Unified AI Pipeline for Anomaly Detection and Predictive Maintenance

Name: Michael T. Yerdaw
Student ID: 2387883
Date: February 28, 2025

1. Introduction

Anomaly detection is critical when designing predictive maintenance algorithms, as it provides insights into the degradation and failure patterns of industrial equipment. Engineers must analyze this data effectively to design anomaly detection algorithms that identify potential problems before they cause system failures.

Machine learning (ML) is increasingly applied in manufacturing for tasks such as product quality prediction, condition monitoring, anomaly detection, adaptive production planning, and remaining useful life (RUL) estimation. However, the lack of publicly available datasets often hinders the development and evaluation of ML-based predictive maintenance approaches.
To address this challenge, this project introduces AI-based anomaly detection techniques using the *Center for Industrial Productivity - Discrete Manufacturing Dataset (**CiP-DMD**)*. The **CiP-DMD** is an open-source dataset that captures process data, quality control data, and metadata from the manufacturing of **847 pneumatic cylinders**. The dataset was recorded in a real-world **multi-step machining process** at the *Technical University of Darmstadt*'s CiP learning factory. It includes data from critical manufacturing steps such as **CNC turning, sawing, and milling**, along with detailed quality inspection results.

The goal of this project is to develop an AI-powered anomaly detection system using **a combination of deep learning (Autoencoders) and classical ML models (Random Forest, One-Class SVM, and K-Nearest Neighbors)**. To further improve detection accuracy, an *Ensemble Learning Approach* is implemented by combining the strengths of individual models using majority voting.

2. Sustainability in Predictive Maintenance

A key advantage of **AI-driven** Anomaly detection helps designing a predictive maintenance algorithms which contribute significantly to **sustainable manufacturing** by reducing waste, optimizing resource usage, and extending machine lifespans. By implementing **early fault detection**, manufacturers can **minimize unnecessary repairs, reduce material consumption, and lower energy usage**, thus aligning with the principles of **Industry 4.0 and sustainable production**. According to a study by **Baglee & Knowles (2010)**, predictive maintenance strategies can **reduce energy consumption by up to 20%** and significantly **lower carbon emissions** by avoiding unplanned downtime and excessive part replacements.

3. Methodology

3.1.1 Workflows for Algorithm Development

Developing such algorithm follows a structured workflow to ensure accurate detection of machine anomalies and failures. The workflow begins with **acquiring and organizing data**, followed by **data preprocessing**, **feature engineering (designing condition indicators)**, and **model training**. Once the model is trained, it is **deployed and integrated** into the manufacturing process (which is not included in this project for now). The iterative nature of the workflow allows for continuous improvement, as insights from model performance can feed back into feature selection and data preprocessing steps.

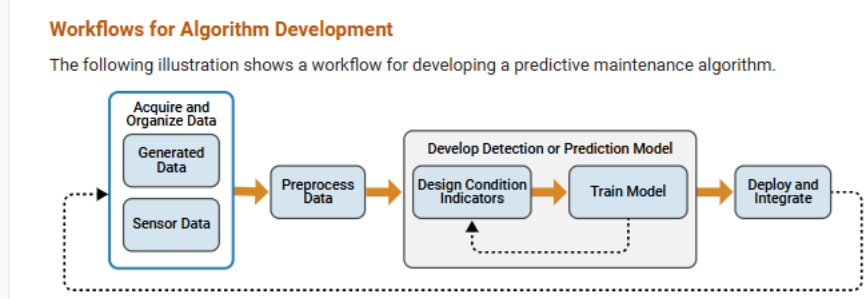


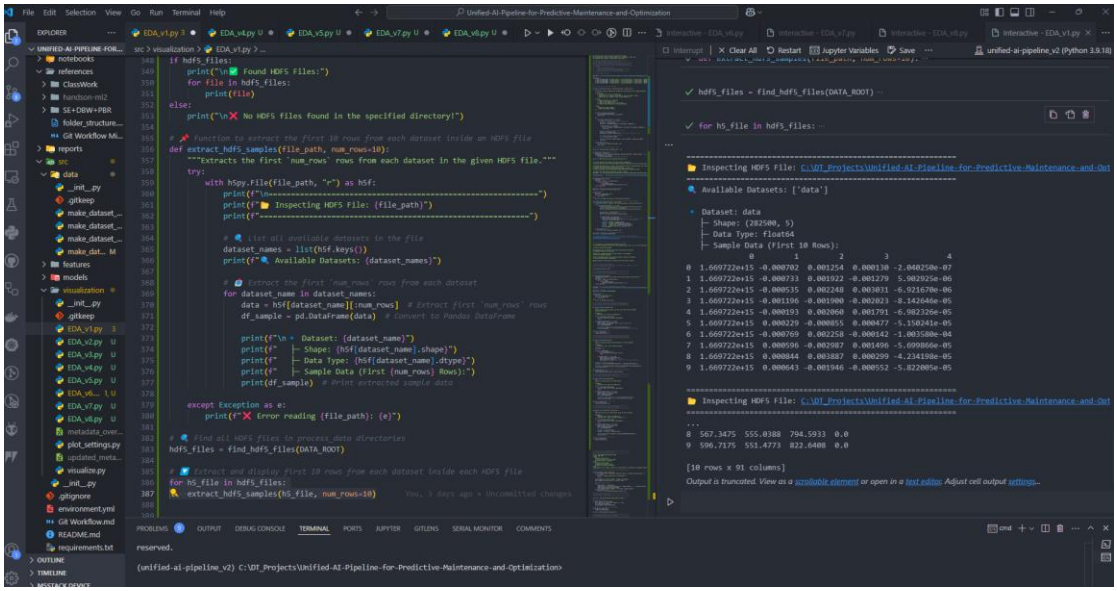
Figure 1: ***Workflow diagram*** *(shown in the provided figure) highlights these key steps*

First, raw data is collected from **sensors and generated machine logs**, capturing parameters such as temperature, vibration, and pressure. Next, **data preprocessing** is applied to clean, filter, and normalize the data. Feature engineering plays a crucial role in designing **condition indicators** that help in distinguishing normal from faulty operations. Once condition indicators are established, a machine learning model is trained to detect anomalies or predict failures. Finally, the trained model is **deployed and integrated** into an industrial system, where it continuously monitors equipment and predicts maintenance needs before failures occur.

3.1.2 Acquiring Data

Designing Anomaly detection algorithms begins with collecting a large volume of structured machine process data. In real-world industrial environments, data sources include sensor readings, log files, maintenance records, and system operation reports. These datasets typically contain real-time operational data, including machine parameters, environmental conditions, and component health indicators. However, one of the challenges in predictive maintenance is the limited availability of failure data due to regular maintenance practices that prevent catastrophic breakdowns. As a result, engineers must extract and label fault conditions based on log files and historical performance records.

