Oil Pump Predictive Failure App for Oil Recovery and Risk Management

A Technical Report

Isaac Opoku Nkansah

(ION)

2025

Table of Contents

1.0 INTRODUCTION

Traditional maintenance techniques though proactive (Mohapatra, 2024), often lead to unplanned downtime and safety hazards (Achouch et al., 2022, Akindote, 2023). This necessitates a modern app which leverages data science to anticipate failures, optimize operations, and enhance safety (Saxena, 2025). This technical report details the design, development, and project management of the Predictive Failure Dashboard for Oil Pumps.

2.0 PRODUCT DESIGN

The product is a powerful user-friendly meeting the specific needs of the oil and gas maintenance domain, ensuring alignment between data capabilities and end-user requirements.

2.1 Data Source and Theme Selection and Specification

The used data's features directly align with parameters commonly monitored in pump systems relevant for predictive maintenance (Saxena, 2025; Elturki & Imqam, 2020).

- ***Theme***: Predictive Maintenance for Industrial Oil Pumps.
- ***Data Source***: Predictive Maintenance Dataset (AI4I 2020) (Kaggle, 2020).
- ***Specification***: The dataset includes features such as `Air temperature [K]`, `Process temperature [K]`, `Rotational speed [rpm]`, `Torque [Nm]`, `Tool wear [min]`, etc., along with product identifiers (`UDI`, `Product ID`, `Type`).

2.2 Application Domain/End User's Requirements Analysis

The target application domain is operational maintenance within the oil and gas sector.

- *End Users*: Users possessing domain expertise but have limited or no data science knowledge.
- *Requirements Analysis*: Based on the literature (Achouch et al., 2022; Saxena, 2025) and the target user profile, the following needs were identified and incorporated:
    - *Ease of Use*: A simple, intuitive interface requiring minimal training was used.
    - *Plausible Predictions*: Clear indication of potential failure for specific equipment (single prediction) and analysis of historical trends (batch prediction).

- *Contextual Understanding*: Visualization of both input data characteristics and prediction results.
- *Reporting*: Ability to export analysis results for documentation and communication (PDF report).
- *Support & Feedback*: Accessible support (Chatbot) and a mechanism for user feedback.
- *Customization*: Ability to tailor visual presentation (color palettes).
- *Proactive Maintenance Enablement*: A tool that supports the shift from reactive or preventive to predictive maintenance strategies.

2.3 Product Functional and Non-Functional Requirements Specifications

Based on the requirements analysis, the following specifications were defined in the design:

Requirement ID	Description
Functional	
FR1	Accept user input for five key parameters for single prediction.
FR2	Display a clear binary prediction result Failure or Normal.
FR3	Allow upload of CSV files containing batch operational data.
FR4	Validate uploaded CSV for required columns.
FR5	Apply the pre-trained machine learning model to predict failure for each row in the batch data.
FR6	Display batch data with 'Predicted Failure' column in a tabular format.
FR7	Provide visualizations of the input batch data
FR8	Provide visualizations of the batch prediction analysis with solutions
FR9	Generate a downloadable PDF report summarizing results and visualizations.

FR10	Embed an external form for user feedback submission.
FR11	Integrate an external coded trained AI support agent accessible via a lightbox.
FR12	Allow user customization of plot palettes and line colours via sidebar controls.
	Non-Functional
NFR1	Intuitive layout, clear labels, minimal clicks for core tasks.
NFR2	Predictions and visualizations renders within seconds for typical inputs/uploads with fair time lags.
NFR3	The dashboard must operate stably; predictions are based on the pre-trained model's accuracy.
NFR4	Should handle CSV files of reasonable size like tens of thousands of rows without significant slowdown.
NFR5	Code structured logically in Python using standard libraries.
NFR6	Relies on encrypted external services app; user data uploaded is processed in-session.

Table 1: Functional and Non-functional Requirement Analysis

2.4 Product Software Architecture Design

The dashboard I used employs a simple, effective web-based architecture leveraging the Streamlit framework, suitable for rapid development of data-centric applications.

- *Architecture Style:* Monolithic web application.
- *Core Components*:
    - Frontend (UI Layer): Built entirely using Streamlit widgets via `st.pyplot`, embedded HTML/iframes via `st.markdown` and `st.components.v1`.
    - Backend Logic (Application Layer): Python scripts executed by Streamlit. Includes:
        - Data loading and preprocessing (Pandas, Scikit-learn `StandardScaler`).

        - Model loading and prediction (Joblib, Scikit-learn model `predict` method).
        - Visualization generation (Matplotlib, Seaborn).
        - PDF report generation (FPDF).
        - Handling user inputs and application flow.
    - Machine Learning Model: A pre-trained classification model using Random Forest Classifier and saved as `trained_model.joblib`.
    - Data Storage:
        - Initial Dataset (`ai4i2020.csv`): Local file used for scaler fitting and implicit model training.
        - User Uploads: Handled in-memory during a user session via Streamlit's `UploadedFile` object. No persistent storage by the application itself.
    - External Services:
        - Jotform: Used for embedding the user feedback form and the AI support agent via iframe/lightbox mechanism.

2.5 Product Use Case Specifications

Use Case	User	Flow
Predict Single Pump Failure and suggest solutions when failure is predicted *(Appendix F)*	Any user or where applicable, Maintenance Engineer/Operator	1. User navigates to the dashboard. 2. User enters values for Air Temp, Process Temp, Speed, Torque, Tool Wear. 3. User clicks "Predict Failure". 4. System processes input, applies model. 5. System displays "Warning: FAIL" or "Operating normally".
Analyze Batch Pump Data *(Appendix G, H)*	Any user or where applicable, Reliability Analyst/Manager	1. User navigates to the dashboard. 2. User clicks "Choose a CSV file" and uploads a valid CSV. 3. User optionally expands "Visualize Uploaded Input Data" to view input characteristics.

		4. System processes data, performs predictions. 5. System displays results table with 'Predicted Failure'. 6. System displays post-prediction visualizations. 7. User optionally expands "Generate Downloadable Report" and clicks button to download PDF.
Customize Colors *(Appendix A, K)*	Any User	1. User interacts with palette controls in the sidebar. 2. If batch data is loaded, relevant plots in the main area update with selected colours. 3. Generated PDF report will use the selected colours.(Appendix J)
Access Chatbot Support *(Appendix E)*	Any User	1. User clicks the "Isaac: Prediction Support Agent" button. 2. A lightbox opens with the Jotform Chatbot interface. 3. User interacts with the agent.
Submit Feedback *(Appendix J)*	Any User	1. User scrolls to the "*Leave Your Feedback*" section. 2. User fills in the embedded Jotform fields. 3. User clicks the submit button within the form.

Table 2: Product Use Case Specifications

3.0 PRODUCT DEVELOPMENT

The development phase focused on translating the design into a functional application prioritizing user experience and rapid iteration.

3.1 Selection of Software Tools/Platforms and Hardware Methodologies

- Software Tools:
    - *Python*: Chosen as the primary language due to its extensive ecosystem for data science (Pandas, NumPy, Scikit-learn) and web development (Streamlit).
    - *Streamlit*: Selected as the web application framework for its speed and ease in creating interactive data applications.
    - *Jotform*: Utilized as an external platform for embedding the Chatbot (via lightbox) and the feedback form (via iframe), reducing development time for these features.
- Platform/Hardware Methodology:
    - *Development*: Standard desktop environment hosted on Streamlit web app.
    - *Deployment Target*: Web-based application accessible via a browser maximizing accessibility for users.

