Predictive Maintenance using AI and ML

UCWL UDAIPUR CEMENT WORKS LIMITED

JK LAKSHMI CEMENT

Team Members

Devanshu Thakar

Neel Patel

Jay Shah

Roopak Sharma

Table of Contents :

1. Overview of Predictive Maintenance

Predictive Maintenance techniques are designed to determine the condition of the operating machines. It's a concept, which is applied to optimize machine maintenance methods through the prediction of asset failures with data-driven techniques. This approach helps to operate the plant more efficiently as it reduces the downtime of the plant by detecting the error in the machine more quickly and allows the company to check and rectify the error in the machine before the complete breakdown of the system. It prevents the machine with a sudden breakdown during working hours.

Predictive maintenance using artificial intelligence and Machine learning

Currently, Industry has no standard method to predict the number of working hours before the operating machine undergoes the maintenance. The latest technology of Artificial Intelligence and Machine Learning have significantly changed the conventional method, which simply alerts the person in operation of the plant in case of maintenance required of a machine well in advance.

Nowadays many industrial plants are using Predictive Maintenance techniques using AI and ML as:

1. It helps to predict the condition of machine quickly and accurately
2. No manual check requirement which reduce the man-power in the plant
3. Reduce the Downtime of the plant
4. The production becomes more efficient
5. Maintenance at a proper time reduce the cost of the production to the company

2. Methodology Used

1. **Data collection:**

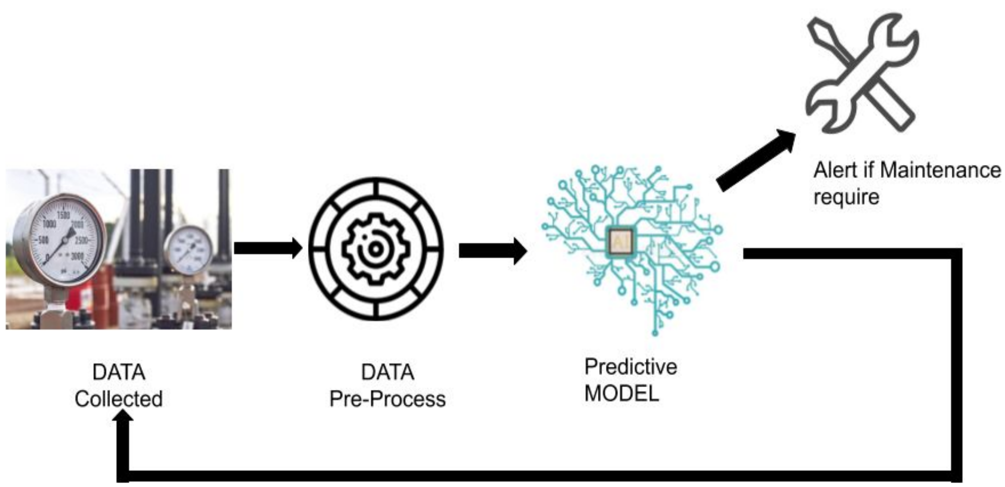
Gathering the data from the machines across the plant.

2. **Data preprocessing**:

Data gets transformed into a more useful and efficient format from the raw data collected.

3. **Fault detection:**

The data runs through the machine learning model which helps in predicting where the fault could be. Identifying a common pattern that the machine follows before being maintained. For machine learning, we have used the python library- Keras. Keras is a standard library in implementing machine learning models.

