels**: *HDF*, *PWF*, and *OSF* showed robust performance, with high precision, recall, and F1-scores. This demonstrates the model's effectiveness in predicting more prevalent failure modes, leveraging the balanced distribution after over-sampling.

2) CatBoost Performance: CatBoost mirrored the performance trends observed in XGBoost, maintaining high accuracy for majority classes while struggling with minority classes:

- **Machine Failure**: Achieved similar accuracy (98.45%) with slightly higher recall (82.14%) compared to XGBoost. This indicates a marginal improvement in identifying true positive machine failures.
- **TWF**: Showed marginal improvement over XGBoost with a precision of 19.23% and recall of 33.33%, though still inadequate for practical applications. This slight improvement suggests that CatBoost may be slightly better at capturing patterns in minority classes but remains insufficient.
- **RNF**: Like XGBoost, CatBoost failed to predict any instances of this class effectively, resulting in zero precision, recall, and F1-score. This highlights a critical limitation in handling extremely rare failure modes.
- **Other Labels**: Maintained strong performance metrics for *HDF*, *PWF*, and *OSF*, albeit with slight variations in precision and recall compared to XGBoost. This consistency reinforces the models' capability in handling majority classes effectively.

3) Analysis of Performance Discrepancies: The stark contrast in model performance between majority and minority classes underscores the pervasive challenge of class imbalance. While both XGBoost and CatBoost excelled in predicting more common failure modes, their inability to effectively identify rare failures (*TWF* and *RNF*) highlights a critical limitation. This discrepancy likely stems from the insufficient representation of these classes in the training data, even after over-sampling, coupled with the inherent difficulty in learning from scarce positive examples.

Additionally, the high ROC-AUC scores for *TWF* suggest that the models possess some discriminatory power, yet this is not translated into practical predictive performance due to thresholding and class distribution dynamics. The ROC-AUC metric alone does not account for the balance between precision and recall, especially in imbalanced datasets, thereby providing an incomplete picture of model performance.

F. Confusion Matrix and Classification Report

Confusion matrices and classification reports were generated for each label to provide detailed insights into model performance. These tools break down the number of true positives, false positives, true negatives, and false negatives, offering a granular view of how well the model is performing across different classes.

For instance, the confusion matrix for *Machine failure* under the XGBoost model showed a high number of true negatives and true positives, with a manageable number of false positives and false negatives. However, for minority classes like *TWF* and *RNF*, the confusion matrices indicated an overwhelming number of false negatives, meaning the model failed to identify actual failures in these categories.

G. Results Summary

The evaluation results for both XGBoost and CatBoost classifiers are summarized in the Table I

H. Discussion

1) Performance on Majority Classes: For majority classes such as *PWF*, *Machine Failure*, *HDF*, and *OSF*, both classifiers achieved high accuracy, precision, recall, and F1-scores. This indicates that the models are effectively capturing the patterns and relationships inherent to these failure modes,

TABLE I
EVALUATION METRICS FOR XGBOOST AND CATBOOST CLASSIFIERS

Label	Classifier	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Machine Failure	XGBoost	0.9845	0.7607	0.7946	0.7773	0.9818
	CatBoost	0.9845	0.747967	0.821429	0.782979	0.974973
TWF	XGBoost	0.9948	0.0000	0.0000	0.0000	0.9589
	CatBoost	0.9906	0.1923	0.3333	0.2439	0.967793
HDF	XGBoost	0.9945	0.7381	0.8158	0.7750	0.9976
	CatBoost	0.9933	0.6667	0.8421	0.7442	0.997354
PWF	XGBoost	0.9994	0.9394	1.0000	0.9688	1.0000
	CatBoost	0.9994	0.939394	1.000000	0.968750	1.000000
OSF	XGBoost	0.9979	0.9032	0.8750	0.8889	0.999708
	CatBoost	0.996970	0.866667	0.812500	0.8387	0.999541
RNF	XGBoost	0.9982	0.0000	0.0000	0.0000	0.616727
	CatBoost	0.996667	0.0000	0.0000	0.0000	0.606760

making them reliable for predicting such failures in real-world scenarios.

