Corteva Test Analysis

May 16, 2022

[61]:
```
# import libraries
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import os
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import joblib
import seaborn as sns
sns.set(color_codes=True)
import matplotlib.pyplot as plt
%matplotlib inline


from numpy.random import seed
import tensorflow as tf
from tensorflow.keras.layers import Input, Dropout, Dense, LSTM,␣
 ↪TimeDistributed, RepeatVector
from tensorflow.keras.models import Model
from tensorflow.keras import regularizers
```

[62]:
```
# Test for the set n.2
dataset = pd.read_csv('/home/developer/Documents/Machine Learning/Corteva/
 ↪dataset/cleaned/dryer_3_fan_1_2020.csv', sep=';', usecols =␣
 ↪['Motor_Frequency_Cmd_Hz','X_Axis_High_Frequency_RMS_Acceleration','X_Axis_RMS_Velocity',␣
 ↪'Z_Axis_High_Frequency_RMS_Acceleration', 'Z_Axis_RMS_Velocity'],)
# dataset.columns =␣
 ↪['Motor_Frequency_Cmd_Hz','Timestamp','X_Axis_High_Frequency_RMS_Acceleration','X_Axis_RMS_
 ↪'Z_Axis_High_Frequency_RMS_Acceleration', 'Z_Axis_RMS_Velocity']
dataset.head()
```

[62]:
```
  Motor_Frequency_Cmd_Hz X_Axis_High_Frequency_RMS_Acceleration  \
0                   32,8                                  0,232
1                   32,8                                  0,233
2                   40,2                                  1,047
3                   40,2                                  1,155
```

```
4                  40,2                                  0,773

  X_Axis_RMS_Velocity Z_Axis_High_Frequency_RMS_Acceleration  \
0               0,326                                  0,169
1                0,38                                  0,171
2               0,538                                  0,511
3               0,572                                  0,516
4               0,605                                   0,47

  Z_Axis_RMS_Velocity
0               3,305
1               1,335
2               0,456
3               0,376
4               0,363
```

[63]:

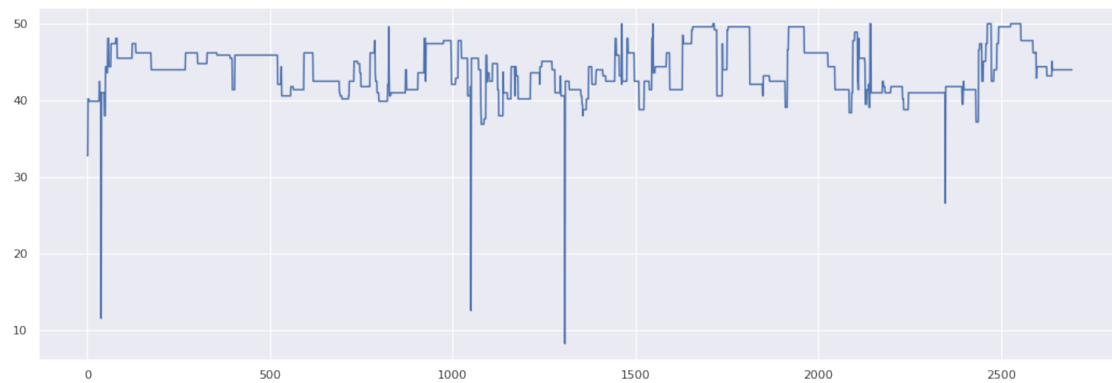
```python
# Convert values with comma into float
dataset['X_Axis_High_Frequency_RMS_Acceleration'] = dataset['X_Axis_High_Frequency_RMS_Acceleration'].astype(str).str.replace(',', '.').astype(float)
dataset['X_Axis_RMS_Velocity'] = dataset['X_Axis_RMS_Velocity'].astype(str).str.replace(',', '.').astype(float)
dataset['Z_Axis_High_Frequency_RMS_Acceleration'] = dataset['Z_Axis_High_Frequency_RMS_Acceleration'].astype(str).str.replace(',', '.').astype(float)
dataset['Z_Axis_RMS_Velocity'] = dataset['Z_Axis_RMS_Velocity'].astype(str).str.replace(',', '.').astype(float)
dataset['Motor_Frequency_Cmd_Hz'] = dataset['Motor_Frequency_Cmd_Hz'].astype(str).str.replace(',', '.').astype(float)
dataset
```