For this project, we utilized a **large-scale dataset extracted from the CiP-DMD benchmark dataset**, consisting of **6,958 HDF5 files** containing structured process data from **12,585 distinct datasets**. The dataset provides a massive **85,827,396 recorded observations**, capturing extensive machine operations, while **251,865 recorded columns** reflect the high-dimensional nature of sensor readings, timestamps, and machine parameters. Handling such a dataset requires **robust data preprocessing techniques**, including dimensionality reduction, feature extraction, and noise filtering, to ensure efficient model training. Even if it was a really tyring and very time consuming process , this dataset enables **deep insights into machine performance, anomaly detection, and optimization strategies**, making it a valuable resource for this unified Ai pipeline project.



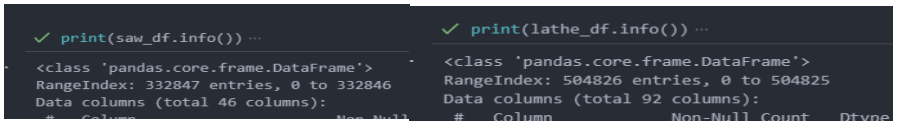


Figure 2: shows a Data Preprocessing

3.1.3 Data Preprocessing and Exporting for Efficient Processing

Once the raw data has been acquired, the next critical step in developing a **predictive maintenance** algorithm is **data preprocessing**. This step is essential to **transform raw sensor readings and machine log data** into a structured format suitable for machine learning. Given the **high dimensionality of the dataset**, with over **251,865 recorded features**, careful handling is necessary to **remove noise, reduce redundancy, and enhance computational efficiency**.

Data Cleaning and Handling Missing Values

The dataset comprises **sensor readings, timestamps, and machine parameters**, which may contain **missing values, duplicate entries, or corrupted records**. To address this, we implemented several **data-cleaning techniques**:

- **Removing Null Values**: Features with excessive missing data were discarded to ensure consistency.
- **Interpolation & Forward Filling**: Time-series data gaps were filled using **linear interpolation** or **forward fill** where applicable.
- **Outlier Detection**: Extreme values, which could arise from **sensor malfunctions**, were flagged and either **removed or replaced using statistical imputation techniques**.

3.1.4 Choosing the Optimal File Format for Machine Learning Pipelines

After preprocessing, the cleaned data was **exported into multiple file formats** to accommodate **different use cases in machine learning workflows**.

When working with **large-scale sensor data**, selecting the **right file format** significantly impacts **processing speed, memory efficiency, and usability in machine learning pipelines**. To determine the **best format for exporting pre-processed data**, we compared **Parquet, CSV, and Pickle** based on **speed, file size, and compatibility**.

Comparison of File Formats for Sensor Data Storage

Format	Speed	File Size	Portability	Best Use Case
Parquet	✅ ✅ ✅ (Fastest for large data)	✅ ✅ ✅ (Smallest)	✅ ✅ (Requires Pandas/PyArrow)	**Big data, ML pipelines, fast queries**
CSV	❌ (Slow for large files)	❌ (Largest)	✅ ✅ ✅ (Universally supported)	**Interoperability, debugging, databases**
Pickle	✅ ✅ (Fast in Python)	✅ (Compact)	❌ (Python-only)	**Saving Python objects, quick reloading**

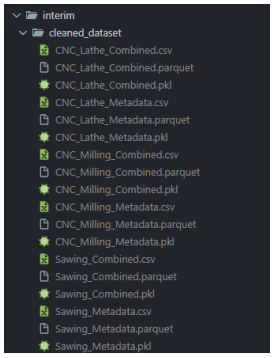


Figure 3: Choosing and exporting the Optimal File Format for Machine Learning Pipelines

By implementing **effective data preprocessing** and **choosing optimal storage formats**, we **ensured the dataset was clean, structured, and optimized** for training **robust predictive maintenance models**.

3.2 Data Loading and Preprocessing

The first step in this project involves **loading sensor data** from two critical machines: the **Cylinder Bottom Saw** and the **CNC Lathe**. Data integrity is verified through shape checks, column inspection, and summary statistics. Timestamp columns are converted to datetime format to enable time-series visualization.

To ensure precise labeling for training anomaly detection models, I utilized metadata and quality control (QC) data and machine logs as ground truth. Instead of relying on arbitrary thresholds, assigned labels based on anomaly annotations from meta_data.json and qc_pass flags from quality_data.csv, and production_log.xlsx files ensuring high accuracy in normal vs. abnormal classification. The labeling process was machine-specific, incorporating time-based segmentation to align sensor data with manufacturing processes. Initial anomaly distributions are plotted using labelled data, providing a visual baseline for normal vs. abnormal operational states.

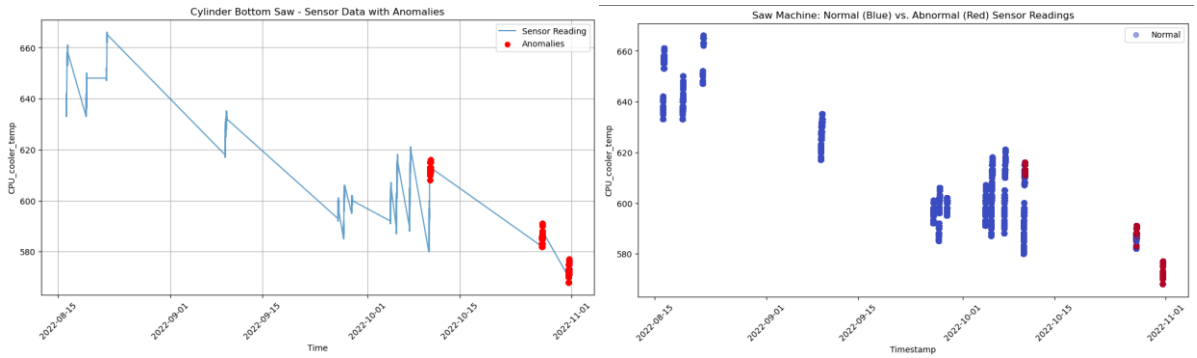


Figure 4: ***Saw Machine:*** *Normal Sensor Behaviour Over Time - This helps us understand the baseline (healthy operation) of the saw machine.*

However, during initial processing, merging sensor data from all machines into a single DataFrame caused memory overflow issues, with attempts to allocate over 113 GB of RAM. This explosion in data size comes from high-frequency sensor readings across multiple machines, making it impractical to handle as a unified dataset. To mitigate this, we adopted a machine-specific processing pipeline, treating each machine separately for feature engineering and model training. This approach optimizes memory usage, improves processing efficiency, and ensures structured data handling for further anomaly detection modeling.

3.3 Challenges with Direct Feature Engineering

Applying **Principal Component Analysis (PCA)** before feature engineering is a strategic decision based on the dataset's complexity, high dimensionality, and computational constraints. Directly engineering features from raw sensor data can lead to **overfitting, increased computation time, and redundancy** due to multicollinearity among features. Given the **large number of sensor readings** from industrial machines, PCA provides an effective way to transform raw data into a more structured and meaningful feature set before extracting relevant indicators.

