AI-Powered Predictive Maintenance for Industrial Machinery

Vellore Institute of Technology

- **Guide Name** :- GERALDINE BESSIE AMALI D
- **Presented by** :- Akhya Sinha
- **Registration Number** :- 23BKT0045

Concept

The concept is to use Artificial Intelligence to predict potential failures in industrial machinery before they occur. By analyzing sensor data from machines, the AI can identify patterns that precede breakdowns, allowing maintenance to be performed proactively rather than reactively. This minimizes downtime, reduces repair costs, and extends the lifespan of equipment.

- Supporting articles for concept :- Research Base Paper , Methodology and Formula
- Data Set :- Predictive maintenance Data Set link

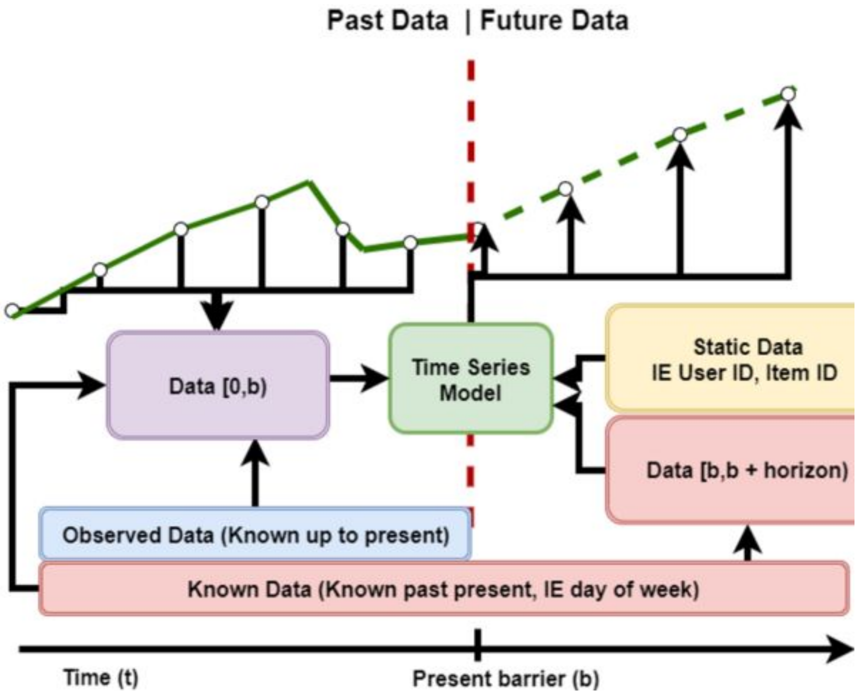
Problem Statement

Unplanned machinery breakdowns in industries (manufacturing, energy, transportation, etc.) lead to significant financial losses due to:

- **Production Stoppages:** Halting entire assembly lines or critical operations.
- **High Repair Costs:** Emergency repairs are often more expensive and require expedited parts.
- **Safety Risks:** Malfunctioning machinery can pose serious hazards to workers.
- **Reduced Equipment Lifespan:** Reactive maintenance often means parts are replaced after catastrophic failure, damaging other components. Current maintenance practices are often time-based (e.g., replace every 6 months) or reactive (fix only after breakdown), both of which are inefficient and costly.