[63]:

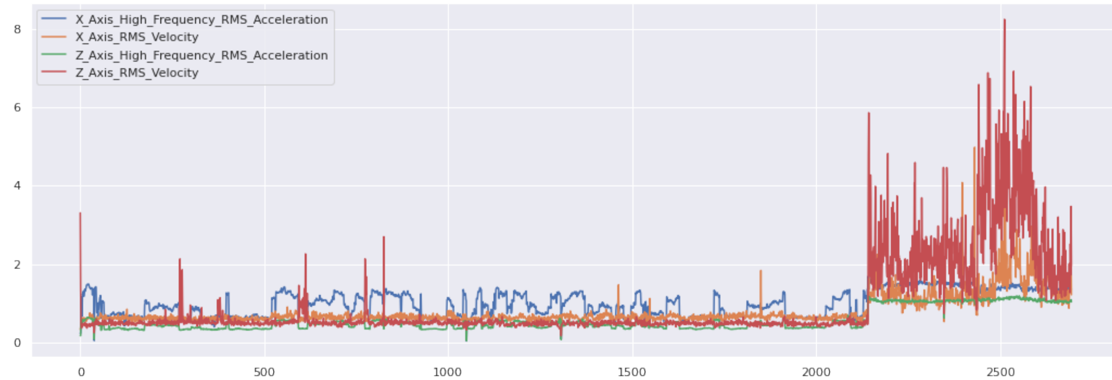
```
      Motor_Frequency_Cmd_Hz  X_Axis_High_Frequency_RMS_Acceleration  \
0                       32.8                                   0.232
1                       32.8                                   0.233
2                       40.2                                   1.047
3                       40.2                                   1.155
4                       40.2                                   0.773
…                        …                                      …
2690                    44.0                                   1.340
2691                    44.0                                   1.441
2692                    44.0                                   1.408
2693                    44.0                                   1.363
2694                    44.0                                   1.353

      X_Axis_RMS_Velocity  Z_Axis_High_Frequency_RMS_Acceleration  \
0                   0.326                                   0.169
1                   0.380                                   0.171
```

```
2                  0.538                                       0.511
3                  0.572                                       0.516
4                  0.605                                       0.470
…                  …                                         …
2690               1.607                                       1.053
2691               1.786                                       1.060
2692               1.395                                       1.047
2693               2.217                                       1.085
2694               1.213                                       1.032

      Z_Axis_RMS_Velocity
0                  3.305
1                  1.335
2                  0.456
3                  0.376
4                  0.363
…                  …
2690               1.264
2691               2.514
2692               1.718
2693               3.470
2694               1.954

[2695 rows x 5 columns]
```

[64]:
```
dataset['Motor_Frequency_Cmd_Hz'].plot(figsize=(18,6));
```



[65]:
```
dataset[['X_Axis_High_Frequency_RMS_Acceleration', 'X_Axis_RMS_Velocity',␣
↪'Z_Axis_High_Frequency_RMS_Acceleration', 'Z_Axis_RMS_Velocity']].
↪plot(figsize=(18,6));
```



[66]:
```
# Split trainig set and test set

# Training set
training_set = dataset[0:2092]
training_set
```

[66]:
```
      Motor_Frequency_Cmd_Hz  X_Axis_High_Frequency_RMS_Acceleration  \
0                       32.8                                   0.232
1                       32.8                                   0.233
2                       40.2                                   1.047
3                       40.2                                   1.155
4                       40.2                                   0.773
…                        …                                      …
2087                    38.4                                   1.252
2088                    38.4                                   1.265
2089                    38.4                                   1.314
2090                    38.4                                   1.301
2091                    38.4                                   1.351

      X_Axis_RMS_Velocity  Z_Axis_High_Frequency_RMS_Acceleration  \
0                   0.326                                   0.169
1                   0.380                                   0.171
2                   0.538                                   0.511
3                   0.572                                   0.516
4                   0.605                                   0.470
…                    …                                      …
2087                0.518                                   0.688
2088                0.619                                   0.626
2089                0.566                                   0.623
2090                0.587                                   0.609
2091                0.618                                   0.627

      Z_Axis_RMS_Velocity
```

```
0                   3.305
1                   1.335
2                   0.456
3                   0.376
4                   0.363
…                   …
2087                0.554
2088                0.505
2089                0.617
2090                0.590
2091                0.582

[2092 rows x 5 columns]
```

