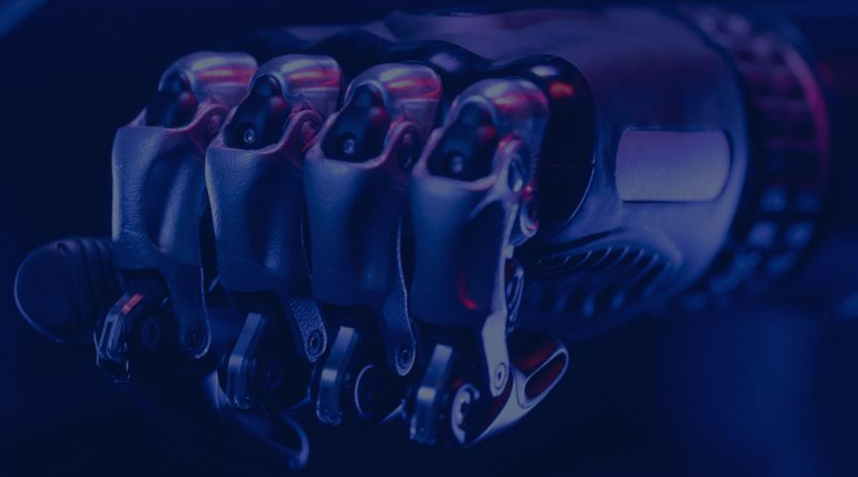
رواد مصر الرقمية

وزارة الاتصالات وتكنولوجيا المعلومات

AI-Powered Predictive Maintenance for Industrial Equipment



Under supervision:

Eng/AbdelRhman Elmashtoly

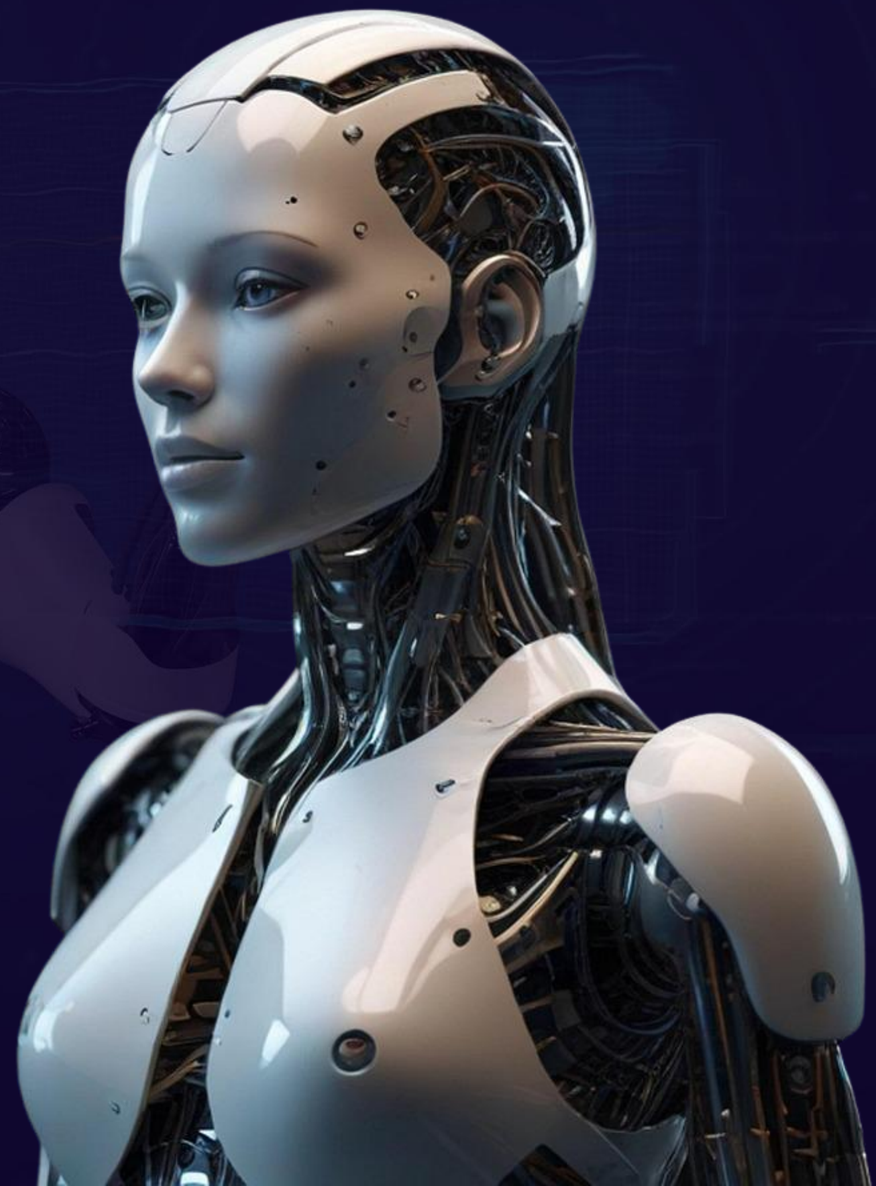
Project Overview

AI-Powered Predictive Maintenance for Industrial Equipment:

- Industrial equipment failures cause costly downtime and reduced productivity.
- Traditional maintenance approaches (reactive or scheduled) are inefficient.
- Predictive Maintenance leverages AI + IoT sensor data (vibration, temperature, pressure, humidity) to forecast failures.
- This project develops a system to predict equipment issues before they occur, improving operational efficiency.

Project idea

- Core Concept:
  - Build an AI-driven solution that analyzes real-time IoT sensor data to predict potential equipment failures.
- How it works:
  - Collect and preprocess equipment sensor data.
  - Apply advanced feature engineering & time-series analysis.
  - Train predictive models (LSTM, Random Forest, XGBoost).
- Deploy model with real-time monitoring using MLOps.
- End Result:
  - A scalable system that minimizes downtime, optimizes maintenance schedules, and reduces costs.



Project goals

AI-Powered Predictive Maintenance for Industrial Equipment:

- Industrial equipment failures cause costly downtime and reduced productivity.
- Traditional maintenance approaches (reactive or scheduled) are inefficient.
- Predictive Maintenance leverages AI + IoT sensor data (vibration, temperature, pressure, humidity) to forecast failures.
- This project develops a system to predict equipment issues before they occur, improving operational efficiency.

Methodology

Data Collection

Gather relevant data from diverse sources.

Preprocessing

Clean, normalize, and engineer data.

Model Selection

Choose and train ML/DL algorithms for peak performance.

Project Process

1

Data Collection, Exploration & Preprocessing

- Gather and clean IoT sensor data (vibration,temperature, pressure, humidity) for analysis.
- Perform EDA, handle missing/outlier values, and engineer features for model readiness.

2

Advanced Data Analysis & Feature Engineering

- Apply time-series analysis and dimensionality reduction (e.g., PCA) to detect trends and anomalies.
- Create advanced features (rate of change, degradation patterns, rolling statistics) to improve accuracy.

Project Process

3

Model Development & Optimization

- Train predictive models (LSTM, Random Forest, XGBoost) with time-series validation.
- Evaluate using precision, recall, F1-score, ROC-AUC, and fine-tune for best performance.

