Project Proposal: AI-Powered Predictive Maintenance System for Industrial Equipment

Executive Summary

This capstone project proposes the development of an AI-powered predictive maintenance system (PdM) for industrial equipment, utilizing machine learning to forecast failures, optimize maintenance schedules, and minimize operational costs. The system integrates real-time IoT sensor data (vibration, temperature, pressure) with historical maintenance logs, utilizing Random Forests for failure classification and LSTM neural networks for time-series forecasting. A Streamlit dashboard will provide real-time visualizations and decision support. The project, spanning 12 weeks, aims to achieve >85% prediction accuracy, reduce downtime by 40%, and save 25% in maintenance costs, as validated on a public dataset (NASA Turbofan Engine Degradation). Deliverables include a GitHub-hosted prototype, comprehensive documentation, and a presentation showcasing alignment with data science and software engineering coursework. Ethical AI practices, scalability, and robust risk mitigation ensure a production-ready solution.

Introduction and Background

Unplanned equipment failures in industries like manufacturing and energy cause significant losses, with downtime costing up to $50 billion annually across sectors (McKinsey, 2023). Traditional maintenance (reactive or scheduled) is inefficient, leading to 20-30% over-maintenance or unexpected breakdowns. This project addresses these challenges by developing an AI-driven PdM system that predicts failures using sensor data, reducing downtime and extending equipment lifespan. It applies machine learning, data processing, and UI development skills from the major, targeting equipment like turbines and pumps.

Competitive Landscape

There are predictive maintenance solutions available in the market (e.g. IBM Maximo, GE Predix), but most of them are costly, need proprietary infrastructure, or the AI model is non-transparent. The solution is differentiated by an open-source basis, explainability (using SHAP), and cost-effective provisions when it comes to small to mid-size businesses.

Objectives

- Develop a machine learning model achieving >85% accuracy in predicting equipment failures.
- Create a Streamlit dashboard for real-time monitoring, failure alerts, and scenario testing.
- Validate system performance on a public dataset, targeting 40% downtime reduction and 25% cost savings.
- Produce a scalable prototype with modular code, documented in a GitHub repository.
- Deliver a portfolio presentation demonstrating technical proficiency and industry relevance.
- Ensure ethical AI deployment with transparency, fairness, and minimal environmental impact.

Technical Approach

Data Acquisition and Processing

- **Data Sources**: NASA Turbofan Engine Degradation Dataset (CMAPSS) for training/validation, supplemented by synthetic data generated via Python's numpy.random for edge cases. Sensors include vibration (Hz), temperature (°C), and pressure (kPa).
- **Preprocessing**:
    - Pandas for data cleaning (remove NaNs, outliers via IQR method).
    - NumPy for feature engineering (e.g., rolling mean, standard deviation over 10-second windows).
    - Normalization using MinMaxScaler to ensure model stability.
- **Volume**: ~100,000 records for training, 20,000 for testing, ensuring robust validation.

Model Development

- **Algorithms**:
    - **Random Forest Classifier**: 100 trees, max_depth=10, for failure type classification (e.g., bearing wear, motor overload). Gini criterion for splits.
    - **LSTM Neural Network**: 2 layers, 64 units each, for time-series forecasting of remaining useful life (RUL). Sequence length=50 timesteps, trained with Adam optimizer (learning rate=0.001).
    - Ensemble: Weighted voting (70% LSTM, 30% RF) for final predictions to balance accuracy and interpretability.
- **Frameworks**: Scikit-learn (v1.5.2) for Random Forest, TensorFlow (v2.17.0) for LSTM. Hyperparameter tuning via GridSearchCV (RF) and Keras Tuner (LSTM).
- **Explainability**: SHAP (Shapley Additive Explanations) to compute feature importance, ensuring transparency for maintenance teams.

System Architecture

- **Backend**: Python pipeline with Apache Kafka (v3.8.0) for streaming sensor data (if scaled), Flask for model inference API, and PostgreSQL for historical data storage.
- **Frontend**: Streamlit (v1.39.0) dashboard displaying:
    - Time-series plots (Matplotlib/Seaborn) of sensor data.
    - Failure probability (0-100%) with alert thresholds (>80% triggers a warning).
    - RUL predictions in hours/days.
- **Deployment**:
    - Prototype runs on Google Colab (16GB RAM, T4 GPU).
    - Production-ready Docker container (Python 3.11 base image) for scalability.

    - Code versioned on GitHub with CI/CD via GitHub Actions for automated testing.

Refinements

- **Bias Mitigation**: Fairness-aware preprocessing using aif360 to detect and correct biases in failure predictions across equipment types.
- **Performance Optimization**: Batch inference for low-latency predictions (<100ms per batch), tested on edge devices (e.g., Raspberry Pi 4 simulation).
- **Scalability**: Modular design supports multi-equipment environments; cloud-ready with AWS ECS compatibility.