[67]:
```python
# Test set
test_set = dataset[2093: ]
test_set
```

[67]:
```
      Motor_Frequency_Cmd_Hz  X_Axis_High_Frequency_RMS_Acceleration  \
2093                    41.0                                   0.870
2094                    41.0                                   0.895
2095                    47.8                                   0.546
2096                    47.8                                   0.541
2097                    47.8                                   0.546
…                        …                                      …
2690                    44.0                                   1.340
2691                    44.0                                   1.441
2692                    44.0                                   1.408
2693                    44.0                                   1.363
2694                    44.0                                   1.353

      X_Axis_RMS_Velocity  Z_Axis_High_Frequency_RMS_Acceleration  \
2093                0.636                                   0.507
2094                0.621                                   0.470
2095                0.746                                   0.369
2096                0.724                                   0.362
2097                0.700                                   0.359
…                   …                                      …
2690                1.607                                   1.053
2691                1.786                                   1.060
2692                1.395                                   1.047
2693                2.217                                   1.085
2694                1.213                                   1.032

      Z_Axis_RMS_Velocity
2093                0.514
2094                0.439
```

```
2095                0.537
2096                0.573
2097                0.497
…                   …
2690                1.264
2691                2.514
2692                1.718
2693                3.470
2694                1.954

[602 rows x 5 columns]
```

[68]:
```python
# normalize the data
scaler = MinMaxScaler()
X_train = scaler.fit_transform(training_set)
X_test = scaler.transform(test_set)
scaler_filename = "scaler_data"
joblib.dump(scaler, scaler_filename)
```

[68]:
```
['scaler_data']
```

[69]:
```python
# max values
np.amax(training_set)
```

[69]:
```
Motor_Frequency_Cmd_Hz                        50.000
X_Axis_High_Frequency_RMS_Acceleration         1.484
X_Axis_RMS_Velocity                            1.835
Z_Axis_High_Frequency_RMS_Acceleration         0.688
Z_Axis_RMS_Velocity                            3.305
dtype: float64
```

[70]:
```python
# min values
np.amin(training_set)
```

[70]:
```
Motor_Frequency_Cmd_Hz                        8.300
X_Axis_High_Frequency_RMS_Acceleration        0.034
X_Axis_RMS_Velocity                           0.189
Z_Axis_High_Frequency_RMS_Acceleration        0.040
Z_Axis_RMS_Velocity                           0.164
dtype: float64
```

[71]:
```python
# training after normalization
X_train
```

[71]:
```
array([[0.58752998, 0.13655172, 0.08323208, 0.19907407, 1.        ],
       [0.58752998, 0.13724138, 0.11603888, 0.20216049, 0.37281121],
       [0.76498801, 0.69862069, 0.21202916, 0.72685185, 0.09296402],
```

```
       …,
       [0.72182254, 0.88275862, 0.2290401 , 0.89969136, 0.14422159],
       [0.72182254, 0.8737931 , 0.2417983 , 0.87808642, 0.1356256 ],
       [0.72182254, 0.90827586, 0.26063183, 0.9058642 , 0.13307864]])
```

[72]:
```python
# test after normalization
X_test
```

[72]:
```
array([[0.78417266, 0.57655172, 0.27156744, 0.72067901, 0.11142948],
       [0.78417266, 0.5937931 , 0.26245443, 0.66358025, 0.08755174],
       [0.94724221, 0.35310345, 0.33839611, 0.50771605, 0.11875199],
       …,
       [0.85611511, 0.94758621, 0.7326853 , 1.55401235, 0.4947469 ],
       [0.85611511, 0.91655172, 1.23207776, 1.61265432, 1.05253104],
       [0.85611511, 0.90965517, 0.62211422, 1.5308642 , 0.5698822 ]])
```

[73]:
```python
# reshape inputs for LSTM [samples, timesteps, features]
X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
print("Training data shape:", X_train.shape)
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])
print("Test data shape:", X_test.shape)
```

```
Training data shape: (2092, 1, 5)
Test data shape: (602, 1, 5)
```

