Practicum Final Report:

Equipment Downtime Prediction

CSE 6748, Summer Semester 2021

Student: Homayoun Gerami

Table of Contents:

Introduction

This report covers the details of the project that I was doing during the 2021 summer semester for the fulfillment of the course CSE 6748, Applied Analytics Practicum. The project was titled as “Equipment Downtime Prediction”, and the required resources for the project was provided jointly by AltaML, an incorporation that offers Machine Learning services, and Spartans Control, a company that provides industrial automation process. The data that was used in this project belongs to Spartans Control and it was collected from a plant that produces Potash, located in Alberta, Canada. I was assigned to this project for four months, April-July 2021 and this report is meant to elaborate on my contribution in the project.

Equipment Downtime Prediction

The objective of the project was to predict industrial equipment failures, in other words downtime or shutdown, preferably one hour ahead of time, and ultimately to reduce the number of unexpected shutdowns causing production losses of one of the Spartan Controls’ plant. It was estimated that these unexpected shutdowns result in over forty-eight thousand tons of potash production losses each month, and the ultimate objective was to minimize the waste. Please note that in this report we used ‘shutdown’ and ‘downtime’ interchangeably, and both mean the same here.

Problem Definition

A shut down is a time where production stops within the plant due to any disturbance in the process; overall shutdown may lead to financial and production losses. In the technical terms, client defines shutdown as any event that Coarse Dryer Feed Speed drops to below 10 rpm. Our goal in this project was to predict such event one hour ahead of happening.

How much we can save?

The production loss due to shut down per month was estimated to be around forty-eight thousand tonnes of potash, that was equivalent of 480 blue whales, each whale roughly weighs around 100 tonnes. The benchmark of our project was to use machine learning

and data driven decisions to reduce 30 percent of the downtimes, which leads to 16 thousand tonnes of savings in production per month, or in terms of whales, 160 blue waves that would be saved. Figure 1 illustrate how much potash can be saved if machine learning model can predict 30% of the downtime events.

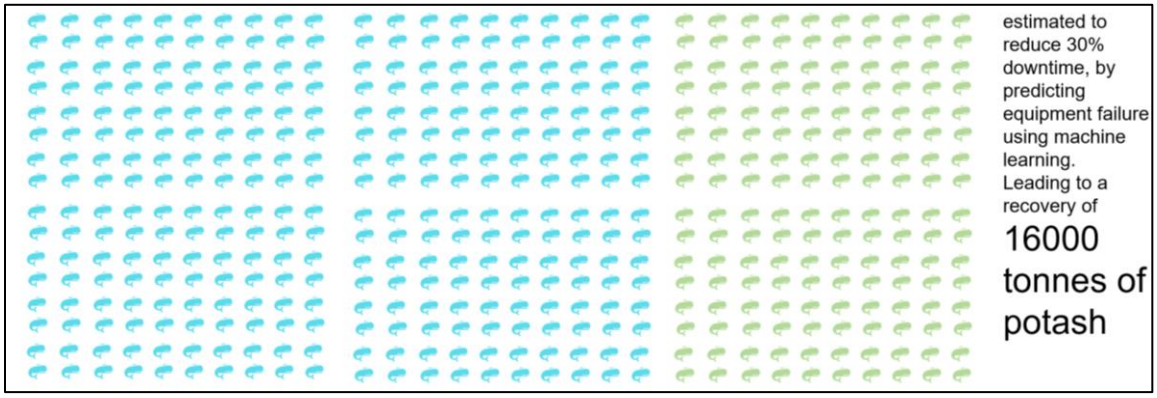

Figure 1, How much potash can be saved if the downtimes drop 30%

If a shutdown would happen at the time equal to T, then we used one hour before as a prediction window to find the likelihood of happening of this outage. Therefore, our goal here was to predict on average, as early as possible within one-hour prediction window to give the operator time to react in case of a shutdown. The prediction window is demonstrated in Figure 2.

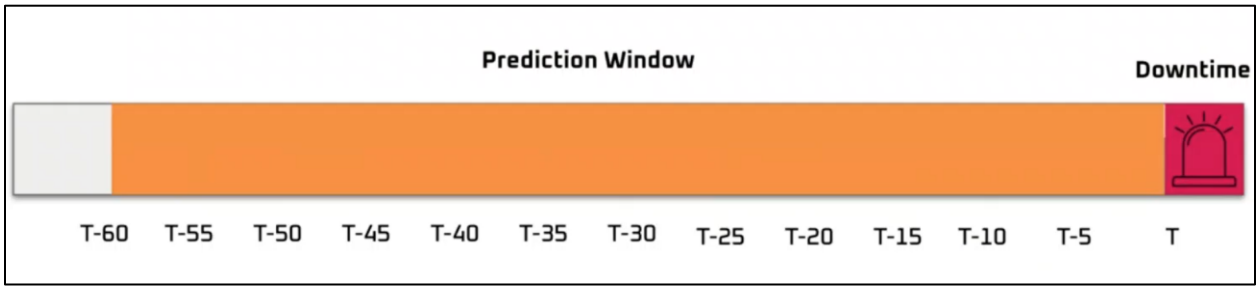

Figure 2, One-hour prediction window

Methodology and Workflow

As illustrated in Figure 3, the below steps were followed in our methodology for this project:

Step-1, Data Acquisition: we collected the data from different sources.

Step-2, Data Processing: we prepared the dataset for data training and performed required data cleaning, label creation and data transformation.

Step-3, Exploratory Data Analysis (EDA): it was the step that we examined and analyzed the relationships within the data, to extract the insights from the dataset.
Step-4, Feature engineering: we manufactured the best and the right features to construct our model in this step.
Step-5, Machine Learning (ML) model development: we established and trained the ML model.
Step-6, Model Evaluation: the model was validated, based on right metrics.
Step-7, Dashboard: we delivered a proof-of-concept model and a dashboard to show how the users may adopt this ML model that we developed within the business process.

