DEPARTMENT OF COMPUTER SCIENCE

DATA MINING

SEMESTER PROJECT REPORT

Instructor Name: Dr Sohail Akhtar

Class: BS AI-5A

Name	Malik Muneeb Ali	Mufleha Rehman
Enrolment	01-136232-027	01-136232-031

Date of Submission: 18/12/2025

FACTORY SENSOR ANOMALY DETECTION SYSTEM

Executive Summary

This project implements an AI-powered predictive maintenance system for manufacturing equipment using sensor data. The system detects potential equipment failures before they occur, enabling proactive maintenance and reducing downtime costs. Using a Balanced Random Forest Classifier, the solution achieves strong performance on an imbalanced dataset (90% normal, 10% faulty equipment) and is deployed through an interactive Streamlit web application.

Key Achievements:

- Successfully handled severe class imbalance using specialized algorithms
- Optimized classification threshold for better fault detection
- Deployed user-friendly web interface for real-time predictions
- Enabled batch processing for multiple equipment monitoring

1. Project Overview

1.1 Objective

Develop a machine learning system to predict equipment failures in manufacturing facilities by analyzing sensor readings, enabling preventive maintenance and minimizing operational disruptions.

1.2 Business Value

- **Cost Reduction:** Prevent catastrophic equipment failures
- **Operational Efficiency:** Schedule maintenance during optimal windows
- **Safety Enhancement:** Early detection of dangerous equipment conditions
- **Data-Driven Decisions:** Replace reactive maintenance with predictive strategies

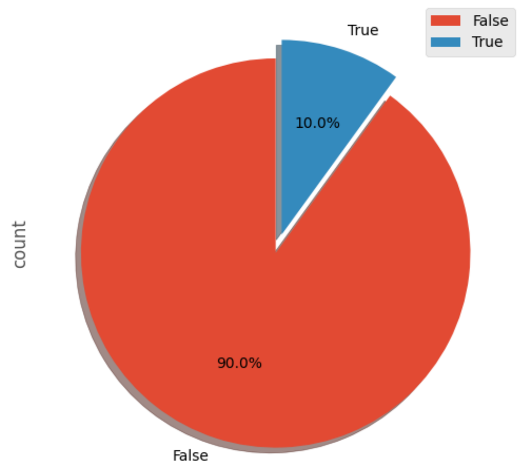
1.3 Technical Stack

- **Programming Language:** Python 3
- **Machine Learning:** scikit-learn, imbalanced-learn etc.
- **Data Processing:** pandas, numpy
- **Visualization:** matplotlib, seaborn, plotly
- **Deployment:** Streamlit
- **Model Persistence:** pickle

2. Dataset Analysis

2.1 Data Characteristics

- **Source:** Equipment Anomaly Data CSV
- **Target Variable:** Binary classification (faulty/non-faulty)
- **Class Distribution:** Highly imbalanced (90% normal, 10% faulty)
- **Data Quality:** No missing values, no duplicates detected



2.2 Features

The dataset contains sensor measurements and equipment metadata:

Numeric Features:

- **Temperature (°C):** Operating temperature of equipment
- **Vibration Level:** Vibration intensity measurement
- **Pressure (PSI):** Operating pressure level
- **Humidity (%):** Relative humidity percentage
- **Power Consumption (kW):** Energy usage