[74]:
```python
X_train
```

[74]:
```
array([[[0.58752998, 0.13655172, 0.08323208, 0.19907407, 1.        ]],

       [[0.58752998, 0.13724138, 0.11603888, 0.20216049, 0.37281121]],

       [[0.76498801, 0.69862069, 0.21202916, 0.72685185, 0.09296402]],

       …,

       [[0.72182254, 0.88275862, 0.2290401 , 0.89969136, 0.14422159]],

       [[0.72182254, 0.8737931 , 0.2417983 , 0.87808642, 0.1356256 ]],

       [[0.72182254, 0.90827586, 0.26063183, 0.9058642 , 0.13307864]]])
```

[75]:
```python
X_test
```

[75]:
```
array([[[0.78417266, 0.57655172, 0.27156744, 0.72067901, 0.11142948]],

       [[0.78417266, 0.5937931 , 0.26245443, 0.66358025, 0.08755174]],
```

```
[[0.94724221, 0.35310345, 0.33839611, 0.50771605, 0.11875199]],

…,

[[0.85611511, 0.94758621, 0.7326853 , 1.55401235, 0.4947469 ]],

[[0.85611511, 0.91655172, 1.23207776, 1.61265432, 1.05253104]],

[[0.85611511, 0.90965517, 0.62211422, 1.5308642 , 0.5698822 ]]])
```

[76]:
```python
# define the autoencoder network model
def autoencoder_model(X):
    inputs = Input(shape=(X.shape[1], X.shape[2]))
    inputs
    L1 = LSTM(16, activation='relu', return_sequences=True,
              kernel_regularizer=regularizers.l2(0.00))(inputs)
    L2 = LSTM(4, activation='relu', return_sequences=False)(L1)
    L3 = RepeatVector(X.shape[1])(L2)
    L4 = LSTM(4, activation='relu', return_sequences=True)(L3)
    L5 = LSTM(16, activation='relu', return_sequences=True)(L4)
    output = TimeDistributed(Dense(X.shape[2]))(L5)
    model = Model(inputs=inputs, outputs=output)
    return model
```

[77]:
```python
# create the autoencoder model
model = autoencoder_model(X_train)
model.compile(optimizer='adam', loss='mae')
model.summary()
```

```
Model: "model_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 1, 5)]            0

 lstm_4 (LSTM)               (None, 1, 16)             1408

 lstm_5 (LSTM)               (None, 4)                 336

 repeat_vector_1 (RepeatVect  (None, 1, 4)             0
 or)

 lstm_6 (LSTM)               (None, 1, 4)              144

 lstm_7 (LSTM)               (None, 1, 16)             1344

 time_distributed_1 (TimeDis  (None, 1, 5)             85
```

```
 tributed)

=================================================================
Total params: 3,317
Trainable params: 3,317
Non-trainable params: 0
_________________________________________________________________
```