Timeline and Milestone Planning

- **Weeks 1-3: Planning and Setup** (08/21/2025 - 09/09/2025)
    - Tasks: Dataset acquisition, literature review, environment setup (Python 3.11, dependencies).
    - Deliverable: Requirements document and data schema (CSV format).
- **Weeks 4-6: Data and Model Development** (09/10/2025 - 10/07/2025)
    - Tasks: EDA, feature engineering, train baseline Random Forest (accuracy >80%) and LSTM (MAE <10 hours for RUL).
    - Deliverable: Jupyter Notebook with EDA and model checkpoints.
- **Weeks 7-10: System Integration** (10/08/2025 - 10/28/2025)
    - Tasks: Build Streamlit dashboard, integrate models, test end-to-end pipeline.
    - Deliverable: Functional prototype demo (video recording).
- **Weeks 9-10: Testing and Optimization** (10/29/2025 - 11/18/2025)
    - Tasks: Validate on test set, achieve >85% accuracy, 80% precision/recall, <2s dashboard latency.
    - Deliverable: Performance report with confusion matrix, ROC-AUC, and SHAP plots.
- **Weeks 11-12: Finalization** (11/19/2025 - 12/05/2025)
    - Tasks: Refine code, document in README.md, prepare 10-minute presentation.
    - Deliverable: GitHub repository, final report (PDF), portfolio slides (PowerPoint).

Resources Required

- **Hardware**: Laptop with 8GB RAM; Google Colab GPU.
- **Software**: Free libraries (Pandas 2.2.3, NumPy 2.1.2, Scikit-learn, TensorFlow, Streamlit, SHAP 0.46.0).

- **Data**: Public NASA dataset (no cost); synthetic data generation if needed.
- **Personnel**: 5 team members (**Olu, Richard, Miquel, and Akinbobola**).
- **Budget**: ~$20 for cloud credits; no other costs.

Success Metrics

- **Quantitative**:
    - Model: Accuracy >85%, Precision/Recall >80%, F1-score >0.82, RUL MAE <20 hours.
    - Operational: 40% downtime reduction, 25% maintenance cost savings (simulated via cost models).
    - System: Dashboard latency <2s, 99% uptime during testing.
- **Qualitative**:
    - User satisfaction: >4.5/5 from mock technician survey (10 users).
    - Documentation: 100% test coverage, clear README with setup instructions.
    - Presentation: Covers technical, ethical, and business value; <10 minutes, >90% audience comprehension (mock panel feedback).

Metrics tracked via MLflow (v2.17.0) for experiments and GitHub milestones.

Educational Value

Other than industry use, this project is a culmination of machine learning, software engineering, and ethical AI practices that are trained during the program. It will be a powerful academic demonstration as team members will have practical experience with end-to-end system design, data preprocessing and model training, and deployment and communication with stakeholders.

Risk Assessment

Risk	Probability	Impact	Mitigation
Poor data quality (missing/noisy)	High (60%)	High	Use multiple datasets; generate synthetic data with numpy.random; validate with domain experts.
Model underperformance	Medium (40%)	High	Apply k-fold cross-validation (k=5), early stopping, ensemble weighting; consult advisors if accuracy <80%.
Integration failures (API/dashboard)	Medium (30%)	Medium	Modular testing with pytest; rollback to Flask-only API if Streamlit fails.
Timeline delays	Low (20%)	Medium	Buffer 1 week; weekly syncs to track progress; prioritize MVP features.

Dependency issues (library versions)	Low (10%)	Low	Pin versions (requirements.txt); test on clean environments via Docker.

Ethical Considerations

- **Data Privacy**: Use anonymized public datasets; simulate GDPR compliance by masking any hypothetical identifiers.
- **Fairness**: Audit for bias using aif360 (e.g., disparate impact ratio >0.8); ensure predictions are equitable across equipment types.
- **Transparency**: SHAP plots and model summaries provided to users, explaining predictions in human-readable terms (e.g., "Vibration spike caused 90% failure probability").
- **Environmental Impact**: Optimize maintenance to reduce energy waste (e.g., 10% less power for over-maintained equipment, per simulation).
- **Workforce Impact**: Include recommendations for technician retraining in documentation, mitigating automation-related job concerns.
- **Compliance**: Adhere to IEEE Ethically Aligned Design principles; obtain mock stakeholder approval for deployment.
- **Industry Impact**: The suggested system can transform predictive maintenance in various industries, including aviation, manufacturing and renewable energy. It can save billions of dollars of losses each year to global economies by reducing unexpected downtime, and improve safety and make industrial operations greener and more sustainable.
- **Collaboration and Stakeholder Engagement**: Close collaboration with the professionals working in the industry, academic consultants, and potential users will also play a major part in the success of this project. The refinements will be informed by the stakeholder workshops and feedbacks to ensure that the solution is compatible with the real world requirements in addition to maximizing the adoption potential.
- **Sustainability Considerations**: The system helps minimize industrial waste and carbon emission by reducing the need to replace parts that do not need replacement and premature maintenance. It is estimated, using simulations, that optimized maintenance would reduce the equipment-related waste by as much as 15 percent each year in line with the world sustainability objectives.

Future Enhancements

The system may be modified to include predictive reinforcement learning in future versions to enable the system to adjust maintenance schedules in response to real-time feedback. The simulated test environments can be offered through the integration with the digital twins and improve the accuracy of the failure detection. Also, blockchain has the potential of being investigated in secure maintenance data logging to guarantee audit trails that are not tampered with.

Conclusion

This AI-Powered Predictive Maintenance System is a technically rigorous and ethically sound capstone project that addresses a critical industry need. By achieving precise failure predictions, scalable deployment, and transparent operations, it maximizes impact while aligning with coursework in machine learning, data processing, and UI development. The detailed timeline, robust metrics, and proactive risk mitigation ensure a well-thought-out proposal, poised for extension into industry applications.