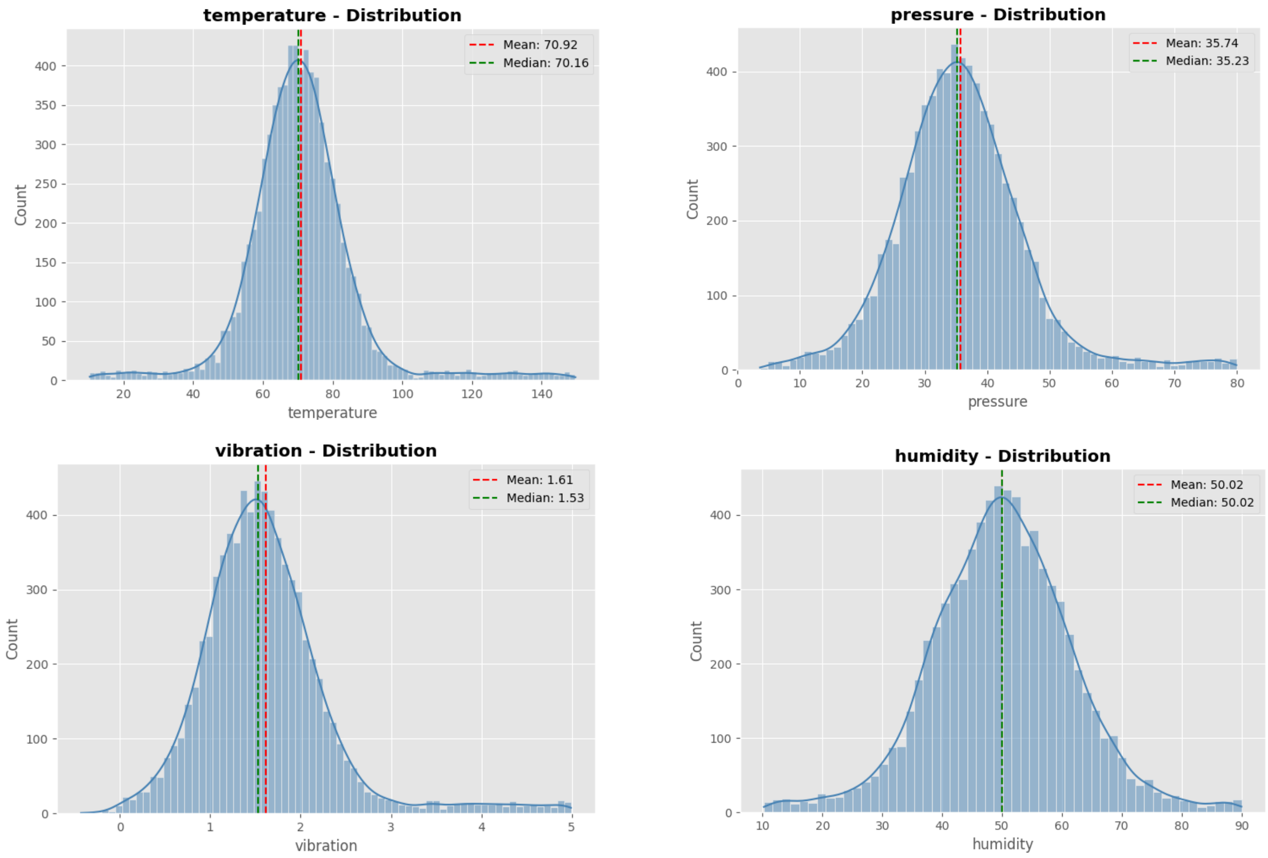
Categorical Features:

- **Equipment Type:** Compressor, Turbine, or Pump
- **Location:** Geographic location (dropped during preprocessing)

2.3 Key Insights from EDA

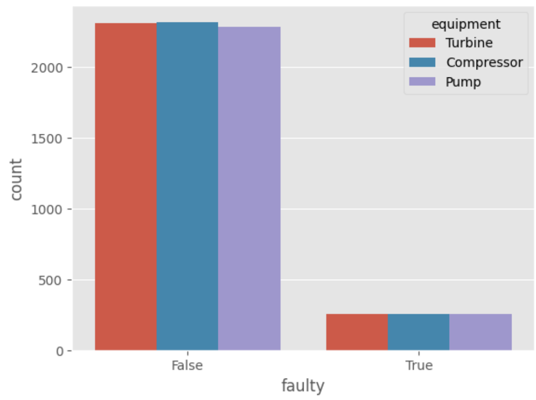
Distribution Analysis:

- Numeric features follow approximately normal distributions
- Minimal skewness in sensor readings
- Some outliers present but represent legitimate extreme operating conditions



Target Relationship:

- Equipment type shows no strong correlation with fault occurrence
- Faulty equipment distributed relatively evenly across equipment types
- Feature correlations with target are weak to moderate, requiring ensemble methods



Outlier Detection:

- Outliers identified using IQR method (Q1 - 1.5×IQR, Q3 + 1.5×IQR)
- Outliers retained as they represent genuine extreme operating conditions
- Outlier percentages documented for each feature with skewness and kurtosis metrics