[78]:

```
# fit the model to the data
nb_epochs = 100
batch_size = 10
history = model.fit(X_train, X_train, epochs=nb_epochs, batch_size=batch_size,
                    validation_split=0.05).history
```

```
Epoch 1/100
199/199 [==============================] - 7s 10ms/step - loss: 0.3126 -
val_loss: 0.0884
Epoch 2/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0681 -
val_loss: 0.0625
Epoch 3/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0605 -
val_loss: 0.0590
Epoch 4/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0543 -
val_loss: 0.0525
Epoch 5/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0484 -
val_loss: 0.0469
Epoch 6/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0417 -
val_loss: 0.0366
Epoch 7/100
199/199 [==============================] - 1s 5ms/step - loss: 0.0291 -
val_loss: 0.0261
Epoch 8/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0248 -
val_loss: 0.0253
Epoch 9/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0232 -
val_loss: 0.0219
Epoch 10/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0220 -
val_loss: 0.0216
Epoch 11/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0216 -
val_loss: 0.0242
Epoch 12/100
```

```
199/199 [==============================] - 1s 6ms/step - loss: 0.0216 -
val_loss: 0.0201
Epoch 13/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0214 -
val_loss: 0.0226
Epoch 14/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0212 -
val_loss: 0.0224
Epoch 15/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0211 -
val_loss: 0.0204
Epoch 16/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0211 -
val_loss: 0.0226
Epoch 17/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0210 -
val_loss: 0.0209
Epoch 18/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0210 -
val_loss: 0.0219
Epoch 19/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0209 -
val_loss: 0.0252
Epoch 20/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0210 -
val_loss: 0.0236
Epoch 21/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0208 -
val_loss: 0.0202
Epoch 22/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0210 -
val_loss: 0.0201
Epoch 23/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0207 -
val_loss: 0.0213
Epoch 24/100
199/199 [==============================] - 1s 5ms/step - loss: 0.0207 -
val_loss: 0.0218
Epoch 25/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0207 -
val_loss: 0.0227
Epoch 26/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0206 -
val_loss: 0.0206
Epoch 27/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0208 -
val_loss: 0.0215
Epoch 28/100
```

```
199/199 [==============================] - 1s 7ms/step - loss: 0.0206 -
val_loss: 0.0211
Epoch 29/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0208 -
val_loss: 0.0211
Epoch 30/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0207 -
val_loss: 0.0222
Epoch 31/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0206 -
val_loss: 0.0207
Epoch 32/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0206 -
val_loss: 0.0194
Epoch 33/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0205 -
val_loss: 0.0212
Epoch 34/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0207 -
val_loss: 0.0216
Epoch 35/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0204 -
val_loss: 0.0214
Epoch 36/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0205 -
val_loss: 0.0217
Epoch 37/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0204 -
val_loss: 0.0227
Epoch 38/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0204 -
val_loss: 0.0207
Epoch 39/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0205 -
val_loss: 0.0190
Epoch 40/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0206 -
val_loss: 0.0212
Epoch 41/100
199/199 [==============================] - 1s 8ms/step - loss: 0.0204 -
val_loss: 0.0241
Epoch 42/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0205 -
val_loss: 0.0217
Epoch 43/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0205 -
val_loss: 0.0239
Epoch 44/100
```

```
199/199 [==============================] - 2s 8ms/step - loss: 0.0205 -
val_loss: 0.0228
Epoch 45/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0204 -
val_loss: 0.0206
Epoch 46/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0203 -
val_loss: 0.0205
Epoch 47/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0204 -
val_loss: 0.0221
Epoch 48/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0203 -
val_loss: 0.0212
Epoch 49/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0203 -
val_loss: 0.0205
Epoch 50/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0204 -
val_loss: 0.0212
Epoch 51/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0203 -
val_loss: 0.0206
Epoch 52/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0203 -
val_loss: 0.0196
Epoch 53/100
199/199 [==============================] - 1s 6ms/step - loss: 0.0203 -
val_loss: 0.0226
Epoch 54/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0204 -
val_loss: 0.0217
Epoch 55/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0204 -
val_loss: 0.0220
Epoch 56/100
199/199 [==============================] - 2s 9ms/step - loss: 0.0202 -
val_loss: 0.0222
Epoch 57/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0226
Epoch 58/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0205
Epoch 59/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0203 -
val_loss: 0.0216
Epoch 60/100
```

