Part 1

1. Problem Definition

Predicting Equipment Failure in a Manufacturing Plant

Objectives

Predictive maintenance before an actual breakdown will help in reducing machinery downtime.

Reduce maintenance costs through optimized repair schedules using predictive analytics.

Improve production efficiency by ensuring continuous, uninterrupted machine uptime.

Stakeholders

Plant Operations Manager: Oversees production and ensures the reliability of equipment.

Maintenance Engineers: The AI predictions could be utilized to schedule and perform timely maintenance.

Key Performance Indicator

Mean Time Between Failures (MTBF) means the average operational time between predicted and actual equipment failures. Higher MTBF reflects higher predictive accuracy and better effectiveness of maintenance.

2. Data Collection & Preprocessing

Data Sources

Sensor Data: Real-time data from IoT sensors that measure temperature, vibration, pressure, and motor speed of machines.

Maintenance Logs: Historical records of repairs, dates of replacement, and equipment issues reported.

Potential Bias

Maintenance Reporting Bias: Some maintenance activities may not be consistently or accurately recorded by technicians. This may lead to incomplete historical data, which makes the AI model biased toward machines that have much more detail and mispredicts failure for those that are underreported.

Preprocessing Steps

Handling Missing Data:

Fill in missing sensor readings using interpolation, or remove records with too many gaps.

Normalization: Scale sensor readings such as temperature, vibration magnitude to a common range of 0 to 1 to train models in a balanced manner. Feature Engineering: Create derived variables such as "average vibration over 24 hours" or "temperature deviation from normal" to capture early signs of wear.

3. Model Development

Model choice:

Random Forest

Justification:

Works well for tabular sensor + log data without heavy feature scaling.

Robust to outliers and noisy sensor signals, and tolerates some missingness.

Provides feature importance for interpretability, which is useful for maintenance engineers.

Quick to train with, and reliable for, datasets of moderate size; perfect for engineering teams needing quick results.

If we had long sequences of raw time-series per machine, and a lot of labeled sequences, we'd consider an LSTM/Transformer; but for typical predictive-maintenance features, and engineered time-window stats, Random Forest is a strong choice.

Data splitting (training / validation / test)

Because this is time-dependent data and failures are rare, avoid random shuffling that leaks future information:

Chronological split (holdout):

Training: earliest 60% of the time range.

Validating the next 20% of the timescale.

Test: final 20% (most recent) — evaluate final generalization on future-like data.

Prevent leakage across machines:

If a single machine produces long contiguous records, ensure its contiguous failure-window sequences don't get split across sets - either assign whole machines to splits where appropriate or use time windows per machine.

Cross-validation for robustness:

Use rolling (time-series) cross-validation / expanding-window CV on the training+validation period for hyperparameter tuning, then lock and evaluate on the final test set.

Class imbalance handling:

If failures are rare, keep the same failure/non-failure ratio in validation and test by selecting windows rather than individual rows (so the evaluation reflects realistic prevalence).

Hyperparameters

n_estimators (Number of Trees)

Controls ensemble size. Increasing trees generally reduces variance and boosts stability but increases computational burden/time. Tune to find a point of diminishing returns for accuracy vs. training time.

max_depth (maximum tree depth)

max_depth limits how deep trees grow, preventing overfitting to noisy sensor fluctuations. Alternatively min_samples_leaf enforces a minimum leaf size to avoid overly specific splits. Tuning either helps balance bias vs variance and improves generalization to unseen machines/time periods.

4. Evaluation & Deployment

Evaluation Metrics

Precision – The ratio of correctly predicted failures to the overall predicted failures.

Relevance: In maintenance, false alarms are expensive to handle (unnecessary inspections), so high precision ensures that alerts will be trustworthy.

Recall – This is the proportion of actual failures correctly identified by the model.

Relevance: Missing a true failure might lead to unplanned downtime and production losses. High recall ensures the model hardly misses real issues.

Precision and Recall together balance the trade-off between avoiding false alarms and preventing missed failures. Often combined using the F1-score for overall assessment.

Concept Drift

Concept drift refers to the change in the underlying statistical data properties or even the linkage between inputs and failures over time. Examples include those after new machinery installation, sensor recalibration, and changes in environmental circumstances that may affect the readings.

Monitoring strategy:

Continuously monitor the distributions of model predictions, comparing them with actual recent failure events.

Identify shifts in key features using data drift detection techniques, such as a KS test or Population Stability Index.

Periodically, retrain the model with more recent data to maintain accuracy.