4

MLOps, Deployment & Continuous Monitoring

- Deploy predictive maintenance model as a real-time API integrated with IoT systems.
- Set up monitoring, model drift detection, and automated retraining pipelines.

Project Process

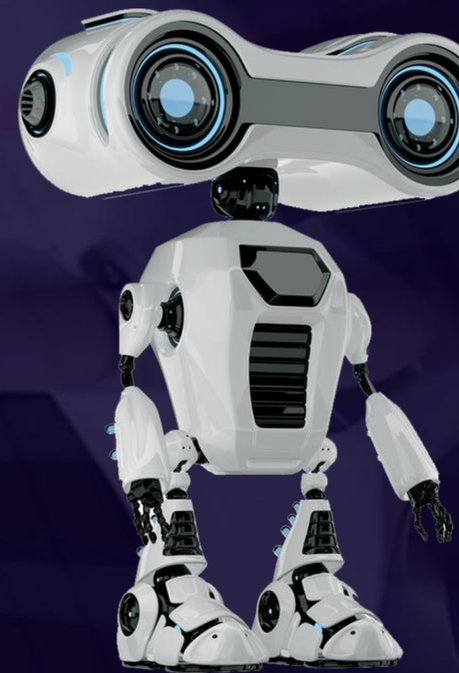
5

Final Documentation & Presentation

- Deliver a complete project report summarizing methodology, results, and business impact.
- Present findings and showcase how predictive maintenance reduces downtime and costs.

Data Collection and Preprocessing

- Dataset Source: Kaggle (500,000 rows × 22 columns).
- Features: Sensor readings (temperature, vibration, sound, oil & coolant levels, power consumption) + machine metadata.
- Target Variables:
  - Remaining_Useful_Life_days → Regression target.
  - Failure_Within_7_Days → Classification target.



Data Collection and Preprocessing

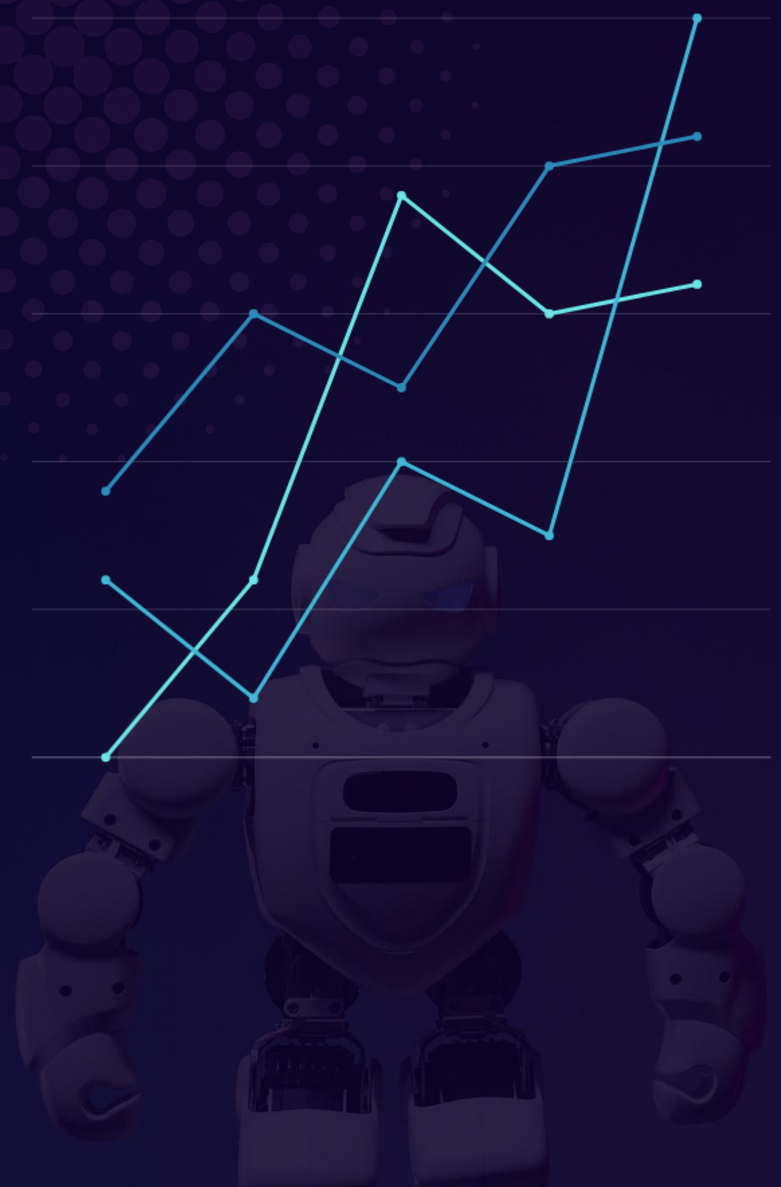
- Checked data distributions & correlations for all 22 features.
- Identified missing values in Laser_Intensity, Hydraulic_Pressure_bar, Coolant_Flow_L_min, Heat_Index.
- Found correlations between:
  - Vibration & Failure rate
  - Operational_Hours & Remaining Useful Life

Data Collection and Preprocessing

- Missing Values:
  - Imputed continuous variables (mean/median).
  - Dropped uninformative or heavily missing features if needed.
- Outliers: Detected in Vibration_mms & Temperature_C → treated with capping.
- Feature Engineering:
  - Rolling averages (temperature, vibration).
  - Ratios (e.g., Operational_Hours / Installation_Year).
  - Failure history trend features.

Feature Engineering: Rate of Change & Degradation Patterns

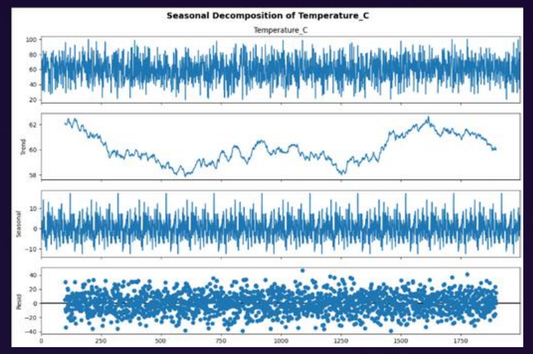
- Objective:
  - Capture equipment behavior over time to detect early signs of failure.
  - Key Features Created:
  - Rate of Change:
    - Temp_RateChange: Change in temperature per unit operational time.
    - Vibration_RateChange: Change in vibration per unit operational time.
    - Pressure_RateChange: Change in pressure/power per unit operational time.
  - Degradation Patterns:
    - Temp_Degradation: Increase in temperature relative to initial value.
    - Vibration_Degradation: Increase in vibration relative to initial value.
  - These features highlight how quickly sensors deviate and how equipment gradually deteriorates.