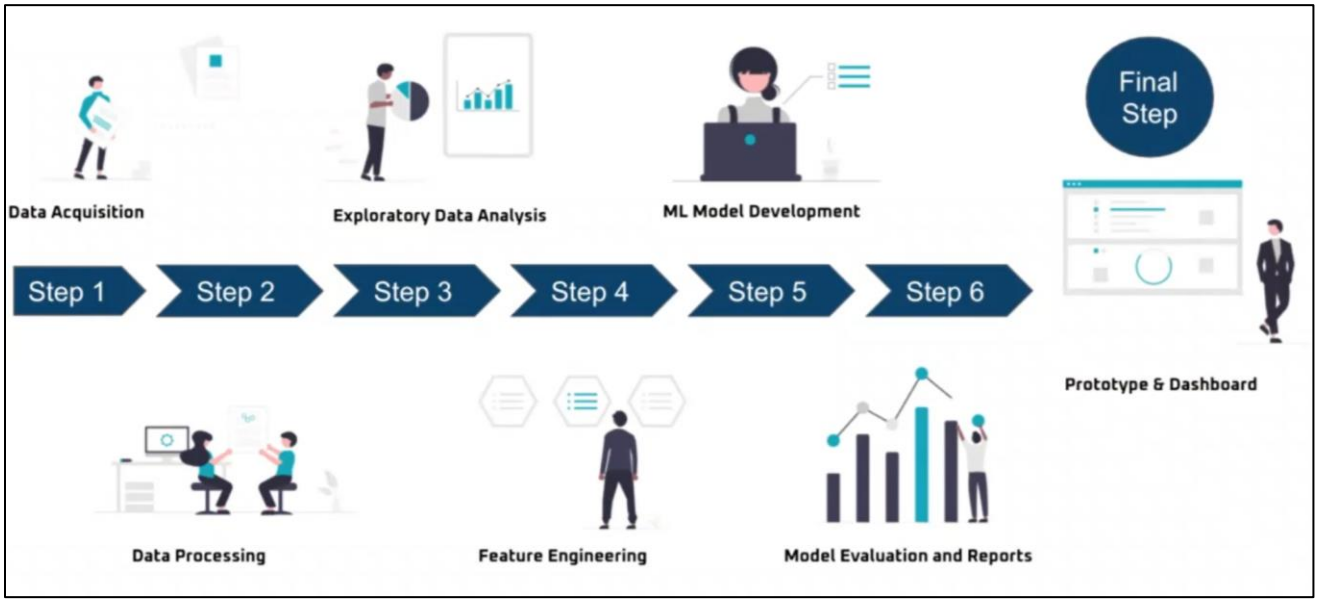

Figure 3, Methodology and workflow in this project

The above steps are elaborated in more details:

Data acquisition:

We have been given 11 tables, time series data with 1 minute of sampling rate, covering time period of 01 Jan 2018 to 31 Dec 2019. There were 95 features associated with these tables under the following categories:

- Unit process instrumentation readings:
  - Drive current, Torque SP and PV, Speed, etc.
- Coarse Dryer parameters:
  - Differential pressure, Speed, Temperature, Bed depth, etc.
  - Set points for the unit processes.

As a glimpse of the dataset, figure 4 shows top three rows of one of the 11 tables, and the figure 5 represents the flow diagram of the plant, with the instrumentation and coarse dryer highlighted.

Timestamp	CENTRFUGE_FEED-TK_L.PV	HYDROFLOAT_PROD-TK_LC.MD_x	HYDROFLOAT_PROD-TK_LC.PV_x	HYDROFLOAT_PROD-TK_LC.SP_x	Centrifuge 1 Torque PV	Centrifuge 1 Torque SP	Centrifuge 1 Drive Current	Centrifuge 1 Drive Speed	Centrifuge 2 Torque PV	..
2018-01-01 00:00:00	45.0336	Auto	29.6249	35	731.651	725	37.6984	56.136	1262.94	..
2018-01-01 00:01:00	45.3468	Auto	29.2124	35	717.425	725	37.6984	56.1638	1261.84	..
2018-01-01 00:02:00	46.2669	Auto	28.7189	35	699.428	725	35.7143	55.312	1293.91	..

Figure 4, Input data top three rows

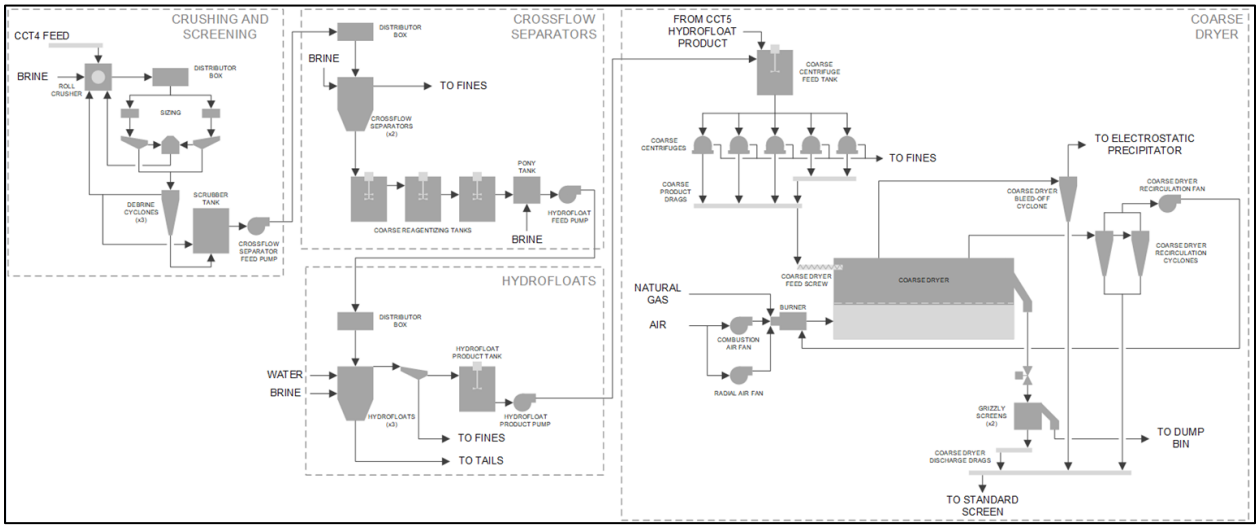

Figure 5, Flow Diagram of the plant, with Coarse Dryer and other instrumentation highlighted

Data Processing:

After we got the raw data from Spartan controls, the Data Processing approach compiles several steps and prepared the final Train and Test Data to build the machine learning model. The major parts of the data processing is further explained:

Data cleaning: The performed data cleaning for this project Included replacing categorical features with encoded ones, converting any object datatypes to float and dropping the columns (features) that the missing values were beyond threshold (5%).

