PitPredict

Tagline: AI-Powered Predictive Maintenance for Formula 1 - Predicting failures before they happen

Built With: python, scikit-learn, streamlit, plotly, pandas, numpy, machine-learning, artificial-intelligence, iot, data-visualization

---

Inspiration

In Formula 1, a single mechanical failure can cost millions of dollars, championship points, and even lives. During the 2021 Azerbaijan Grand Prix, Max Verstappen's tire failure while leading the race cost Red Bull Racing crucial championship points. In the 2020 British Grand Prix, Lewis Hamilton's tire degradation nearly cost him the win. These incidents highlight a critical problem: traditional maintenance relies on fixed schedules or manual inspections, which cannot catch sudden failures in real-time.

The problem is multifaceted:

- Components fail unexpectedly during races, leading to DNF (Did Not Finish) results
- Manual inspections miss early warning signs hidden in sensor data patterns
- Fixed maintenance schedules are inefficient and can't adapt to varying usage conditions
- Teams collect massive amounts of telemetry data but lack tools to predict failures proactively

We were inspired to create an AI system that learns from sensor patterns to predict failures BEFORE they become catastrophic, giving teams precious time to prevent disasters and optimize performance.

What it does

PitPredict is an intelligent predictive maintenance system specifically designed for Formula 1 and high-performance engineering applications. The system:

Monitors Real-Time Sensor Data: Continuously tracks 9 critical parameters including vibration (mm/s), temperature (°C), pressure (PSI), RPM, noise levels (dB), runtime hours, load cycles, vibration variance, and temperature gradients.

Predicts Component Failures: Uses a trained Random Forest machine learning model to calculate failure probability on a scale of 0-100% with over 95% accuracy. The model analyzes patterns in sensor data that human operators might miss.

Provides Smart Alerts: Implements a three-tier warning system:

- Healthy (0-50% failure risk): Component operating within normal parameters
- Warning (50-70% failure risk): Schedule preventive maintenance within 24 hours
- Critical (70-100% failure risk): Immediate action required, replace component before next session

Visualizes Trends: Features an interactive dashboard that displays historical patterns through line charts and gauge visualizations, helping teams identify gradual degradation before catastrophic failure.

Supports Multiple Components: Monitors critical F1 systems including Turbocharger, Gearbox, Brake System, Suspension, and Cooling System, each with component-specific failure patterns and thresholds.

The system runs continuously during practice, qualifying, and race sessions, updating predictions every 1-10 seconds based on live telemetry data.

How we built it

Technology Stack: We built PitPredict using Python 3.8+ as the core programming language. For machine learning, we leveraged scikit-learn's Random Forest classifier. The real-time dashboard was developed using Streamlit for its ability to create interactive web applications quickly. Data visualization utilized Plotly for dynamic gauge charts and trend lines, while pandas and numpy handled data processing and feature engineering. Model persistence was managed through joblib for efficient serialization.

System Architecture: The system follows a pipeline architecture: Sensor Data → Feature Engineering → ML Model → Predictions → Live Dashboard. Raw sensor data from 9 sensors flows through feature engineering that creates 15+ derived features, which feed into the Random Forest model. The model outputs failure probabilities that trigger alerts and update the dashboard visualizations in real-time.

Development Process:

First, we tackled Data Simulation. Since real F1 telemetry data is highly confidential, we created realistic synthetic sensor data that mimics actual component behavior. We modeled normal operation patterns and failure modes based on engineering principles, ensuring sensors showed realistic correlations (e.g., increased vibration correlates with higher temperature).

Second, Feature Engineering was critical. We developed 15+ engineered features including vibration-to-temperature ratios (indicating thermal stress), pressure-to-RPM efficiency metrics (hydraulic performance), polynomial features like vibration squared and temperature squared (non-linear failure indicators), and composite risk scores weighted by criticality. These features improved model accuracy by 15% compared to using raw sensor data alone.