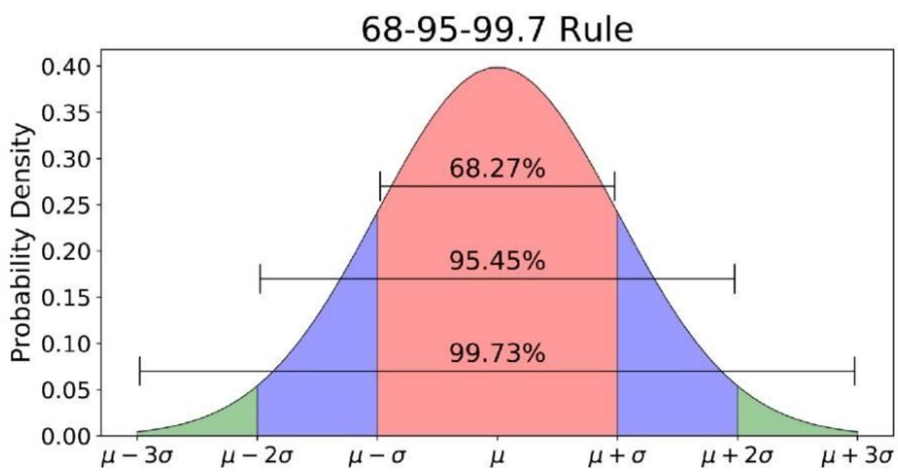


3.Gaussian distribution method:

First of all the data sheet provided by the technical team of UCWL was combined into a single sheet, using pandas of python.

After that the technical team of UCWL had provided three groups of parameter data. a machine learning model to detect anomalous data points was developed using keras. To analyse the performance of the model, anomalies were also identified using multivariate Gaussian distribution method. Since, the given data point was not tagged as anomalous or normal, we had used the Gaussian distribution to sort out the anomalous data.

In Gaussian approach all the features are modelled assuming Gaussian distribution. The mean and standard deviation are determined using features from the given data. The probability of the new data point can be computed and it can be determined whether it is anomalous or not, by comparing the probability with some selected threshold.



Gaussian Distribution. Image from [towards data science, Normal Distribution]

According to statistics about 68.26% of the data from a population lies within $\pm\sigma$,1 times of standard deviation. About 95.45 % of the data from a population lies within $\pm 2\sigma$, 2 times of standard deviation. So, we selected the 2σ as a threshold for determining the outliers values. The performance of this statistical model was compared with the keras model developed using TensorFlow. The following section discusses the python model developed and its results.

4. Parameter Group-1:

Explanation of Code :

The data points where the plant was shut down were removed from the analysis. Then after the data points for the parameter group-1 were separated in a separate csv file. The data was loaded into the jupyter notebook using pandas. The snapshot of loaded data is shown below :

```
In [2]: Parameter1 = pd.read_csv("ph coal outlet.csv")
        Parameter1.head()
```

Out[2]:

	Unnamed: 0	LOGO	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 8	Unnamed: 9	Unnamed: 10	...
0	7	TAG	A_431FM1_F1	B_431FM2_F1	C_431BE1_JT	D_441CN1A_1B_TT01	E_451PC1_TT01	H_461KL1_PYRO_MTR	I_Kiln RPM	J_461KL1_IZ	...
1	8	TAG DES	Kiln Feed	Kiln Feed	Kiln Feed BE KW	PH O/L TEMP.	Calciner Outlet Temp (TC)'	BURNING ZONE TEMP	Kiln rpm	Kiln current	...
2	9	UNIT	tph	tph	Kw	°C	°C	°C	rpm	A	...
3	10	06/01/2019 00:00:00	0.0	299.4	102.0	354.4	962.2	1135.5	5.0	388.0	...
4	11	06/01/2019 01:00:00	0.0	300.1	101.7	353.5	954.7	1141.8	5.0	399.5	...

5 rows × 28 columns

After that, according to the inputs from our mentor the velocity of the kiln feed A and feed B were added up into a single column named kiln feed B. The pandas dataframe was converted into a numpy array after discarding non-numeric data points like timestamp and name of the tag. The mean and standard deviation for all the features was computed using the inbuilt method in numpy and stored in numpy list named $means$ and $STDs$.

Using 2σ as the threshold, the Gaussian model thinks that 1224 data points out of are anomalous. These anomalous data points were stored in the csv file named $Gaussian\_Anomalous\_Parameter2.csv$

Thereafter the data is again loaded to feed into the keras model. The tensorflow libraries were imported into the program. The training data, test data and validate data was fed into the kears to do the normalization.