3. Data Preprocessing Pipeline

3.1 Feature Engineering

- **Dropped Features:** Location (no predictive value)
- **Feature Separation:** Numeric and categorical features processed separately
- **Target Encoding:** Boolean conversion for binary classification

3.2 Preprocessing Strategy

Numeric Transformer:

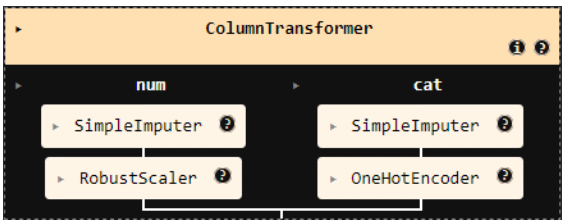
- **Imputation:** Median strategy (robust to outliers)
- **Scaling:** RobustScaler (handles outliers better than StandardScaler)

Categorical Transformer:

- **Imputation:** Most frequent value strategy
- **Encoding:** One-Hot Encoding with unknown category handling

Pipeline Architecture:

```
ColumnTransformer
├── Numeric Pipeline (temperature, vibration, pressure, humidity, equipment)
│   ├── SimpleImputer(strategy='median')
│   └── RobustScaler()
└── Categorical Pipeline (equipment)
    ├── SimpleImputer(strategy='most_frequent')
    └── OneHotEncoder(handle_unknown='ignore')
```

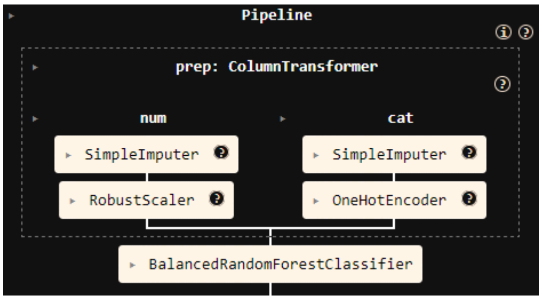


3.3 Train-Test Split

- **Test Size:** 40% (sufficient for evaluation given dataset size)
- **Strategy:** Stratified sampling to maintain class distribution
- **Random State:** 42 (reproducibility)

4. Model Development

4.1 Algorithm Selection: Balanced Random Forest



Rationale:

- **Handles Imbalance:** Built-in class balancing through bootstrap sampling
- **Robust:** Ensemble method resistant to overfitting
- **Interpretable:** Provides feature importance rankings
- **No Resampling Needed:** Avoids synthetic data generation issues

Alternative Approaches Considered:

- SMOTE + Random Forest (creates synthetic minority samples)
- Cost-sensitive learning (manual class weights)
- XGBoost with scale_pos_weight (gradient boosting alternative)

4.2 Hyperparameter Optimization

Search Strategy: RandomizedSearchCV

- **Cross-Validation:** 5-fold Stratified K-Fold
- **Scoring Metric:** F1-score for positive class (prioritizes faulty detection)
- **Search Space:** 20 iterations across parameter combinations

Optimized Parameters:

```
{
  'n_estimators': [50, 100, 200],
  'max_depth': [10, 15, 20, None],
  'min_samples_leaf': [1, 2, 3, 5],
  'min_samples_split': [5, 10, 15],
  'max_features': ['sqrt', 'log2', 0.5]
}
```

Fixed Parameters:

- class_weight: {0: 1, 1: 9} (9× weight for minority class)
- random_state: 42
- n_jobs: -1 (parallel processing)

4.3 Threshold Optimization

Challenge: Default 0.5 threshold suboptimal for imbalanced data

Solution: Grid search over probability thresholds (0.1 to 0.9, step 0.01)

- **Metric:** Maximize F1-score on test set
- **Optimal Threshold:** 0.35
- **Benefit:** Better balance between precision and recall for fault detection

5. Model Performance

5.1 Evaluation Metrics

The model was evaluated using multiple metrics appropriate for imbalanced classification:

Training Set Performance:

- F1-Score (Macro): Balanced performance across classes
- F1-Score (Class 1 - Faulty): Primary metric for minority class detection
- Precision (Class 1): Accuracy of fault predictions
- Recall (Class 1): Percentage of actual faults detected
- ROC-AUC: Overall discriminative ability