Data Labeling: The original dataset did not have the shutdowns events labeled; it was then necessary to perform such labeling manually. Shutdown was recognized once the coarse dryer speed fall below 10 rpm, and such events were labelled as 1, representing shutdown, and similarly, 0 means no shutdown. 231 shutdown events were identified within the dataset and labeled accordingly. As the goal of the project was to predict a shutdown one hour prior it happens, the 60 minutes prior of each outage event was labeled as shutdown as well. Figure 6 represented how we labeled the shutdown events.

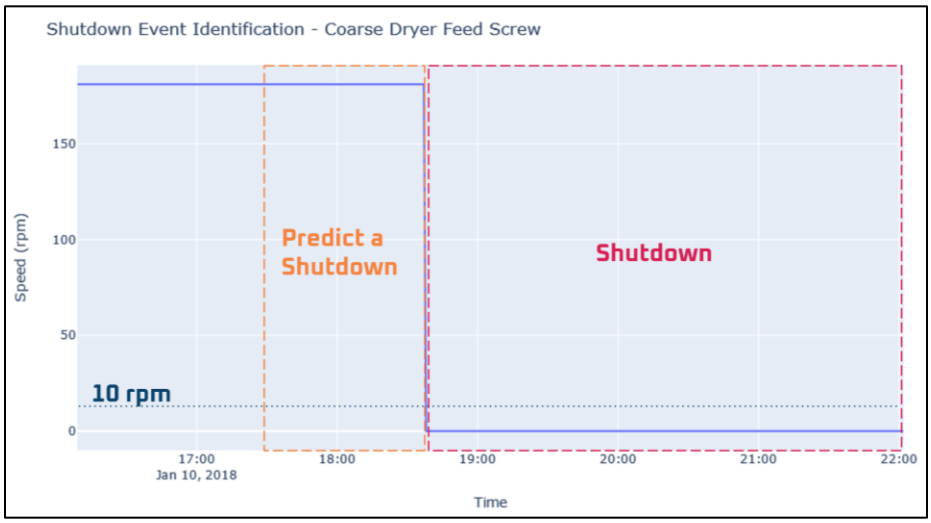

Figure 6, Shutdown event identification and labelling

Exploratory Data Analysis (EDA) and Feature Engineering:

The major highlights among performed EDA tasks were Downtime clustering and correlation analysis:

- **Downtime clustering:** We speculated that the downtimes most probably were in different types, as a few associated features manifested very different behaviors for number of downtimes, and we further investigated that through clustering

techniques and reported our finding to the client. We used the following features for the clustering:

- Rate of Coarse Dryer Pressure, Temperature, Motor Current, and Tanks Levels changes
- Error between Set Point (SP) variables and Processed Variables (PV) one hour ahead of downtime

Figures 7 shows cross-plots of two major principal components, color coded with the associated clusters, derived from Spectral clustering method.

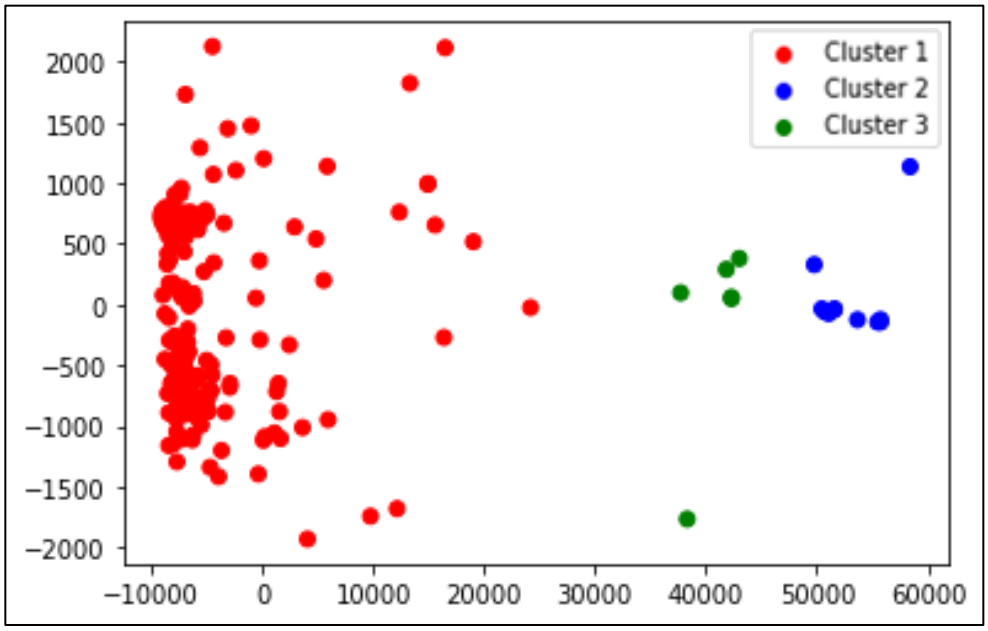

Figure 7, spectral clustering results, suggest three different downtime natures

Spectral clustering was able to sepearte those downtimes effectively, suggesting downtimes with three different natures, and based on our investigation we chracterized the three clusters as follows:

Cluster 1: Contains the shutdown events that a few features manifested tanglible changes within an hour leading to the shutdown, an example of that is demonstarted in the Figure 8, we called this group as unplanned shutdowns.