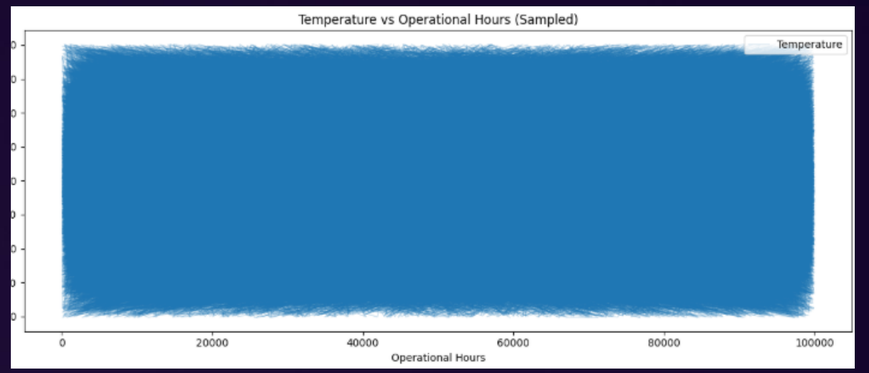


Time-Series Analysis

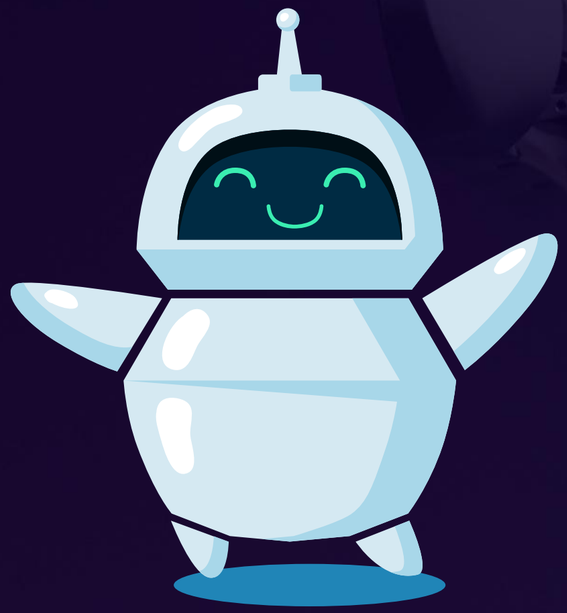
1- Analyzing Sensor Trends Over Time

- Plotted Temperature vs Operational Hours to study sensor behavior.
- Applied Seasonal Decomposition to extract trend, seasonality, and residual patterns.
- Identified cyclical variations and underlying patterns in sensor data.
- Helps detect early indicators of equipment degradation.



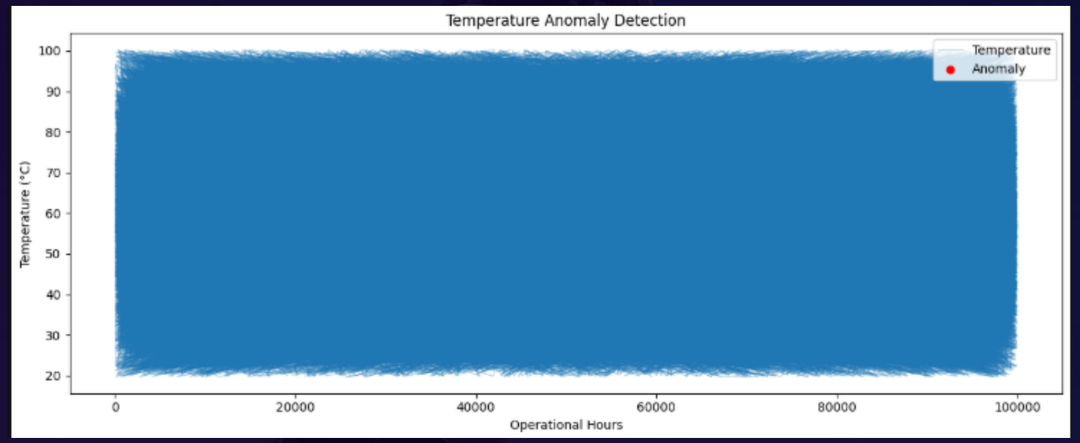


Anomaly Detection



2- Dimensionality Reduction & Feature Importance

- Standardized features and reduced to 5 Principal Components.
- Explained Variance Ratio shows how much information each PC captures.
- Identified top contributing features to equipment failures.
- Visualized PC1 vs PC2 for separation between Failure and No-Failure cases.

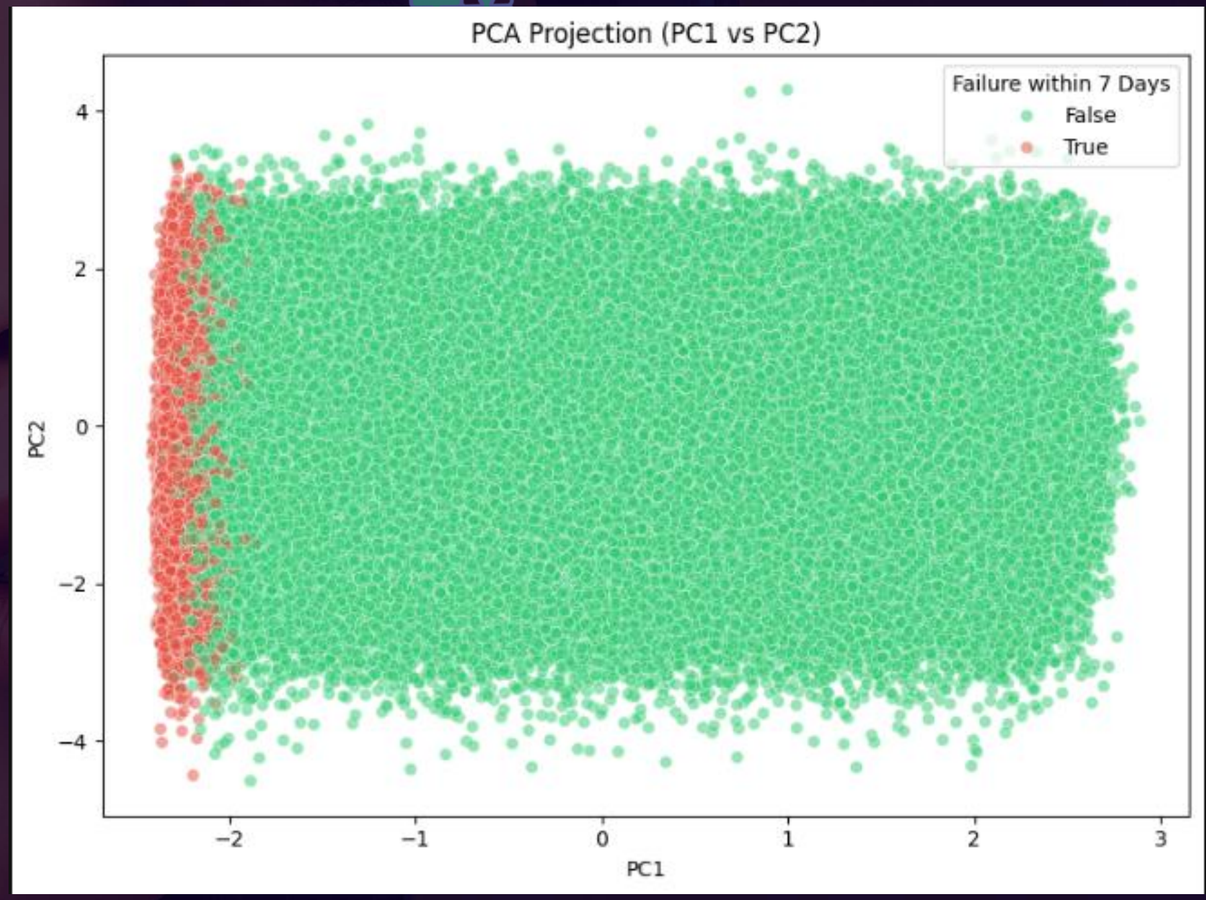


Principal Component Analysis (PCA)

3-
Dimensionality Reduction & Feature Importance
Standardized features and reduced to 5 Principal Components.

