ECOLE SUPÉRIEURE D'INGÉNIEURS LÉONARD DE VINCI

Advanced Machine Learning I: Project Report

Smart Predictive Maintenance for Industrial Equipment

AUTHOR
ESILV A5, Data Science & Artificial Intelligence – DIA2, Class of 2025

LAPILUS Joyce

2025-01-08

Contents

1 Introduction

The purpose of this project was to design and implement a predictive maintenance system capable of forecasting machinery failures before they occur. The ultimate goal was to optimize maintenance schedules, reduce unexpected downtime, and minimize operational costs in an industrial setting. By leveraging data-driven techniques, the system enables proactive decision-making for maintenance, which is critical in industries where equipment failures can have significant economic and operational consequences.

The project utilized NASA's C-MAPSS dataset [1] which consists of multivariate time-series data collected from turbofan engines operating under varying conditions. The data reflects the degradation of engines over time, ultimately leading to failure. This unique feature of the dataset allowed training and validation of machine learning models tailored for predictive maintenance tasks.

Key components of this system include ensemble learning, strategies for handling imbalanced data, reinforcement learning for maintenance scheduling, and the use of AutoML to streamline model selection and tuning. Each of these components was carefully designed and implemented to address specific challenges associated with predictive maintenance in real-world scenarios.

2 Objectives

The project aimed to achieve the following:

1. Identify machine failures within a specific time window, based on historical sensor data.
2. Address challenges associated with highly imbalanced datasets, where failures represent rare occurrences.
3. Implement reinforcement learning to optimize maintenance scheduling, striking a balance between downtime and maintenance costs.
4. Automate the model development process through AutoML to ensure efficiency and scalability.

3 Problem Definition

The goal was to predict whether a machine will fail within a specific time window based on historical sensor data. Additionally, a maintenance scheduling policy was developed using reinforcement learning to minimize costs associated with downtime and unexpected failures.

Key challenges included:

1. Class imbalance in the dataset due to rare failure events.
2. Handling high-dimensional, multivariate time-series data.
3. Framing the time-series data into a classification problem for effective prediction.
4. Balancing maintenance costs and machine uptime through reinforcement learning.

4 Methodology

This project followed a systematic workflow designed to achieve the outlined objectives, incorporating state-of-the-art machine learning methodologies and tools.

4.1 Data Exploration and Preprocessing

The NASA C-MAPSS dataset, consisting of multivariate time-series data, was analyzed to understand its structure and variability. So, The preprocessing pipeline was developed to ensure the dataset was clean, consistent, and suitable for predictive modeling.
Remaining Useful Life (RUL) was calculated for each engine to facilitate failure prediction. Then, based on the RUL, the failure status has been added to each row; a failure is considered true if the RUL is below 50. Missing values were addressed using imputation techniques, while feature engineering extracted temporal patterns, such as rolling averages, to better capture degradation trends over time. Scaling was applied to ensure uniformity across features, facilitating the training of machine learning models. The dataset was normalized, and feature selection identified the most predictive variables.

4.2 Modeling and Imbalance Handling

Predictive modeling was carried out using Random Forest (RF) and XGBoost (XGB), two robust ensemble techniques. To address the imbalance in the dataset, the following approaches were explored:

1. **Cost-sensitive (CS) learning** by assigning higher weights to the minority class during training.
2. **Synthetic Minority Oversampling Technique (SMOTE)** to generate synthetic samples for the minority class.
3. **Random Under-Sampling (RUS)** to balance the dataset by reducing instances of the majority class.

4.3 Ensemble Learning

Two ensemble methods were employed to improve predictive accuracy:

1. **RF (Bagging)**: This model was utilized to enhance robustness and reduce variance.
2. **XGB (Boosting)**: This model leveraged iterative training to focus on difficult-to-classify samples, achieving high accuracy in imbalanced settings.

Both methods were evaluated using metrics such as accuracy, precision, recall, F1-score, and AUC-ROC.

4.4 Reinforcement Learning for Maintenance Optimization

To optimize maintenance schedules, a reinforcement learning framework based on **Q-learning** was implemented. The framework defined:

- States: Representing the health of the machinery (e.g., healthy, moderate wear, severe wear, failed).
- Actions: Maintenance or no maintenance.
- Rewards: Structured to encourage healthy operation and penalize failure or unnecessary maintenance. For example:
  - Healthy + No Maintenance: +10 points
  - Moderate Wear + No Maintenance: -20 points
  - Severe Wear + No Maintenance: -50 points
  - Maintenance: Penalty proportional to maintenance costs.

