Bearing_Anomaly_Autoencoder_Model

April 25, 2021

[31]:
```
#from google.colab import files
#uploaded = files.upload()
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[32]:
```
# Common imports
import pandas as pd
import numpy as np
from sklearn import preprocessing
import seaborn as sns
sns.set(color_codes=True)
import matplotlib.pyplot as plt
```

[33]:
```
from numpy.random import seed
from tensorflow.random import set_seed
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras import regularizers
```

[34]:
```
merged_data = pd.read_excel('/content/drive/My Drive/Colab Notebooks/vibration.
↪xlsx',index_col = 0)
#merged_data = pd.read_excel('vibration.xlsx',index_col = 0)
```

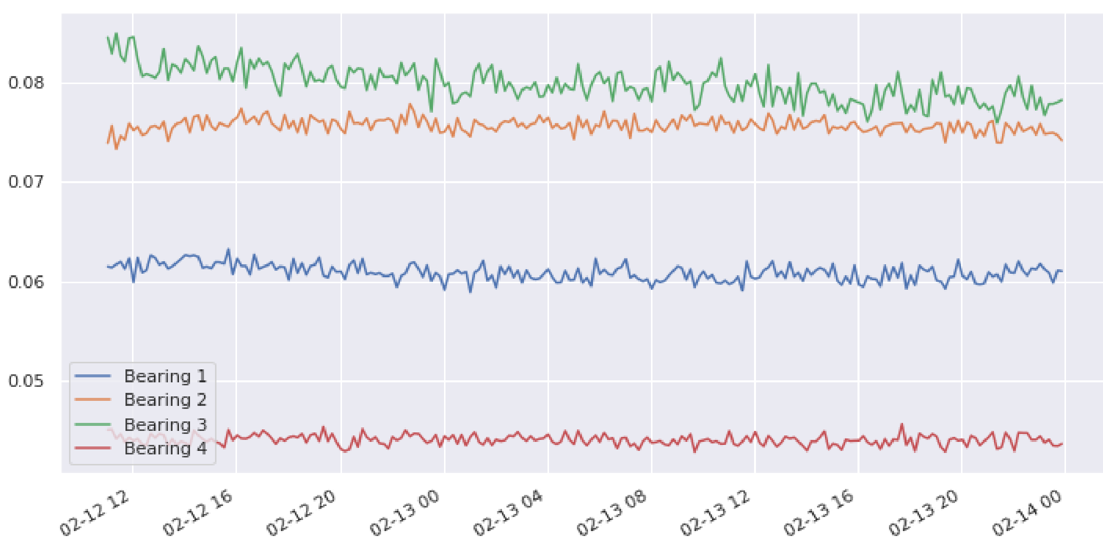
[35]:
```
#  Before setting up the models, we need to define train/test data. To do this,␣
↪we perform a
#  simple split where we train on the first part of the dataset (which should␣
↪represent normal
#  operating conditions), and test on the remaining parts of the dataset␣
↪leading up to the
#  bearing failure.
```

[36]:
```
merged_data
```

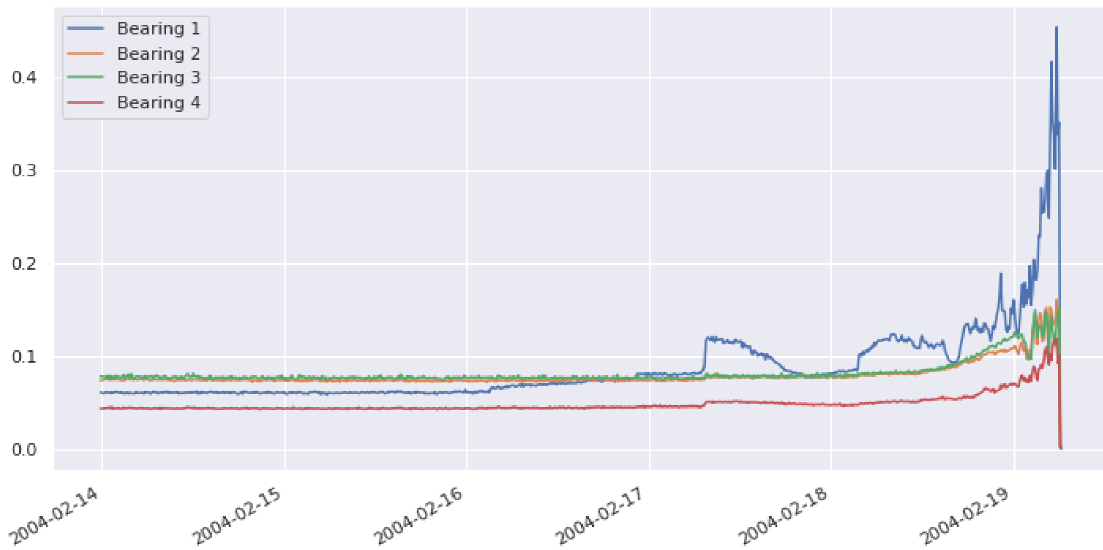
[36]:
```
                     Bearing 1  Bearing 2  Bearing 3  Bearing 4
2004-02-12 10:32:39   0.058333   0.071832   0.083242   0.043067
```

