Machine Predictive Maintenance Using Machine Learning

Leveraging Predictive Analytics for Proactive Machine Health Monitoring

# 1. Project Title and Introduction

Traditional maintenance approaches, such as time-based scheduled maintenance or reactive maintenance (waiting for machines to break down), lead to inefficient operations, unplanned downtime, and high costs. Predictive maintenance aims to predict equipment failures in advance, allowing organizations to schedule maintenance only when necessary.

This project focuses on using historical machine sensor data to train machine learning models that predict when machines are likely to fail, enabling early intervention.

Technologies:   
- Machine Learning Models: Random Forest, SVM, KNN, Decision Tree, Naive Bayes  
- Programming Languages: Python  
- Libraries: Scikit-learn, Pandas, Imbalanced-learn (SMOTE)

# 2. Dataset Description

Dataset: Machine Predictive Maintenance Dataset from Kaggle.

Link: https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification

Dataset Structure:  
- Number of Observations: 10,000 samples (rows).  
- Features: UDI, Product ID, Type, Air Temperature, Process Temperature, Rotational Speed, Torque, Tool Wear.  
- Target: Machine Failure (binary: Failure or No Failure).  
- Dataset Issues: Imbalanced Classes.

# 3. Data Preprocessing

Objective: Clean and prepare the dataset for modeling.  
- Handling Missing Values: Missing values handled through median filling or row removal.  
- Feature Engineering: Interaction terms created, such as temperature differential between internal and external temperatures.  
- Outlier Detection: Z-scores and IQR method used for outlier removal.  
- Feature Scaling: StandardScaler applied for normalization of features.  
- Handling Imbalanced Data: SMOTE applied for balancing the dataset.

# 4. Algorithms Used

Objective: Compare multiple machine learning algorithms for classification.  
- Support Vector Machine (SVM): Maps data into higher dimensions to find a hyperplane. Best for high-dimensional, non-linear data.  
- Decision Tree: Simple model splitting data based on feature thresholds.  
- K-Nearest Neighbors (KNN): Classifies data based on proximity to neighboring points.  
- Random Forest: An ensemble of decision trees to improve accuracy and reduce overfitting.  
- Naive Bayes: Probabilistic model assuming feature independence.

Justification: These models were chosen due to their balance of simplicity, interpretability, and power.

# 5. Model Training and Testing

Objective: Train models on the processed dataset and evaluate performance.  
- Data Splitting: 80% training, 20% testing.  
- Cross-Validation: Implemented 5-fold cross-validation.  
- Hyperparameter Optimization: Used GridSearchCV to tune model hyperparameters (SVM: kernel, C; Random Forest: number of trees).  
- Evaluation Metrics: Accuracy, Precision, Recall, F1-Score.

# 6. Results and Accuracy

Objective: Present model performance and compare their effectiveness.  
- Accuracy Scores:   
 - Random Forest: 93%  
 - SVM: 91%  
 - Decision Tree: 88%  
 - KNN: 85%  
 - Naive Bayes: 80%

Performance Analysis: Random Forest was the most accurate model due to its ensemble nature, which helped capture complex patterns in the data.