Third, Model Training involved selecting Random Forest as our algorithm due to its robustness with non-linear relationships, resistance to outliers, built-in feature importance

rankings, and minimal hyperparameter tuning requirements. We trained on 10,000 synthetic sensor readings with 200 decision trees, achieving 95%+ accuracy and 0.96 ROC-AUC score. The model balances precision (avoiding false alarms) with recall (catching real failures).

Fourth, Dashboard Development created a professional interface with real-time data streaming and configurable auto-refresh intervals, interactive gauge charts with color-coded danger zones (green/yellow/red), historical trend analysis showing sensor behavior over time, and component-specific monitoring with customizable alert thresholds.

Challenges we ran into

Challenge 1: Realistic Data Simulation Without access to actual F1 telemetry data due to confidentiality agreements, we had to engineer realistic sensor patterns from scratch. The solution involved extensive research into mechanical failure modes, studying how vibration, temperature, and pressure interact during component degradation. We implemented correlation matrices to ensure sensors showed realistic relationships and validated our synthetic data against published engineering studies.

Challenge 2: Feature Engineering Complexity Raw sensor readings alone weren't sufficiently predictive. Temperature might spike for benign reasons, or vibration could vary based on track conditions. We solved this by creating interaction features that capture relationships between sensors, such as vibration normalized by temperature (accounting for thermal expansion) and pressure efficiency at different RPMs (hydraulic performance degradation). These composite features dramatically improved prediction accuracy.

Challenge 3: Real-Time Performance The system needed to process sensor data and generate predictions fast enough for live race monitoring without lag. We optimized the feature extraction pipeline to vectorize operations, chose Random Forest over deep learning for faster inference times (predictions in milliseconds), and implemented efficient data structures to minimize memory usage during continuous operation.

Challenge 4: Alert Threshold Optimization Finding the right balance was difficult: too sensitive and teams get alert fatigue from false positives, too conservative and we miss critical failures. We implemented a configurable three-tier system with adjustable thresholds, allowing teams to tune sensitivity based on risk tolerance. Historical probability distributions helped us identify optimal default thresholds of 50% for warnings and 70% for critical alerts.

Challenge 5: User Experience Design Complex ML predictions needed to be understandable for pit crew engineers who aren't data scientists. We solved this through visual gauge charts with intuitive color coding (everyone understands red means danger), clear maintenance recommendations written in plain language, historical trends that show context (is this sensor always high or just recently?), and component-specific alerts that tell users exactly what to inspect.

Accomplishments that we're proud of

High Model Accuracy: Our Random Forest classifier achieves 97% precision on healthy components and 91% precision on failure detection, with an overall ROC-AUC score of 0.96. This level of accuracy makes the system trustworthy for real-world deployment where false alarms are costly.

Production-Ready Dashboard: We built a fully functional, professional monitoring interface with real-time updates, not just a proof-of-concept. The dashboard handles continuous data streams, gracefully manages errors, and provides an intuitive user experience that requires minimal training.

Comprehensive Feature Engineering: Creating 15+ meaningful features from just 9 raw sensors demonstrates deep understanding of the problem domain. Features like vibration-temperature ratios and runtime-load ratios capture nuanced failure patterns that simple thresholding would miss.

Clean, Modular Code Architecture: The codebase separates data simulation, feature engineering, model training, and dashboard visualization into distinct modules. This makes the system maintainable, testable, and extensible for future enhancements. Proper error handling ensures robustness in production environments.

Practical Alert System: The three-tier warning system with actionable recommendations bridges the gap between ML predictions and real-world maintenance decisions. Engineers receive specific guidance (e.g., "Schedule maintenance within 24 hours") rather than just probability scores.

Open Source Contribution: Full documentation and clean code published on GitHub enable the community to learn from, extend, and adapt the system for other predictive maintenance applications beyond Formula 1.

What we learned

Technical Skills Acquired: We gained deep expertise in advanced feature engineering techniques for time-series sensor data, including creating meaningful interaction features and polynomial terms. We learned Random Forest hyperparameter optimization for real-time predictions, balancing model complexity with inference speed. Streamlit dashboard development taught us how to build responsive web applications with live data streaming. We mastered model persistence strategies using joblib for efficient serialization and deployment.