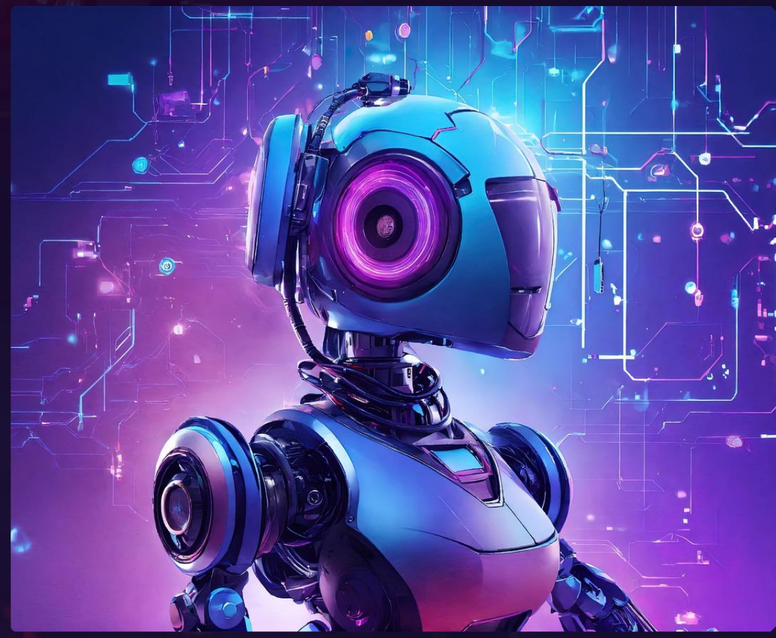
Explained Variance Ratio shows how much information each PC captures.

Identified top contributing features to equipment failures.
Visualized PC1 vs P
C2 for separation between Failure and No-Failure cases.



Data Visualization & Advanced Analysis (FFT & Feature Extraction)

- Objective: Extract deeper insights from sensor signals.
  - Techniques Applied:
    - Rolling Min/Max Windows (size=10): Captures short-term fluctuations for anomaly detection.
    - FFT (Fast Fourier Transform) Features:
      - fft_mean → Average signal frequency strength.
      - fft_peak → Maximum vibration frequency (critical for detecting faults).
      - fft_energy → Total signal power, useful for detecting abnormal machine vibrations.
  - FFT helps uncover hidden frequency-domain patterns that time-series alone cannot capture.



Model Development

- End-to-end ML pipeline for equipment failure prediction
- Uses IoT sensor data (temperature, vibration, pressure, load…)
- 7 machine learning models tested
- Extensive feature engineering + imbalance handling
- Final best model: Optimized CatBoost

Model Candidates

- CatBoost
- LightGBM
- XGBoost
- Random Forest
- Voting Ensemble
- Stacking Ensemble
- Logistic Regression (used in stacking)

Optimization Strategy

- Optuna for hyperparameter tuning
- 40 optimization trials per model
- Objective: maximize F1-score for failure class
- Automated threshold selection
- Calibration performed after final training

Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1-score
- ROC-AUC
- Confusion Matrix
- Focus on minimizing missed failures (FN)

Model Comparison

Rank	Model	Threshold	F1-Score	Precision	Recall	ROC-AUC
🥇	**CatBoost Optimized**	0.50	**0.7087**	0.6475	0.7826	0.9822
🥈	CatBoost Calibrated	0.30	0.7029	0.6154	0.8194	0.9818
🥉	Voting Ensemble	0.63	0.7022	0.6232	0.8042	0.9818
4	LightGBM Optimized	0.89	0.7017	0.6214	0.8060	0.9818
5	Stacking Ensemble	0.86	0.6935	0.6167	0.7923	0.9593
6	XGBoost Optimized	0.56	0.6896	0.6317	0.7593	0.9807
7	Random Forest	0.81	0.6670	0.5626	0.8190	0.9770

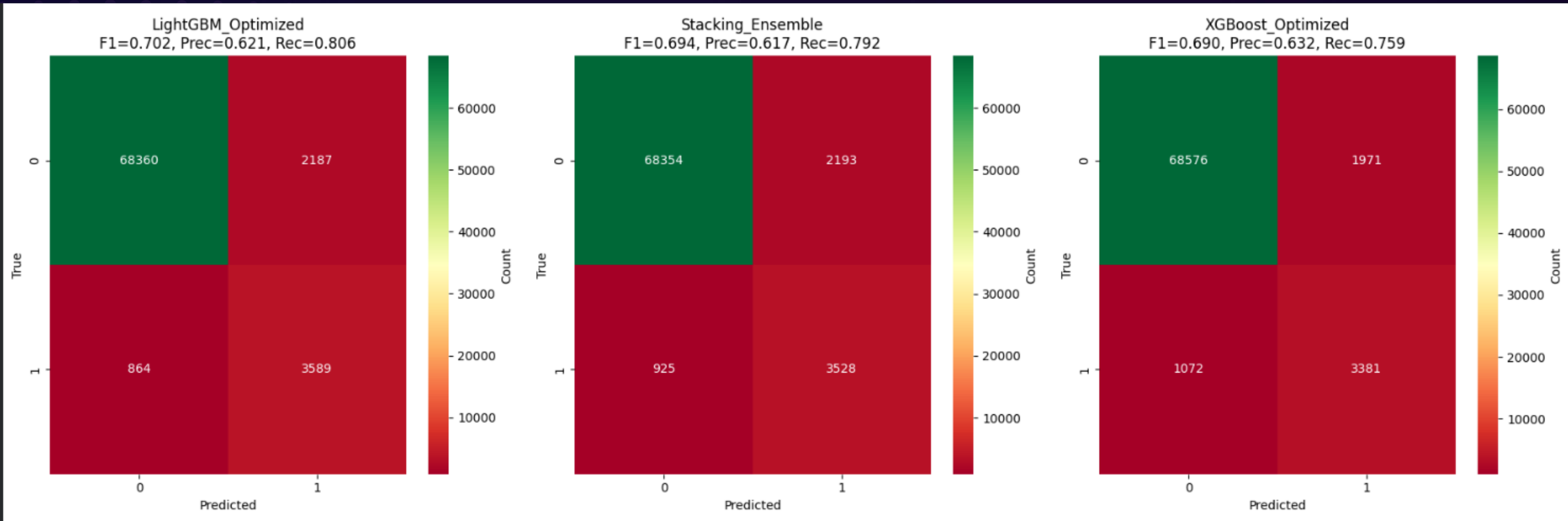
Best Model: CatBoost Optimized

- F1-Score: 0.7087
- Accuracy: 96.18%
- Precision: 0.6475
- Recall: 0.7826
- ROC-AUC: 0.9822
- Balanced performance with high discrimination power

