Assignment – 2

1. Architecture Diagram : Uploaded in Github in image format (Name: Assignment2_Architecture And Data Flow Diagram.jpeg)
2. Data Flow Diagram: Uploaded in Github in image format (Name: Assignment2_Architecture And Data Flow Diagram.jpeg)
3. Implementation Process: Uploaded in Github in image format (Name: Assignment2_Architecture And Data Flow Diagram.jpeg)
4. Performance of the solution: Need to discuss. Since after applying the data into model, actual performance can be noted and depending on that the performance improvement can be decided.
5. Next steps to improve solution is to apply neural networks or complex ensemble models so as to get good values of recalls.
6. Timeline to deliver the project:
   a. The complete data pipeline implementation is expected to be completed in 1 sprint (Considering 2 data engineering resources- 1 senior and 1 junior and 1 sprint = 2 weeks)
   b. The condition monitoring will take a maximum of 2 sprints (considering 2 data science resources with mid-level experience)
   c. The Predictive maintenance will take a maximum of 3 sprints (again considering 2 data scientists with mid-level experiences)
   d. Testing and deployment to dev, then to QA and testing in QA will take 1.5 sprints with 2 testers and 1 MLOps engineer
   e. The final deployment and prod test will take 1 sprint with 1 tester and 1 MLOps member
7. The above high-level tasks (Epic) will further be divided into tasks.

Architecture and Data Flow Diagram of a Motor - Condition Monitoring and Predictive Analysis

Sensor - 1

Sensor - 2

Gateway

PLC

Cloud Storage

Sensor Data → Streaming → Data Ingestion → Data Storage

Store the Flag

Condition Monitoring Results

Condition Monitoring

Code to access data from cloud in local machine

Data Availability Check → Data Preprocessing

Data Analysis

Model to Check if Vibration of DE/NDE > Threshold

Yes   No

Store the Flag = Danger

Store the Flag = No_Danger

Predictive Maintenance

Fetch the flags where Alarm = Danger → Get the respective Vibration Data and other features at the same timestamps → Model Development ← Data Preprocessing

Model Training ← Data Split into Train/Test/Val

Trained Model Evaluation on Train and Val Sets

Store the Scores after inferencing on history data ← Hyper Parameter Tuning

Final Model Evaluation on Test Set

Model Scoring and Saving

Model Deployment
Make Docker Container, Push code in Git, Configure CI/CD, Start Deployment

Deployed Model

Saving the Scores in Cloud DB ← Scoring in Real-Time ←