2) Performance on Minority Classes: Conversely, for the minority classes *TWF* and *RNF*, both classifiers exhibited poor performance metrics

3) Implications of Class Imbalance: The high accuracy observed in minority classes is misleading, as it primarily reflects the models' proficiency in predicting the majority class correctly. In imbalanced datasets, accuracy can be a deceptive metric, as a model that consistently predicts the majority class will achieve high accuracy without truly capturing the underlying patterns of the minority classes. This phenomenon underscores the necessity of employing additional evaluation metrics, such as precision, recall, and F1-score, which provide a more nuanced understanding of model performance, especially in the context of class imbalance.

4) Model Sensitivity to Data Splits: The variability in performance metrics across different cross-validation folds suggests that the models' ability to generalize is sensitive to the specific data splits used during training and testing. This sensitivity can lead to inconsistent performance, particularly for minority classes, where even slight variations in data distribution across folds can significantly impact the models' ability to learn and predict accurately.

5) Conclusion: The model training and evaluation process revealed that while ensemble classifiers like XGBoost and CatBoost are highly effective for balanced datasets, their performance diminishes in the face of severe class imbalance, particularly for exceedingly rare classes. The models demonstrated excellent predictive capabilities for majority classes but failed to identify minority classes (*TWF* and *RNF*), resulting in poor recall and F1-scores despite high accuracy. This limitation poses significant challenges in real-world applications where predicting rare but critical failure modes is essential for maintenance and operational reliability.

I. Integration of Trained Models into the Processing Architecture

Following the training phase, the models exhibiting satisfactory performance, specifically those for predicting **Machine Failure**, **HDF (Heat Dissipation Failure)**, **OSF (Overstrain Failure)**, and **PWF (Power Failure)**, were serialized using the Joblib library and stored locally. These models were subsequently integrated into the real-time processing architecture by copying them to the 'spark-app' directory of the Lambda Architecture. This directory ensures the models are readily accessible for prediction tasks within the Spark environment. The models for TWF (Tool Wear Failure) and RNF (Random Failure) were excluded due to their poor predictive performance and the extreme imbalance of these labels in the dataset.

The prediction process leverages PySpark's distributed data processing capabilities and is implemented as part of the Spark Streaming pipeline. Below is a detailed description of the data transformation and prediction workflow:

- **Data Ingestion:** Synthetic data is ingested via Apache Kafka and passed into the Spark application as a continuous stream.
- **Data Parsing:** The raw JSON data is parsed and exploded into individual records, adhering to a pre-defined schema that captures essential sensor readings, including air temperature, process temperature, rotational speed, torque, and tool wear.
- **Feature Engineering:** Key features for prediction are computed:
  - **Power:** Calculated as the product of rotational speed and torque.
  - **Power Wear:** A feature capturing the interaction between power and tool wear.
  - **Temperature Difference:** Difference between process temperature and air temperature.
  - **Temperature Power:** Ratio of temperature difference to power, providing a normalized measure of thermal efficiency.
- **Model Broadcasting:** The trained models are broadcast to all Spark worker nodes, ensuring distributed access for prediction tasks. This minimizes communication overhead and enhances scalability.
- **Prediction:** A user-defined function (UDF) is implemented using PySpark's Pandas UDF functionality. This UDF takes the computed features as input and applies each broadcast model (HDF, Machine Failure, OSF, and PWF) to produce predictions.
- **Real-Time Data Storage:** The predictions are appended

to the original data and stored in Elasticsearch, with indices organized for seamless real-time querying and visualization.

This integrated approach ensures that the system processes incoming data streams in real-time while utilizing the advanced prediction capabilities of the trained models. The exclusion of TWF and RNF models is justified by their inadequate performance, as identified during the training phase. This decision avoids introducing unreliable predictions into the system, thereby maintaining the overall integrity of the real-time monitoring solution.