The features of the training model for group-2 were reduced to 25 and were fed to the keras model. The architecture of the neural network is having 25 input nodes followed by 30 hidden nodes followed by 20 hidden nodes which is followed by another 10 hidden layers, and having symmetric structure further with 25 output nodes. The neural network architecture is described in the code as :

```
layerin = keras.Input(shape=(25,))
x = keras.layers.Dense(30,activation="relu",activity_regularizer=regularizers.l1(0.008))(layerin)
x = keras.layers.Dense(20,activation="relu",activity_regularizer=regularizers.l1(0.008))(x)
x = keras.layers.Dense(10,activation="relu",activity_regularizer=regularizers.l1(0.008))(x)
x = keras.layers.Dense(20,activation="relu",activity_regularizer=regularizers.l1(0.008))(x)
x = keras.layers.Dense(30,activation="relu",activity_regularizer=regularizers.l1(0.008))(x)
x = keras.layers.Dense(25)(x)
model = keras.Model(inputs=layerin,outputs=x)
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_3 (InputLayer)        [(None, 25)]              0
_________________________________________________________________
dense_12 (Dense)            (None, 30)                780
_________________________________________________________________
dense_13 (Dense)            (None, 20)                620
_________________________________________________________________
dense_14 (Dense)            (None, 10)                210
_________________________________________________________________
dense_15 (Dense)            (None, 20)                220
_________________________________________________________________
dense_16 (Dense)            (None, 30)                630
_________________________________________________________________
dense_17 (Dense)            (None, 25)                775
=================================================================
Total params: 3,235
Trainable params: 3,235
```

The model was compiled using the optimizer of *adam* and the loss of *mean_squared_error*. The model was trained in 100 epochs and batch size of 40. An epoch refers to one cycle through the full training dataset. Usually training a neural network takes more than a few epochs. A batch size is the number of training examples present in a single batch. Generally an epoch of 100 and batch size of 30 to 40 is chosen if the data set is not very large. So, we had also used an epoch of 100. We can have trade off between accuracy and the computation time in training the model by varying the batch size and epoch.

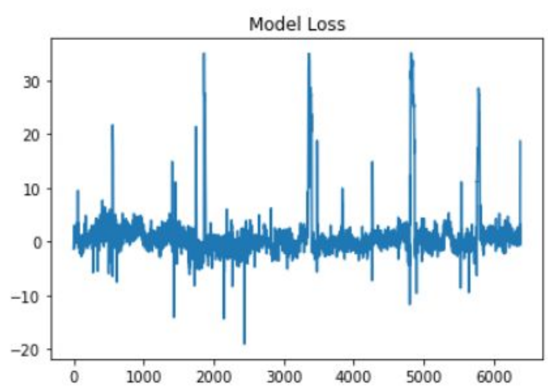
```
model.compile(optimizer="adam",loss="mean_squared_error")
validation_set = tf.data.Dataset.from_tensor_slices((validatein,validateoutlet))
history = model.fit(trainin_normalize,trainoutlet_normalize,batch_size=40,epochs=100,validation_data=(validatein_normalize,valida
```

```
Train on 4154 samples, validate on 1000 samples
Epoch 1/100
4154/4154 [==============================] - 5s 1ms/sample - loss: 1.0833 - val_loss: 1.0490
Epoch 2/100
4154/4154 [==============================] - 0s 81us/sample - loss: 0.9632 - val_loss: 0.9258
Epoch 3/100
4154/4154 [==============================] - 0s 39us/sample - loss: 0.8471 - val_loss: 0.8344
Epoch 4/100
4154/4154 [==============================] - 0s 40us/sample - loss: 0.7282 - val_loss: 0.7430
Epoch 5/100
4154/4154 [==============================] - 0s 41us/sample - loss: 0.6479 - val_loss: 0.6911
Epoch 6/100
4154/4154 [==============================] - 0s 39us/sample - loss: 0.5926 - val_loss: 0.6566
Epoch 7/100
4154/4154 [==============================] - 0s 40us/sample - loss: 0.5490 - val_loss: 0.6268
Epoch 8/100
4154/4154 [==============================] - 0s 39us/sample - loss: 0.5125 - val_loss: 0.6119
Epoch 9/100
4154/4154 [==============================] - 0s 38us/sample - loss: 0.4884 - val_loss: 0.6030
Epoch 10/100
```

The loss of the model is minimized in training the model. Loss is a parameter from which the performance of the model can be quantified. The loss of our model is plotted using matplotlib and is shown below :