Technical Challenge - Scalability

Challenge: Prediction has to be done in real-time, processing continuous streams of high-frequency sensor data. Handling multiple machines simultaneously can be computationally and memory-intensive. Mitigation: Deploy model at the edge (lightweight inference near sensors) to reduce data transmission. Use frameworks for stream processing (e.g., Apache Kafka, Spark Streaming) to process large-scale, real-time input.

Part 2

1. Problem Scope

Problem Definition

A hospital wants to develop an AI system that predicts the likelihood of a patient being readmitted within 30 days after discharge.

The goal is to identify high-risk patients early so that preventive care and follow-up actions can be taken to reduce readmission rates.

Objectives

Predict readmission risk accurately using patient medical records, demographics, and treatment history.

Support clinical decision-making by flagging patients who may require additional monitoring or follow-up care.

Reduce hospital readmission rates to improve patient outcomes and lower healthcare costs.

Stakeholders

Doctors and Nurses – Use the system's predictions to plan follow-up visits or adjust treatment plans.

Hospital Administration – Monitors performance metrics, resource allocation, and cost savings from reduced readmissions.

2. Data Strategy

Data Sources

Electronic Health Records:

Clinical data can include diagnosis codes, lab results, vital signs, medications, length of stay, and discharge notes.

Patient Demographic Data:

Age, sex, socioeconomic status, insurance type, and living arrangements (such as living alone or with a family).

Historical Readmission Data:

Records of patients being readmitted within 30 days, for supervised learning using such labels.

Ethical Considerations

Patient Privacy and Data Security:

Health information is sensitive data and must be treated within the requirements for protection conditions, such as HIPAA/GDPR. Encryption, anonymization, and limited access will help prevent misuse.

Bias and Fairness in Predictions:

Such underrepresentation in some demographic groups, such as low-income or minority patients, may well lead to unfair model predictions and unequal quality of care. Testing bias and audits of models should be done continuously.

3. Preprocessing Pipeline

Step 1: Cleaning Data

Handle missing lab results or incomplete records by imputation, such as using mean/median or clinical default values.

Remove duplicate entries or inconsistent patient IDs

Step 2: Data Normalization & Encoding

Normalize numerical variables, such as lab and vital sign values, to a common range of 0–1.

one-hot or embedding representations for categorical features such as diagnosis codes and medication types.

Step 3: Feature Engineering

The following derivatives could be created:

Number of hospital visits in the last 6 months.

Average length of hospital stay.

3. Model Development

Model Selection

Chosen Model: Logistic Regression

Justification:

Interpretability: Understanding by doctors and administrators of how much different factors contribute to readmission risk-from the age, comorbidities, lab results-is easily understood.

Performance on tabular data: It works well on structured medical datasets comprising both numeric and categorical features.

Probabilistic output: Provides a clear risk score, within the range of 0-1, which shows the likelihood of readmission, guiding decision thresholds for when to intervene.

Other models, such as Random Forests or Gradient Boosting, might be tried in pursuit of better performance later on, but Logistic Regression supplies a clear baseline.

Hypothetical Confusion Matrix

Predicted: Readmit

Actual: Readmit (Yes) 70 (True Positives, TP) 30 (False Negatives, FN)

Actual: No Readmit 20 (False Positives, FP) 180 (True Negatives, TN)

Total samples = 300

Precision and Recall Calculations

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{70}{70 + 20} = \frac{70}{90} = 0.78$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{70}{70 + 30} = \frac{70}{100} = 0.70$$

4. Deployment

Integration Steps

Model Packaging and API Development:

Deploy the trained AI model as a RESTful API or microservice that the EHR system within the hospital setting can access.

The API will take in patient discharge data and return a readmission risk score in real time.

Integration with EHR System:

Connect the model to the EHR interface of the hospital to enable clinicians to directly view the risk predictions in a patient's profile.

Role-based access should be implemented to ensure that only those with the proper permission can view predictions, such as doctors and nurses.

Real-Time Prediction Dashboard:

Develop a dashboard that visualizes risk levels, highlights high-risk patients, and follows readmission trends.

Feedback Loop for Continuous Learning:

Store actual outcomes or results of whether readmission occurred for future retraining of the model.

Performance metrics like precision, recall, and drift should be monitored over time.

Healthcare Compliance Issues, such as HIPAA

Data Anonymization: identifiable patient information should be removed before training or distribution- names, IDs, addresses.

Encryption: All patient data should be encrypted while in transit and in rest, using TLS/SSL.

Access Control: Limiting sensitive data access through authentication and role-based permissions.