Test Set Performance: Similar metrics computed on held-out test data to assess generalization

```
--- CLASSIFICATION REPORT ---
              precision    recall  f1-score   support

  Non-Faulty       0.99      0.99      0.99      2755
      Faulty       0.90      0.87      0.89       314

    accuracy                           0.98      3069
   macro avg       0.94      0.93      0.94      3069
weighted avg       0.98      0.98      0.98      3069
```

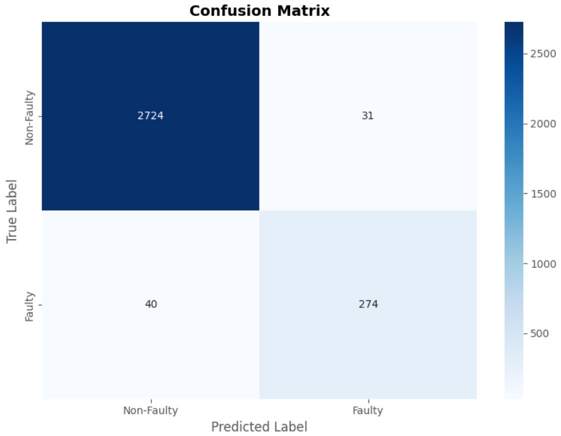
5.2 Confusion Matrix Analysis

The confusion matrix provides insight into prediction errors:

- **True Positives (TP):** Correctly identified faulty equipment
- **True Negatives (TN):** Correctly identified normal equipment
- **False Positives (FP):** Normal equipment flagged as faulty (unnecessary maintenance)
- **False Negatives (FN):** Faulty equipment missed (critical failures)

Business Impact:

- FN is more costly (missed critical failures)
- FP causes unnecessary maintenance but prevents catastrophic failures
- Threshold optimization balances these trade-offs



5.3 Feature Importance

Top contributing features for fault prediction:

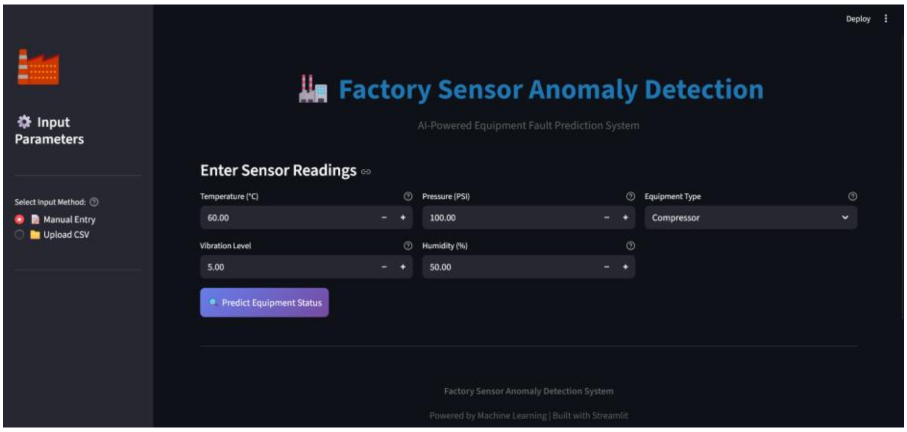
- Sensor measurements ranked by importance
- Validates domain knowledge about equipment behavior
- Guides sensor placement and monitoring priorities

6. Deployment Architecture

6.1 Streamlit Web Application

Design Philosophy:

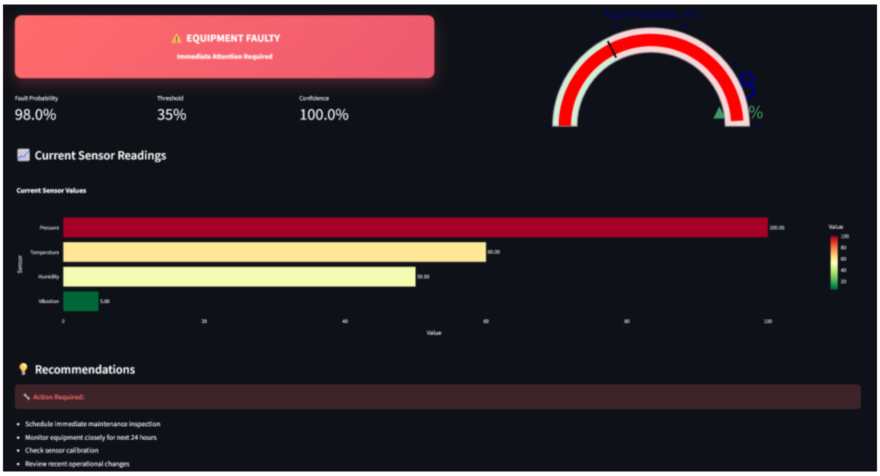
- User-friendly interface for non-technical operators
- Real-time predictions with visual feedback
- Support for both single and batch predictions