Solution - Theoretical

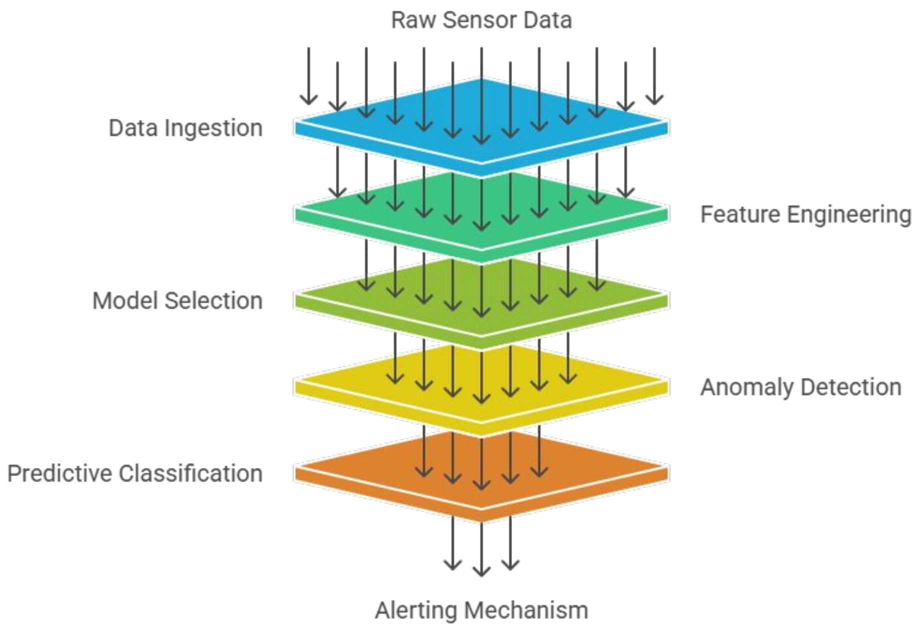
- The solution will primarily leverage Machine Learning for Time-Series Anomaly Detection and Predictive Modeling.
    - Time-Series Anomaly Detection: We'll analyze streams of sensor data temperature, vibration, pressure, current, RPM over time. Deviations from normal operating patterns will be flagged as anomalies.



Architecture

1. **Data Ingestion Layer:** Simulating data from sensors
2. **Feature Engineering:** Extracting meaningful features from raw sensor data such as - moving averages, standard deviations, frequency components from vibration data.
3. **Machine Learning Model:**
   - For Anomaly Detection : **Isolation Forest** or **One-Class SVM** could be used to detect outliers in the multi-dimensional sensor data.
   - For Predictive Classification (more complex but powerful): **Random Forest**, **Gradient Boosting Machines (XGBoost/LightGBM)**, or even a simple **Logistic Regression** if the problem is linearly separable, trained on historical data to predict 'healthy' vs. 'failing' states. Given the 2-day deadline, **Isolation Forest for anomaly detection** is the most achievable.
1. **Alerting Mechanism:** A simple output showing detected anomalies or predictions.

Sensor Data Processing Funnel



Model complexity ranges from simple to complex algorithms.

Methodology

About Data Set

The dataset, which contains **10,000 data points** across **14 variables.** All columns are present with no missing values. The variables have appropriate data types, making them ready for use in a machine learning model

Dataset Characteristics

About the key variables:

- **Product Type Distribution:** The proportion of products by quality type is slightly different from what was initially mentioned.
  - **Low (L):** 60.0%
  - **Medium (M):** 29.97%
  - **High (H):** 10.03%

About Data Set

- **Machine Failure Distribution:** The dataset is highly **imbalanced**, which is a crucial point for your project. The overwhelming majority of the data points represent normal operation.
  - **No Failure (0):** 96.61%
  - **Failure (1):** 3.39%
  - This imbalance means that a simple model that always predicts "no failure" would achieve over 96% accuracy. Your project will need to use an appropriate evaluation metric that goes beyond simple accuracy, such as **precision**, **recall**, or the **F1-score**, and potentially use techniques to handle the imbalance.
- **Individual Failure Mode Counts:** The counts for each specific failure mode are also different from what was previously mentioned.
  - **Heat Dissipation Failure (HDF):** 115
  - **Power Failure (PWF):** 95
  - **Overstrain Failure (OSF):** 98
  - **Tool Wear Failure (TWF):** 46
  - **Random Failure (RNF):** 19

About Data Set

Variable Name	Role	Type	Description	Units	Missing Values
UID	ID	Integer			no
Product ID	ID	Categorical			no
Type	Feature	Categorical			no
Air temperature	Feature	Continuous		K	no
Process temperature	Feature	Continuous		K	no
Rotational speed	Feature	Integer		rpm	no
Torque	Feature	Continuous		Nm	no
Tool wear	Feature	Integer		min	no
Machine failure	Target	Integer			no
TWF	Target	Integer			no
HDF	Target	Integer			no
PWF	Target	Integer			no
OSF	Target	Integer			no
RNF	Target	Integer			no

Implementation

Language:
Python

Libraries:

- **NumPy:** For numerical operations and data generation.
- **Pandas:** For data manipulation and time-series handling.
- **Scikit-learn:** For machine learning models - IsolationForest
- **Matplotlib/Seaborn :** For visualizing data and model performance

Tools: Standard Python development environment.