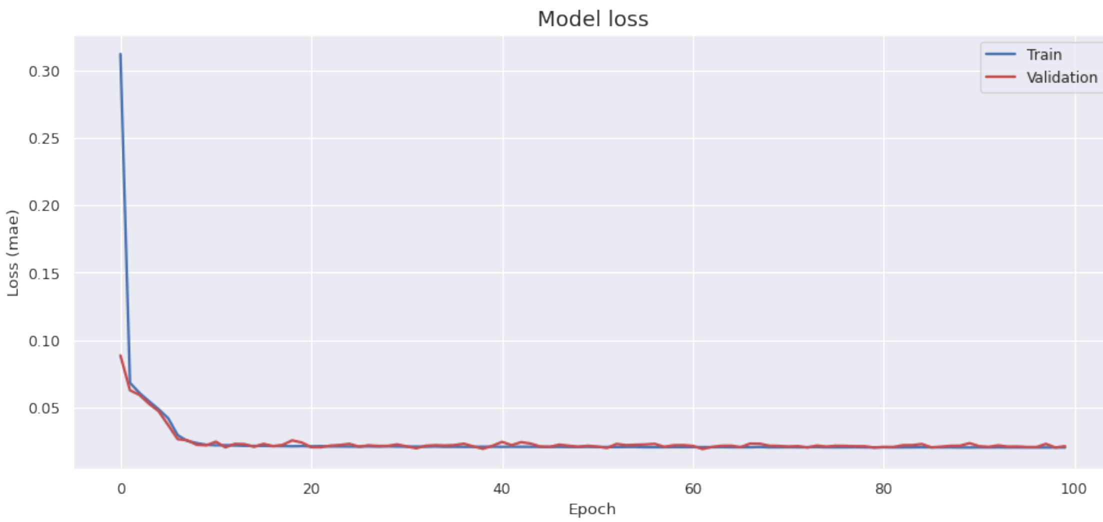
```
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0217
Epoch 61/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0211
Epoch 62/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0188
Epoch 63/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0204
Epoch 64/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0202 -
val_loss: 0.0212
Epoch 65/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0213
Epoch 66/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0202
Epoch 67/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0201 -
val_loss: 0.0228
Epoch 68/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0204 -
val_loss: 0.0228
Epoch 69/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0201 -
val_loss: 0.0213
Epoch 70/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0201 -
val_loss: 0.0211
Epoch 71/100
199/199 [==============================] - 2s 9ms/step - loss: 0.0202 -
val_loss: 0.0206
Epoch 72/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0201 -
val_loss: 0.0210
Epoch 73/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0201 -
val_loss: 0.0200
Epoch 74/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0203 -
val_loss: 0.0213
Epoch 75/100
199/199 [==============================] - 2s 9ms/step - loss: 0.0201 -
val_loss: 0.0206
Epoch 76/100
```

```
199/199 [==============================] - 2s 8ms/step - loss: 0.0201 -
val_loss: 0.0212
Epoch 77/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0201 -
val_loss: 0.0211
Epoch 78/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0202 -
val_loss: 0.0209
Epoch 79/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0200 -
val_loss: 0.0209
Epoch 80/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0200 -
val_loss: 0.0199
Epoch 81/100
199/199 [==============================] - 2s 9ms/step - loss: 0.0201 -
val_loss: 0.0204
Epoch 82/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0200 -
val_loss: 0.0203
Epoch 83/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0200 -
val_loss: 0.0216
Epoch 84/100
199/199 [==============================] - 2s 9ms/step - loss: 0.0201 -
val_loss: 0.0217
Epoch 85/100
199/199 [==============================] - 2s 9ms/step - loss: 0.0201 -
val_loss: 0.0225
Epoch 86/100
199/199 [==============================] - 1s 7ms/step - loss: 0.0201 -
val_loss: 0.0200
Epoch 87/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0200 -
val_loss: 0.0205
Epoch 88/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0201 -
val_loss: 0.0212
Epoch 89/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0200 -
val_loss: 0.0212
Epoch 90/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0199 -
val_loss: 0.0231
Epoch 91/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0200 -
val_loss: 0.0209
Epoch 92/100
```

```
199/199 [==============================] - 2s 8ms/step - loss: 0.0200 -
val_loss: 0.0204
Epoch 93/100
199/199 [==============================] - 2s 10ms/step - loss: 0.0201 -
val_loss: 0.0215
Epoch 94/100
199/199 [==============================] - 2s 12ms/step - loss: 0.0200 -
val_loss: 0.0206
Epoch 95/100
199/199 [==============================] - 2s 11ms/step - loss: 0.0200 -
val_loss: 0.0207
Epoch 96/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0199 -
val_loss: 0.0203
Epoch 97/100
199/199 [==============================] - 2s 10ms/step - loss: 0.0199 -
val_loss: 0.0203
Epoch 98/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0199 -
val_loss: 0.0226
Epoch 99/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0200 -
val_loss: 0.0199
Epoch 100/100
199/199 [==============================] - 2s 8ms/step - loss: 0.0199 -
val_loss: 0.0209
```

[79]:

```
# plot the training losses
fig, ax = plt.subplots(figsize=(14, 6), dpi=80)
ax.plot(history['loss'], 'b', label='Train', linewidth=2)
ax.plot(history['val_loss'], 'r', label='Validation', linewidth=2)
ax.set_title('Model loss', fontsize=16)
ax.set_ylabel('Loss (mae)')
ax.set_xlabel('Epoch')
ax.legend(loc='upper right')
plt.show()
```



[80]:
```python
# Distribution of Loss Function

# By plotting the distribution of the calculated loss in the training set, one
↪can use this to identify a suitable threshold value for identifying an
↪anomaly.


# plot the loss distribution of the training set
X_pred = model.predict(X_train)
X_pred = X_pred.reshape(X_pred.shape[0], X_pred.shape[2])
X_pred = pd.DataFrame(X_pred, columns=training_set.columns)
X_pred.index = training_set.index

scored = pd.DataFrame(index=training_set.index)
Xtrain = X_train.reshape(X_train.shape[0], X_train.shape[2])
scored['Loss_mae'] = np.mean(np.abs(X_pred-Xtrain), axis = 1)
```