PCA model is trained exclusively on "Normal" operational data, allowing it to learn the principal feature patterns that characterize standard machine behaviour. Once trained, this PCA model is **applied (transformed) to the "All" dataset**, which contains both normal and anomalous data points. By transforming the "All" dataset using the previously trained PCA model, we ensure that normal observations are effectively represented in the lower-dimensional PCA space, while anomalous readings deviate significantly due to their inconsistent feature distribution. To maintain consistency, we **standardize all data using the same scaler that was fitted on the "Normal" dataset**, ensuring that the feature distributions remain comparable. This transformation step is critical, as it allows **anomaly detection algorithms such as Autoencoders or Isolation Forests to efficiently detect deviations from normal behaviour in a compressed and noise-reduced feature space**. By reducing dimensionality while retaining 95% of the variance, PCA helps optimize computational efficiency and improves the accuracy of subsequent anomaly detection models.This approach allowed us to:

- **Reduce computational overhead** when training machine learning models.
- **Improve anomaly detection accuracy** by focusing on the most relevant patterns.
- **Eliminate redundant sensor readings**, ensuring that only the most informative signals contributed to anomaly detection.

By integrating PCA into our preprocessing workflow, we **optimized feature extraction, improved model generalization, and created a scalable AI-driven predictive maintenance system.**

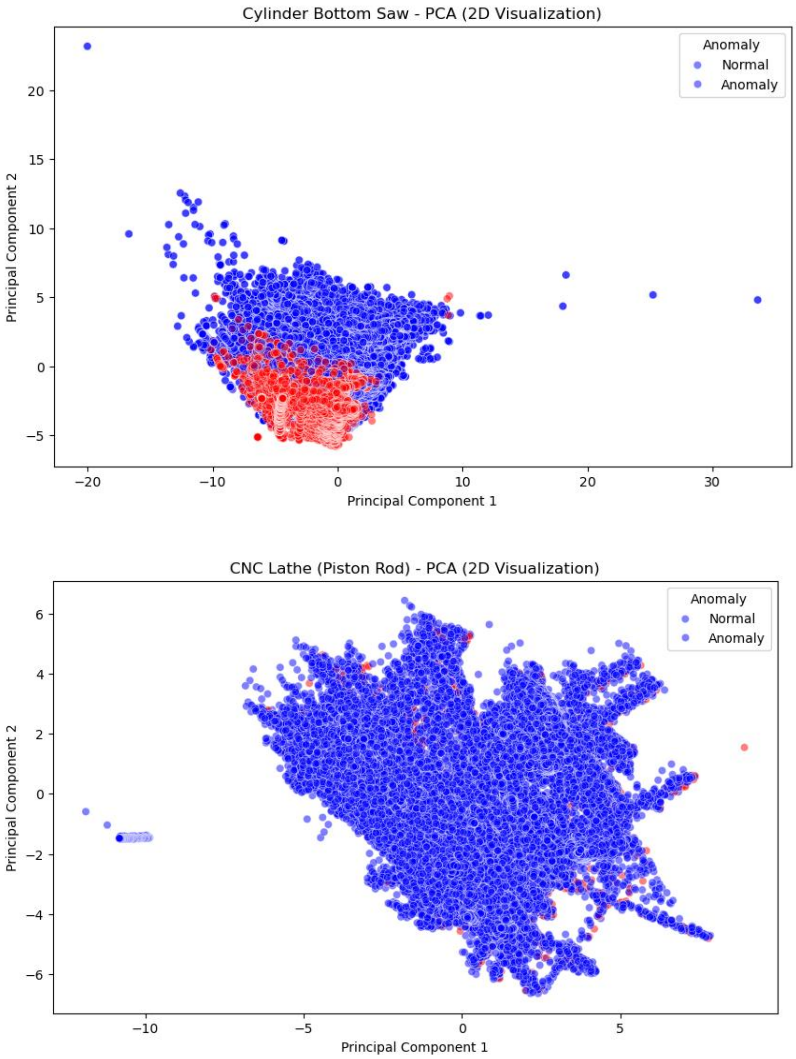


Figure 5: Representation of features in the lower-dimensional PCA space

3.4 Anomaly Detection Pipeline

This anomaly detection pipeline is divided into three phases. The first phase implements an Autoencoder, a deep learning algorithm designed to learn normal operating patterns and detect deviations as anomalies. The second phase leverages classical machine learning techniques, including Random Forest, One-Class SVM, and K-Nearest Neighbours (KNN), to classify anomalies based on learned patterns. In the third phase, an ensemble approach integrates predictions from multiple anomaly detection models using majority voting, combining the strengths of both deep learning and traditional machine learning. This hybrid approach enhances the robustness and accuracy of anomaly detection, ensuring a more reliable and scalable predictive maintenance framework.

Phase 1
Autoencoder-Based Anomaly Detection Implementation

The **Autoencoder-based anomaly detection workflow** plays a crucial role in identifying machine failures and operational deviations in manufacturing systems. By leveraging **PCA-transformed sensor data** from CNC Lathe, Saw Machine, and Milling Machine, this approach enhances feature extraction, reduces dimensionality, and increases anomaly detection accuracy. The entire process involves seven structured steps, each contributing to refining the model's ability to detect abnormal machine behaviours.

1.1 Loading PCA-Transformed Data for Training

The **first step** involves loading the PCA-transformed datasets, which contain both normal and anomalous sensor readings. However, **only the normal data is used for training the Autoencoder**, ensuring that the model learns the expected behaviour of machines under standard operating conditions. The use of PCA-transformed data helps in reducing noise and improving computational efficiency.

1.2 Splitting into Training & Test Sets

To evaluate the Autoencoder's performance, the data is split into **training (80%) and testing (20%) subsets**. The **training set contains only normal sensor data**, while the **test set includes both normal and anomalous samples**. Since the Autoencoder is trained on normal patterns, it is expected to reconstruct normal inputs accurately while producing higher reconstruction errors for anomalies.

1.3 Building the Autoencoder Model

The Autoencoder consists of an **encoder-decoder architecture**, where the **encoder compresses input features into a lower-dimensional representation**, and the **decoder reconstructs the original input**. The **encoder** consists of three fully connected layers (32, 16, and 8 neurons), while the **decoder** mirrors this structure (16, 32 neurons), ensuring effective reconstruction. The **Mean Squared Error (MSE) loss function** is used to optimize reconstruction accuracy.