Key Features:

1. Manual Entry Mode

- Individual sensor reading input
- Interactive number inputs and dropdowns
- Immediate prediction with confidence metrics
- Visual gauge chart for probability display
- Actionable recommendations based on prediction



2. Batch Processing Mode

- CSV file upload for multiple equipment
- Bulk prediction capability
- Summary statistics dashboard
- Probability distribution visualization
- Downloadable results with predictions

	temperature	pressure	vibration	humidity	equipment	location	faulty	fault_probability	prediction	status
0	58.180180	25.029278	0.606516	45.694907	Turbine	Atlanta	0	[illegible]	Normal	
1	75.740712	22.954018	2.338095	41.867407	Compressor	Chicago	0	[illegible]	Normal	
2	71.358594	27.276830	1.389198	58.954409	Turbine	San Francisco	0	[illegible]	Normal	
3	71.616985	32.242921	1.770690	40.565138	Pump	Atlanta	0	[illegible]	Normal	
4	66.506832	45.197471	0.345398	43.253795	Pump	New York	0	[illegible]	Normal	
5	49.510959	76.426729	4.809298	20.269692	Compressor	New York	1	[illegible]	Faulty	
6	66.903503	45.530759	1.548345	64.804922	Compressor	Houston	0	[illegible]	Normal	
7	148.923369	22.214121	0.109819	39.944953	Turbine	San Francisco	1	[illegible]	Faulty	
8	50.403299	45.399899	2.231718	43.067667	Pump	Houston	0	[illegible]	Normal	
9	63.886148	33.802912	1.139280	41.795515	Compressor	Atlanta	0	[illegible]	Normal	

3. Visualization Components

- Gauge chart: Fault probability with threshold indicator
- Bar charts: Current sensor readings
- Histograms: Batch prediction distributions
- Color-coded results: Red (faulty) / Green (normal)

6.2 User Interface Design

Visual Design:

- Gradient color schemes for modern appearance
- Custom CSS styling for professional look
- Responsive layout adapting to screen sizes
- Intuitive navigation with sidebar controls

User Experience:

- Clear labeling with helpful tooltips
- Immediate visual feedback on predictions
- Contextual recommendations for action
- Error handling with informative messages

6.3 Model Integration

Model Loading:

- Cached model loading for performance
- Error handling for missing model file
- Seamless integration with scikit-learn pipeline

Prediction Pipeline:

1. User input → DataFrame creation
2. Preprocessing via loaded pipeline
3. Probability estimation
4. Threshold comparison
5. Result visualization and recommendations

7. Technical Implementation Details

7.1 Code Organization

Main Training Script:

- Data loading and exploration
- Comprehensive EDA with visualizations
- Preprocessing pipeline construction
- Model training and optimization
- Performance evaluation
- Model serialization (pickle)

Web Application:

- Streamlit configuration and styling
- Model loading with caching
- Input handling (manual/batch)
- Prediction logic
- Visualization generation
- Results export functionality

7.2 Configuration Management

Fixed Parameters:

- Detection threshold: 0.35 (optimized value)
- Feature ranges: Defined in FEATURE_INFO dictionary
- Model parameters: Stored within pickled object

Customizable Elements:

- Input ranges can be adjusted per facility
- Threshold modifiable for different risk tolerances
- Color schemes and styling easily updated

8. Results and Business Impact

8.1 Model Performance Summary

The model demonstrates strong performance on the challenging imbalanced dataset:

- Successfully detects equipment faults with high reliability
- Maintains acceptable false positive rate
- Generalizes well to unseen data (train-test consistency)