```
2004-02-12 10:42:39   0.058995   0.074006   0.084435   0.044541
2004-02-12 10:52:39   0.060236   0.074227   0.083926   0.044443
2004-02-12 11:02:39   0.061455   0.073844   0.084457   0.045081
2004-02-12 11:12:39   0.061361   0.075609   0.082837   0.045118
…                          …          …          …          …
2004-02-19 05:42:39   0.453335   0.161016   0.137440   0.119047
2004-02-19 05:52:39   0.337583   0.132400   0.144992   0.092125
2004-02-19 06:02:39   0.351111   0.152266   0.151299   0.100817
2004-02-19 06:12:39   0.001857   0.003732   0.003656   0.001786
2004-02-19 06:22:39   0.001168   0.000767   0.000716   0.001699

[984 rows x 4 columns]
```

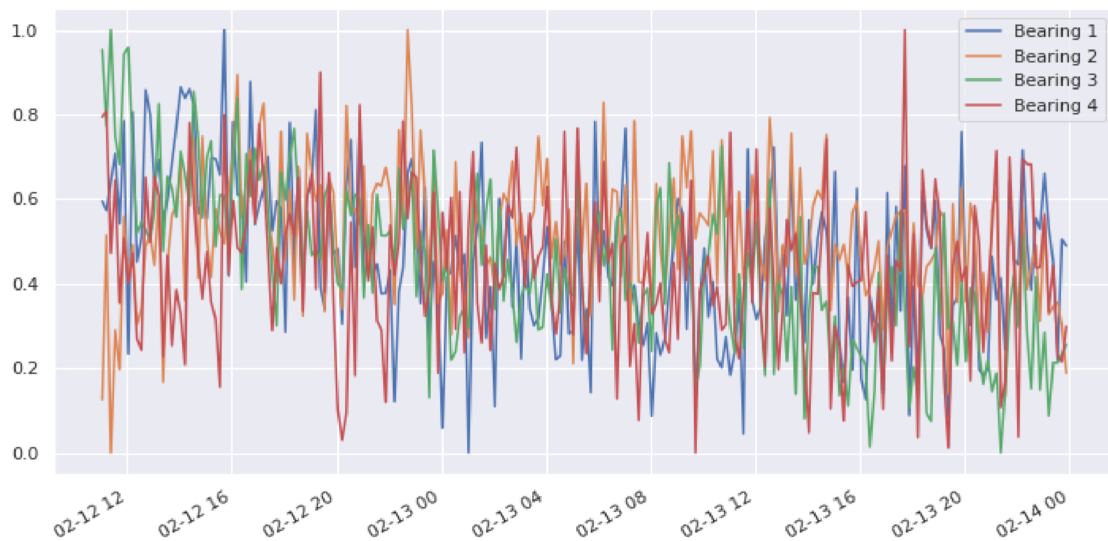
[37]:
```
#Define train
dataset_train = merged_data['2004-02-12 11:02:39':'2004-02-13 23:52:39']
dataset_train.plot(figsize = (12,6))
plt.show()
```



[38]:
```
#Define test data:
dataset_test = merged_data['2004-02-13 23:52:39':]
dataset_test.plot(figsize = (12,6))
plt.show()
```

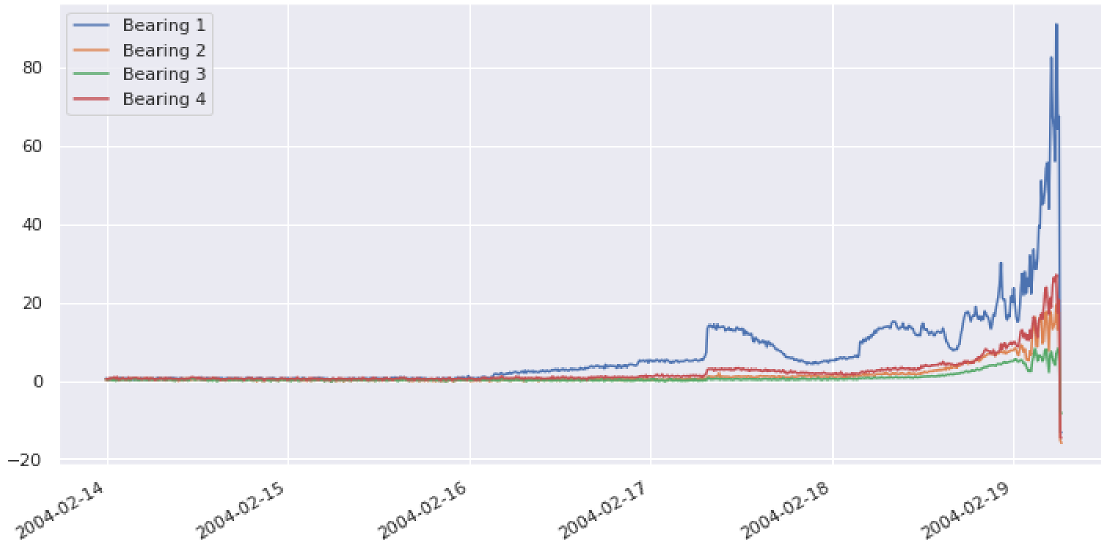


[39]:
```
#Normalize data:
scaler = preprocessing.MinMaxScaler()
X_train = pd.DataFrame(scaler.fit_transform(dataset_train),
                              columns=dataset_train.columns,
                              index=dataset_train.index)
X_train.plot(figsize = (12,6))
plt.show()
```



[40]:
```
# Random shuffle training data
X_train.sample(frac=1)
```

```
X_test = pd.DataFrame(scaler.transform(dataset_test),
                              columns=dataset_test.columns,
                              index=dataset_test.index)
X_test.plot(figsize = (12,6))
plt.show()
```



1 Defining the Autoencoder network:

We use a 3 layer neural network: First layer has 10 nodes, middle layer has 2 nodes, and third layer has 10 nodes. We use the mean square error as loss function, and train the model using the "Adam" optimizer.

[41]:
```
seed(10)
set_seed(10)
act_func = 'elu'
# Input layer:
model=Sequential()
# First hidden layer, connected to input vector X.
model.add(Dense(10,activation=act_func,
                kernel_initializer='glorot_uniform',
                kernel_regularizer=regularizers.l2(0.0),
                input_shape=(X_train.shape[1],)
               )
         )

model.add(Dense(2,activation=act_func,
                kernel_initializer='glorot_uniform'))
```

```
model.add(Dense(10,activation=act_func,
                kernel_initializer='glorot_uniform'))

model.add(Dense(X_train.shape[1],
                kernel_initializer='glorot_uniform'))

model.compile(loss='mse',optimizer='adam')
# Train model for 100 epochs, batch size of 10:
NUM_EPOCHS=100
BATCH_SIZE=10
```

2 Fitting the model:

To keep track of the accuracy during training, we use 5% of the training data for validation after each epoch (validation_split = 0.05)

[42]:
```
len(X_train)
```

[42]: 222

[43]:
```
history=model.fit(np.array(X_train),np.array(X_train),
                  batch_size=BATCH_SIZE,
                  epochs=NUM_EPOCHS,
                  validation_split=0.05,
                  verbose = 1)
```

```
Epoch 1/100
21/21 [==============================] - 1s 10ms/step - loss: 0.1806 - val_loss:
0.0879
Epoch 2/100
21/21 [==============================] - 0s 3ms/step - loss: 0.1220 - val_loss:
0.0590
Epoch 3/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0836 - val_loss:
0.0365
Epoch 4/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0526 - val_loss:
0.0248
Epoch 5/100
21/21 [==============================] - 0s 4ms/step - loss: 0.0322 - val_loss:
0.0212
Epoch 6/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0204 - val_loss:
0.0221
Epoch 7/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0182 - val_loss:
0.0233
```

```
Epoch 8/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0175 - val_loss:
0.0222
Epoch 9/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0188 - val_loss:
0.0218
Epoch 10/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0164 - val_loss:
0.0208
Epoch 11/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0161 - val_loss:
0.0199
Epoch 12/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0161 - val_loss:
0.0198
Epoch 13/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0156 - val_loss:
0.0192
Epoch 14/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0149 - val_loss:
0.0185
Epoch 15/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0137 - val_loss:
0.0184
Epoch 16/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0143 - val_loss:
0.0181
Epoch 17/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0152 - val_loss:
0.0175
Epoch 18/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0144 - val_loss:
0.0172
Epoch 19/100
21/21 [==============================] - 0s 4ms/step - loss: 0.0132 - val_loss:
0.0175
Epoch 20/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0132 - val_loss:
0.0161
Epoch 21/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0154 - val_loss:
0.0166
Epoch 22/100
21/21 [==============================] - 0s 4ms/step - loss: 0.0115 - val_loss:
0.0162
Epoch 23/100
21/21 [==============================] - 0s 4ms/step - loss: 0.0130 - val_loss:
0.0156
```

```
Epoch 24/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0123 - val_loss:
0.0157
Epoch 25/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0128 - val_loss:
0.0151
Epoch 26/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0121 - val_loss:
0.0155
Epoch 27/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0119 - val_loss:
0.0144
Epoch 28/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0122 - val_loss:
0.0149
Epoch 29/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0133 - val_loss:
0.0145
Epoch 30/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0118 - val_loss:
0.0141
Epoch 31/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0125 - val_loss:
0.0145
Epoch 32/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0114 - val_loss:
0.0139
Epoch 33/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0128 - val_loss:
0.0135
Epoch 34/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0119 - val_loss:
0.0140
Epoch 35/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0118 - val_loss:
0.0140
Epoch 36/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0105 - val_loss:
0.0140
Epoch 37/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0121 - val_loss:
0.0133
Epoch 38/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0108 - val_loss:
0.0132
Epoch 39/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0116 - val_loss:
0.0142
```

```
Epoch 40/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0123 - val_loss:
0.0131
Epoch 41/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0114 - val_loss:
0.0135
Epoch 42/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0116 - val_loss:
0.0135
Epoch 43/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0112 - val_loss:
0.0131
Epoch 44/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0121 - val_loss:
0.0133
Epoch 45/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0112 - val_loss:
0.0138
Epoch 46/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0109 - val_loss:
0.0135
Epoch 47/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0108 - val_loss:
0.0130
Epoch 48/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0110 - val_loss:
0.0141
Epoch 49/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0105 - val_loss:
0.0139
Epoch 50/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0118 - val_loss:
0.0131
Epoch 51/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0115 - val_loss:
0.0139
Epoch 52/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0093 - val_loss:
0.0141
Epoch 53/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0104 - val_loss:
0.0133
Epoch 54/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0098 - val_loss:
0.0140
Epoch 55/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0113 - val_loss:
0.0140
```