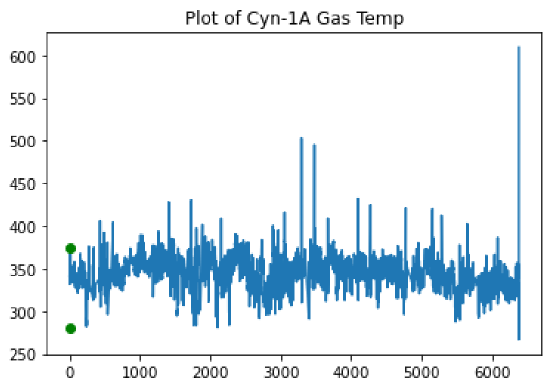
```
plt.plot(loss_model)
plt.title('Model Loss')
```

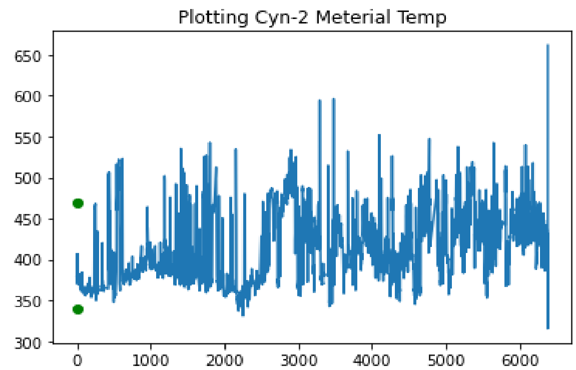
```
Text(0.5, 1.0, 'Model Loss')
```

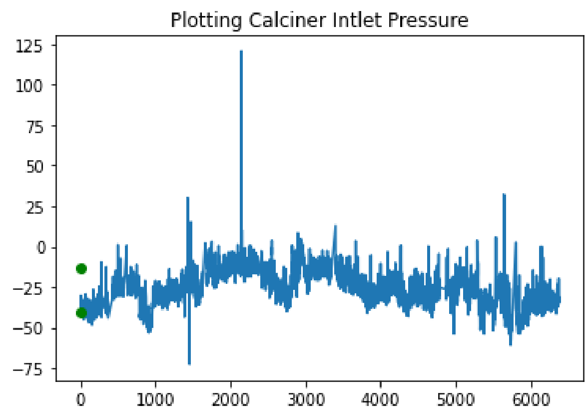


The data points having large jumps or dips corresponds to the anomalous data points according to the model.

The graph of some other parameters like Cyn-1A Gas Temp, Cyn-2 Material Temp, Calciner Inlet Pressure etc. was also plotted using matplotlib, in the jupyter notebook. Some of the Graphs are shown below :







Results

By using 2X(standard deviation) there are 1224 anomalous cases.

Out of 6378 test examples, the model thinks that 282 are anomalous.

Out of 6374 test examples, the model truly detected anomaly in 274 cases. TRUE POSITIVE(Anomalous)

Out of 6374 test examples, the model falsely detected anomaly on 8 cases.FALSE POSITIVE(Anomalous)

Out of 6374 test examples, the model falsely detected normalcy on 1224-282 = 942 cases. FALSE NEGATIVE(normalcy)

Since we were not provided with the exact anomalous data points. We had used the statistical method to determine anomalous data points. The accuracy of the model in detecting truly anomalous is quite good. However, according to 2 times of standard deviation, there are still 942 anomalous according to statistics, where the model detected normalcy. These can be improved if the actual anomalous data was used, instead of statistically determining anomalous data.