Domain Knowledge Gained: Understanding F1 component failure patterns required studying mechanical engineering principles we hadn't encountered before. We learned how vibration signatures indicate bearing wear, how temperature gradients reveal thermal stress, and how pressure drops signal hydraulic system degradation. This domain expertise was crucial for effective feature engineering.

Software Engineering Practices: We implemented modular code architecture for machine learning projects, separating concerns cleanly between data processing, model training, and inference. We learned Git version control workflows including branching strategies and

commit message conventions. Writing comprehensive documentation taught us to explain technical concepts clearly for diverse audiences. User experience design showed us that the best ML model is worthless without an intuitive interface.

Critical Insight: The most important lesson was that model accuracy alone doesn't determine success in real-world applications. A 99% accurate model that engineers don't trust or understand will be ignored. We learned that visualization, clear communication of uncertainty, and actionable recommendations are just as important as algorithmic performance. Building trust through transparent explanations and intuitive interfaces is essential for AI adoption.

Data Science Realities: Working with simulated data taught us the importance of understanding data generation processes. Every assumption in our simulation (e.g., how vibration correlates with temperature) directly impacted model behavior. This experience will make us better at working with real data by questioning data quality, understanding collection methods, and validating assumptions rigorously.

What's next for PitPredict - AI-Powered Predictive Maintenance System

Phase 1: Hardware Integration (Immediate - 3 months) Connect the system to real IoT sensors using Arduino or Raspberry Pi platforms. Implement support for CAN bus protocol, the standard automotive communication system used in F1 cars. Build real-time data ingestion pipelines that handle multiple sensor streams simultaneously. Conduct validation testing with racing simulators before track deployment.

Phase 2: Advanced Machine Learning (6-12 months) Implement LSTM (Long Short-Term Memory) neural networks for time-series forecasting that can predict failure probability hours or days in advance. Add anomaly detection algorithms to identify unknown failure modes not seen during training. Develop XGBoost ensemble models to improve prediction accuracy further through gradient boosting. Integrate explainable AI techniques (SHAP values) to provide transparency into which sensor patterns drive each prediction, building engineer trust.

Phase 3: Enterprise Features (12-18 months) Deploy the system on cloud infrastructure (AWS or Azure) for remote monitoring, enabling team members anywhere to access live data. Develop native mobile applications for iOS and Android, giving pit crew and strategists on-the-go access to alerts. Build an automated email and SMS notification system for critical alerts sent to designated team members. Create multi-team collaboration dashboards with role-based access control. Implement PostgreSQL database for long-term historical analysis and trend identification across seasons. Establish integration pathways with existing F1 telemetry systems through documented APIs.

Phase 4: Industry Expansion (18-24 months) Adapt the system for aerospace predictive maintenance, monitoring aircraft engine components and hydraulic systems. Customize for manufacturing equipment monitoring in production lines where downtime is costly. Develop a medical device reliability application for hospital equipment. Create a power plant version for

turbine and generator monitoring. Partner with fleet management companies for commercial vehicle maintenance.

Immediate Next Steps: Establish partnerships with amateur racing teams willing to pilot the technology. Our first goal is collecting real telemetry data during practice sessions at local racing events. We will use this data to refine our models and validate accuracy against actual component failures. Simultaneously, we plan to develop a mobile companion app for alerts and quick status checks. We are also exploring commercial deployment models and consulting with racing team engineers to understand additional feature requirements.

Technical Roadmap: Migrate from simulated to real sensor data while maintaining model performance. Implement continuous learning pipelines where the model improves as it encounters more failure cases. Build A/B testing infrastructure to evaluate model updates safely. Develop comprehensive monitoring and logging systems for production deployment. Create automated retraining schedules triggered by model performance degradation.

Business Development: We plan to present PitPredict at motorsport engineering conferences and hackathons. Our target is securing a pilot program with one racing team in the next six months. We will gather feedback, iterate rapidly, and build case studies demonstrating cost savings and performance improvements. Long-term, we envision PitPredict becoming a standard tool across motorsport and expanding into adjacent industries where predictive maintenance creates value.