3.2 Product Development Software Engineering Methodology

An Agile/Iterative methodology with elements of Rapid Prototyping was implicitly followed during development.

- *Justification*: This approach is well-suited for data science projects where requirements can evolve, and user feedback is valuable.
- *Process*: This is perfectly captured in the figure below

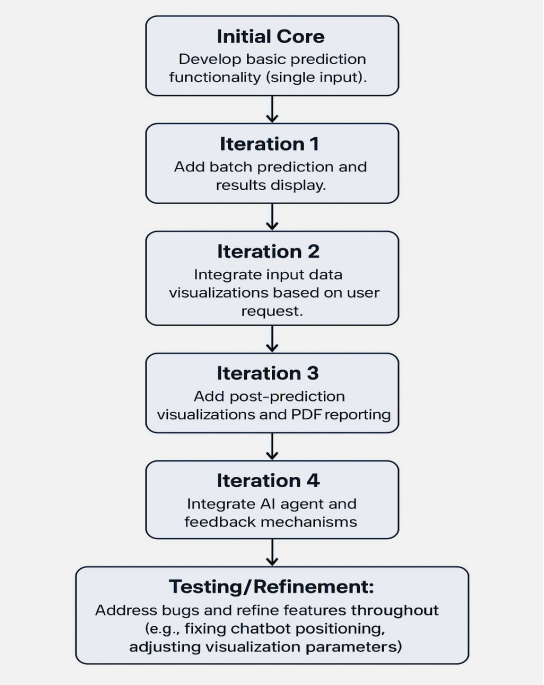


Figure 1: Agile Product Development Flow

- *Contrast*: This contrasts with a traditional Waterfall model, which would require defining all requirements upfront.

3.3 System Testing Method

Testing was conducted throughout the iterative development process, focusing on functionality and usability:

- *Manual Functional Testing or User Acceptance Testing (UAT)*

The task involved running the Streamlit application, testing its features.

- *Error Handling Testing*

Specifically, tested scenarios like file not found errors for model/data, incorrect CSV formats, and potential issues during plot generation or prediction.

- *Integration Testing*

Implicitly tested by verifying the end-to-end workflows to prevent data quality challenges (Dahmani, 2024); including, CSV upload -> input viz -> prediction -> output viz -> PDF report.

- *Future Improvement Unit Testing*

Formal unit tests using frameworks like `pytest` were added to test individual functions like data scaling and PDF generation logic in isolation.

3.4 User Evaluation Plan and Methods

Formal user evaluation is relevant for post-development. These are the propositions I give:

- ***Target Users***: Recruit 3-5 representatives from the target user groups (Maintenance Managers, Reliability Engineers).
- ***Evaluation Methods***
    - *Task-Based Observation*: Assign users' specific tasks with prompts like "Predict the status of a pump with these parameters". This is done and time records and any difficulties encountered recorded.
    - *Heuristic Evaluation*: Assess the dashboard against established usability heuristics to identify potential design flaws.
    - *Feedback Form (Jotform):* Utilize the already embedded feedback form to collect structured feedback on ease of use, clarity of results, usefulness of visualizations, and satisfaction with customization.
    - *System Usability Scale (SUS)*: A standardized questionnaire to get a quantitative measure of perceived usability.

- ***Plan***:
    1. *Preparation:* Define tasks, prepare evaluation materials, recruit users.
    2. *Execution:* Conduct individual evaluation sessions, observing users and collecting data. Ensure users complete the feedback form.
    3. *Analysis:* Analyze observation notes, task completion rates, and survey responses to identify usability issues and areas for improvement.
    4. *Iteration:* Prioritize feedback and plan necessary revisions to the dashboard design and functionality.

4.0 PROJECT MANAGEMENT

I used a timed project management that ensures the delivery of a quality product while addressing potential risks.

4.1 Time Management with Gantt Chart

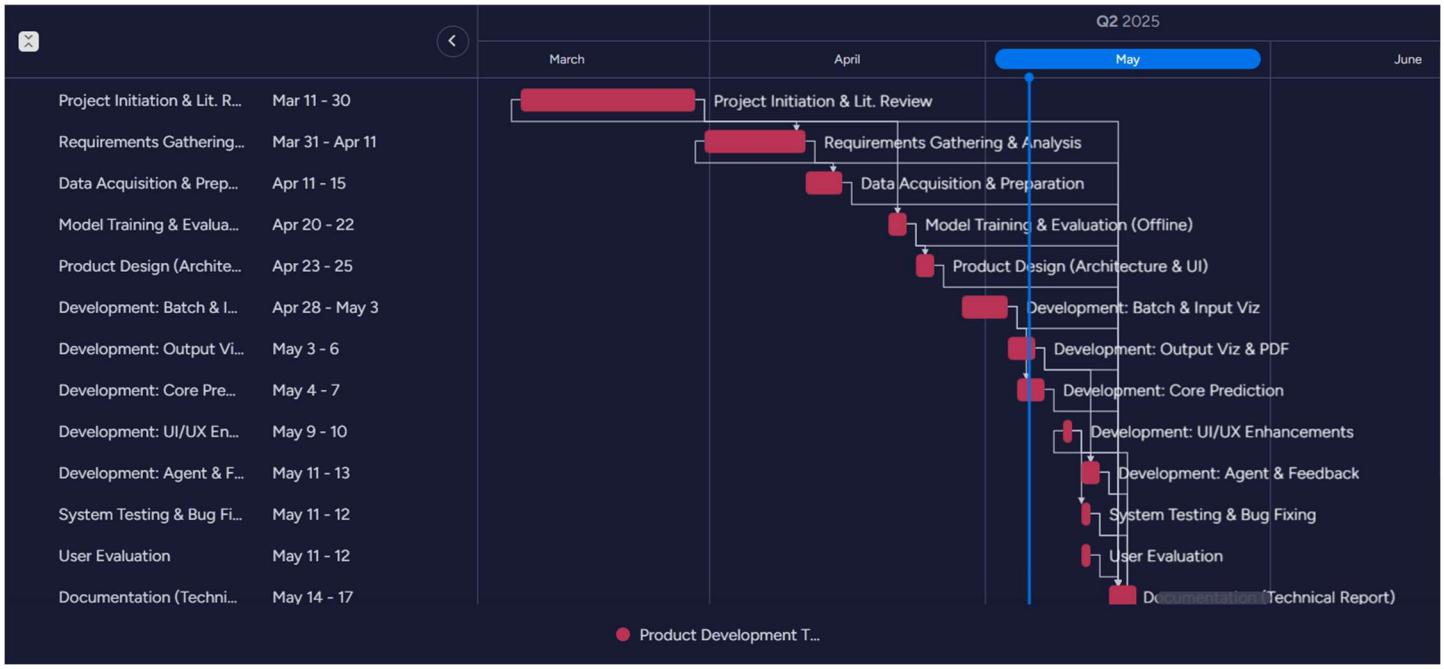


Figure 2: Gantt chart representation of Product Development Time Allocations

4.2 Risk Assessment on Personal Information Protection and Data Security

Managing data responsibly is paramount. Here are tailored strategies for probable risks.

Identified Risk	Possible Impact Level	Mitigation Strategy
Sensitive Operational Data Upload	High	The application uploads data to in-session memory, doesn't store it persistently, and is recommended to be run within a secure network or deployed securely.
Feedback/Agent	Medium	The approach relies on Jotform's security and privacy policies, and disclosure of third-party reliance in a privacy notice.
Data Transmission Insecurity	Medium	My recommendation is to use secure deployment practices (HTTPS).
Unauthorized Access	High	I recommend implementing user authentication, authorization, and secure deployment practices (HTTPS).
Model Bias:	Medium	Periodic retraining with client-specific data with ongoing monitoring for performance disparities.
Compliance Violations	High	Add a privacy notice for Jotform integration and consult legal advice for compliance requirements based on deployment region and data types.

4.3 Quality Control on Software Development

Maintaining software quality throughout the development lifecycle was approached through several practices:

- *Version Control*: Utilising Git for tracking changes, branching for features, and managing code versions is standard best practice.
- *Coding Standards*: Adhering to Python best practices like PEP 8 for readability and maintainability.

- *Iterative Testing*: Incorporating manual functional testing and error checking at each development iteration.
- *Code Structure:* Organizing the Python code logically as seen in processes for separating functions like `set_background`, `generate_pdf_report`.
- *Dependency Management (Appendix A)*: Using standard libraries to ensure compatibility and ease of deployment managed via Python environment.
- *Documentation*: Creating this technical report serves as key project documentation as well as the Inline code comments and Readme.

4.4 Basic Customer/User Relationship Management (CRM)

Mechanisms were integrated to manage user interaction and feedback:

- *Feedback Collection*: The securely embedded Jotform allows users to submit feedback directly to identify and gather improvement suggestions.
- *User Support*:
    - *Primary:* The Jotform Chatbot provides immediate, automated support for queries regarding dashboard usage.
    - *Secondary:* Contact details are provided in the sidebar for direct inquiries.
- *Communication*: The sidebar links provide channels for announcing updates or new features.

4.5 Basic Product Marketing Strategy

Targeting the niche but critical oil and gas maintenance sector requires a focused strategy:

- ***Target Audience***: Maintenance Managers, Reliability Engineers, Operations Supervisors in Oil & Gas companies.
- ***Value Proposition***: An intuitive, easy-to-use dashboard reducing costly downtime and enabling data-driven maintenance decisions for non-data scientists.
- ***Key Features to Highlight***: Single & Batch Prediction, Visualizations, Recommendations, PDF Reports, AI Support, Customization, User-Friendliness.
- ***Marketing Channels***: Utilize digital marketing, direct outreach, and content marketing to promote predictive maintenance.
- *Initial Target Market*: Focus on building relationships with a few early adopters or pilot customers to gather testimonials.

5.0 CONCLUSION

The product successfully demonstrates data science principles application to address challenges in the oil and gas industry. Future work could integrate real-time data streams via IoT platforms for undisturbed data transmissions (Aendikov & Azayeva, 2024) and enhancing drilling operations (Zulfaqar et al., 2024). Continued development, guided by user feedback and emerging technological trends (Nayak et al., 2022), further enhances the dashboard's capabilities.

6.0 REFERENCES

Achouch, M. *et al.* (2022) 'On Predictive Maintenance in Industry 4.0: Overview, models, and challenges,' *Applied Sciences*, 12(16), p. 8081. https://doi.org/10.3390/app12168081.

Aendikov, N. and Azayeva, A. (2024) 'Integration of GIS and machine learning analytics into Streamlit application,' *Procedia Computer Science*, 231, pp. 691–696. https://doi.org/10.1016/j.procs.2023.12.160.

Akindote, O.J., Egieya, Z.E., Ewuga, S.K., Omotosho, A., & Adegbite, A.O. (2023). A review of data-driven business optimization strategies in the US economy. *International Journal of Management & Entrepreneurship Research*, 5(12), 1124-1138.

Dahmani, S. (2024) 'Computational intelligence for green cloud computing and digital waste management,' in *Advances in computational intelligence and robotics book series*, pp. 248–266. https://doi.org/10.4018/979-8-3693-1552-1.ch013.

Elturki, M., & Imqam, A. (2020, June). Application of Enhanced Oil Recovery Methods In Unconventional Reservoirs: A Review And Data Analysis. In *ARMA US Rock Mechanics/Geomechanics Symposium* (pp. ARMA-2020). ARMA.

Mohapatra, A. (2024) 'Generative AI for predictive maintenance: Predicting equipment failures and optimizing maintenance schedules using AI,' *International Journal of Scientific Research and Management (IJSRM)*, 12(11), pp. 1648–1672. https://doi.org/10.18535/ijsrm/v12i11.ec03.

Nayak, S. *et al.* (2022) 'A review on edge analytics: Issues, challenges, opportunities, promises, future directions, and applications,' *Digital Communications and Networks*, 10(3), pp. 783–804. https://doi.org/10.1016/j.dcan.2022.10.016.

Kaggle, *Predictive Maintenance Dataset (AI4I 2020)* (2022). Retrieved from https://www.kaggle.com/datasets/stephanmatzka/predictive-maintenance-dataset-ai4i-2020.

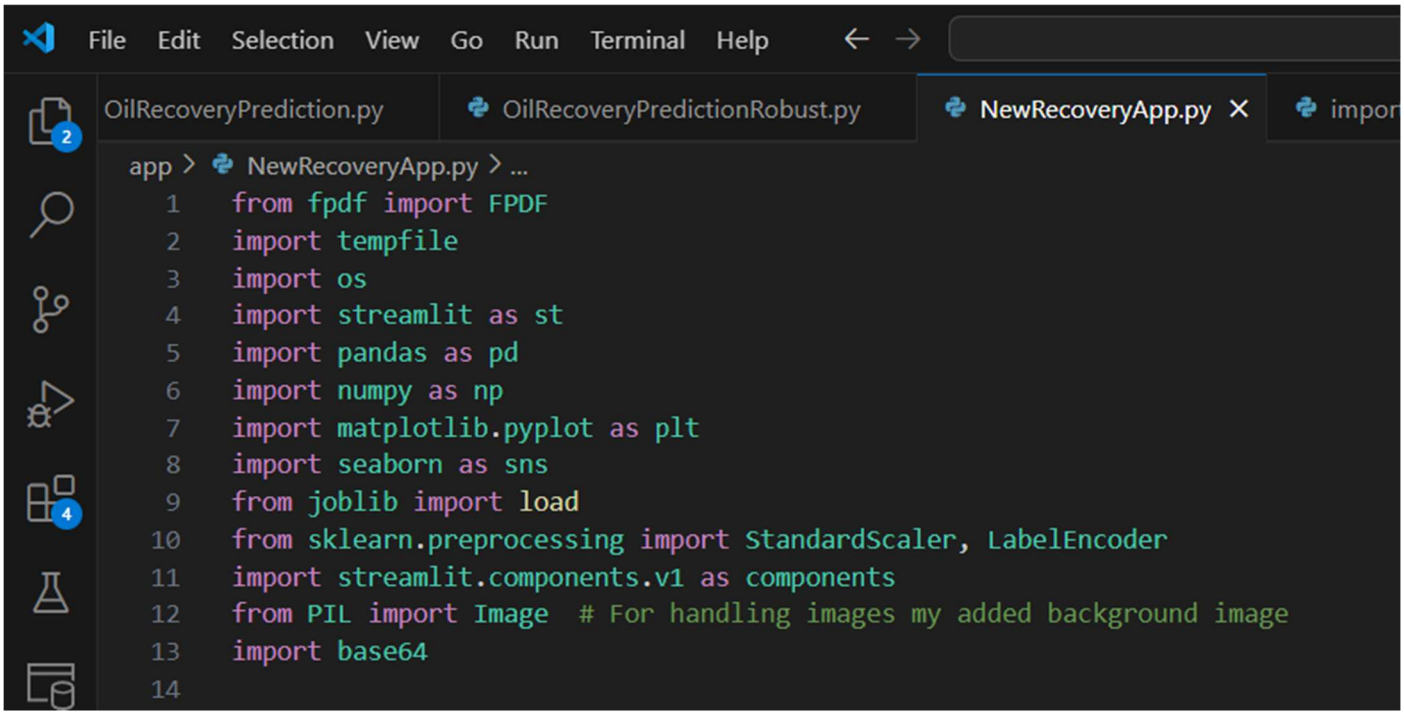
Saxena, A. (2025) 'Application of data science in pump maintenance,' *International Journal for Multidisciplinary Research*, 7(1). https://doi.org/10.36948/ijfmr.2025.v07i01.37055.

Zulfaqar, S.A., Alias, N.E., Yusop, Z., Chow, M.F., Muhammad, M.K.I., Mazilamani, L.S., Ramli, M.W.A., Shiru, M.S., Mohamad, N.A., Rohmat, F.I.W., & Khambali, M.H.M. (2024). Spatiotemporal assessment of rainfall and drought projection for integrated

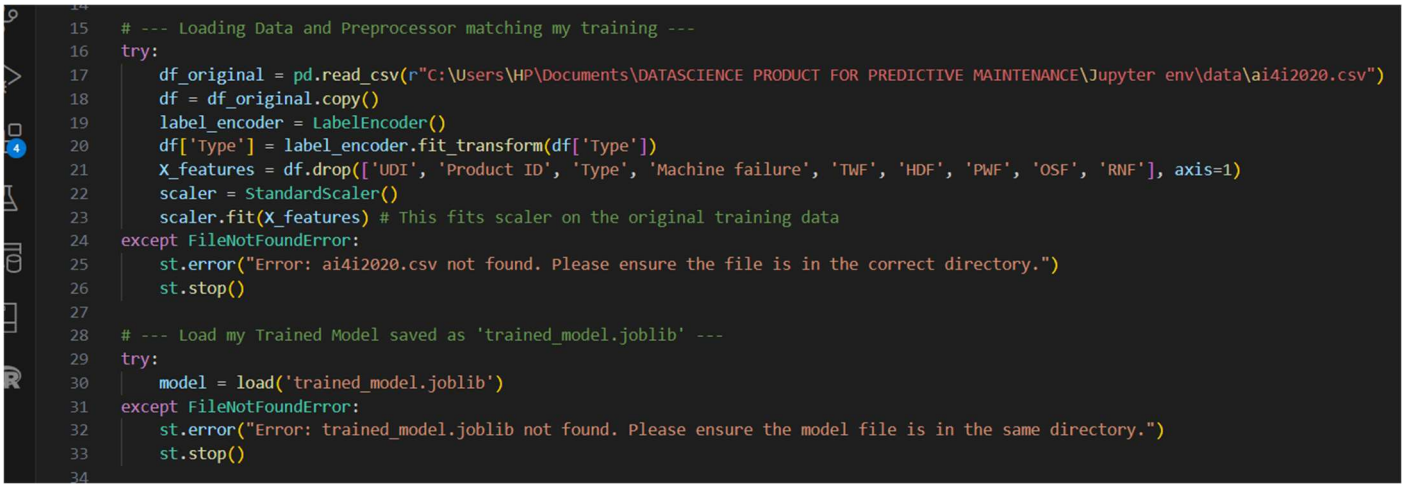
dam management in Benut River Basin, Malaysia under CMIP6 Scenarios. *Environmental Challenges*, 100892.

7.0 APPENDICES

Appendix A—Environment and Product Code Dependencies



Appendix B—Loading Data and Incorporating Trained Model

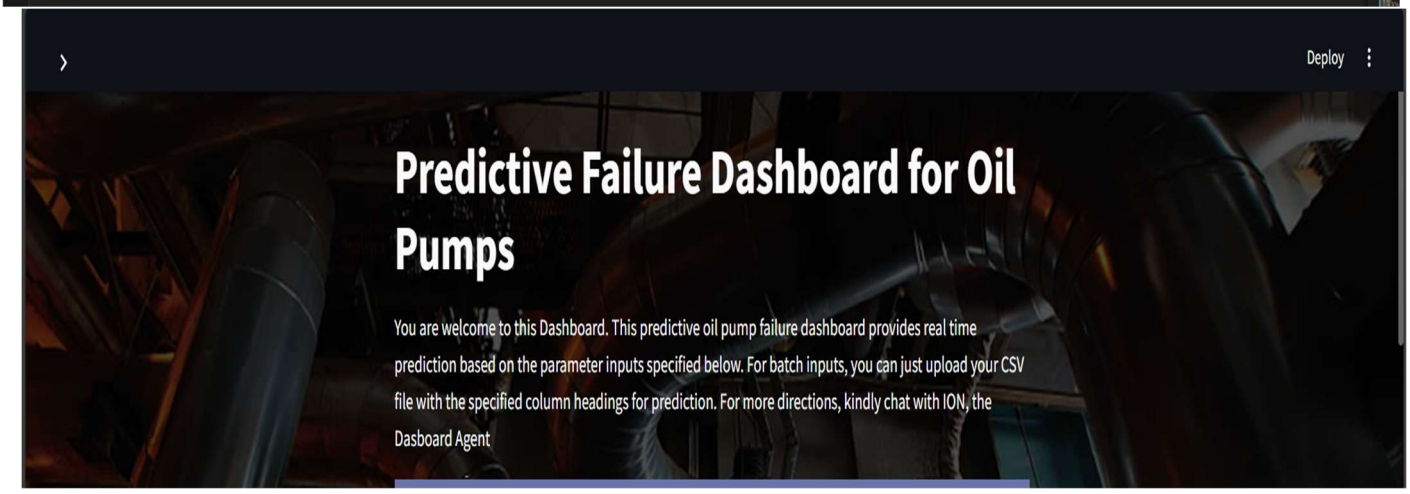


Appendix C—Sidebar Customisations and Effect Controls for Plots

```python
# --- My preferred Sidebar Styling ---
st.sidebar.title("IONARTS Projects Consult")
st.sidebar.markdown("**✨ Dream it..We Deliver it ✨**")
st.sidebar.markdown("Designed by **bi95cz**. Providing cutting-edge Data Science solutions for analytics and risk management in the oil and g
st.sidebar.subheader("Contact")
# Using Font Awesome icons requires internet access for the user's browser
st.sidebar.markdown("<link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css'>", unsafe_
st.sidebar.markdown("<i class='fas fa-envelope'></i> Email: isaac3g@outlook.com", unsafe_allow_html=True)
st.sidebar.markdown("<i class='fas fa-phone'></i> Phone: +447392615042", unsafe_allow_html=True)
st.sidebar.subheader("Links")
st.sidebar.markdown("[<i class='fab fa-github'></i> Our Blog](https://github.com/Ionkansah)", unsafe_allow_html=True)
st.sidebar.markdown("[<i class='fas fa-globe'></i> Company Website](https://github.com/Ionkansah)", unsafe_allow_html=True)


# --- ** Added sidebar Visualization Customization** ---
st.sidebar.subheader("🎨 Visualization Customization")

# Define available palettes
qualitative_palettes = ['Set1', 'Set2', 'Set3', 'Pastel1', 'Pastel2', 'Paired', 'Accent', 'Dark2', 'tab10', 'tab20', 'tab20b', 'tab20c']
sequential_palettes = ['coolwarm', 'viridis', 'plasma', 'inferno', 'magma', 'cividis', 'Greys', 'Purples', 'Blues', 'Greens', 'Oranges', 'Red
diverging_palettes = ['coolwarm', 'bwr', 'seismic', 'Spectral', 'RdBu', 'RdGy', 'PiYG', 'PRGn', 'BrBG', 'PuOr']

# Selectbox for plots using categorical palettes (hist with hue, boxplot)
selected_palette_categorical = st.sidebar.selectbox(
    "Categorical Palette (Histograms/Boxplots):",
    qualitative_palettes + sequential_palettes, # Offer sequential too just in case
    index=qualitative_palettes.index('Set1') # Default to Set1
)
```

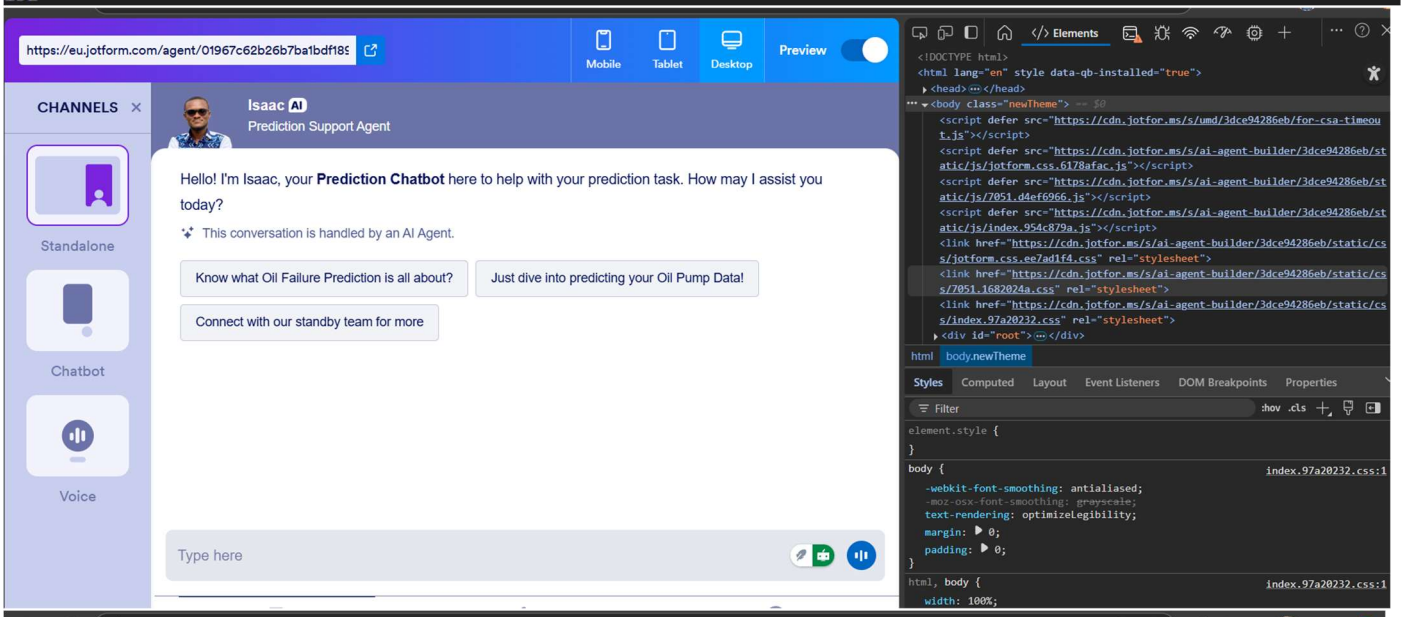
Appendix D—Code and Result for Setting Background Image

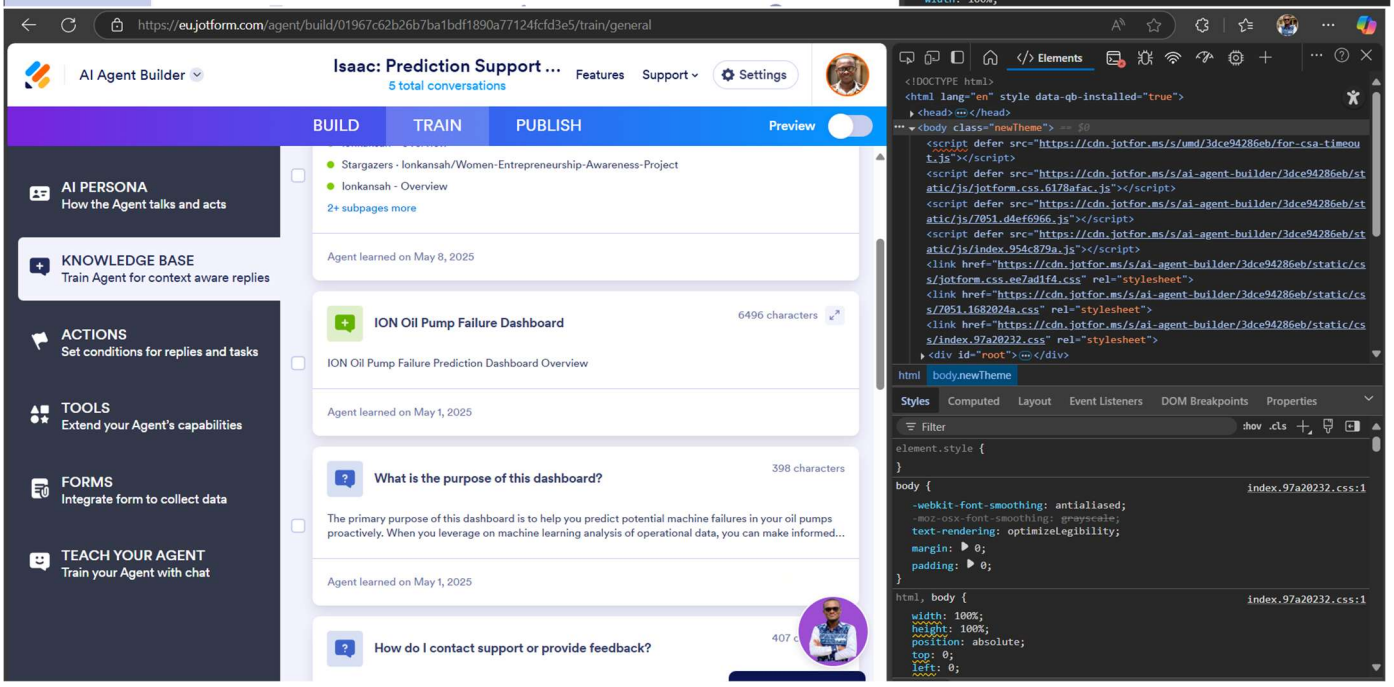
```python
# --- Set background image ---
try:
    image_path = r"C:\Users\HP\Documents\DATASCIENCE PRODUCT FOR PREDICTIVE MAINTENANCE\Jupyter env\data\Pipes7.jpeg"
    image = Image.open(image_path)
    # target_width = 300 # Adjust this width value
    # aspect_ratio = image.height / image.width
    # target_height = int(target_width * aspect_ratio)
    # image = image.resize((target_width, target_height)) # Optional resize
    resized_image_path = 'pump_network_background_resized.png' # Use a consistent name
    image.save(resized_image_path) # Save the original or resized image
    set_background(resized_image_path)
except FileNotFoundError:
    st.warning("Warning: Background image not found. Background will be default.")
except Exception as e:
    st.error(f"Error loading or setting background image: {e}")

# --- Main Dashboard ---
st.title("Predictive Failure Dashboard for Oil Pumps")
st.markdown("You are welcome to this Dashboard. This predictive oil pump failure dashboard provides real time prediction based on the paramet

```



Appendix E—Embed Code and result for Externally Trained Chatbot (Isaac Ai)

```
# --- AI Agent Integration (Lightbox Button) ---
jotform_lightbox_html = """
<iframe id="JotFormIFrame-01967c62b26b7ba1bdf1890a77124fcfd3e5"
  title="Isaac: Prediction Support Agent" onload="window.parent.scrollTo(0,0)"
  allowtransparency="true" allow="geolocation; microphone; camera; fullscreen"
  src="https://eu.jotform.com/agent/01967c62b26b7ba1bdf1890a77124fcfd3e5?embedMode=iframe&background=1&shadow=1"
  frameborder="0" style="
    min-width:100%;
    max-width:100%;
    height:600px;
    border:none;
    width:40%;
  " scrolling="no">
</iframe>
<script src='https://cdn.jotfor.ms/s/umd/latest/for-form-embed-handler.js'></script>
<script>
  window.jotformEmbedHandler("iframe[id='JotFormIFrame-01967c62b26b7ba1bdf1890a77124fcfd3e5']",
    "https://eu.jotform.com")
</script>
"""

st.markdown(jotform_lightbox_html, unsafe_allow_html=True)
```



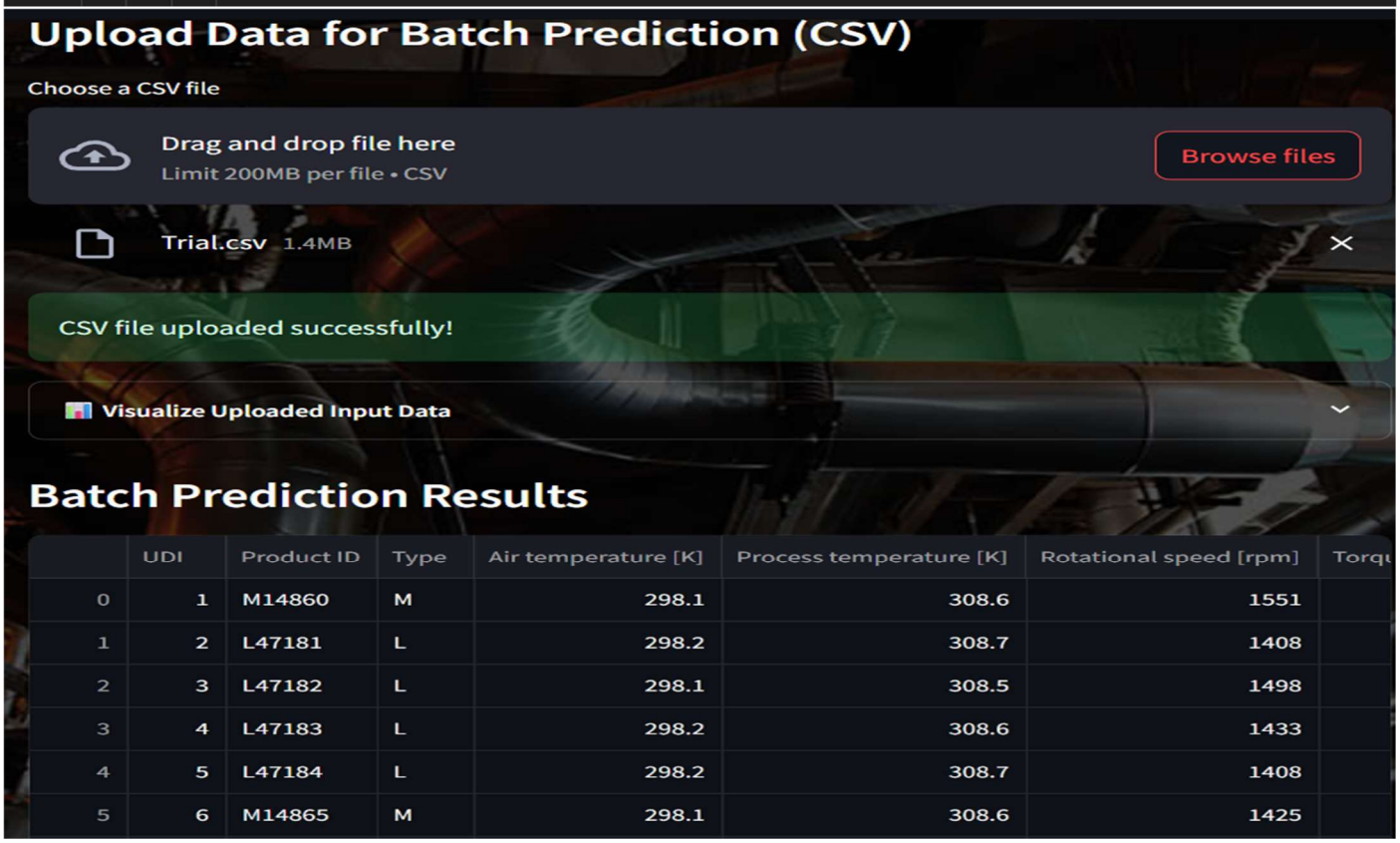


Appendix F—Code for Initial Input Prediction and Result

```python
# --- Prediction Section ---
st.header("Predict Machine Failure")
st.subheader("Enter Pump Parameters")

col1, col2, col3 = st.columns(3)
air_temperature = col1.number_input("Air Temperature [K]", value=298.0)
process_temperature = col2.number_input("Process Temperature [K]", value=308.0)
rotational_speed = col3.number_input("Rotational Speed [rpm]", value=1500.0)

col4, col5 = st.columns(2)
torque = col4.number_input("Torque [Nm]", value=40.0)
tool_wear = col5.number_input("Tool Wear [min]", value=10.0)

predict_button = st.button("Predict Failure")

if predict_button:
    input_data = pd.DataFrame({
        'Air temperature [K]': [air_temperature],
        'Process temperature [K]': [process_temperature],
        'Rotational speed [rpm]': [rotational_speed],
        'Torque [Nm]': [torque],
        'Tool wear [min]': [tool_wear],
        'TWF': [0], 'HDF': [0], 'PWF': [0], 'OSF': [0], 'RNF': [0]
    })
    numerical_features = ['Air temperature [K]', 'Process temperature [K]',
                          'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]']
    binary_features = ['TWF', 'HDF', 'PWF', 'OSF', 'RNF']
    scaled_numerical = scaler.transform(input_data[numerical_features])
    final_input = np.concatenate([scaled_numerical, input_data[binary_features].values], axis=1)
    if final_input.shape[1] == model.n_features_in_:
        prediction = model.predict(final_input)[0]
        st.subheader("Prediction Result")
        if prediction == 1: st.error("⚠️ Warning: Machine is predicted to FAIL!")
        else: st.success("✅ Machine is predicted to be operating normally. Please maintain similar conditions")
    else: st.error(f"Error: Input data shape mismatch.")
```

Appendix G—Code and Result for Batch Failure/Normal Prediction

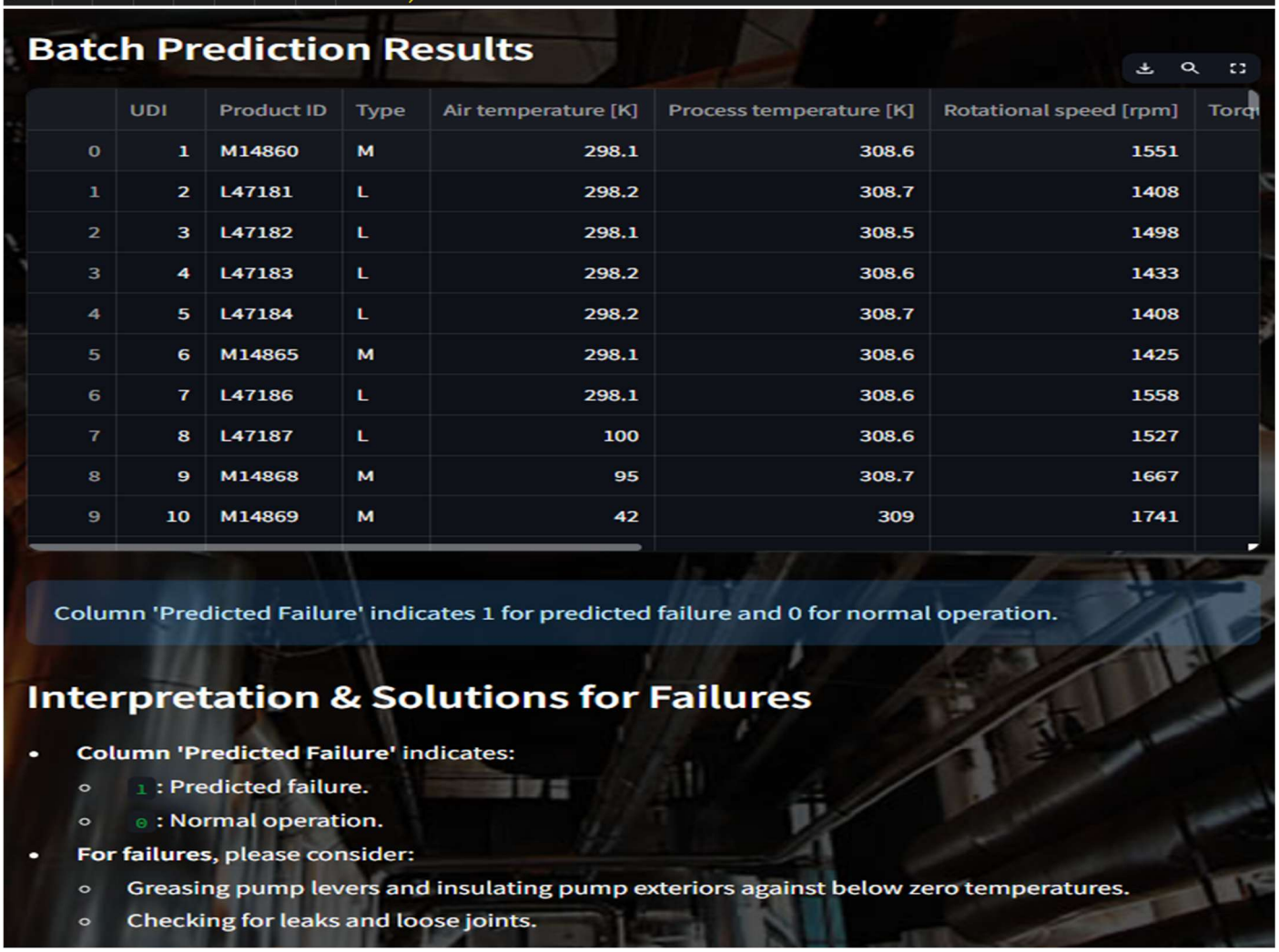
```python
# --- ION(bi95cz) Poduct for Batch Prediction Section ---
st.subheader("Upload Data for Batch Prediction (CSV)")
uploaded_file = st.file_uploader("Choose a CSV file", type="csv")

batch_data_display = None # Initialize

if uploaded_file is not None:
    try:
        batch_data_input = pd.read_csv(uploaded_file)
        st.success("CSV file uploaded successfully!")

        # --- Input Data Visualization Section ---
        with st.expander("📊 **Visualize Uploaded Input Data**"):
            st.markdown("Explore the characteristics of the data you uploaded.")
            required_viz_cols = ['Product ID', 'Type', 'Air temperature [K]', 'Process temperature [K]',
                                 'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]']
            if all(col in batch_data_input.columns for col in required_viz_cols):
                st.markdown("### Product ID Distribution (Top 20)")
                if batch_data_input['Product ID'].nunique() > 20:
                    st.bar_chart(batch_data_input['Product ID'].value_counts().nlargest(20))
                else:
                    st.bar_chart(batch_data_input['Product ID'].value_counts())
                st.markdown("### Product Type Distribution")
                st.bar_chart(batch_data_input['Type'].value_counts())
                st.markdown("### Numerical Feature Trends (vs. Record Index)")
                numerical_to_plot = ['Air temperature [K]', 'Process temperature [K]',
                                     'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]']
                selected_num_feature = st.selectbox("Select numerical feature to plot:", numerical_to_plot)
                fig_line, ax_line = plt.subplots(figsize=(10, 4))
```



	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torqu
0	1	M14860	M	298.1	308.6	1551	
1	2	L47181	L	298.2	308.7	1408	
2	3	L47182	L	298.1	308.5	1498	
3	4	L47183	L	298.2	308.6	1433	
4	5	L47184	L	298.2	308.7	1408	
5	6	M14865	M	298.1	308.6	1425	

Appendix H—Code and Result for Prediction Logic and Output and Solution Recommended