Cluster 2: Include the shutdowns that the system was back into normal process for a short time period and again another downtime occurred. We called this group as douple dip shutdowns, an instance of such downtimes is shown in Figure 9.

Cluster 3: Covers the shutdown events that occurred with no noticeable change in the features during the hours prior to the shutdowns, and we suspected that these downtimes were maybe caused intentionaly, and those might probabaly plannned in advance for any required maintenance or update in the system. An example of this group is displayed in Figure 10.

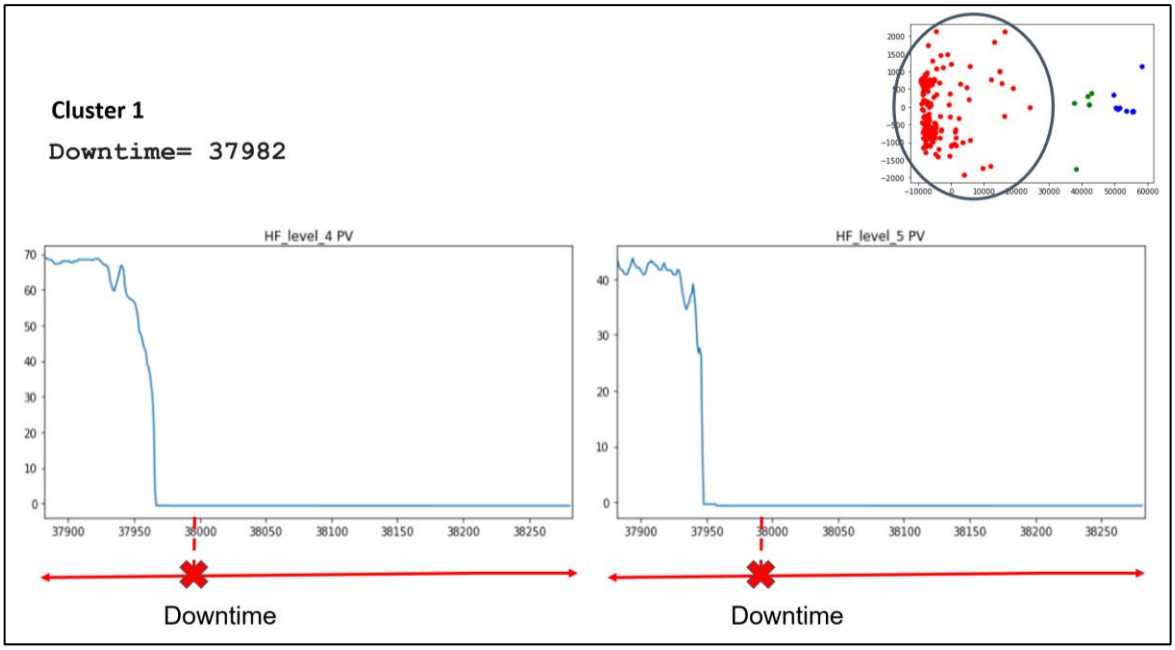

Figure 8, An example of a shutdown from the unplanned group. The displayed features gone through rapid changes before the shutdown is occurred.

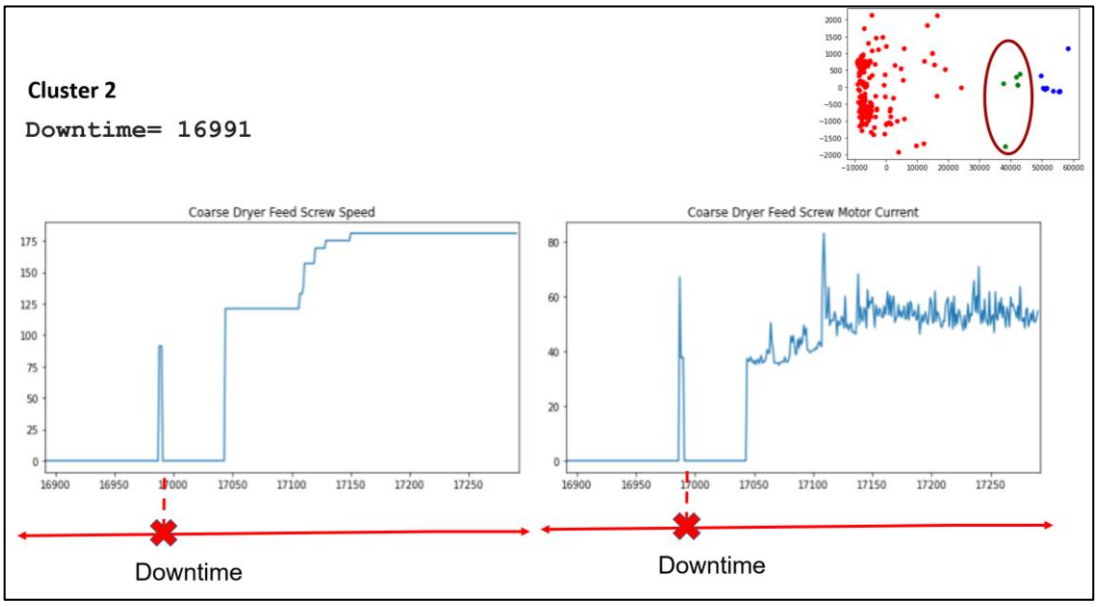

Figure 9, An example of a shutdown from the double dip group, after the 1st shutdown occurred, the course dryer speed back to normal for short time, before the 2nd shutdown strike