[81]:
```python
X_pred
```

[81]:
```
      Motor_Frequency_Cmd_Hz  X_Axis_High_Frequency_RMS_Acceleration  \
0                   0.882420                                0.191387
1                   0.884089                                0.197339
2                   0.798753                                0.708779
3                   0.784880                                0.769590
4                   0.857167                                0.541086
…                        …                                     …
2087                0.756971                                0.907774
2088                0.761227                                0.884896
2089                0.756366                                0.911066
```

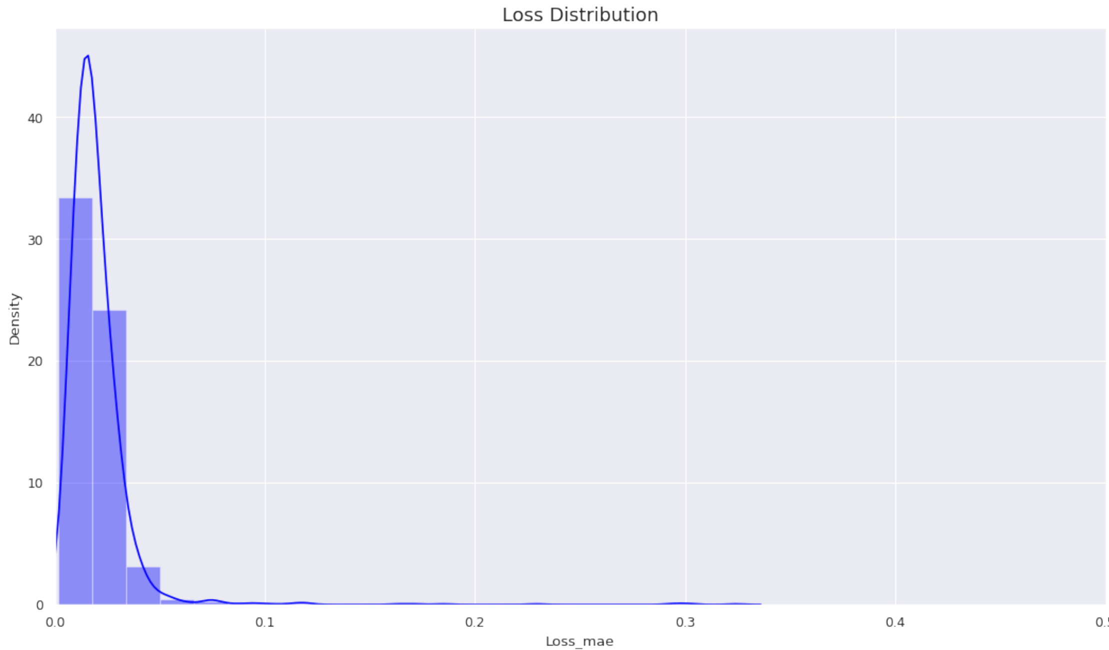
```
2090             0.759042                                     0.896537
2091             0.751957                                     0.935694

      X_Axis_RMS_Velocity  Z_Axis_High_Frequency_RMS_Acceleration  \
0                0.211240                                0.351147
1                0.212841                                0.353823
2                0.277096                                0.680479
3                0.273810                                0.724386
4                0.267967                                0.562863
…                   …                                      …
2087             0.258218                                0.827530
2088             0.261612                                0.810116
2089             0.257769                                0.830033
2090             0.259958                                0.818953
2091             0.253827                                0.848924

      Z_Axis_RMS_Velocity
0                0.119372
1                0.119559
2                0.099500
3                0.098109
4                0.101095
…                   …
2087             0.093171
2088             0.094180
2089             0.093042
2090             0.093688
2091             0.091894

[2092 rows x 5 columns]
```

[82]:
```
scored
```

[82]:
```
      Loss_mae
0     0.302087
1     0.171675
2     0.032380
3     0.021065
4     0.055460
…        …
2087  0.072945
2088  0.036859
2089  0.042484
2090  0.035839
2091  0.032497

[2092 rows x 1 columns]
```