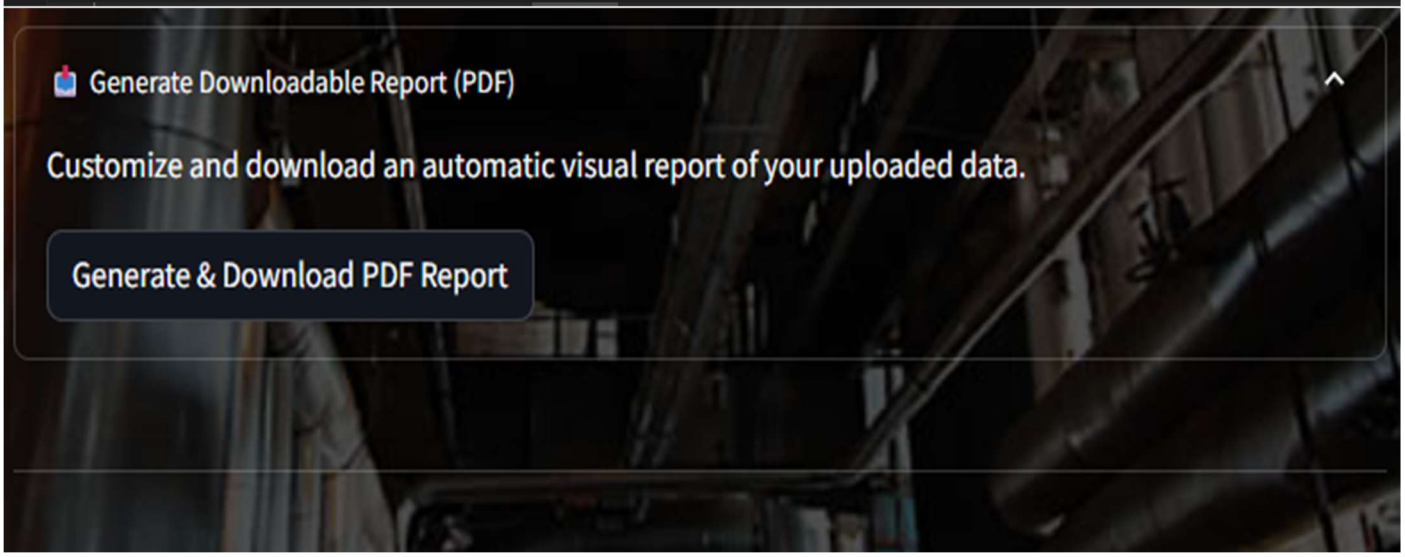
```
        # --- Prediction Logic and Output ---
        required_numerical = ['Air temperature [K]', 'Process temperature [K]',
                              'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]']
        required_binary = ['TWF', 'HDF', 'PWF', 'OSF', 'RNF']
        all_required = required_numerical + required_binary
        if all(col in batch_data_input.columns for col in all_required):
            scaled_numerical = scaler.transform(batch_data_input[required_numerical])
            binary_data = batch_data_input[required_binary].values
            final_batch_input = np.concatenate([scaled_numerical, binary_data], axis=1)
            if final_batch_input.shape[1] == model.n_features_in_:
                batch_predictions = model.predict(final_batch_input)
                batch_data_display = batch_data_input.copy()
                batch_data_display['Predicted Failure'] = batch_predictions
                st.subheader("Batch Prediction Results")
                st.dataframe(batch_data_display)
                st.info("Column 'Predicted Failure' indicates 1 for predicted failure and 0 for normal operation.")
                st.subheader("Interpretation & Solutions for Failures")
                st.markdown("""
                            - **Column 'Predicted Failure'** indicates:
                              - `1`: Predicted failure.
                              - `0`: Normal operation.
                            - **For failures**, please consider:
                              - Greasing pump levers and insulating pump exteriors against below zero temperatures.
                              - Checking for leaks and loose joints.
                              """)
```



	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torq
0	1	M14860	M	298.1	308.6	1551	
1	2	L47181	L	298.2	308.7	1408	
2	3	L47182	L	298.1	308.5	1498	
3	4	L47183	L	298.2	308.6	1433	
4	5	L47184	L	298.2	308.7	1408	
5	6	M14865	M	298.1	308.6	1425	
6	7	L47186	L	298.1	308.6	1558	
7	8	L47187	L	100	308.6	1527	
8	9	M14868	M	95	308.7	1667	
9	10	M14869	M	42	309	1741	

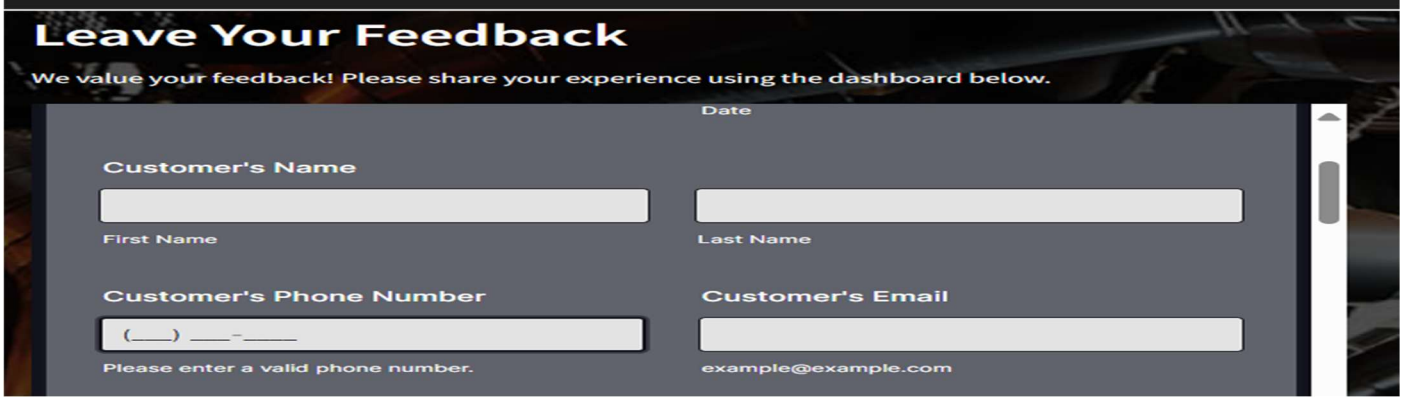
Appendix I—PDF Report Generation

```
def generate_pdf_report(data, failure_counts, selected_feature, correlation_matrix, box_feature,
                        palette_categorical, palette_heatmap): # Pass palettes here
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt="Machine Failure Prediction Report", ln=True, align='C')
    pdf.ln(5)
    pdf.set_font("Arial", size=10)
    pdf.cell(200, 10, txt="Failure Summary:", ln=True)
    for key, value in failure_counts.items():
        label = 'Failure' if key == 1 else 'No Failure'
        pdf.cell(200, 8, txt=f"{label}: {value} instances", ln=True)
    with tempfile.TemporaryDirectory() as tmpdir:
```



Appendix J—Comments and Feedback Section

```
# --- User Review Section (Embedded Jotform) ---
st.markdown("---")
st.header("Leave Your Feedback")
st.markdown("We value your feedback! Please share your experience using the dashboard below.")
jotform_embed_url = "https://form.jotform.com/251204859375361"
components.iframe(jotform_embed_url, height=400, scrolling=True)

# --- Footer ---
st.markdown("---")
st.markdown("© Developed by **bi95cz** with pseudonym IONARTS PROJECTS CONSULT- 2025")
```



Appendix K—Dashboard and Some Visualisations