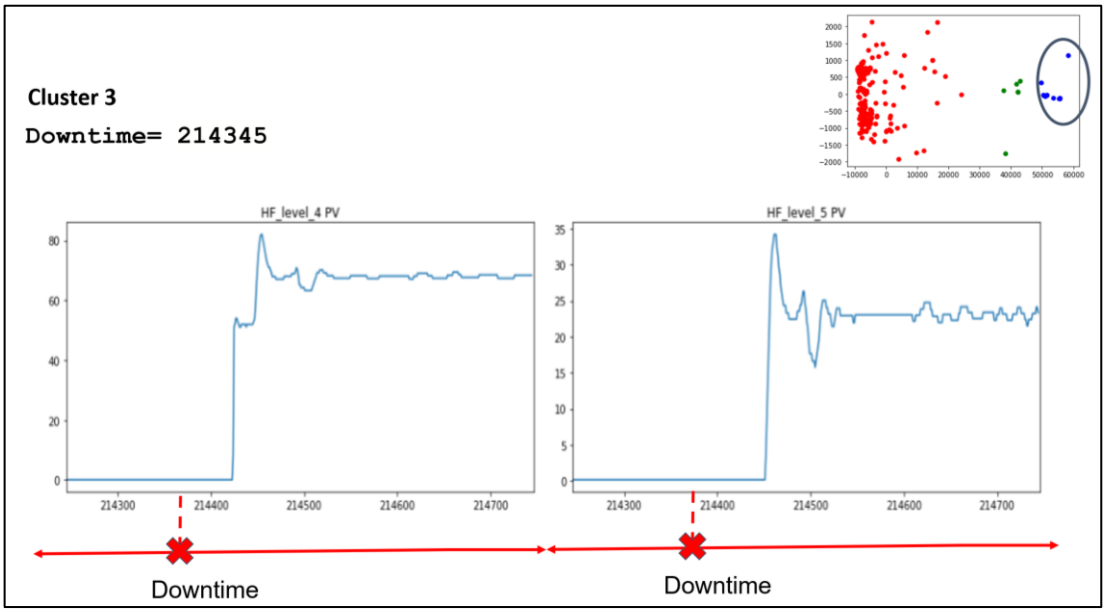

Figure 10, An example of a shutdown from the planned group. The related features didn't show any tangible changes before occurring of the shutdown.

It should be noted that while client appreicated our effort for categorizing the shutdown events, however they asked us to treat all the shudown equally the same, for the execution of this project.

- **Correlation analysis:** The figure 11 shows the features with highest correlation and the associated correlation features. In addition to this basic correlation analysis, we also performed the followings:
    - Extracted the correlations between the features and the delayed copy of themselves.
    - Identified new relationships between process values and set points (steady state errors) to be used in the model.
    - Identified the distribution of features change rate, to be used in the model as additional features.

The Feature Engineering process created new features using moving average and lags of the selected sensor readings and ultimately, we composed the dataset of 698401 rows (time samples) and 108 columns (features) for the train-test splitting and model building.

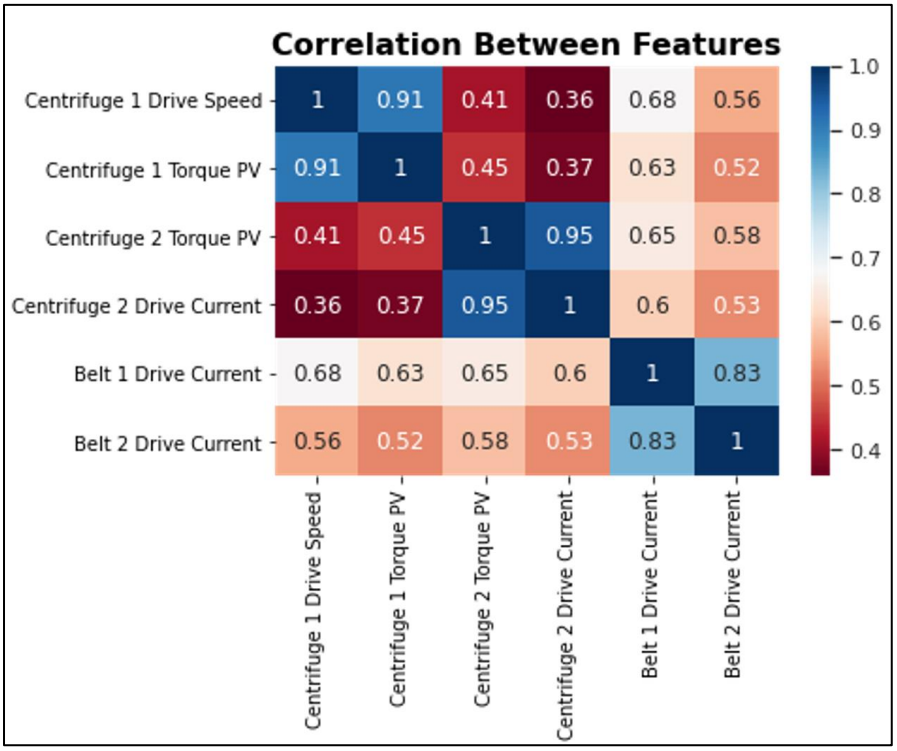

Figure 11, Correlation between features

Model Development:

Train-Test-Split: The Train-Test-Split created the train set for model training and test set for model validation. We tested two different approaches that the dataset can be splitted for this purpose:

1. Splitting dataset purely based on the date: This approach was simple to implement, but was not well representative of the dataset, as the downtime events were not distributed uniformly across the dataset.
2. Splitting dataset based on downtime test-portion: the most recent downtimes were selected to compose the test set, based on the portion of the total downtime discrete events that must be included in the test set, like 20% of the total downtimes, this was the approach that we followed.

Model Training:

"XGBoost" stands for "Extreme Gradient Boosting", and is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework.

The Modeling approach compiled the subsequent steps to train the model:

1) Finding the best features in the dataset: One of the benefits of using gradient boosting was after the boosted trees were constructed, it was relatively straightforward to retrieve importance scores for each attribute. We used 'RFECV', that is feature ranking with recursive feature elimination and cross-validated selection of the best number of features, and 'Time-Series-Split' for the cross validation.