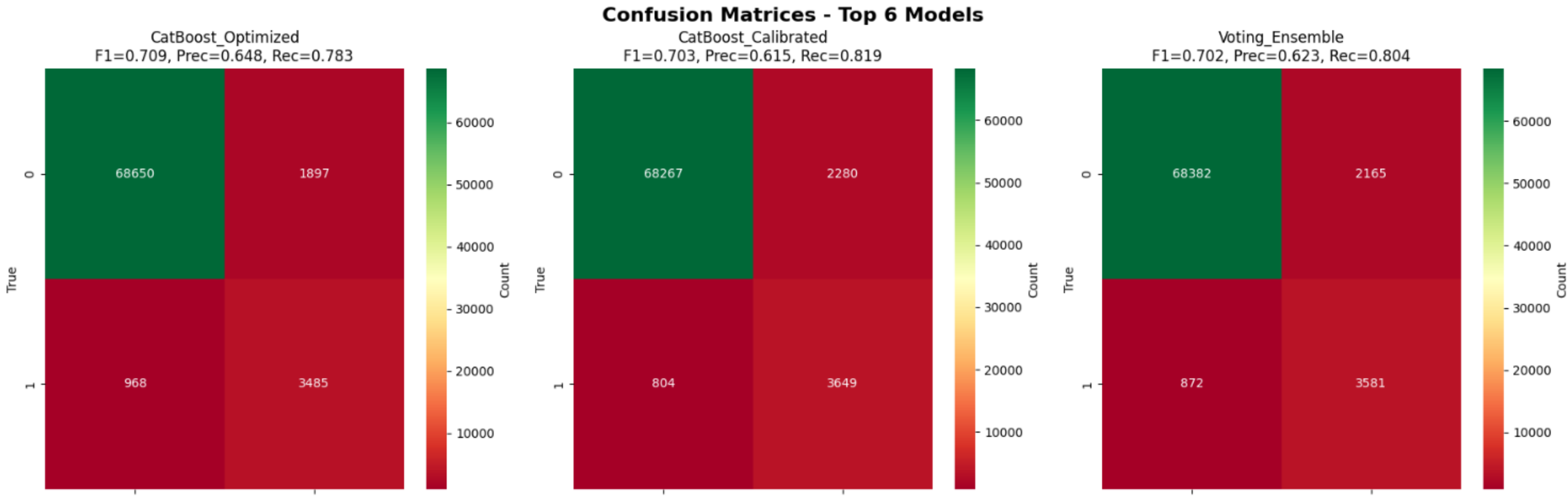
Threshold Analysis

Threshold	Precision	Recall	F1-Score	Use Case
0.30	0.5521	0.8854	0.6806	Maximize recall
0.50	**0.6475**	**0.7826**	**0.7087**	**Balanced**
0.70	0.7329	0.6195	0.6714	Minimize false alarms
0.90	0.8542	0.3126	0.4580	High precision