By coupling the trained models with the robust Lambda Architecture, this implementation provides a scalable and efficient fault detection mechanism, enabling proactive decision-making in industrial environments.

VI. Conclusion

This study presented a comprehensive fault detection system for milling machines, leveraging advanced machine learning models and a robust Lambda Architecture for real-time data processing and predictive maintenance. By integrating machine learning techniques, such as XGBoost and CatBoost, with distributed processing frameworks, the proposed solution effectively addressed the complexities of high-velocity and high-volume data generated by industrial equipment. The architecture demonstrated scalability and reliability, employing tools like Apache Kafka, Apache Spark, Elasticsearch, and Kibana to ensure seamless data ingestion, processing, and visualization.

The trained models successfully predicted critical failure modes such as *Machine Failure*, *Heat Dissipation Failure (HDF)*, *Power Failure (PWF)*, and *Overstrain Failure (OSF)* with high accuracy, precision, and recall. However, the system faced challenges with minority classes such as *Tool Wear Failure (TWF)* and *Random Failure (RNF)*, primarily due to extreme class imbalance and the inherent difficulty in learning patterns from limited samples. Consequently, these labels were excluded from the real-time prediction pipeline to maintain the overall reliability and robustness of the system.

The integration of the trained models into the real-time processing pipeline showcased the potential for predictive maintenance applications in industrial settings. The exclusion of unreliable predictions for minority classes further reinforced the system's integrity, ensuring that the results remain actionable and meaningful for end-users. By providing real-time fault detection and visualization, this system enables proactive decision-making, ultimately reducing downtime, optimizing maintenance schedules, and enhancing operational efficiency.

VII. Real-World Applicability

Although the system developed demonstrates a robust architecture and good models for fault detection in milling machines, it is essential to acknowledge that real-world applications bring additional challenges beyond the scope of this work. The complexity of industrial environments and the variations in operational parameters can limit the effectiveness of the model trained in a controlled environment with synthetic data.

One of the main challenges in practical applications lies in the need for a **continuous feedback layer** for the machine learning models. In real-world scenarios, new failure patterns or changes in sensor data may emerge due to process modifications or equipment updates. Without a feedback mechanism enabling the continuous reassessment and updating of models, the system may become outdated or inaccurate over time.

Moreover, the variable conditions of industrial environments require **more flexible models** capable of adapting to different configurations, operational parameters and machines specifications. For example, variations in sensor calibration, differences in tool wear, or changes in the type of processed material can directly impact model performance. Solutions such as incremental training, online learning, or even transfer learning techniques may be necessary to address these dynamics effectively.

Another critical aspect is that, in real implementations, fault detections often need to be **manually evaluated** before being used as feedback for the models. This process demands significant effort from data science, machine learning engineering, and MLOps teams, which must implement robust pipelines to retrain the models with newly labeled data and reintegrate them into the production system. Additionally, dealing with delays in validation or errors in labeled data can introduce bottlenecks that affect the system's overall efficiency.

Finally, it is crucial to highlight that scaling a predictive maintenance system requires more than just well-performing models and architectures. Issues related to data reliability, IT infrastructure, and cultural acceptance by operational teams are equally critical for the success of such systems. Therefore, while this work provides a solid foundation, it should be seen as a starting point for broader and more integrated efforts involving multiple disciplines and areas of expertise.

VIII. Future Work

While the proposed system achieved significant milestones, several areas for improvement and future research remain:

- **Addressing Class Imbalance:** The poor performance of models for minority classes such as *TWF* and *RNF* highlights the need for advanced techniques to handle class imbalance. Future work could explore synthetic data generation methods, such as Generative Adversarial Networks (GANs), to augment training data for rare failure modes.
- **Explainability and Interpretability:** Implementing explainable AI (XAI) techniques, such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations), could enhance the interpretability of model predictions. This would help domain experts understand the reasoning behind failure predictions and improve trust in the system.
- **Dynamic Model Updates:** Incorporating online learning or incremental model training could enable the system

to adapt to new data patterns and evolving conditions without requiring complete retraining.

- **Integration with Legacy Systems:** Future work could explore the integration of the proposed system with existing industrial control systems (ICS) to provide seamless operational support and automated decision-making.
- **Extending to Multi-Machine Scenarios:** Expanding the architecture to handle multiple types of industrial machines simultaneously would enhance its versatility and applicability to broader manufacturing contexts.
- **Advanced Visualization and Alerts:** Enhancing the Kibana dashboards with predictive insights, such as failure probability over time or recommended actions, could provide additional value. Integrating real-time alert systems via messaging platforms or mobile notifications would further improve responsiveness.
- **Robustness to Noisy Data:** Developing noise-resilient preprocessing techniques or robust learning models to handle sensor anomalies and missing data could improve system reliability in real-world applications.
- **Sustainability and Cost Analysis:** Evaluating the energy consumption and operational costs of deploying such a system at scale would provide valuable insights into its feasibility for industrial adoption.

By addressing these areas, the proposed fault detection system can be further refined to meet the dynamic requirements of modern industrial environments. The advancements in predictive maintenance will not only enhance equipment reliability but also contribute to the broader goals of sustainability and efficiency in manufacturing processes.

REFERENCES

[1] "AI4I 2020 Predictive Maintenance Dataset," UCI Machine Learning Repository, 2020, DOI: https://doi.org/10.24432/C5HS5C.

[2] J. Warren and N. Marz, 2015.

[3] G. G. Samatas, S. S. Moumgiakmas, and G. A. Papakostas, "Predictive maintenance - bridging artificial intelligence and iot," in *2021 IEEE World AI IoT Congress (AIIoT)*, May 2021, p. 0413–0419. [Online]. Available: https://ieeexplore.ieee.org/document/9454173

[4] L. Magadán, F. J. Suárez, J. C. Granda, and D. F. García, "Low-cost real-time monitoring of electric motors for the industry 4.0," *Procedia Manufacturing*, vol. 42, p. 393–398, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978920306211

[5] S. F. Chevtchenko, E. D. S. Rocha, M. C. M. D. Santos, R. L. Mota, D. M. Vieira, E. C. De Andrade, and D. R. B. De Araújo, "Anomaly detection in industrial machinery using iot devices and machine learning: A systematic mapping," *IEEE Access*, vol. 11, p. 128288–128305, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10318838/?arnumber=10318838

[6] P. Angelov, E. Soares, R. Jiang, N. Arnold, and P. Atkinson, "Explainable artificial intelligence: an analytical review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 5, 2021. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85110049592&doi=10.1002%2fwidm.1424&partnerID=40&md5=f15cac4f7a392e14767297ab686721d6

[7] T. Mansouri and S. Vadera, "A deep explainable model for fault prediction using iot sensors," *IEEE Access*, vol. 10, p. 66933–66942, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9801817

[8] M. B. Imani, M. Heydarzadeh, L. Khan, and M. Nourani, "A scalable spark-based fault diagnosis platform for gearbox fault diagnosis in wind farms," in *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, Aug. 2017, p. 100–107. [Online]. Available: https://ieeexplore.ieee.org/document/8102925

[9] M. Gribaudo, M. Iacono, and M. Kiran, "A performance modeling framework for lambda architecture based applications," *Future Generation Computer Systems*, vol. 86, p. 1032–1041, Sep. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17315364

[10] J. Patel, "An effective and scalable data modeling for enterprise big data platform," in *2019 IEEE International Conference on Big Data (Big Data)*, Dec. 2019, p. 2691–2697. [Online]. Available: https://ieeexplore.ieee.org/document/9005614

[11] S. Matzka, "Explainable artificial intelligence for predictive maintenance applications," in *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, Sep. 2020, p. 69–74. [Online]. Available: https://ieeexplore.ieee.org/document/9253083

[12] Vinícius, "italovinicius18/fault-detection," Dec. 2024. [Online]. Available: https://github.com/italovinicius18/fault_detection