2) Selecting the best parameters for the 'XGBoost': We used the Hyper-parameter tuning to finds the best configuration set for the 'XGBoost' model. we considered two different approaches for parameters selection:

    a. The Manual method, first uses 'Grid-Search-CV' and take all the combinations within the parameters.
    b. Bayesian optimization

We used the 1st method in this project and the table-1 shows the steps for the 'Grid-Search-CV':

Table 1, Grid-Search steps for 'XGBoost' parameters selection

Steps:	Tested Parameters:
Set-1	Max Depth: [9,12,15,18] Min Child Weight: [20,80,150, 250, 300]
Set-2	Gamma: [0.0,0.1,0.2,0.3] Scale Pos Weight: [10,20,40,80]
Set-3	Subsample: [0.6,0.7,0.8,1.0] Col-sample by tree: [0.6.0.7,0.8,1.0]
Set-4	Alpha: [0.0,0.1,0.05,0.1,1]

The Grid-Search started with the first set and once the best parameters for that set were selected, it then went to next set, and it continued until all the set of parameters were tuned. We used F1 score for the parameters tuning.

3) Train the model: We then used the selected features and parameters for the model training.

Model Performance Evaluation:

In our 'Model Evaluation' we adjusted the default Precision and Recall metrics for a fair evaluation of the model performance. We considered two different thresholds: 'Probability threshold' and' In-a-row threshold'. The first one was used to create the final 1 or 0 labels based on the model prediction likelihood and the second one for number of consistent predictions that appeared in a row, and both contributed for shutdown classification. In summary, these are the steps that we took to calculate the adjusted metric for the model evaluation:

1) Calculated the shutdown probability of each timestamp using the trained model
2) Created a label for the groups of predictions (either 1 or 0 based on probability threshold) that appeared sequentially in a row.
3) Calculated a new Precision, Recall, F1 score, and accuracy. Additionally, True positives/negatives and False positives/negatives were also calculated.

It should also be noted that our model was tested under a strict condition, which was a static or offline situation, and validated over a two-month period without retraining. We believe in the real practice, the model would be continuously updated and hence the performance of the model would gradually be improved.
Figure 12, shows the calculated Precision, Recall and Average Prediction Lead Time for our model. Our model Precision and Recalls were 69% and 48% respectively. Also, the average prediction lead time for the model was 38 minutes.

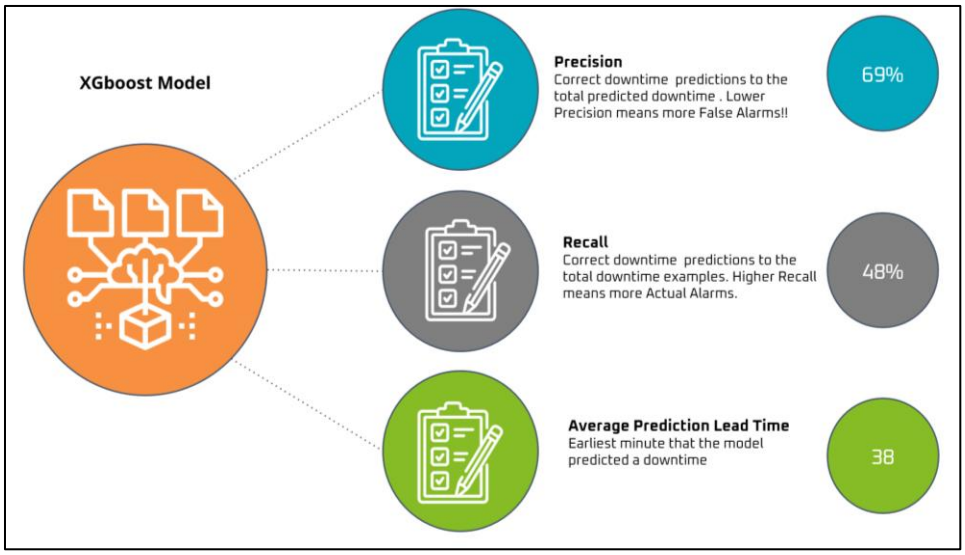

Figure 12, The established model Precision, Recall and Average Prediction Lead Time

False Alarm Rate: shows the number of the model predicted downtime events that were actually not a downtime. Over a period of two months, the model raised 16 alarms, that 11 of them were in fact true downtimes, and five of them were false, as demonstrated in Figure 10.

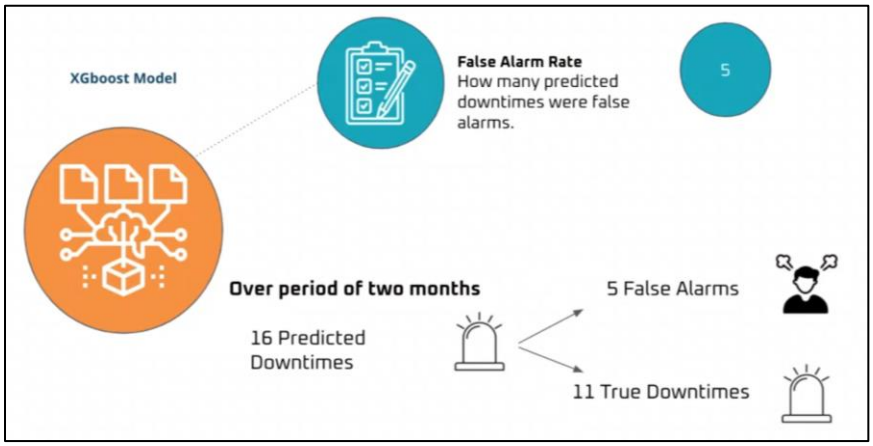

Figure 7, The model False Alarm details

True Alarms Rate: denotes on number of true downtime events the model was able to predict. We had twenty-three downtimes in our test sample, and eleven of them were correctly identified, which was better than the benchmark. Figure 11 shows the true alarm rate of the model.

