Final Report

Predicting Bearing Defects Using Machine Learning and Deep Learning Techniques

Name: J. Rejuvesh

Company: Infinite Uptime India Pvt. Ltd.

Internship Duration: 10-06-2025 to 11-07-2025

Final Report: Predicting Bearing Defects Using Machine Learning and Deep Learning Techniques

Project Overview

The goal of this project is to develop a predictive system that can identify potential bearing defects in industrial machines before they lead to system failure. This early detection mechanism enables timely maintenance, reducing downtime and preventing costly breakdowns.

The dataset used for this project includes both numerical and categorical features representing operational and material conditions, with a binary target variable `defect` (1 = Defective, 0 = Normal). Importantly, the dataset reflects real-world conditions and is highly imbalanced, with only ~1% of samples labeled as defective.

Task-Wise Breakdown and Methodology

1. Exploratory Data Analysis (EDA)

- Loaded and inspected dataset structure, datatypes, and completeness.
- Confirmed the presence of 200,000 records with no missing values.
- Identified a significant class imbalance (only ~1% defective samples).
- **Visualized:**
- Class distribution using count plots.
- Summary statistics of numerical features via .describe().T.
- Correlations using a heatmap.
- Boxplots to compare feature distributions against defect status.
- Category-wise defect breakdown.
- Monthly trend of defects using timestamp column.

Outcome: The data was clean and rich with patterns, but class imbalance and rare anomalies posed a major modeling challenge.

2. Data Preprocessing

- Dropped `timestamp` for modeling but used it in EDA.
- Separated features into numerical and categorical types.
- **Applied imputation:**

- median for numerical
- most_frequent for categorical
- Scaled numerical features using StandardScaler.
- Encoded categorical features using OneHotEncoder(drop='first').
- Used SMOTE to synthetically balance the training data.
- Split dataset into 80% training and 20% testing, preserving class distribution with stratify=y.

Outcome: Created a balanced and fully preprocessed dataset ready for training machine learning and deep learning models.

3. Feature Engineering
- Identified important features using correlation analysis.
- Grouped and handled features by type for specialized transformation.
- (Optional) No additional derived features were added as base features showed sufficient discriminatory power.

Outcome: Feature engineering aligned with model needs and preserved interpretability.

4. Model Building

a. XGBoost Classifier (Supervised)
- Trained on SMOTE-balanced dataset.
- Evaluated using classification metrics and ROC-AUC.
- Resulted in:
- High accuracy (~99%)
- But zero recall for defects, indicating poor detection of rare class.

b. Autoencoder (Unsupervised Anomaly Detection)
- Trained only on non-defective samples to learn normal patterns.
- Used reconstruction error (MSE) to flag anomalies.
- Chose the threshold using 95th percentile of reconstruction errors.
- Evaluated predictions with confusion matrix, classification report, and ROC curve.
- Resulted in:

- Lower accuracy (~94%)
- Slightly better recall (6%) for defect class than XGBoost.

Outcome: Both models performed well on normal samples, but the autoencoder slightly outperformed XGBoost in detecting rare defects.

5. Model Evaluation

- Used:

- Confusion Matrix
- Precision, Recall, F1-Score
- ROC-AUC Score

- Compared ROC curves of both models:

- XGBoost AUC = 0.52
- Autoencoder AUC = 0.53

Outcome: While both models had limited ability to separate defective from normal cases, the autoencoder showed better sensitivity (recall) and slightly higher AUC.

6. Insights and Conclusion

- XGBoost: High precision for class 0, but failed to identify actual defects (recall = 0). Effective only in clean datasets.
- Autoencoder: Better suited for this anomaly detection problem. Identified rare cases, though performance was still limited by data imbalance and subtle defect signatures.

Industrial Relevance & Deployment Potential

This solution, especially the autoencoder model, can be integrated into real-time machine health monitoring systems to:

- Continuously assess sensor and operational data
- Trigger maintenance alerts when reconstruction error crosses a defined threshold
- Improve uptime and reduce maintenance costs

With enhancements like:

- Threshold tuning using domain expert feedback

- Ensemble of anomaly detection models
- Real-time streaming integration

this system can evolve into a robust, intelligent diagnostic tool for industrial equipment.

Final Deliverables

- Jupyter notebooks for:

- EDA
- Preprocessing and Feature Engineering
- Model Training and Evaluation
- Model files: xgboost_bearing_model.pkl
- Saved datasets: X_train_res.npy, y_train_res.npy, etc.
- ROC Curve comparison plots and confusion matrices
- This final report (markdown/pdf format)