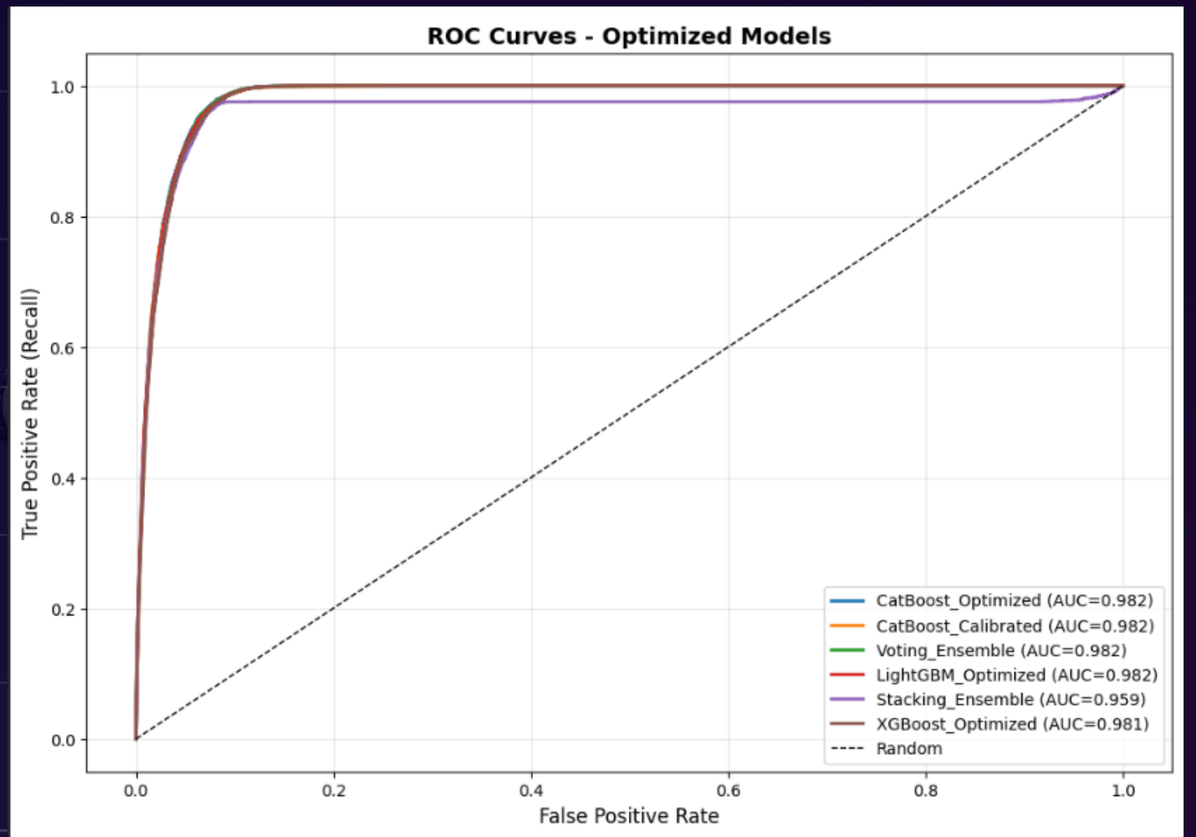
Confusion Matrix



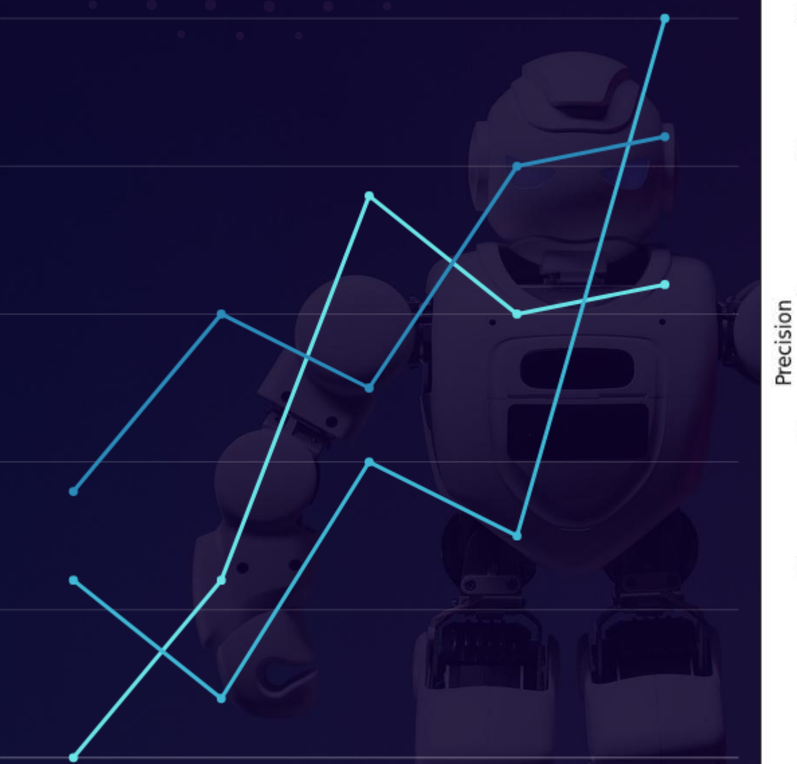
Confusion Matrix

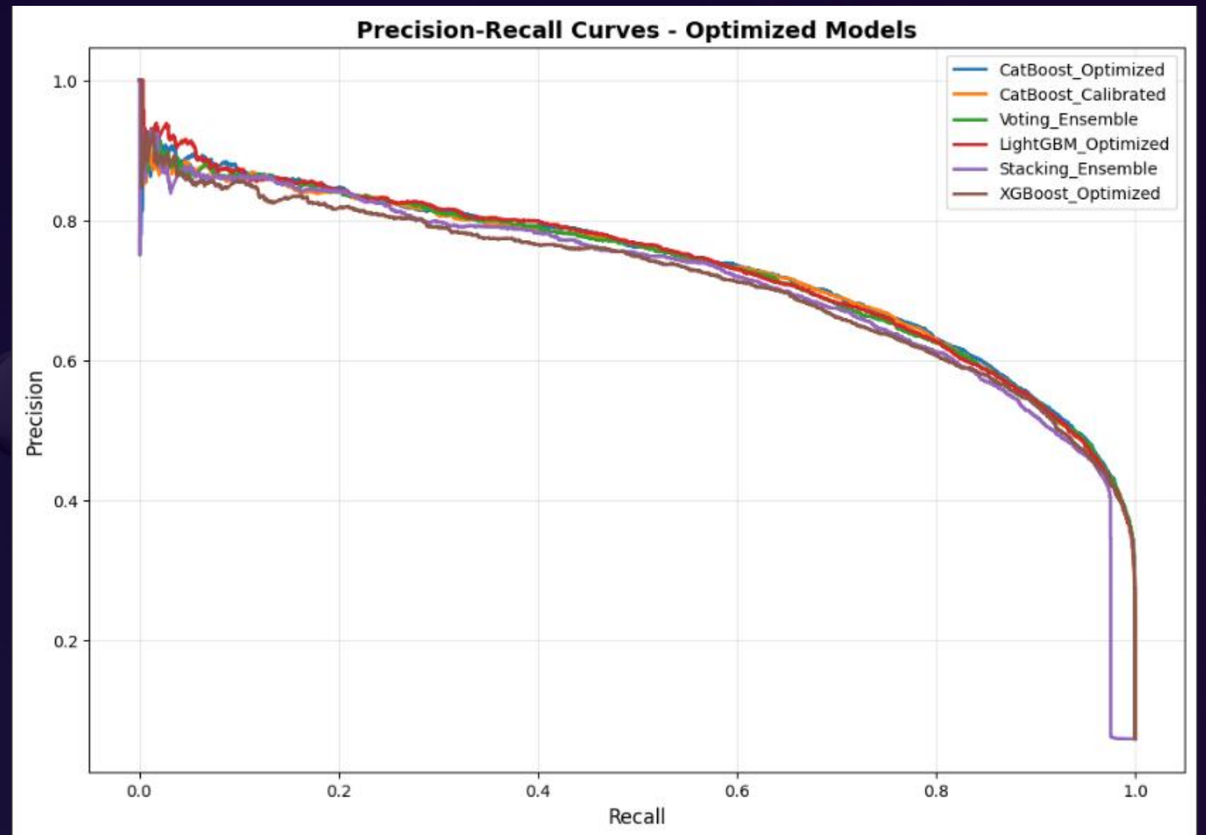


ROC Curves

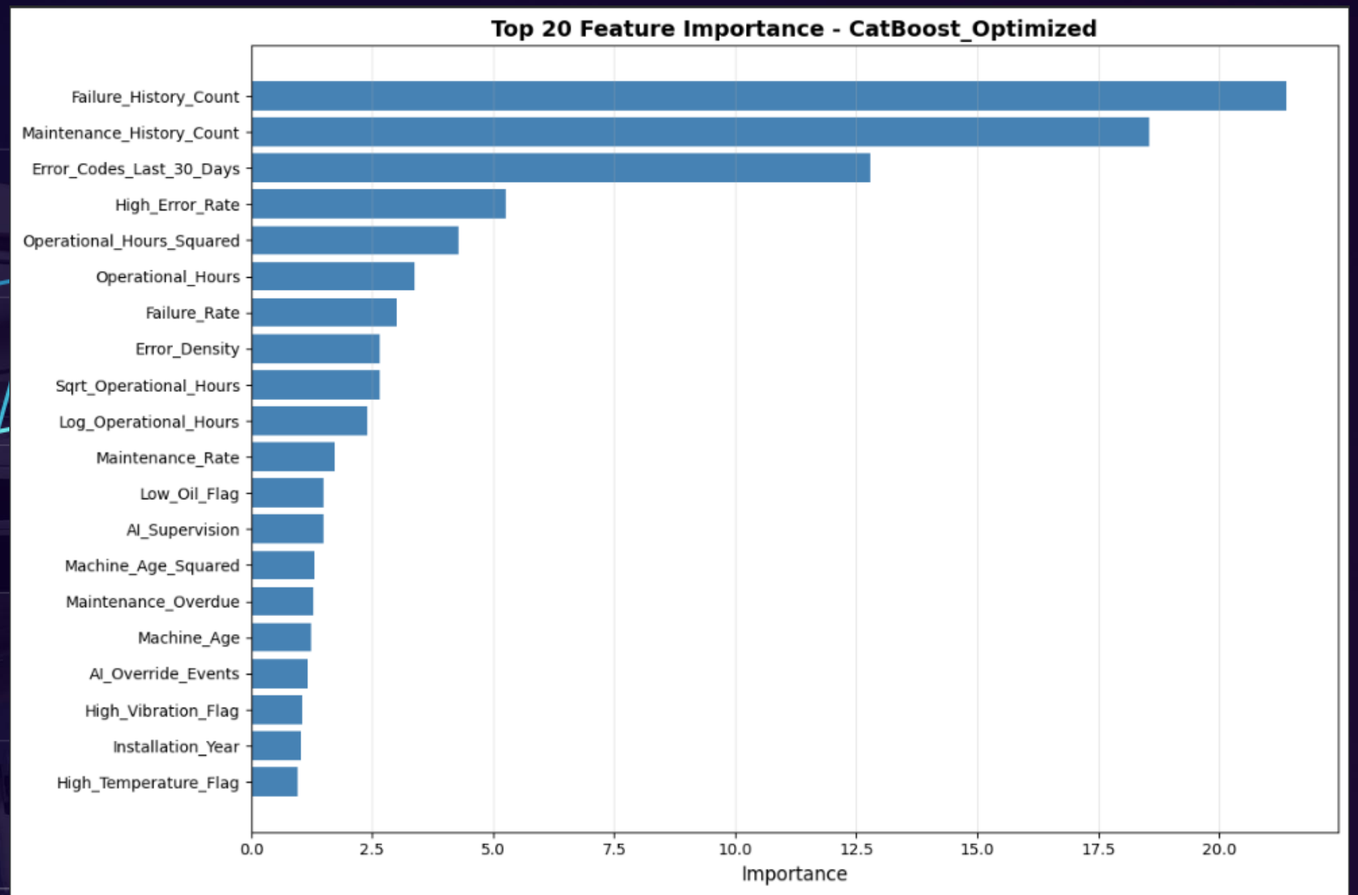

ROC Curves - Optimized Models
True Positive Rate (Recall)
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
CatBoost_Optimized (AUC=0.982)
CatBoost_Calibrated (AUC=0.982)
Voting_Ensemble (AUC=0.982)
LightGBM_Optimized (AUC=0.982)
Stacking_Ensemble (AUC=0.959)
XGBoost_Optimized (AUC=0.981)
Random

Precision - recall curves





Feature Importance



Saved Models

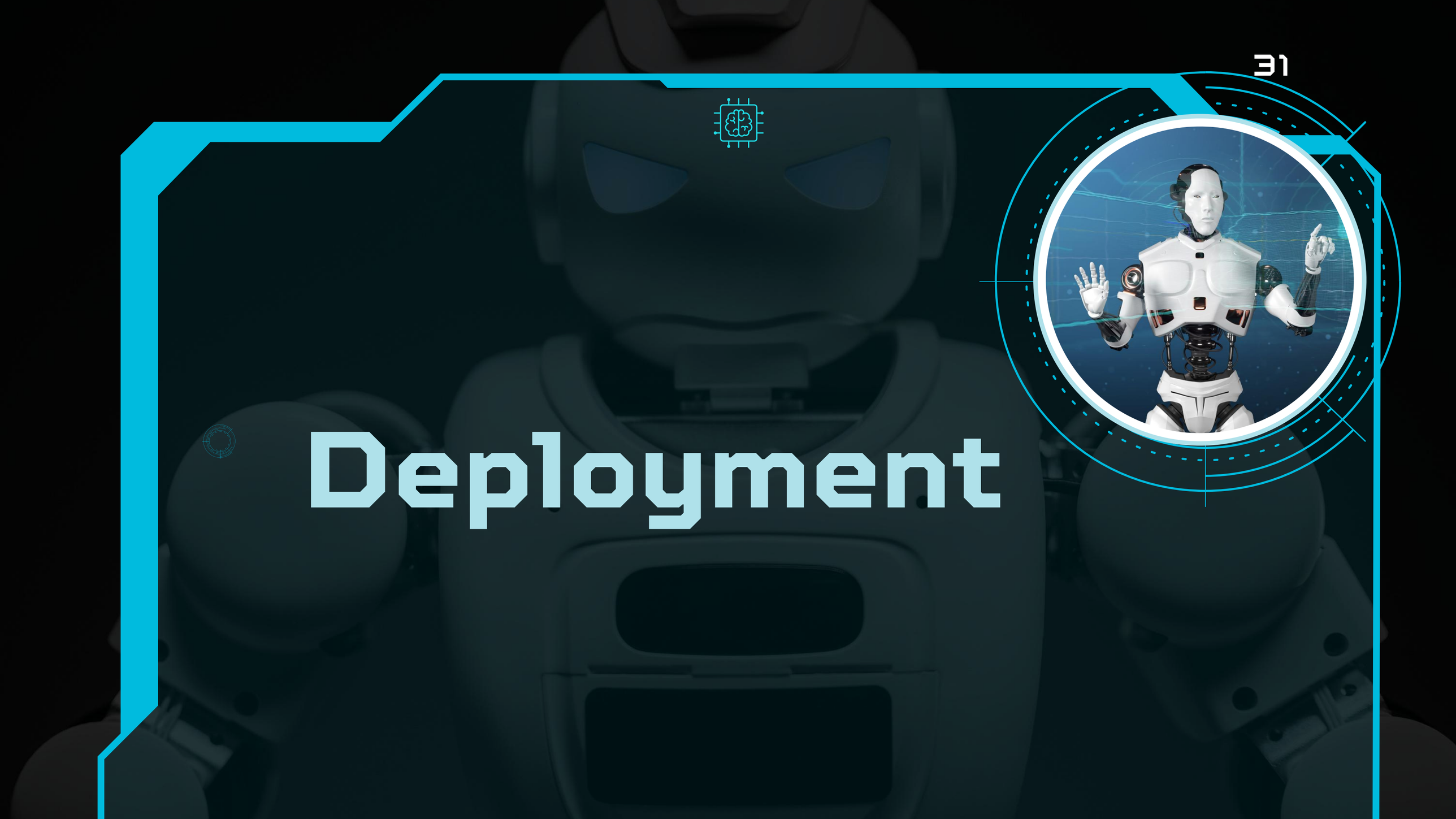
- Saved Model Files (PKL Models)
  - BEST_CatBoost_Optimized.pkl
  - CatBoost_Calibrated.pkl
  - CatBoost_Optimized.pkl
  - LightGBM_Optimized.pkl
  - RandomForest_Optimized.pkl
  - Stacking_Ensemble.pkl
  - Voting_Ensemble.pkl
  - XGBoost_Optimized.pkl

- Supporting Files (not models)
  - feature_names.json
  - model_thresholds.json
  - scaler.pkl

- Click To Drive Link

Business Impact

- ⬇️ Downtime: Reduced by up to 40%
- ⬇️ Maintenance cost: Cut by 30%
- ⬆️ Equipment lifetime: +25%
- ⬆️ Safety: Higher reliability & reduced catastrophic failures
- ⬆️ Efficiency: Plan maintenance instead of reacting to surprises