The Q-learning algorithm learned an optimal policy by iteratively exploring the action space and updating Q-values based on rewards. The learned policy alternated between maintenance and no maintenance actions based on the current state, effectively balancing operational efficiency with cost minimization.

4.5 AutoML for Workflow Automation

The AutoML framework TPOT was employed to automate model selection and hyperparameter tuning. TPOT explored various algorithms and parameter combinations, ultimately selecting K-Nearest Neighbors (KNN) as the best-performing model. KNN outperformed manually tuned models in terms of both accuracy and recall, as it will be showcased in Results.

The AutoML workflow demonstrated the value of automation in optimizing machine learning pipelines, reducing manual effort and improving model performance. But it also proved to be really time-consuming, taking more than 2 hours to fully execute, even when using the GPU configuration.

5 Results

5.1 Model Performance Metrics

Table 1 summarizes the performance of different models, imbalance-handling techniques, and the AutoML-selected pipeline.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC
RF	94.9	99.48	94.9	97.13	0.48
XGB	94.67	99.48	94.67	97.02	0.47
RF with SMOTE	93.59	99.48	93.59	96.45	0.47
XGB with SMOTE	94.32	99.48	94.32	96.83	0.47
RF with RUS	92.17	99.48	92.17	95.69	0.46
XGB with RUS	91.97	99.48	91.97	95.58	0.46
RF with CS	95.04	99.48	95.04	97.21	0.48
XGB with CS	93.85	99.48	93.85	96.59	0.47
KNN	**95.2**	99.48	**95.2**	**97.3**	0.48

Table 1: Model Performance Metrics

The KNN model achieved the highest F1-score, recall, and accuracy, making it the preferred model for deployment.

5.2 Analysis of Results

- The **cost-sensitive learning approach** consistently outperformed other imbalance-handling methods across all models, achieving higher recall and F1-scores.
- The **AutoML-selected K-Neighbors Classifier** (KNC) delivered the best overall performance, achieving 95.2% accuracy and an F1-score of 97.3%.
- AUC-ROC values were relatively low, attributed to dataset characteristics but remained consistent across models.

5.3 Visualizations

5.3.1 Feature Importance

The most influential features are exclusively related to sensor values such as `sensor_4` and `sensor_11`, which captured critical operational metrics.

