Group Members:

HAdiya farooq 20I-0579 Zuha Umar 20i-0603 Fatima abdul wahid 20i-0711

MLOPs Project

**MLOPs Project Pipeline**

**Data Acquisition**

Data Acquisition was done through the following steps:

1. A dataset was generated by running the given script. It was a Time Series Dataset that had the following columns: Timestamp, Machine\_ID, Sensor\_ID and Reading.
2. The generated data was stored in a csv file named “dummy\_sensor\_data.csv”.
3. This dataset was also maintained using DVC. For that, we first set up the environment using ‘dvc init’. Then the data was added to dvc tracking using the ‘dvc add’ command. Lastly, the latest data file was pushed to google drive storage.

**Data Pre-Processing**

Data Pre-Processing was done through the following steps:

1. The data previously saved in the csv file had to be pre processed and normalized in order to be fed into the model. For that, we had to perform mean normalization to the ‘Reading’ feature values so that they don’t vary much and are normalized around the mean to make it easy to understand what the reading actually depicts.
2. Remaining features were also pre-processed to convert them into numeric values. The ‘Timestamp’ feature was divided into Day, Hour and Month. In addition to this, the features ‘Machine\_ID’ and ‘Sensor\_ID’ were converted into numeric ids (e.g Machine\_1 was converted into 1 to be a numeric value).
3. The resultant features ( ‘Hour’, ‘Month’, ‘Day’, ‘Machine\_ID’, ‘Sensor\_ID’,’NormalizedReading’) were finalized and stored in another csv file named “preprocesseddata.csv” to save changes made.

**Model Training**

Model Training was done through the following steps:

1. The data was split into a training and testing set using 80/20 ratio. The suggested models i.e Random Forest and Xgboost were selected for training.
2. After splitting the data, we performed hyperparameter tuning using Grid Search CV. After the grid search, it identifies the best performing Random Forest and Xgboost model with optimal hyperparameters.
3. These parameters and metrics are then tracked using the MLflow library.
4. The DVC pipeline was also made in dvc.yaml where all steps of the pipeline were defined and used to train the above-mentioned models and get predictions.
5. The best model is searched among all experiments keeping in mind the metrics calculated (MSE in our case).

**Model Evaluation**

Model Evaluation was done through the following steps:

1. Model Evaluation was done using MLFlow library. The resultant metric (Mean Square Error) for both the models is compared.
2. The model having less Mean Square Error is considered better. The best model out of the two is logged, saved and then registered for the production stage.

**Deployment**

Deployment was done through the following steps:

1. We first trained our models i.e. Random Forest and XGBoost (more on this is explained in the above section).
2. After determining the best model, we created a Flask web application to display our results. The user selects a .csv file containing time stamp, machine ID and sensor ID and our web application displays the results.
3. In order to deploy our web app, we first created a Dockerfile and then used github workflows to build the docker image. After checking if our container was running without any anomalies, we pushed the image on our Docker Hub account.

**Testing**

Model Testing was done through the following steps:

1. The Best Model that was saved and logged earlier was loaded.
2. The new test data that has been pre-processed already was loaded into a dataframe from a csv.
3. The features and the target variable are extracted from the new test data.
4. The loaded model is then used to make predictions on the new test data. The performance is evaluated by calculating Mean Square Error.

**Concept Drift Monitoring**

Concept Drift Monitoring was done through the following steps:

1. During the training of the models, after the evaluation is done. We’ve set a threshold for the metric that we are evaluating our models on i.e Mean Square Error. If the Mean Square Error calculated by either of the models increases the threshold value, it is sent for retraining.
2. The retraining is done the same as the training, but the parameter grid is tweaked a little for the retraining part.