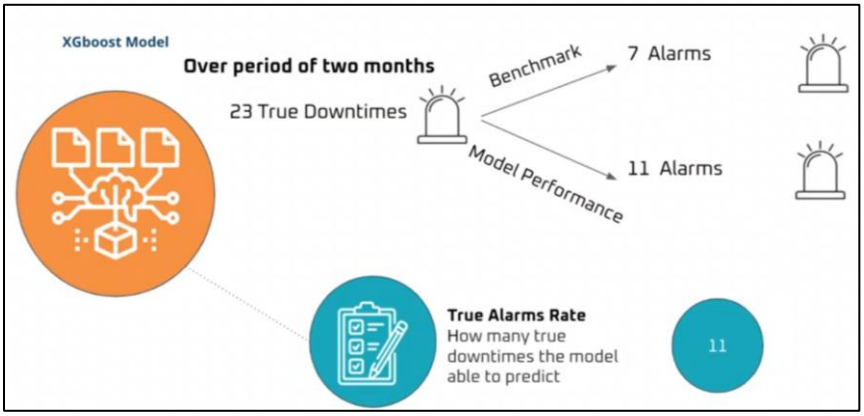

Figure 8, The model True Alarm details, compared with the benchmark

As it was mentioned earlier, the benchmark for us at the beginning of the project was to predict 30 percent of the true downtimes in the process, that is equivalent of seven downtimes out of twenty-three ones. The model that we developed, could predict 11 out of twenty-three, equivalent of forty eight percent of downtimes ahead of time, which was superior than the benchmark, and by doing this, as it is illustrated in figure 12, we would be able to save twenty-five thousand tonnes of potash from being wasted.

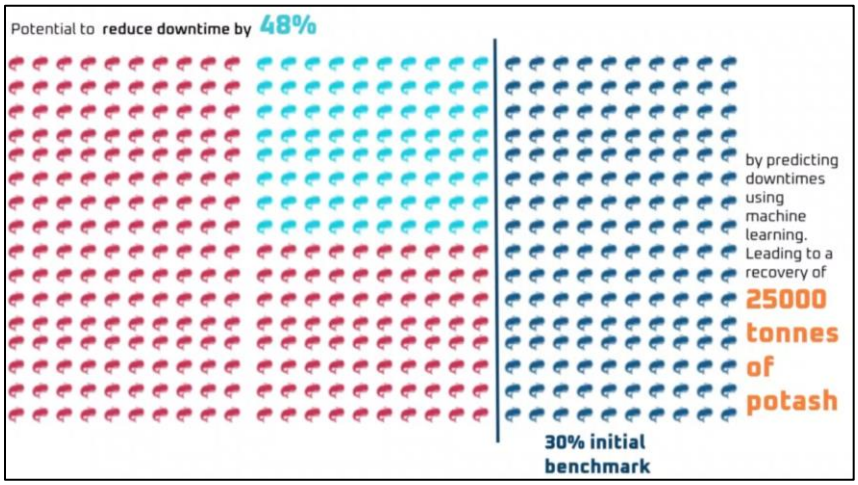

Figure 9, 25000 tonnes of potash would be saved with using our model, that is beyond of 16000 tonnes, that was earlier set as the benchmark for us

Prediction Structure

The final stage of the project pipeline was the prediction structure, and the goal here was to answer one fundamental question, and that was how the machine learning model would fit into the business process? Our proposed structure looked like the following:

Once the input data, extracted from the production process, was fed into the machine learning pipeline, the model would indicate the probability of a downtime occurring within one hour ahead, and then, the Decision-Making control would be transferred to the system operator or technical personnel or the 'human in the loop architecture', and all of this would be accomplished through the interactive dashboard, that was developed by us to complete this project.

The dashboard would give the human greater visibility into the model and would allow them to leverage their subject matter expertise to determine how imminent of a threat the shutdown really would be. The operator would then examine the current model performance and the system operating conditions to readjust the probability of a downtime in a continuous feedback loop to improve the model performance over time. Ultimately, this would give the operator the final decision of whether or not to continue operations or to re-adjust the system parameters before a downtime occurs. This was to utilize machine learning to enhance the decision-making ability of the user by delivering fast and accurate insights that otherwise would not be possible.

The above suggested Prediction structure is illustrated in Figure 13.

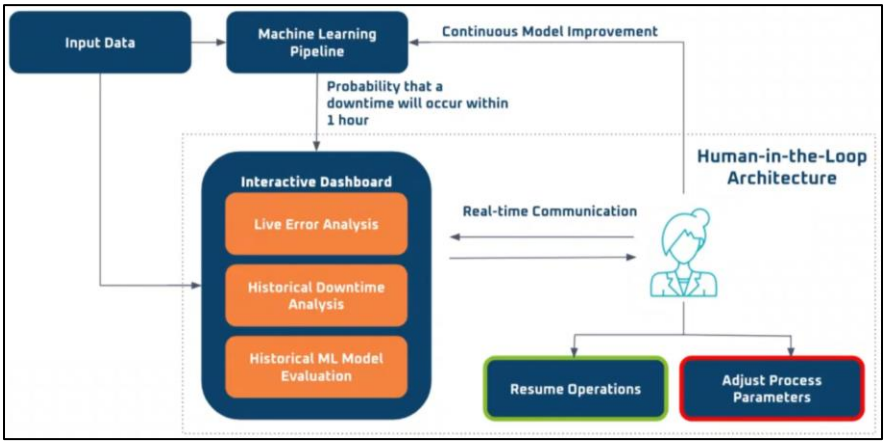

Figure 10, Downtime prediction structure

Dashboard

To demonstrate how the prediction structure might look like in practice, we have created a custom dashboard for our client, and let's follow a scenario in which the site engineer, who's monitoring the production process, is about to experience a downtime event. Under normal condition the status of the dashboard would be something similar to Figure 14, the feature signals and the data that's coming from the process system all looks healthy and the green check mark suggest that everything is normal. The user also has the option to select and visualize other features from the dropdown menu.