Audit Logging: Detailed logs of who and how data was accessed, including modification.

Regulatory Review: Conduct compliance audits, and align processes to HIPAA or, if applicable, the General Data Protection Regulation to meet standards on privacy and protection of data.

5. Optimization

Overfitting Control Method

Regularization (L2 or Ridge Regularization)

Add a penalty term to the logistic regression loss function that discourages coefficients from being too large.

This prevents the model from fitting noise or irrelevant correlations in the training data. The strength of regularization can be tuned using cross-validation in order to find the best trade-off between bias and variance.

Part 3

Ethics & Bias

Impact of Biased Training Data on Patient Outcomes

If the training data contains biases for example, underrepresentation of certain patient groups (e.g., elderly, rural, or low-income patients) the AI model may learn inaccurate patterns.

Possible negative effects include:

- Unequal predictions: The model may underestimate readmission risk for some groups, causing them to miss essential follow-up care.
- Reduced quality of care: Vulnerable populations might receive less post-discharge support, leading to higher readmission rates or complications.
- Erosion of trust: Patients and clinicians may lose confidence in AI-driven recommendations if they notice systematic unfairness.

Strategy to Mitigate Bias

Fairness-Aware Data Sampling and Evaluation

- Step 1: Analyze the dataset for demographic imbalances (e.g., age, gender, ethnicity, socioeconomic status).
- Step 2: Use re-sampling techniques such as *stratified sampling* or *re-weighting* to ensure all key groups are well-represented in training.
- Step 3: Evaluate model performance separately across subgroups (e.g., recall for male vs. female patients) to detect bias.
- Step 4: If disparities exist, apply fairness constraints or re-train with adjusted class weights to equalize performance across groups.

Result:
This approach promotes equitable treatment recommendations, improves trustworthiness, and ensures that the AI system supports all patients fairly, regardless of their background.

Trade-offs

Interpretability vs. Accuracy in Healthcare

In healthcare, there is often a trade-off between model interpretability and predictive accuracy:

- Highly interpretable models (e.g., Logistic Regression, Decision Trees)
    - Advantages:
        - Clinicians can easily understand why the model made a specific prediction.
        - Easier to justify medical decisions ethically and legally.
        - Supports transparency and trust among healthcare professionals.
    - Disadvantages:
        - May capture fewer complex relationships in the data, leading to slightly lower accuracy.
- Highly accurate models (e.g., Neural Networks, Gradient Boosting Machines)
    - Advantages:
        - Can identify nonlinear patterns and subtle risk factors that simpler models might miss.
    - Disadvantages:
        - Harder to interpret — clinicians may not understand why the model predicts high risk for certain patients.
        - Lower transparency could lead to resistance in clinical adoption or difficulty in regulatory approval.

Balance:
In healthcare, interpretability often takes priority over small gains in accuracy, since decisions affect real patients. Therefore, a slightly less accurate but explainable model is usually preferred for clinical use.

Impact of Limited Computational Resources on Model Choice

If the hospital has limited computational power or lacks advanced servers:

- Complex models (e.g., deep neural networks) might be impractical due to high training and inference costs.
- The hospital should favor lightweight, resource-efficient models such as:
    - Logistic Regression
    - Random Forests with limited depth
    - Gradient Boosting with few estimators

These models can run efficiently on standard hospital systems while still delivering reliable performance.
They also enable faster predictions, which is critical for real-time clinical decision support.

Conclusion:
Hospitals should strike a balance by selecting models that are interpretable, computationally feasible, and clinically accurate enough to support safe, fair, and trustworthy patient care.

Part 4

Reflection

Most Challenging Part of the Workflow

The most challenging part was data preprocessing and bias mitigation.
Medical data is often incomplete, inconsistent, and highly sensitive, making it difficult to clean and prepare without losing important information.
Additionally, ensuring fairness across diverse patient groups required careful analysis and ethical consideration balancing technical performance with clinical responsibility.

Improvements with More Time/Resources

With more time and resources, I would:

- Collect more comprehensive datasets from multiple hospitals to improve model generalization.
- Implement advanced models (e.g., Gradient Boosting, Explainable AI methods like SHAP or LIME) to balance accuracy and interpretability.

- Collaborate with medical professionals to refine features and validate model outputs in real-world clinical settings.
- Develop automated bias detection tools to continuously monitor fairness after deployment.

Diagram

Problem Definition

↓

Data Strategy

↓

Preprocessing

↓

Model Development

↓

Evaluation & Bias Mitigation

↓

Deployment

↓

Optimization

↓

Reflection