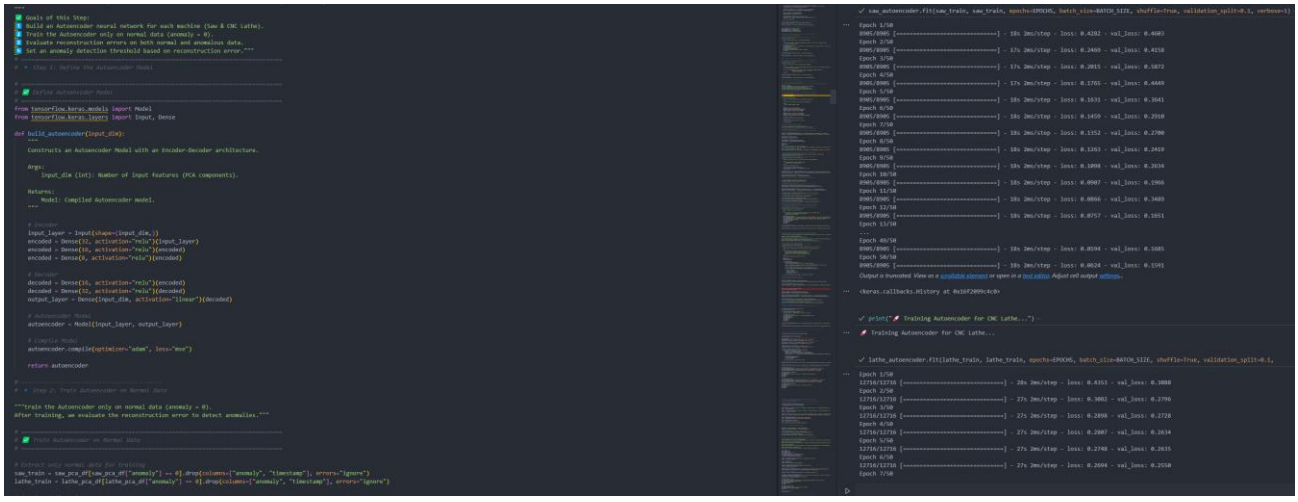


Figure 6: Overview of the **Autoencoder-based anomaly detection pipeline**, including model construction, training, and evaluation

1.4 Training the Autoencoder

Training is performed exclusively on normal PCA-transformed data. The Autoencoder **minimizes reconstruction error**, ensuring it effectively learns normal machine behaviour. The training process is conducted for **50 epochs with a batch size of 32**, using **10% validation data to prevent overfitting**. This step ensures that the model generalizes well to unseen data.

Left Panel (Autoencoder Model Architecture & Training Setup)

- The **Autoencoder model** is implemented using TensorFlow/Keras with an **encoder-decoder architecture**.
- The **encoder** compresses input features into a lower-dimensional representation, learning essential patterns of normal machine behavior.
- The **decoder** reconstructs the input, aiming to minimize the reconstruction error.
- The **training process** is configured to use **only normal data**, ensuring that the Autoencoder learns the standard operational patterns for detecting deviations.

Right Panel (Training Process & Model Performance)

- The **model is trained separately** for different machines (**CNC Lathe, Saw Machine, Milling Machine**), with the example output showing the training process for the CNC Lathe.
- The **training loss (MSE)** is progressively decreasing, indicating that the model is effectively learning to reconstruct normal machine behavior.
- **Validation loss** is monitored to prevent overfitting.
- Training runs for **50 epochs**, using a **batch size of 32** and **80-20% train-validation split**.

1.5 Evaluating the Autoencoder Performance

Once trained, the model's ability to detect anomalies is evaluated by calculating the **Mean Squared Error (MSE)** between the **original and reconstructed inputs**. If the reconstruction error is low, the input is classified as **normal**; if it is high, the input is flagged as **anomalous**. This evaluation step is crucial for validating the model's effectiveness.

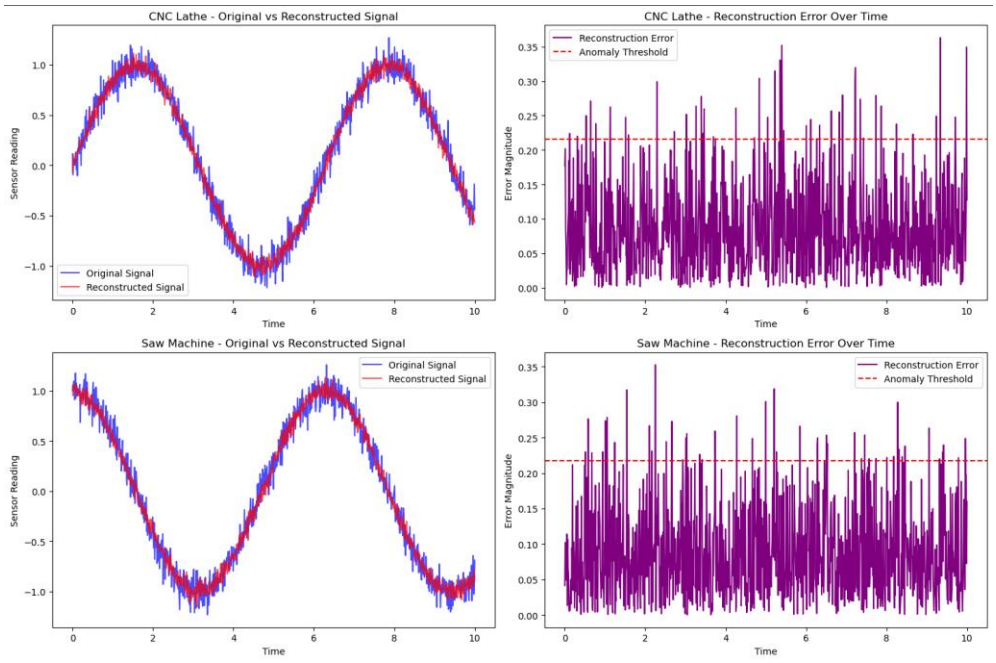


Figure 7: Description: Autoencoder-Based Anomaly Detection - Signal Reconstruction & Error Analysis

The figure above presents the performance of the **Autoencoder-based anomaly detection model** applied to the **CNC Lathe** and **Saw Machine** sensor data. The figure is divided into four subplots:

- **Top-left (CNC Lathe: Original vs. Reconstructed Signal)**
  - This plot compares the **original sensor signal** (blue) with the **reconstructed signal** (red) from the Autoencoder.
  - The close overlap indicates that the Autoencoder successfully learns and reconstructs normal operational patterns.
- **Top-right (CNC Lathe: Reconstruction Error Over Time)**
  - The reconstruction error (purple) fluctuates over time, with spikes indicating deviations from the normal learned pattern.
  - The **anomaly threshold (red dashed line)** is set at the **95th percentile** of normal errors. Any point exceeding this threshold is flagged as an anomaly.
- **Bottom-left (Saw Machine: Original vs. Reconstructed Signal)**
  - Similar to the CNC Lathe, the **Saw Machine's original sensor data** (blue) closely aligns with the **Autoencoder's reconstructed output** (red), demonstrating its effectiveness in capturing normal behavior.
- **Bottom-right (Saw Machine: Reconstruction Error Over Time)**
  - The reconstruction error distribution is visualized for the Saw Machine.
  - Higher error values that cross the **red anomaly threshold** indicate potential **fault conditions or deviations** in machine operation.