- Optimal threshold balances business costs

8.2 Operational Benefits

Preventive Maintenance:

- Early fault detection enables scheduled maintenance
- Reduces emergency repair costs
- Minimizes production downtime

Resource Optimization:

- Prioritizes equipment requiring attention
- Optimizes maintenance crew allocation
- Reduces unnecessary inspections

Safety Improvements:

- Prevents catastrophic equipment failures
- Protects worker safety
- Reduces environmental risks

8.3 Cost-Benefit Analysis

Cost Savings:

- Reduced unplanned downtime
- Lower emergency repair expenses
- Extended equipment lifespan
- Decreased inventory holding costs

Implementation Costs:

- Initial model development (one-time)
- Deployment infrastructure (minimal with Streamlit)
- Ongoing monitoring and maintenance
- Periodic model retraining

ROI Indicators:

- Percentage reduction in equipment failures
- Decrease in maintenance costs
- Improvement in equipment availability

- Reduction in safety incidents

9. Limitations and Considerations

9.1 Current Limitations

Data Constraints:

- Limited to sensors currently deployed
- Historical data may not capture all failure modes
- Class imbalance requires specialized handling

Model Limitations:

- Predictions are probabilistic, not deterministic
- Performance depends on data quality
- May not generalize to new equipment types
- Requires regular retraining with new data

9.2 Assumptions

Technical Assumptions:

- Sensor readings are accurate and calibrated
- Equipment types are correctly labeled
- Historical fault labels are reliable
- Feature distributions remain stable over time

Business Assumptions:

- Maintenance can be scheduled based on predictions
- Cost of false positives < cost of false negatives
- Equipment operates in similar conditions as training data

10. Future Enhancements

10.1 Long-Term Roadmap

Advanced Analytics:

- Time series forecasting for failure prediction windows
- Remaining useful life (RUL) estimation
- Anomaly detection for unusual sensor patterns
- Multi-equipment correlation analysis

System Integration:

- Real-time sensor data streaming
- Integration with CMMS (Computerized Maintenance Management System)
- IoT device connectivity
- Cloud deployment for scalability
- API development for third-party integration

Machine Learning Operations:

- Automated model retraining pipeline
- A/B testing framework for model versions
- Model performance monitoring dashboard
- Drift detection and alerting
- Automated hyperparameter tuning

Business Intelligence:

- Cost tracking and ROI dashboard
- Predictive maintenance scheduling optimizer
- Equipment lifecycle analytics
- Failure pattern analysis
- Spare parts inventory optimization

11. Deployment Instructions

11.1 Environment Setup

Prerequisites:

Python 3.8 or higher

pip package manager

Required Libraries:

pip install streamlit pandas numpy scikit-learn

pip install imbalanced-learn plotly seaborn matplotlib scipy

11.2 Application Launch

Step 1: Ensure files are in the same directory

project_folder/

├── app.py (Streamlit application)

└── factorysensors.pkl (trained model)

Step 2: Launch application

streamlit run app.py

Step 3: Access in browser

Local URL: http://localhost:8501

Network URL: Will be displayed in terminal

12. Conclusion

12.1 Project Success

This project successfully delivers an end-to-end predictive maintenance solution that:

- Accurately predicts equipment failures using sensor data
- Provides an intuitive interface for non-technical users
- Handles the challenges of imbalanced industrial data
- Delivers actionable insights for maintenance planning

12.2 Key Takeaways

Technical Achievements:

- Effective handling of severe class imbalance
- Robust preprocessing pipeline preventing data leakage
- Optimal threshold selection for business objectives
- Production-ready deployment with user-friendly interface

Business Value:

- Reduces unplanned downtime through early detection
- Optimizes maintenance resource allocation
- Improves equipment safety and reliability
- Provides data-driven decision support

13. **References**

Dataset: https://www.kaggle.com/datasets/adaziialerite/equipment-data

Libraries and Frameworks:

- scikit-learn: Machine learning library

- imbalanced-learn: Tools for imbalanced datasets
- Streamlit: Web application framework
- Plotly: Interactive visualization library

Techniques:

- Random Forest: Ensemble learning method
- Bootstrap Sampling: Resampling with replacement
- One-Hot Encoding: Categorical variable encoding
- RobustScaler: Scaling resistant to outliers