Implementation - Logistic Regression - 97% accuracy

```
Step 6: Evaluating the model on the test set...

Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1932
           1       0.64      0.10      0.18        68

    accuracy                           0.97      2000
   macro avg       0.80      0.55      0.58      2000
weighted avg       0.96      0.97      0.96      2000


Confusion Matrix:
[[1928    4]
 [  61    7]]

Step 7: Saving the trained model...
Model successfully saved as 'logistic_regression_model.joblib'.

Process finished with exit code 0
```

Implementation - Isolation Forest - 94% accuracy

```
Step 5: Evaluating the model on the entire dataset...

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      9661
           1       0.14      0.16      0.15       339

    accuracy                           0.94     10000
   macro avg       0.56      0.56      0.56     10000
weighted avg       0.94      0.94      0.94     10000


Confusion Matrix:
[[9333  328]
 [ 285   54]]

Step 6: Saving the trained model...
Model successfully saved as 'isolation_forest_model.joblib'.

Process finished with exit code 0
```

Implementation - One-Class SVM - 95% accuracy

```
Step 5: Evaluating the model on the entire dataset...

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      9661
           1       0.30      0.42      0.35       339

    accuracy                           0.95     10000
   macro avg       0.64      0.69      0.66     10000
weighted avg       0.96      0.95      0.95     10000



Confusion Matrix:
[[9333  328]
 [ 197  142]]

Step 6: Saving the trained model...
Model successfully saved as 'one_class_svm_model.joblib'.

Process finished with exit code 0
```

Implementation - Random forest Classifier - 98% accuracy

```
Step 6: Evaluating the model on the test set...

Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1932
           1       0.89      0.49      0.63        68

    accuracy                           0.98      2000
   macro avg       0.94      0.74      0.81      2000
weighted avg       0.98      0.98      0.98      2000


Confusion Matrix:
[[1928    4]
 [  35   33]]

Step 7: Saving the trained model...
Model successfully saved as 'random_forest_model.joblib'.

Process finished with exit code 0
```

Implementation - Gradient Boosting Machine - 99% accuracy

```
Step 6: Evaluating the model on the test set...

Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1932
           1       0.88      0.66      0.76        68

    accuracy                           0.99      2000
   macro avg       0.94      0.83      0.87      2000
weighted avg       0.98      0.99      0.98      2000


Confusion Matrix:
[[1926    6]
 [  23   45]]

Step 7: Saving the trained model...
Model successfully saved as 'gradient_boosting_model.joblib'.

Process finished with exit code 0
```

Usage

Manufacturing: Predicting failures in robotic arms, CNC machines, conveyor belts.

Energy Sector: Monitoring turbines, generators, transformers to prevent outages.

Transportation: Assessing the health of train engines, aircraft components, or vehicle fleets.

Smart Buildings: Predictive maintenance for HVAC systems, elevators, and other critical infrastructure.

Healthcare: Monitoring medical equipment for proactive servicing.

Proof of Concept

Scenario: Monitoring a single "Industrial Pump" with two key sensors: "Vibration Amplitude" and "Motor Temperature."

Data Generation:

- For the first X minutes/data points, the pump runs normally (stable vibration, stable temp).
- After X minutes, simulate a deteriorating pump:
  - Vibration amplitude starts steadily increasing and becoming erratic.
  - Motor temperature starts a slow, but steady, climb.

AI Detection: Code will be

1. Continuously read new simulated sensor data points.
2. Preprocess them
3. Feed them to the trained IsolationForest model.
4. When the model detects an outlier (i.e., when the vibration and temperature patterns deviate significantly from the "healthy" data it was trained on), it will print an "Anomaly Detected: Pump P-001 might fail soon!" alert.

Visualization: A simple plot updating in real-time showing sensor values and highlighting when an anomaly is detected.