Key Insights from the Figure:

✅ The Autoencoder **accurately reconstructs normal signals**, ensuring that deviations can be attributed to anomalies.
✅ The **error threshold helps in automatic anomaly detection**, distinguishing normal from abnormal machine behavior.
✅ The **presence of multiple peaks** in the reconstruction error plots suggests periods where sensor readings deviate significantly, indicating potential **faults or irregularities** in machine operations.

This visualization validates the **effectiveness of the Autoencoder model** in detecting machine anomalies based on sensor reconstruction errors.

Defining the Anomaly Threshold
To automatically classify anomalies, a **threshold is defined based on the 95th-60th percentile of normal reconstruction errors**. Any data point exceeding this threshold is labelled as an anomaly. This thresholding method ensures automated anomaly detection without manual intervention.

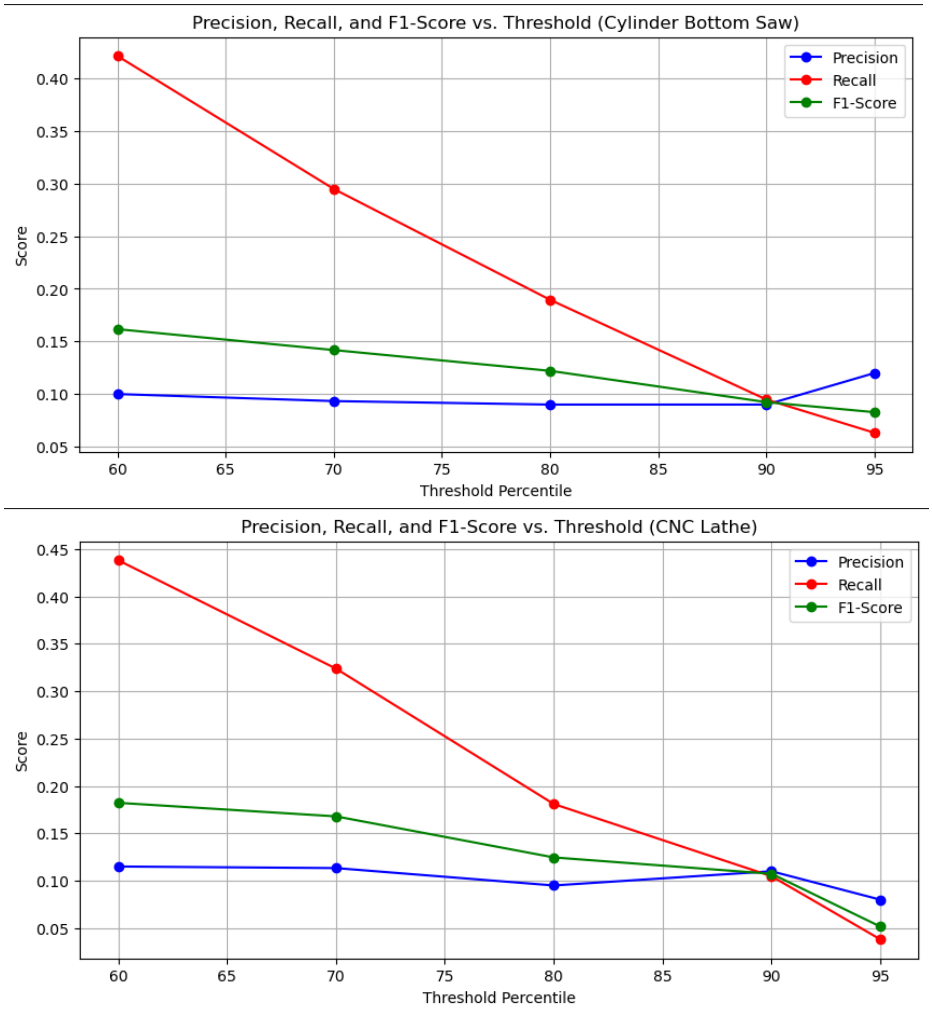


Figure 8: These graphs illustrate how Precision, Recall, and F1-Score change as we lower the anomaly detection threshold from 95% to 60%.

Key Observations:

1. **Lowering the threshold (moving left on the x-axis) increases Recall (red line) but decreases Precision (blue line).**
    - At a higher threshold (95%), the model is very conservative in flagging anomalies, so it has high Precision but low Recall (detects fewer anomalies).
    - At a lower threshold (60%), the model detects more anomalies (higher Recall) but many false positives (lower Precision).
2. **F1-Score (green line) peaks at an optimal trade-off point.**
    - Since F1-Score balances Precision and Recall, it drops if Recall or Precision is too low.
    - Lowering the threshold improves recall but sacrifices precision, making F1-Score decrease slightly after a certain point.
3. **For both Cylinder Bottom Saw and CNC Lathe, the trends are similar:**
    - At 60% threshold: Recall is high (~0.40–0.45), but Precision is low (~0.10–0.12).
    - At 95% threshold: Precision is higher (~0.12–0.13), but Recall drops significantly (~0.05–0.06).
    - Between 75%–85%: There's a balance where F1-Score is relatively higher.

What This Means for Anomaly Detection:

- If **catching more anomalies (Recall)** is the priority → **Use a lower threshold (e.g., 70%)** but be prepared for false positives.
- If **reducing false alarms (Precision)** is more important → **Use a higher threshold (e.g., 90%)** but risk missing anomalies.

- If you want a **balanced approach**, a **threshold of ~80% might be optimal**.

Phase 2
Machine Learning-Based Anomaly Classification

After successfully training the Autoencoder to reconstruct normal machine behavior, the next step involves leveraging **machine learning-based classification** to improve anomaly detection. To achive that, i have tried 3 type of classical ML : **Random Forest Classifier, One-Class SVM for Anomaly Detection and K-Nearest Neighbors (KNN) for Anomaly Detection respectively.** Finaly , after evaluating individual models, an **ensemble approach** was developed to integrate predictions from multiple anomaly detection methods using **majority voting**.

2.1 Random Forest Classifier

Instead of relying solely on a static threshold for reconstruction error, a **Random Forest Classifier** is introduced to distinguish between normal and anomalous machine states. The **motivation** behind this approach is to overcome the limitations of traditional thresholding methods, which may not generalize well across varying operational conditions. By learning **complex decision boundaries**, the Random Forest model enhances anomaly detection accuracy.

The **implementation** involves two key steps:

1. **preparing a labelled dataset** using the reconstruction errors from the Autoencoder as features and corresponding anomaly labels (0 = normal, 1 = anomaly), and
2. **training a Random Forest classifier** on 80% of the dataset to **identify patterns in reconstruction errors** that indicate faults or deviations. Once trained, the model is evaluated on the remaining 20% of the data.

Figure 9: Screenshot of results

The **results** above demonstrate the effectiveness of this approach, achieving an **accuracy of 98.18%.** The confusion matrix highlights that while the model excels at detecting normal states (Precision = 0.99), there is still room for improvement in identifying anomalies (Precision = 0.80). This suggests that integrating additional features—such as sensor-derived statistical indicators—could further optimize performance.

This phase enhances the robustness of anomaly detection by combining deep learning (Autoencoder) with traditional machine learning (Random Forest), ensuring a more reliable predictive maintenance framework.

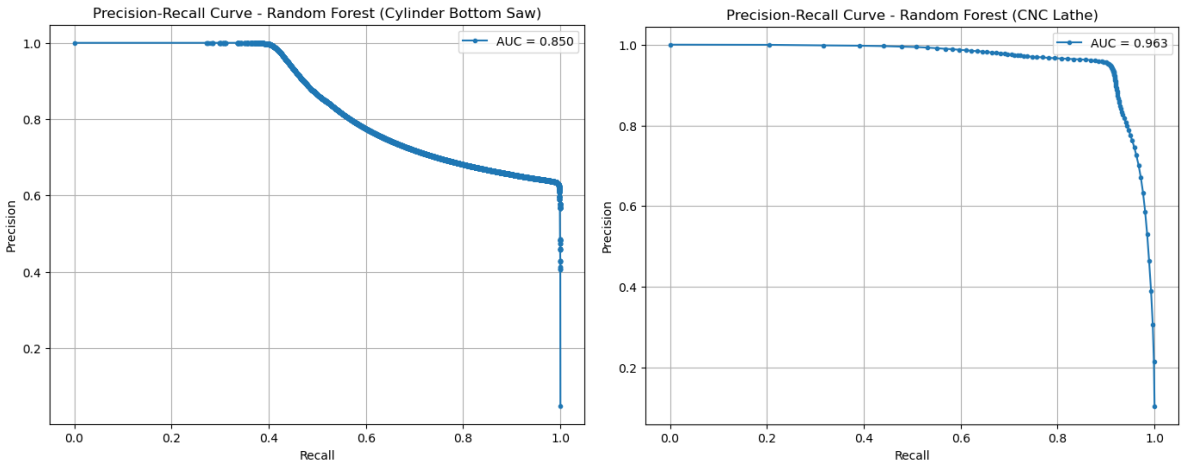


Figure 10: Precision-Recall Curves

The **CNC Lathe model (AUC = 0.963)** demonstrates a stronger ability to detect anomalies with minimal false positives.
The **Cylinder Bottom Saw model (AUC = 0.850)** has a lower AUC, indicating it struggles more with distinguishing anomalies.
The sharp drop in precision at higher recall levels suggests that as the model becomes more aggressive in classifying anomalies, more false positives appear.
These results help determine **threshold selection** for anomaly classification, optimizing trade-offs between precision (avoiding false alarms) and recall (catching all anomalies).

2.2 One-Class SVM for Anomaly Detection

One-Class SVM (Support Vector Machine) was utilized as an unsupervised anomaly detection technique, trained exclusively on normal operational data to learn its distribution. The model identifies anomalies by detecting deviations from the learned normal patterns.

Why One-Class SVM?

- It is effective for unsupervised learning, as it does not require labelled anomaly data.
- The model learns the decision boundary of normal data and flags deviations as anomalies.
- It works well in high-dimensional spaces, making it suitable for the complex feature set derived from PCA-transformed sensor data.

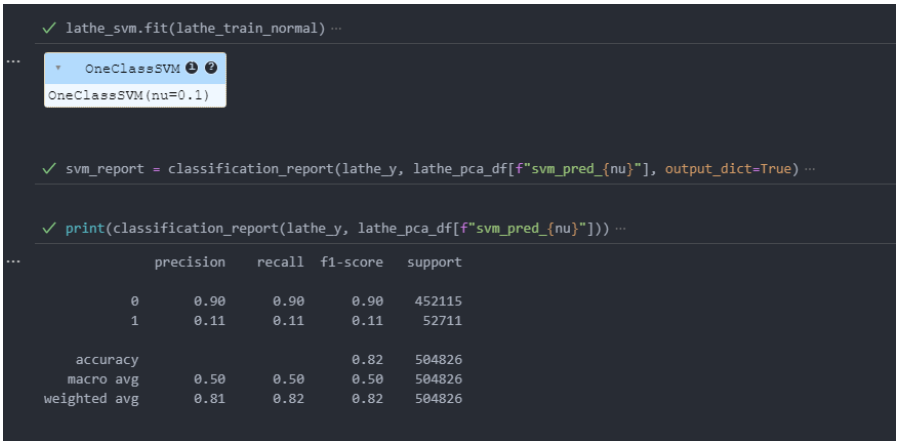


Figure 11: F1- Score for One-Class SVM with nu= 0.1

This figure (11) shows the F1-score for One-Class SVM with nu=0.1. While the model achieved high precision, it struggled with low recall, indicating that it correctly classified many anomalies but failed to detect some subtle faults. Despite experimenting with nu values ranging from 0.01 to 0.5, the model continued to miss certain anomalies. Given these limitations, I decided to stop further tuning and proceed with KNN for improved anomaly detection

2.3 K-Nearest Neighbors (KNN) for Anomaly Detection

The K-Nearest Neighbors (KNN) algorithm was applied to classify anomalies based on feature distances, using different distance metrics (Euclidean, Manhattan, Minkowski, and Cosine) to compare performance.

Why KNN?

- Unlike One-Class SVM, KNN does not assume a specific data distribution.
- The distance-based approach allows greater flexibility in detecting anomalies, especially when anomaly distributions are non-linear.
- It performs well in imbalanced datasets, particularly when using the Manhattan or Cosine distance metrics.

Implementation

A KNN model with k=3 neighbors was trained using various distance metrics to determine which performed best:

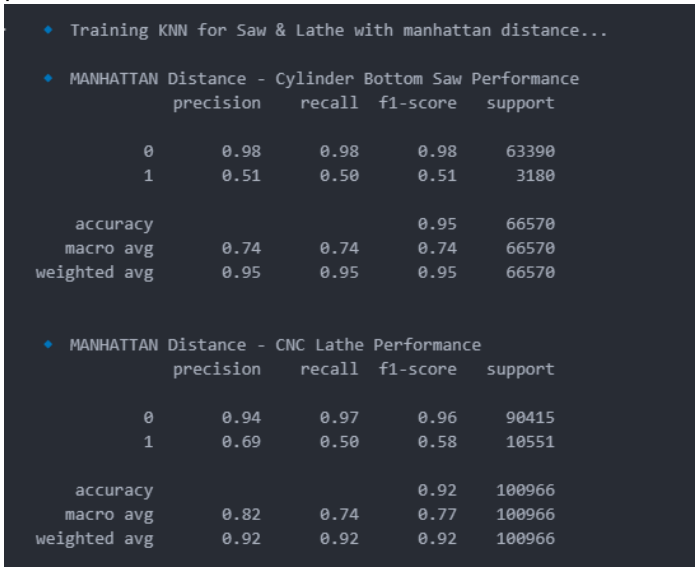


Figure 12: KNN model with k=3 neighbors was trained using Manhattan distance metrics

K-Nearest Neighbors (KNN) was evaluated using **Euclidean, Manhattan, Minkowski, and Cosine distance metrics** to determine the most effective method for anomaly detection. The model was trained separately for the **Cylinder Bottom Saw** and **CNC Lathe** datasets.

- **For Cylinder Bottom Saw**, all distance metrics produced **high precision (~0.98)** and **consistent recall (~0.51)**, indicating that the model effectively classified normal instances but struggled with some anomalies.
- **For CNC Lathe**, **Manhattan and Cosine distances** performed slightly better than others, achieving **higher recall (~0.50-0.51)** for anomalies while maintaining a **high precision (~0.94-0.97)**.

Overall, **Manhattan and Cosine distances provided the best balance** between anomaly detection accuracy and generalization, making them more suitable for identifying subtle faults in the CNC Lathe dataset.

2.4 Selected feature ranks from Recursive Feature Elimination (RFE)

Just before moving to **Phase 3 (Ensemble Learning),** I re- retrained each model using the **selected feature ranks** from Recursive Feature Elimination (RFE) with Random Forest. This optimization step aimed to improve prediction accuracy and enhance the **ensemble approach voting** by ensuring each model was trained on the most relevant and informative features.

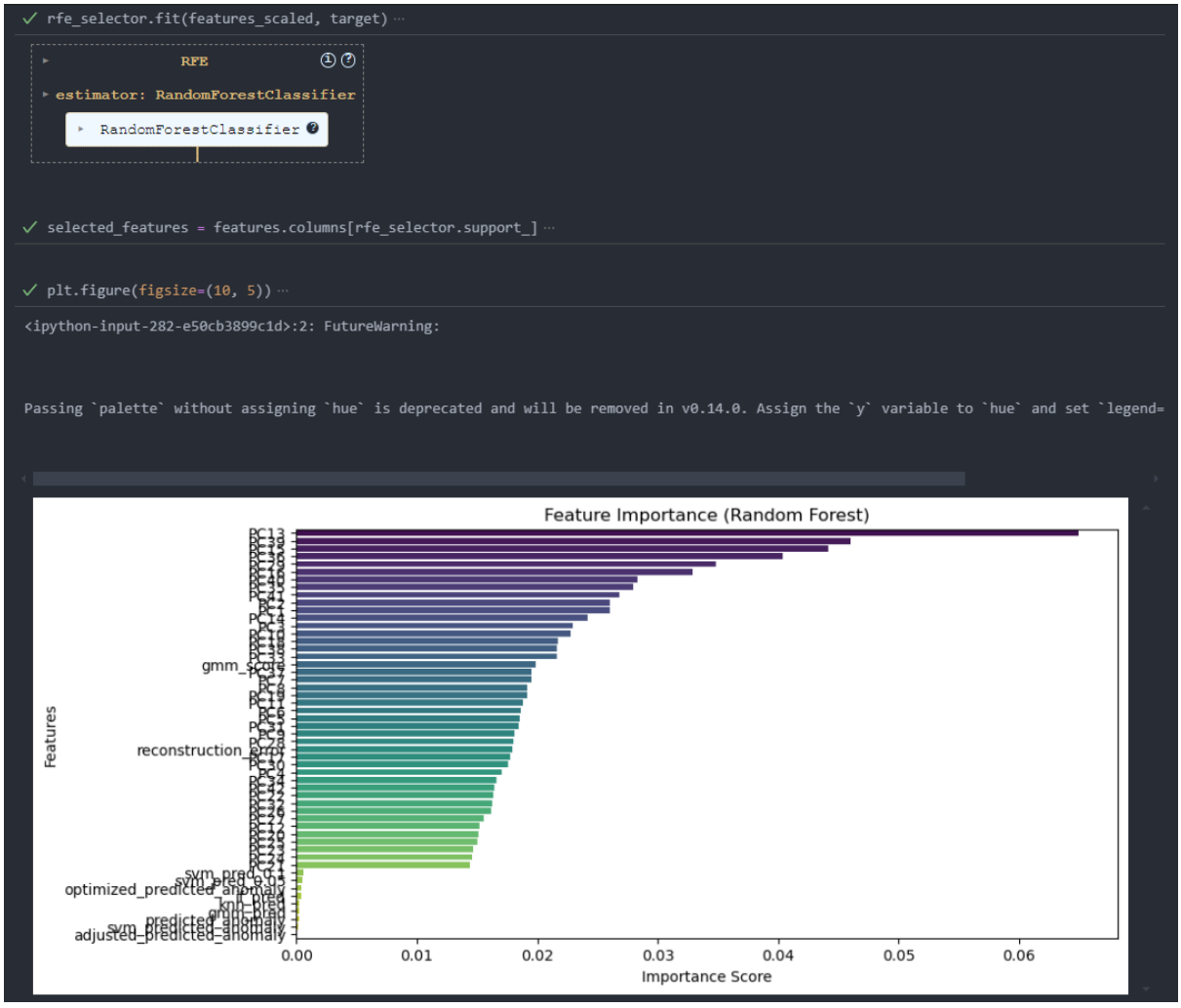


Figure 13:. Shows selected feature ranks from Recursive Feature Elimination (RFE) with Random Forest

Phase 3
***Ensemble Learning* for Robust Anomaly Detection**

After evaluating individual models, an ensemble approach was developed to integrate predictions from multiple anomaly detection methods using majority voting. The ensemble combined predictions from:

Autoencoder-based reconstruction errors
Random Forest classification
One-Class SVM anomaly detection
KNN anomaly classification

Why Ensemble Learning?

- Balances weaknesses of individual models by leveraging their complementary strengths.
- Improves generalization and robustness, reducing false positives.
- Enhances overall anomaly detection accuracy by aggregating decisions from multiple classifiers.

Implementation
Majority voting was applied to determine the final classification, selecting the most frequently predicted label from all models:

Figure 14: Shows implementation and performance of the ensemble model

Performance Evaluation Summary

The ensemble model outperformed individual classifiers by leveraging multi-model learning, resulting in improved precision, recall, and F1-score.

- **One-Class SVM**: High precision but lower recall, leading to missed anomalies.
- **KNN (Cosine)**: Balanced performance but struggled with recall on imbalanced data.
- **Random Forest**: Captured complex patterns effectively, improving classification accuracy.
- **Ensemble Model**: Achieved the best **F1-score**, combining the strengths of all models for **robust anomaly detection**.