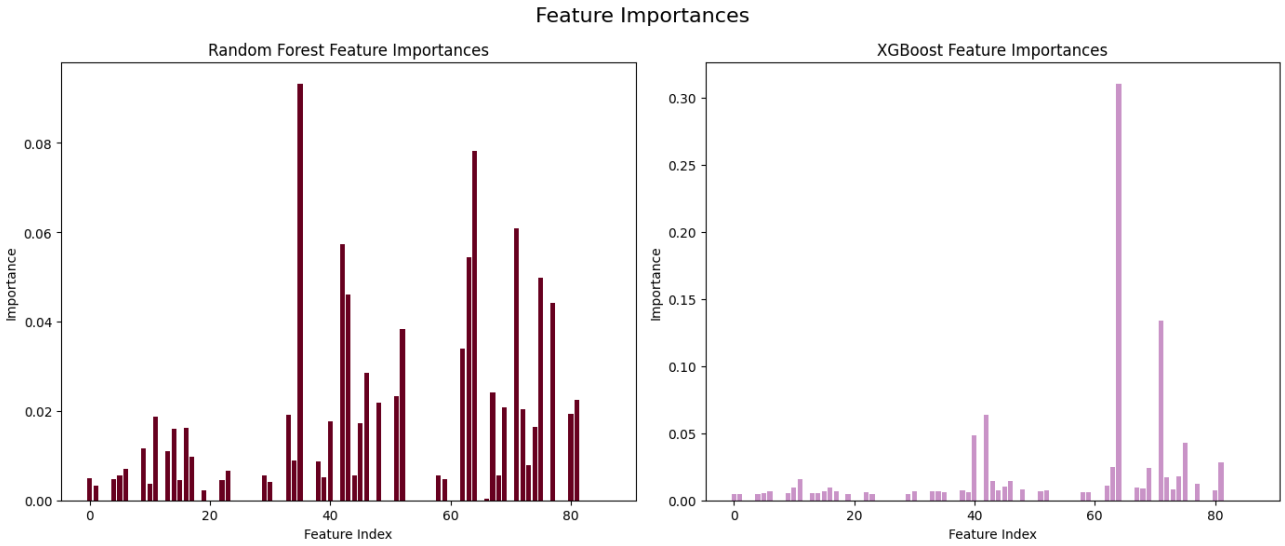

Figure 1: Models' Feature Importances

Ranking	Models	
	RF	XGB
1	`sensor_measurement_4_rolling_mean_3`	`sensor_measurement_4_rolling_mean_5`
2	`sensor_measurement_4_rolling_mean_5`	`sensor_measurement_11_rolling_mean_5`
3	`sensor_measurement_11_rolling_mean_5`	`sensor_measurement_11_rolling_mean_3`
4	`sensor_measurement_11_rolling_mean_3`	`sensor_measurement_9_rolling_mean_3`
5	`sensor_measurement_3_rolling_mean_5`	`sensor_measurement_15_rolling_mean_5`

Table 2: Models' Top 5 Important Features

It is worth to notice that both models share important features, such as `sensor_measurement_4_rolling_mean_5`, `sensor_measurement_11_rolling_mean_5` and `sensor_measurement_11_rolling_mean_3`. Their order stays the same, but RF does seem to prioritize another feature instead of the aforementioned columns.

5.3.2 Decision Boundaries

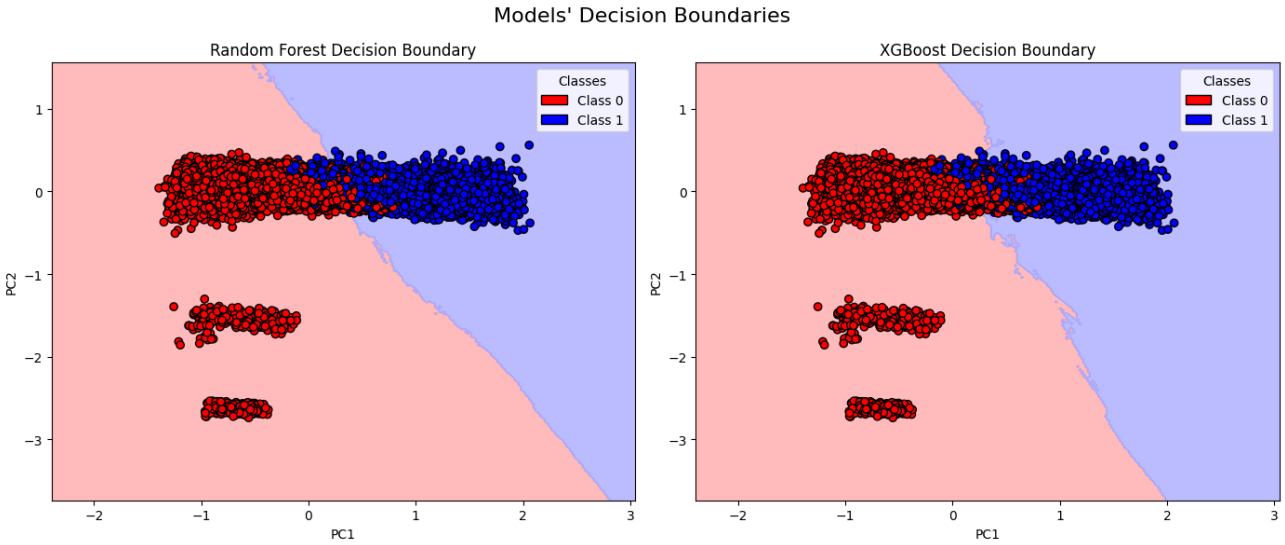

Figure 2: Models' Decision Boundaries

These visualizations depict the decision boundaries of RF and XGB on the first two principal components of the dataset.

The decision boundary is more rigid and piecewise-linear, characteristic of an ensemble of decision trees. RF seems to divide the space cleanly, but it may fail to capture more nuanced patterns in the data.
The decision boundary is smoother and appears to adapt better to the dataset's underlying structure. XGB likely captures more complex relationships between features due to its gradient-boosting approach.

The red and blue scatter points represent two classes (e.g., "failure" and "no failure"). There is considerable overlap between the classes, which aligns with the low ROC AUC values observed earlier in the Model Performance Metrics.

This overall showcases that XGB shows better generalization in handling overlapping regions compared to RF.

5.3.3 AUC-ROC Curve

The AUC-ROC curve for the KNC highlighted its robust discriminatory ability.

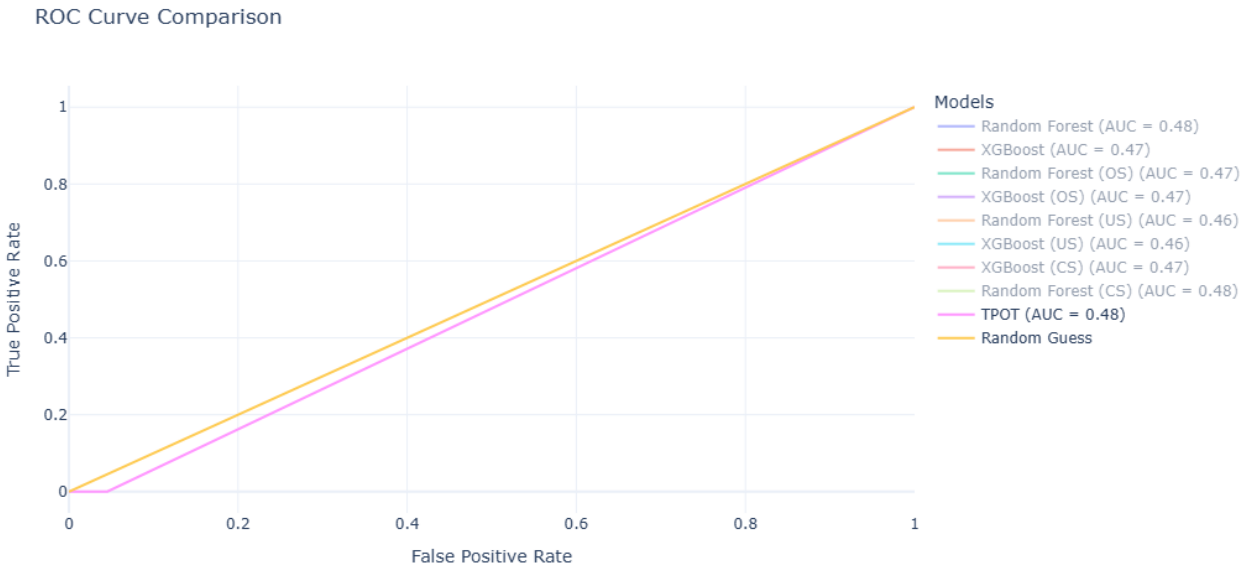

Figure 3: ROC Curve for the AutoML-selected pipeline: K-Neighbors Classifier

The ROC curve provides a comparison of the true positive rate (sensitivity) against the false positive rate for the AutoML pipeline (TPOT).

The ROC curves are almost overlapping, indicating that the discriminative power of these models is quite similar. The model hovers around the diagonal (random guess), suggesting that there might be challenges in model generalization or imbalanced data handling. KNC performs similarly to manually tuned models, indicating its effectiveness in automating the pipeline but not necessarily outperforming the best manually tuned models.

Further improvements might be needed, such as fine-tuning feature engineering or exploring advanced imbalance-handling methods, to significantly boost ROC AUC values. Class imbalance remains a key bottleneck in improving model performance, as also shown in Figure 4.

5.3.4 Confusion Matrix

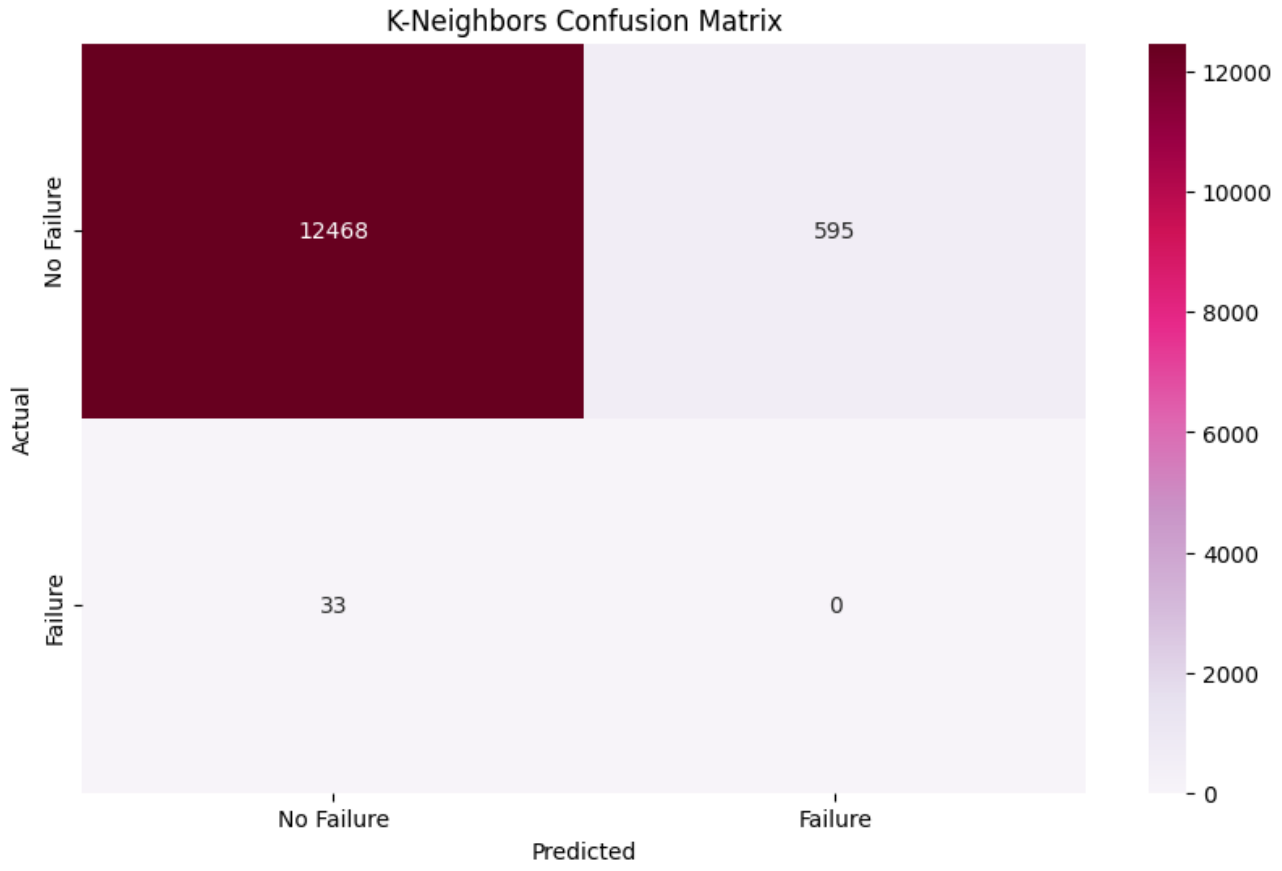

Figure 4: Confusion Matrix of KNC's Predictions

This confusion matrix suggests that while the K-Neighbors model achieves high accuracy overall, it is unsuitable for scenarios where detecting failures is critical. To address this limitation, techniques such as oversampling the minority class, undersampling the majority class, or using class-sensitive algorithms have been considered, but to no avail, the performances were actually poorer. These approaches could help improve the model's performance on the minority class while maintaining reasonable accuracy for the majority class.

5.3.5 Maintenance Schedule Visualization

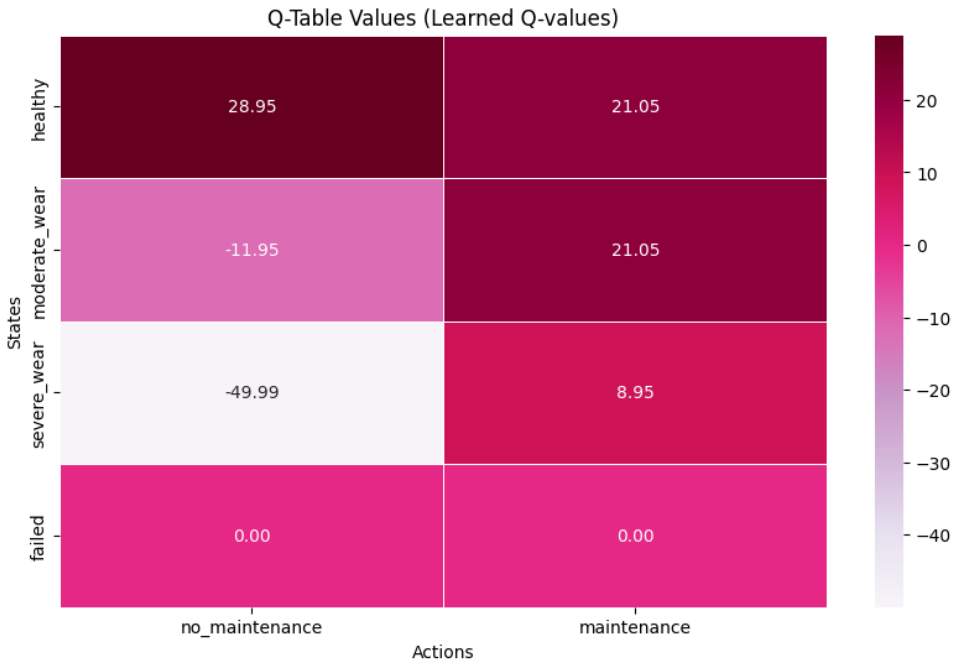

Figure 5: Learned Q-Table Values

The Q-table heatmap illustrates the learned Q-values for each state-action pair in the reinforcement learning model. These values guide optimal maintenance decisions based on the state of the system, balancing maintenance costs and operational risks.
The main takeaways to observe from this Q-Table heatmap are the following:

- **Healthy State**: The highest Q-value (28.95) corresponds to `no_maintenance`, indicating that maintenance is unnecessary and costly when the system is healthy.
- **Moderate Wear State**: `maintenance` has a significantly higher Q-value (21.05) than `no_maintenance` (-11.95), recommending timely maintenance to prevent further degradation.
- **Severe Wear State**: Immediate maintenance is critical, as its Q-value (8.95) far exceeds that of `no_maintenance` (-50.00), which risks failure.
- **Failed State**: Both actions have a Q-value of 0.00, reflecting no benefit from further actions once failure occurs.

The Q-values reinforce a logical policy: avoid unnecessary maintenance when the system is healthy, perform maintenance in moderate and severe wear states, and recognize the futility of

actions in a failed state. This approach minimizes costs while preventing failures.

For improvement, these steps could be applied:

- Validate the policy under varying conditions.
- Integrate predictive models for real-time state estimation.
- Extend the model for complex systems with additional operational states.

The Q-table effectively captures a cost-efficient maintenance strategy. By optimizing maintenance schedules, it reduces downtime and aligns with operational reliability goals, providing a solid foundation for advanced predictive maintenance systems.

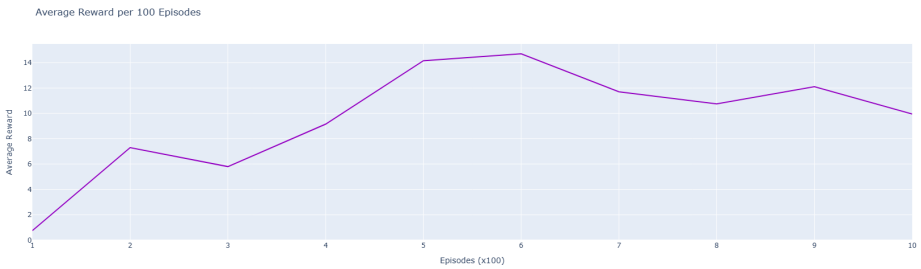

Figure 6: Average reward per 100 episodes for the learned Q-Learning table

This plot shows the evolution of the average reward across episodes during the training of the Q-learning algorithm for maintenance scheduling.

As training progresses, the rewards continue to increase steadily, peaking around the midpoint of the episodes. This period indicates significant learning and optimization as the agent balances the trade-offs between maintenance costs and avoiding failures.
In the later episodes, the rewards show minor fluctuations but remain relatively stable, indicating that the agent has reached a level of convergence. These fluctuations are likely due to the stochastic nature of exploration during training. However, the overall stability of the rewards highlights the robustness of the learned policy, which effectively minimizes costs while maximizing operational efficiency.

Reinforcement learning shows promise for optimizing maintenance schedules by balancing the trade-off between maintenance costs and minimizing downtime. Future work could involve extending this to more complex state-action spaces and incorporating deep reinforcement learning for scalability.

5.4 Discussion

The project demonstrated the integration of advanced machine learning techniques to address the challenges of predictive maintenance. The ensemble models, particularly RF with class-sensitive learning, achieved excellent results when it comes to accuracy. AutoML further enhanced model performance by automating the pipeline.

The Q-learning approach to maintenance scheduling provided a practical and scalable solution for balancing operational efficiency with maintenance costs.

5.4.1 Challenges and Limitations

- The relatively low AUC-ROC values, visualized for all models in Figure 7, indicate that further refinement is needed to improve the separation between true positives and false positives.
- The reinforcement learning framework assumed fixed state transitions, which may not fully capture real-world variability.

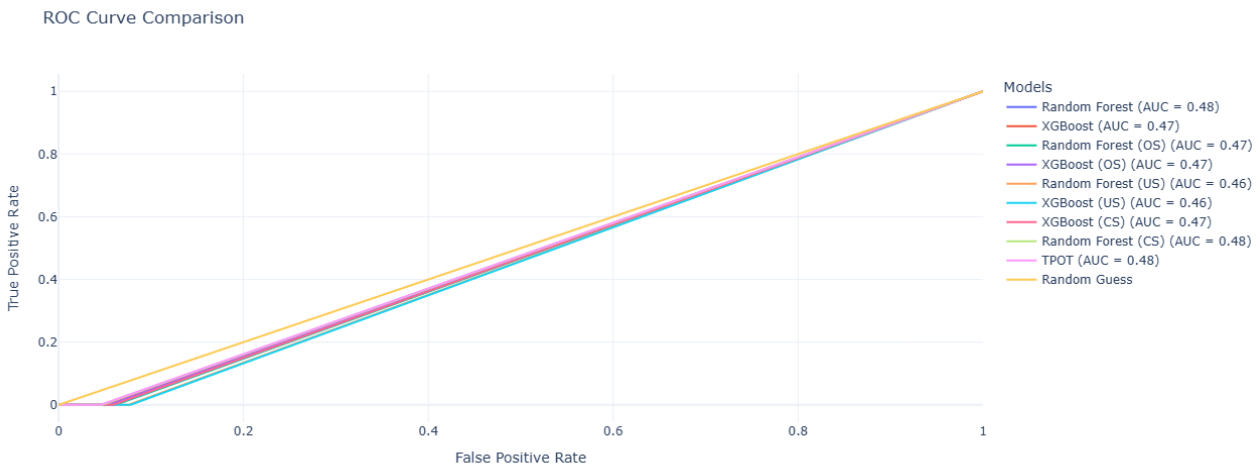

Figure 7: ROC Curves for All Tested Models

6 Conclusion

This project successfully developed a predictive maintenance system for industrial machinery. Key achievements include:

1. High predictive accuracy for machine failures, with the best model achieving 95.2% accuracy and an F1-score of 97.3%.
2. Effective imbalance handling using cost-sensitive learning, outperforming traditional resampling techniques.
3. Automated workflow optimization via AutoML, streamlining model selection and hyperparameter tuning.
4. Reinforcement learning-driven maintenance scheduling, providing actionable insights to reduce downtime and costs.

6.1 Future Work

To further enhance the solution, the following directions could be applied:

1. **Explainable AI**: Implement SHAP or LIME for interpretable predictions, aiding decision-makers in understanding model behavior.
2. **Scalability Testing**: Apply the solution to additional datasets from various industries to validate robustness and generalizability.
3. **Deep Learning Exploration**: Investigate RNNs and transformers to improve predictions on time-series data.

This project demonstrates the potential of machine learning in revolutionizing predictive maintenance, paving the way for cost-effective and reliable industrial operations.

List of Figures

List of Tables

Nomenclature

AUC Area Under the Curve

AutoML Automated Machine Learning

CS Class-Sensitive

GPU Graphics Processing Unit

KNC K-Neighbors Classifier

KNN K-Nearest Neighbors

RF Random Forest

ROC Receiver Operating Characteristic

RUL Remaining Useful Life

RUS Random Under-Sampling

SMOTE Synthetic Minority Oversampling Technique

TPOT Tree-based Pipeline Optimization Tool

XGB XGBoost

References

[1] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation." https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6/about_data, oct 2008. Accessed: 2024-12-31.