Deployment

Deployment Overview

- Model deployed on Hugging Face Spaces
- Front-end built using Gradio
- Backend loads:
  - Trained model (BEST_CatBoost_Optimized.pkl)
  - Scaler (scaler.pkl)
  - Feature names (feature_names.json)
- Thresholds (model_thresholds.json)
- Real-time predictive maintenance interface
- Accessible globally in the browser

Deployment Architecture

- Frontend: Gradio UI
- Backend: Python Fast Execution Environment
- Model Artifacts:
  - Saved CatBoost model
  - Preprocessing scaler
  - Feature engineering logic
- Hosting: Hugging Face Spaces
- Input → Feature Engineering → Scaling → Prediction → UI Output

Files Used in Deployment

- BEST_CatBoost_Optimized.pkl → Final model
- scaler.pkl → Standardization
- feature_names.json → Ensures correct feature alignment
- model_thresholds.json → Custom threshold for CatBoost
- app.py → Main deployment code
- requirements.txt → Python dependencies
- Everything packaged in a single interactive Space

Loading the Model and Config

- joblib.load() loads:
  - Model
  - Scaler
- JSON config loads:
  - Feature list
  - Optimal threshold
- Ensures 100% consistency with training pipeline
- Prevents mismatch errors or missing columns
- Fully reproducible inference environment