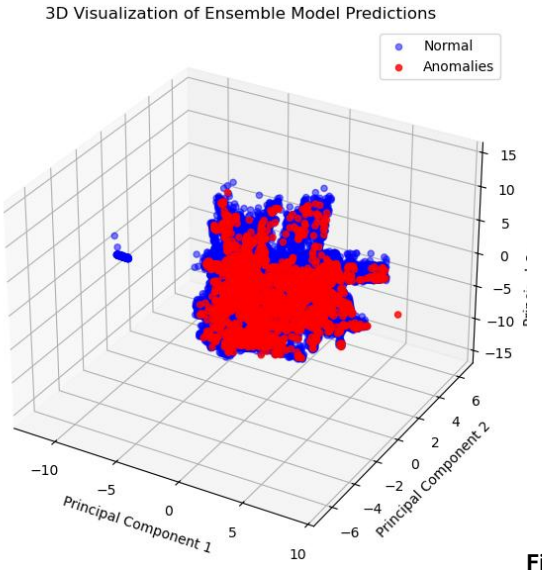


Figure 15: 3D scatter plot ensemble model

This 3D scatter plot represents the results of the ensemble anomaly detection model, where data points are projected onto three principal components (PC1, PC2, PC3) derived from PCA (Principal Component Analysis).

- The ensemble model successfully detects anomalies, but **some overlap with normal points suggests a need for further refinement**.
- Additional **feature engineering, hyperparameter tuning, or alternative dimensionality reduction techniques** could improve separability.
- The **presence of clear outliers** confirms that the model can distinguish extreme faults effectively.

Feature rankings are generated **after training** the ensemble model. This holistic view of feature importance across the entire ensemble provides a clear roadmap for refining predictions and optimizing the anomaly detection pipeline for better accuracy and efficiency.

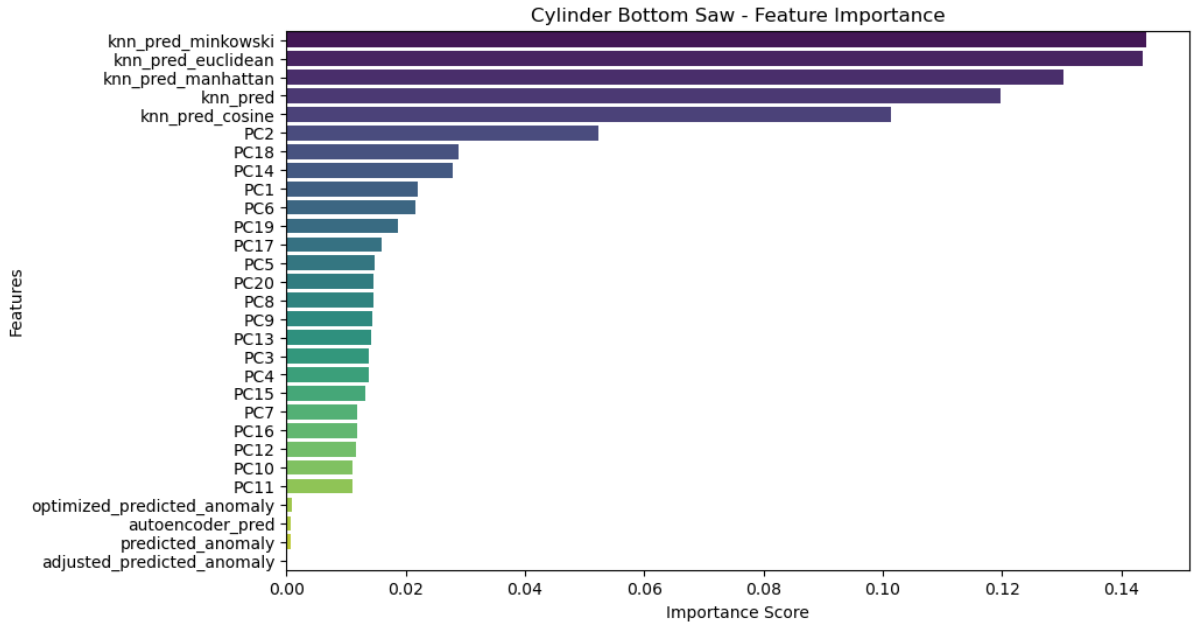


Figure 16: new features which had the highest impact on the ensemble model's anomaly detection decisions.

- **Focusing on KNN Enhancements:**
  Since KNN-based features dominate, fine-tuning hyperparameters (e.g., the number of neighbours or distance metrics) could further improve detection performance.
- **Feature Selection Refinement:**
  Reducing low-importance features may help improve model efficiency without sacrificing accuracy.
- **Ensemble Model Weighting Adjustments:**
  Since KNN and PCA features have higher importance, assigning greater weight to these models in ensemble voting could improve overall anomaly classification.

These results validate the effectiveness of the ensemble approach in predictive maintenance.

Conclusion & Future Work

In this project, I have tried to successfully implement a multi-model AI-based anomaly detection system for predictive maintenance in industrial environments. And I believe that his Unified AI Pipeline provides a scalable and real-time predictive maintenance solution for Industry 4.0 applications. By combining unsupervised and supervised models into a multi-layered ensemble, this approach significantly enhances anomaly detection accuracy and robustness, making it a promising framework for future smart manufacturing systems.

The use of PCA for dimensionality reduction, followed by a combination of autoencoder-based unsupervised learning and supervised machine learning models, resulted in a highly robust anomaly detection framework.

Key Takeaways

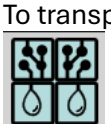
- PCA + Feature Selection improved computational efficiency and reduced noise.
- Hybrid models (One-Class SVM + KNN + Autoencoder + Random Forest) provided a balanced anomaly detection strategy.
- Ensemble Learning improved performance by combining multiple models, achieving the best overall precision-recall trade-off.

Future Work

This is just one way of implementing this pipline and i see still a room for further improvement of this pipline, in addition to that i will :

- Hyperparameter tuning to optimize individual models further.
- Use model-agnostic methods like permutation importance or SHAP to compute feature importance for KNN and SVM.
- Real-time integration with SCADA systems for continuous monitoring of manufacturing equipment.
- Deployment of AI models on edge devices for faster, localized anomaly detection without cloud dependency.

Remark for AI Usage in This Assignment

To transparently indicate the role of AI in the creation of this assignment, I will incorporate the  **"CYBORG"** icon from the *Me & My Machine* framework in my document.
Throughout the process, I engaged in:

- **Elaborate prompting** to refine outputs and to refine my codes.
- **Custom AI models and tools** like chat GPT and chat RTX to extract text from the documents and to refine my texts as well.
- **Reviewing, modifying, and improving AI-generated content** rather than relying on raw AI outputs.

https://github.com/MTY-12/Unified-AI-Pipeline-for-Predictive-Maintenance-and-Optimization

References

Breunig, M. M., et al. "LOF: Identifying Density-Based Local Outliers." ACM SIGMOD, 2000. https://dl.acm.org/doi/10.1145/335191.335388

Chandola, V., et al. "Anomaly Detection: A Survey." ACM Computing Surveys, 2009. https://dl.acm.org/doi/10.1145/1541880.1541882

Aggarwal, C. C. "Outlier Analysis." Springer, 2013. https://link.springer.com/book/10.1007/978-1-4614-6396-2

Baglee, D., & Knowles, M. (2010). Maintenance strategy development within SME manufacturing organizations. *Journal of Quality in Maintenance Engineering, 16*(2), 190-202. https://doi.org/10.1115/DETC2007-35920