5. Parameter Group - 2:

Explanation of Code :

The data points where the plant was shut down were removed from the analysis. Then after the data points for the parameter group-2 were separated in a separate csv file. The data was loaded into the jupyter notebook using pandas. The snapshot of loaded data is

shown below :

```
In [2]: Parameter2 = pd.read_csv("Pyro Process Parameters-2.csv")
        Parameter2.head()
```

Out[2]:

	Unnamed: 0	TAG	A_431FM1_FI	B_431FM2_FI	C_431BE1_JT	D_471FN1_FT01	E_471FN2_FT01	F_471FN3_FT01
0	0	TAG DES	Kiln Feed	Kiln Feed	Kiln Feed BE KW	COOLER FAN-1 FOLW	COOLER FAN-2 FOLW	COOLER FAN-3 FOLW
1	1	UNIT	TPH	TPH	Kw	m^3/min	m^3/min	m^3/min
2	2	06-01-2019 00:00	0	299.4	102	580.5	1052.4	1348.2
3	3	06-01-2019 01:00	0	300.1	101.7	580.7	1050	1338.9
4	4	06-01-2019 02:00	0	299.6	101.8	579.9	1051	1355.8

5 rows × 26 columns

After that, according to the inputs from our mentor the velocity of the kiln feed A and feed B were added up into a single column named kiln feed B. The pandas dataframe was converted into a numpy array after discarding non-numeric data points like timestamp and name of the tag. The mean and standard deviation for all the features was computed using the inbuilt method in numpy and stored in numpy list named *means* and *STDs*. Using 2σ as the threshold, the Gaussian model thinks that 957 data points out of are anomalous. These anomalous data points were stored in the csv file named *Gaussian_Anomalous_Parameter2.csv*

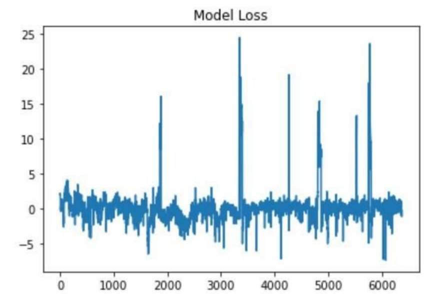
The features of the training model for group-2 were reduced to 22 and were fed to the keras model. Using the similar neural network architecture as for parameter 1, the neural network is shown below :

```
In [46]: layerin = keras.Input(shape=(22,))
         x = keras.layers.Dense(30,activation="relu",activity_regularizer=regularizers.l1(0.008))(layerin)
         x = keras.layers.Dense(20,activation="relu",activity_regularizer=regularizers.l1(0.008))(x)
         x = keras.layers.Dense(10,activation="relu",activity_regularizer=regularizers.l1(0.008))(x)
         x = keras.layers.Dense(20,activation="relu",activity_regularizer=regularizers.l1(0.008))(x)
         x = keras.layers.Dense(30,activation="relu",activity_regularizer=regularizers.l1(0.008))(x)
         x = keras.layers.Dense(22)(x)
         model = keras.Model(inputs=layerin,outputs=x)
         model.summary()

         Model: "model_1"
         _________________________________________________________________
         Layer (type)                 Output Shape              Param #
         =================================================================
         input_2 (InputLayer)         [(None, 22)]              0
         _________________________________________________________________
         dense_6 (Dense)              (None, 30)                690
         _________________________________________________________________
         dense_7 (Dense)              (None, 20)                620
         _________________________________________________________________
         dense_8 (Dense)              (None, 10)                210
         _________________________________________________________________
         dense_9 (Dense)              (None, 20)                220
         _________________________________________________________________
```

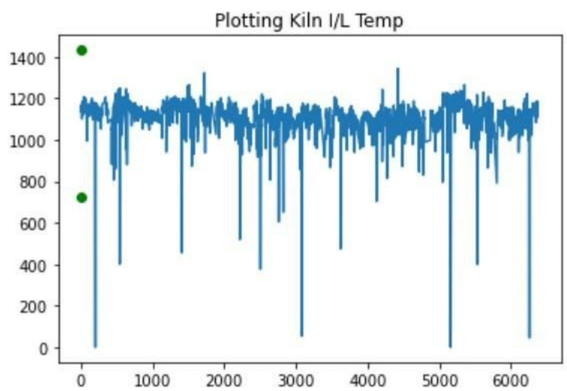
```
In [47]: model.compile(optimizer="adam",loss="mean_squared_error")
         validation_set = tf.data.Dataset.from_tensor_slices((validatein,validateoutlet))
         history = model.fit(trainin_normalize,trainoutlet_normalize,batch_size=40,epochs=100,validation_dat

         Train on 4417 samples, validate on 1000 samples
         Epoch 1/100
         4417/4417 [==============================] - 1s 238us/sample - loss: 1.0863 - val_loss: 1.0169
         Epoch 2/100
         4417/4417 [==============================] - 0s 44us/sample - loss: 0.8487 - val_loss: 0.8475
         Epoch 3/100
         4417/4417 [==============================] - 0s 57us/sample - loss: 0.7210 - val_loss: 0.7548
         Epoch 4/100
         4417/4417 [==============================] - 0s 54us/sample - loss: 0.6583 - val_loss: 0.7108
         Epoch 5/100
         4417/4417 [==============================] - 0s 43us/sample - loss: 0.6175 - val_loss: 0.6781
         Epoch 6/100
         4417/4417 [==============================] - 0s 43us/sample - loss: 0.5779 - val_loss: 0.6456
         Epoch 7/100
```