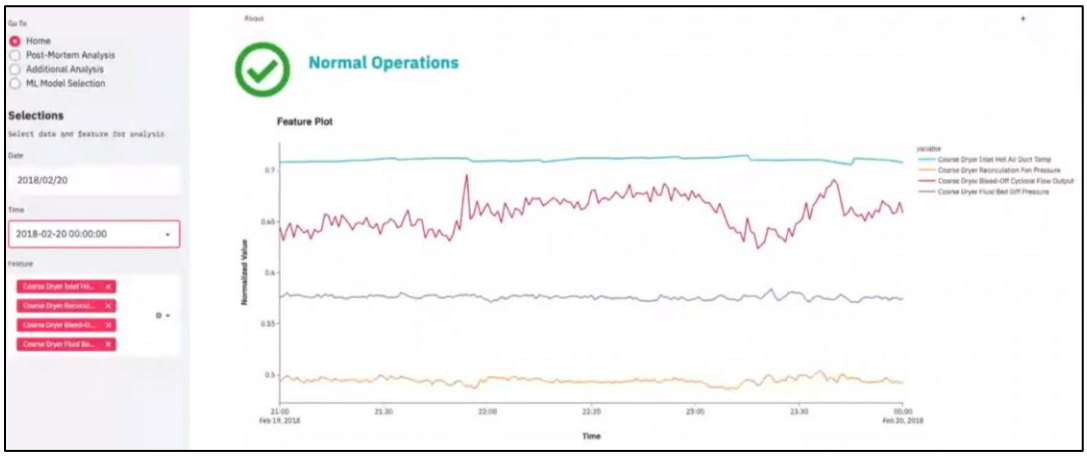

Figure 11, Dashboard demonstration: how dashboard main display look like, when there is no shutdown predicted ahead

Figure 15 demonstrates a situation that model foresees that a shutdown may occurs within an hour ahead, the operator then can observe how the signals have changed and examine the features and the corresponding values, and eventually, decides how to act from here.

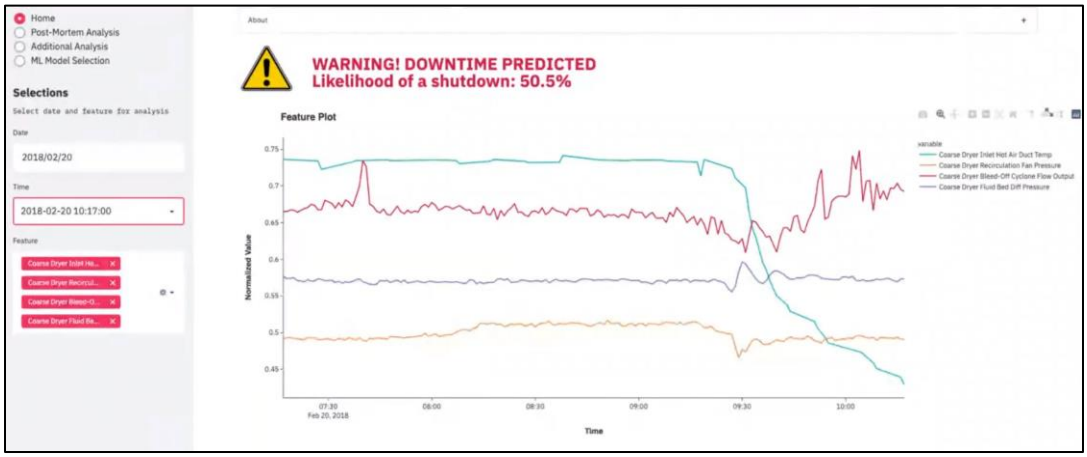

Figure 12, Dashboard demonstration: when the model suggests that a shutdown would be underway

The current version of the dashboard has two other functionalities, one is to analyze any past shutdown event, that is called Post-mortem Analysis, and the other provides the model performance report and exhibits the model parameters and the statistics of the model performance for the train and test datasets. The Figure 16 shows, the generated model performance report.

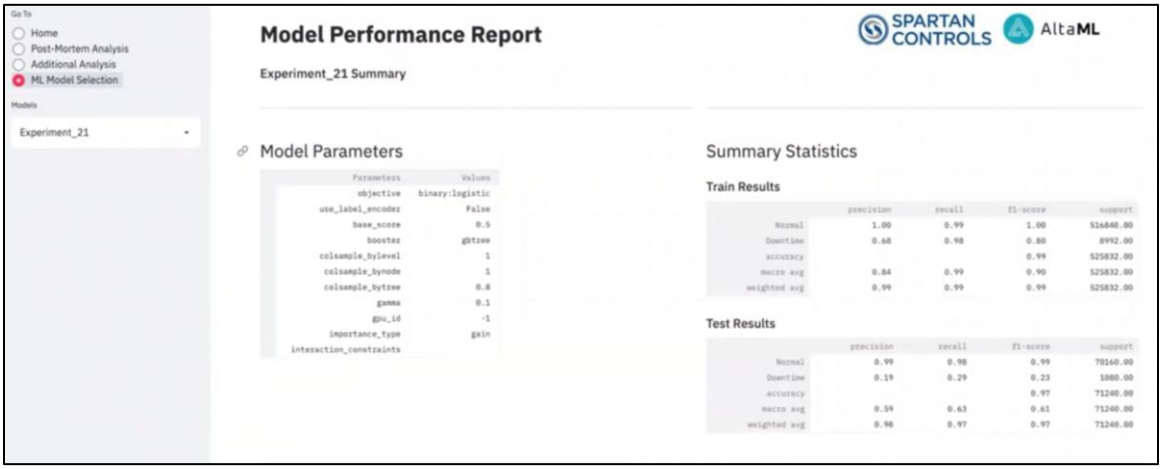

Figure 13, Model Performance Report

Conclusion and Recommendation

Finally, after execution of this project, we recommended to implement more data centric techniques as a method to enhance the model performance, and to better prepare and organize the data before going into the modeling part. The key initiatives were:

1) Re-evaluating how the shutdowns were labeled and to classify shutdowns by plan versus unplanned. This was something that's currently not being implemented, and it would be vital to do.
2) It was also recommended to test the model with real time data to assess its performance and it was hypothesized that this would improve the model performance.
3) The model could always be improved by techniques such as generating additional features that were more suitable to the process and by frequently updating the model with new insights derived from the production.

References:

1) https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html
2) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
3) https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d
4) https://xgboost.readthedocs.io/en/latest/tutorials/model.html
5) https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/
6) https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html
7) AltaML internal presentation templates and materials