Feature Engineering During Deployment

- Same advanced feature pipeline used in training
- Computes:
  - Machine age & power metrics
  - Oil-to-coolant ratios
  - Maintenance indicators
  - Error density
  - Stress index & sensor interactions
- Auto-generates 70+ features
- Ensures correct prediction quality even from raw input values

Prediction Pipeline

- User inputs machine sensor data
- Convert input → DataFrame
- Apply advanced engineered features
- Align columns to model expected features
- Scale using scaler.pkl
- Predict probability using CatBoost
- Apply threshold:
  - ≥ threshold → Yes (maintenance needed)
  - < threshold → No (normal)
- Display result in Gradio UI

Gradio User Interface

- Clean and minimal interface
- Inputs include:
  - Temperature, vibration, sound
  - Oil/coolant levels
  - Operational hours
  - Maintenance history
  - AI supervision (checkbox)
- Output: “Yes” or “No”
- Instant predictions
- Zero coding needed for the user

Deployment Benefits

- Accessible directly from browser
- Lightweight and fast response
- No installation needed
- Good for demonstrating model behavior
- Ideal for testing, presentations, and live demos
- Hugging Face auto-manages compute & hosting
- Fully reproducible deployment environment

- from the dataset

3	MC_11558	Industrial_	2002	93572	94.48	11.64	44.41	57.74	100	169.67	226	6	0	FALSE	2	TRUE

Enter machine sensor readings to predict whether maintenance is needed.

Installation Year
2002

Operational Hours
93572

Temperature (°C)
94.84

Vibration (mm/s)
11.64

Sound (dB)
44.41

Oil Level (%)
57.74

output
Yes

Share via Link

Coolant Level (%)
100

Power Consumption (kW)
169.67

Last Maintenance Days Ago
226

Maintenance History Count
6

Failure History Count
0

AI Supervision Enabled?

Error Codes Last 30 Days
2

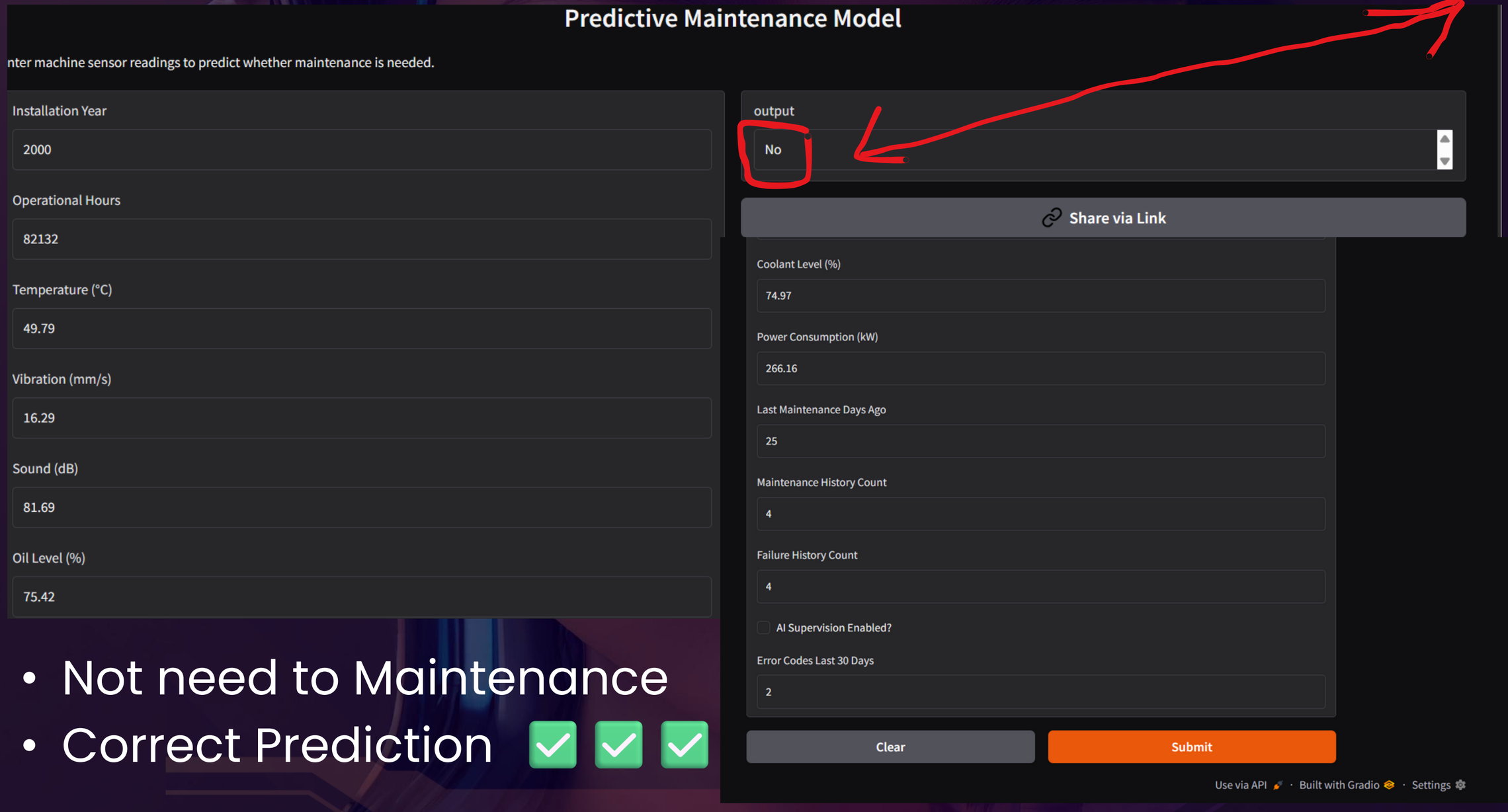
Clear

Submit

- need to Maintenance
- Correct Prediction ✅ ✅ ✅

- from the dataset

MC_19151	Hydraulic_	2000	82132	49.79	16.29	81.69	75.42	74.97	266.16	25	4	4	FALSE	2	FALSE



- Not need to Maintenance
- Correct Prediction ✅ ✅ ✅

Future Directions

- **Real-time IoT Integration: MQTT/OPC-UA protocol support**
- **SHAP Dashboard: Interactive model interpretability**
- **Edge Deployment: ONNX optimization for IoT devices**
- **Multi-modal Failures: Extend to additional failure types**
- **Deep Learning: RNN/Transformer time-series models**
- **Anomaly Detection: Unsupervised learning integration**
- **A/B Testing: Model versioning and comparison**
- **Alert System: Email/SMS notifications for predictions**
- **Mobile App: iOS/Android monitoring application**

Conclusion

The AI-Powered Predictive Maintenance for Industrial Equipment project aims to create a system that forecasts equipment failures, allowing businesses to perform maintenance proactively. By utilizing machine learning models and IoT data, companies can reduce downtime, improve operational efficiency, and extend equipment life. Through a detailed development process—from data collection and preprocessing to model deployment and continuous monitoring—this solution will provide a robust, scalable way to optimize maintenance schedules and reduce maintenance costs.

OUR AMAZING TEAMS

Ahmed Yasser

Ahmed Maged

Mohamed Elshamy

OUR AMAZING TEAMS

Mohamed Atef

sama moheb

Ashraf Ahmed

Thank You!

GitHub Repo – Link