[83]:
```
plt.figure(figsize=(16,9), dpi=80)
plt.title('Loss Distribution', fontsize=16)
sns.distplot(scored['Loss_mae'], bins = 20, kde= True, color = 'blue');
plt.xlim([0.0,.5])
```

[83]: (0.0, 0.5)



[87]:
```
# calculate the loss on the test set
X_pred = model.predict(X_test)
X_pred = X_pred.reshape(X_pred.shape[0], X_pred.shape[2])
X_pred = pd.DataFrame(X_pred, columns=test_set.columns)
X_pred.index = test_set.index

scored = pd.DataFrame(index=test_set.index)
Xtest = X_test.reshape(X_test.shape[0], X_test.shape[2])
scored['Loss_mae'] = np.mean(np.abs(X_pred-Xtest), axis = 1)
scored['Threshold'] = 0.05
scored['Anomaly'] = scored['Loss_mae'] > scored['Threshold']
scored.head()
```

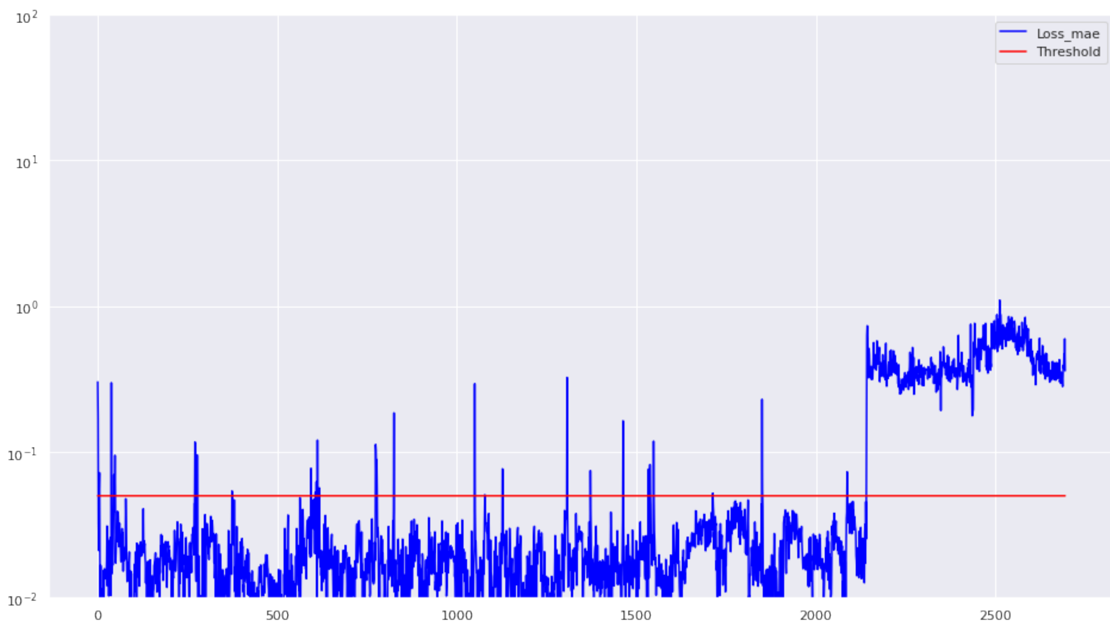
[87]:

	Loss_mae	Threshold	Anomaly
2093	0.040154	0.05	False
2094	0.027680	0.05	False
2095	0.044195	0.05	False
2096	0.042094	0.05	False
2097	0.033008	0.05	False

[88]:
```
# calculate the same metrics for the training set
# and merge all data in a single dataframe for plotting
X_pred_train = model.predict(X_train)
X_pred_train = X_pred_train.reshape(X_pred_train.shape[0], X_pred_train.
↪shape[2])
X_pred_train = pd.DataFrame(X_pred_train, columns=training_set.columns)
X_pred_train.index = training_set.index

scored_train = pd.DataFrame(index=training_set.index)
scored_train['Loss_mae'] = np.mean(np.abs(X_pred_train-Xtrain), axis = 1)
scored_train['Threshold'] = 0.05
scored_train['Anomaly'] = scored_train['Loss_mae'] > scored_train['Threshold']
scored = pd.concat([scored_train, scored])
```

[89]:
```
# plot bearing failure time plot
scored.plot(logy=True,  figsize=(16,9), ylim=[1e-2,1e2], color=['blue','red'])
```

[89]: <AxesSubplot:>



[ ]: