**Scania Predictive Maintenance Project**

**1. Introduction**

Modern industrial and automotive systems generate a massive amount of sensor data that can be used to monitor performance and detect early signs of failure. Unexpected breakdowns not only lead to operational disruptions but also increase maintenance costs and reduce efficiency.

This project focuses on building a machine learning model to predict potential equipment or component failures based on sensor readings. The dataset contains anonymized operational variables recorded under different conditions. The objective is to develop a predictive model that can help organizations shift from reactive maintenance to predictive maintenance.

**2. Business Challenge**

Unplanned equipment failures can result in:

* Increased maintenance and repair costs
* Downtime and productivity loss
* Safety risks and reduced reliability

By using sensor data to identify potential failures in advance, businesses can:

* Schedule maintenance proactively
* Optimize resource allocation
* Improve operational efficiency and equipment lifespan

Thus, the key challenge is to analyze complex sensor data, detect patterns linked to failure events, and build a reliable prediction system.

**3. Dataset Overview**

* Source: Scania sensor dataset
* Total Features: 171 anonymized operational variables
* Target Variable: class
  + 1 (Positive): Indicates a failure
  + 0 (Negative): Normal operation

Data Characteristics

* Each feature represents an aggregated or processed sensor measurement.
* Several columns are histogram-based (e.g., component\_XX\_bin\_Y), representing the frequency of certain operational states.
* Missing values are denoted by "na".
* The dataset is highly imbalanced, with very few failure cases compared to normal ones.

**4. Data Preprocessing**

Steps Performed

1. Missing Value Imputation
   * Replaced "na" values with column means to retain as much data as possible.
2. Normalization
   * Standardized features using z-score normalization to ensure equal contribution of all sensors to the model.
3. Handling Class Imbalance
   * Implemented a custom SMOTE (Synthetic Minority Oversampling Technique) from scratch without using scikit-learn.
   * Oversampled minority class (failures) by generating synthetic samples based on existing feature space.
   * Balanced class distribution improved the model’s ability to learn from failure cases.
4. Data Shuffling
   * Used shuffle() from NumPy to randomize data order, ensuring unbiased model training.

**5. Exploratory Data Analysis (EDA)**

EDA was performed to understand sensor behaviour and its relation to failures.

* Bar Plots: Plotted feature missing values so
* These findings helped guide model selection — Gaussian-based models can perform well.

**6. Model Selection**

Models Considered

1. Gaussian Naive Bayes (GNB):
   * Assumes each feature follows a Gaussian distribution.
   * Simple, interpretable, and computationally efficient.
2. Logistic Regression (Custom Implementation):
   * Estimates the probability of failure using a linear relationship between features and outcome.
   * Good at balancing recall and precision.

**Decision**

Both Logistic Regression and Gaussian Naive Bayes were tested.  
GNB performed reasonably well but was outperformed by Logistic Regression in balancing recall and precision.

**7. Model Training and Evaluation**

Training Process

* Implemented both models without using scikit-learn, relying only on NumPy and Pandas.
* Split dataset into 80% training and 20% testing.
* Evaluated using the following metrics:

|  |  |  |
| --- | --- | --- |
| Metric | Formula | Purpose |
| Accuracy | (TP + TN) / Total | Overall correctness of predictions |
| Precision | TP / (TP + FP) | Quality of positive predictions |
| Recall | TP / (TP + FN) | Ability to detect failures |
| F1-Score | 2 × (Precision × Recall) / (Precision + Recall) | Balance between precision and recall |

**8. Results Summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-Score |
| Gaussian Naive Bayes | 0.9639 | 0.3848 | 0.9040 | 0.5398 |
| Logistic Regression (Custom) | 0.9871 | 0.6764 | 0.8640 | 0.7588 |

**Interpretation**

* The Gaussian Naive Bayes model achieved high recall, meaning it detected most failures, but precision was low — indicating more false alarms.
* Logistic Regression provided a better balance between precision and recall, achieving a higher F1-score and overall stability.
* Therefore, Logistic Regression was chosen as the preferred model for deployment.

**9. Visualization**

* Pie Chart: Showed class imbalance (failure vs normal cases).
* Bar chart Plots: Displayed missing values in features.
* Confusion Matrix: Used to visualize correct and incorrect predictions.

**10. Tools and Libraries Used**

|  |  |
| --- | --- |
| Category | Tools |
| Programming | Python |
| Libraries | pandas, numpy, matplotlib |
| Modeling | Custom Logistic Regression, Gaussian Naive Bayes |
| Oversampling | Custom SMOTE (no scikit-learn) |
| Visualization | Matplotlib |

**11. Key Insights**

* Synthetic oversampling significantly improved model recall for minority class.
* Logistic Regression provided robust results with a good balance between recall and precision.
* Gaussian Naive Bayes was effective for distributions
* The project demonstrated that data preprocessing and class balancing have a stronger impact on results than model complexity.

**12. Conclusion**

This project successfully demonstrated how sensor-based predictive maintenance can be implemented using fundamental machine learning principles and minimal libraries.

By analyzing anonymized sensor readings:

* We identified meaningful patterns linked to failure events.
* Built a balanced model capable of predicting failures with high accuracy and reliability.
* Proved that simple, interpretable models can perform well when data is properly preprocessed and balanced.

**Future Scope**

* Apply Random Forest and XGBoost like tree based models to capture high metrics.
* Explore feature selection and dimensionality reduction techniques to simplify the model.
* Extend the project to real-time failure prediction using live sensor streams.