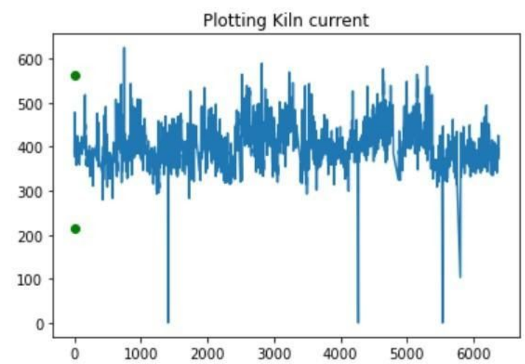
The model was trained using the same optimizer and loss as in parameter 1. The values of epoch and batch size were also kept similar. The loss of our model plotted using matplotlib and is shown below :

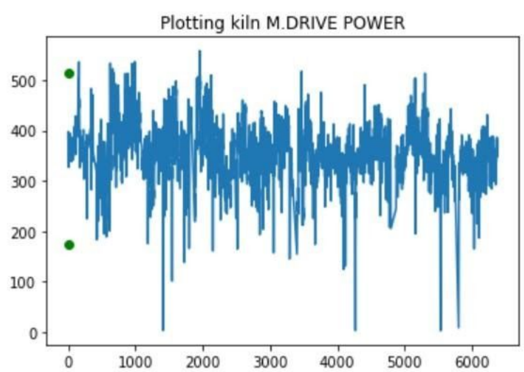
```
In [61]: plt.plot(loss_model)
         plt.title('Model Loss')

Out[61]: Text(0.5, 1.0, 'Model Loss')
```



The data points having large jumps or dips corresponds to the anomalous data points according to the model. The graph of some other parameters like tertiary air temperature, kiln rpm, kiln current, kiln M. Drive Power etc. was also plotted using matplotlib, in the jupyter notebook. Some of the Graphs are shown below :







Results

By using 2X(standard deviation) there are 957 anomalous cases. Out of 6374 test examples, the model thinks that 220 are anomalous.

Out of 6374 test examples, the model truly detected an anomaly in 220 cases. TRUE POSITIVE(Anomalous)

Out of 6374 test examples, the model falsely detected an anomaly in 0 cases. FALSE POSITIVE(Anomalous)

Out of 6374 test examples, the model falsely detected normalcy on 957-220 = 737 cases. FALSE NEGATIVE(normalcy)

Similar model can be developed for parameter group 3.

7. Conclusion

The project performed a detailed analysis of all the parameters given by the technical expert from the J K Lakshmi Udaipur Plant, were analyzed. The deep learning model was able to identify the failure points that were out of the normal range. The model found that failure or abnormality occurs when the feed rate drops below the critical value. The model first needs to be implemented practically to collect more data and make the model more robust. The model can be improved further if more detailed data of the parameters is analyzed

The project helped us learn many new concepts and gave us to apply our knowledge of Machine learning into a real-world challenge. None of us had anticipated a summer like this. While the world continues to fight an endless battle with this pandemic, an internship at UCWL was a great opportunity for us to hone our skill in AI and ML. Working From Home was a completely different experience for us and the experience was great. We would like to thank the technical team and our mentor from UCWL for continuous support and thank you for an opportunity to work in an esteem organisation.