```
Epoch 56/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0106 - val_loss:
0.0139
Epoch 57/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0094 - val_loss:
0.0134
Epoch 58/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0115 - val_loss:
0.0141
Epoch 59/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0106 - val_loss:
0.0136
Epoch 60/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0115 - val_loss:
0.0138
Epoch 61/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0107 - val_loss:
0.0134
Epoch 62/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0094 - val_loss:
0.0142
Epoch 63/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0101 - val_loss:
0.0134
Epoch 64/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0096 - val_loss:
0.0142
Epoch 65/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0110 - val_loss:
0.0137
Epoch 66/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0120 - val_loss:
0.0134
Epoch 67/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0107 - val_loss:
0.0140
Epoch 68/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0104 - val_loss:
0.0149
Epoch 69/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0105 - val_loss:
0.0130
Epoch 70/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0093 - val_loss:
0.0135
Epoch 71/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0107 - val_loss:
0.0142
```

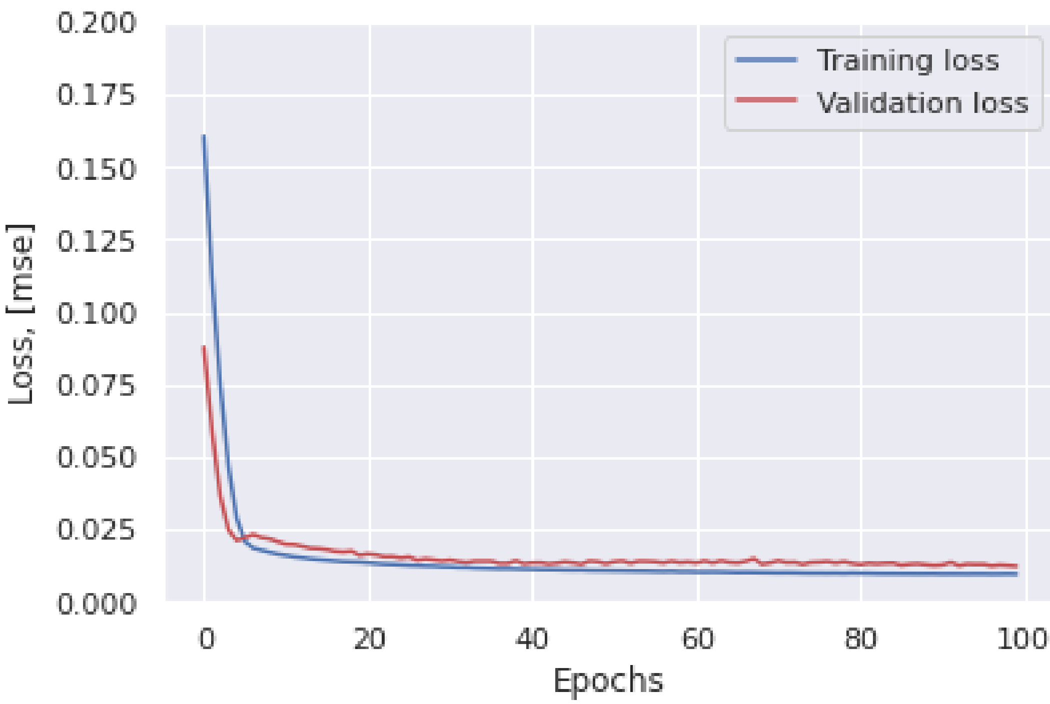
```
Epoch 72/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0095 - val_loss:
0.0136
Epoch 73/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0097 - val_loss:
0.0136
Epoch 74/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0093 - val_loss:
0.0130
Epoch 75/100
21/21 [==============================] - 0s 4ms/step - loss: 0.0096 - val_loss:
0.0136
Epoch 76/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0088 - val_loss:
0.0137
Epoch 77/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0112 - val_loss:
0.0139
Epoch 78/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0108 - val_loss:
0.0134
Epoch 79/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0118 - val_loss:
0.0139
Epoch 80/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0103 - val_loss:
0.0133
Epoch 81/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0086 - val_loss:
0.0129
Epoch 82/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0104 - val_loss:
0.0133
Epoch 83/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0100 - val_loss:
0.0131
Epoch 84/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0099 - val_loss:
0.0133
Epoch 85/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0093 - val_loss:
0.0134
Epoch 86/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0111 - val_loss:
0.0126
Epoch 87/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0097 - val_loss:
0.0130
```

```
Epoch 88/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0101 - val_loss:
0.0132
Epoch 89/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0097 - val_loss:
0.0128
Epoch 90/100
21/21 [==============================] - 0s 4ms/step - loss: 0.0094 - val_loss:
0.0126
Epoch 91/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0084 - val_loss:
0.0128
Epoch 92/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0093 - val_loss:
0.0136
Epoch 93/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0093 - val_loss:
0.0125
Epoch 94/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0098 - val_loss:
0.0131
Epoch 95/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0099 - val_loss:
0.0130
Epoch 96/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0109 - val_loss:
0.0130
Epoch 97/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0112 - val_loss:
0.0124
Epoch 98/100
21/21 [==============================] - 0s 4ms/step - loss: 0.0089 - val_loss:
0.0127
Epoch 99/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0099 - val_loss:
0.0125
Epoch 100/100
21/21 [==============================] - 0s 3ms/step - loss: 0.0083 - val_loss:
0.0123
```

3 Visualize training/validation loss:

[44]:
```
plt.plot(history.history['loss'],
         'b',
         label='Training loss')
plt.plot(history.history['val_loss'],
         'r',
```

```
        label='Validation loss')

plt.legend(loc='upper right')
plt.xlabel('Epochs')
plt.ylabel('Loss, [mse]')
plt.ylim([0,.2])
plt.show()
```



4 Distribution of loss function in the training set:

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. In doing this, one can make sure that this threshold is set above the "noise level", and that any flagged anomalies should be statistically significant above the noise background.

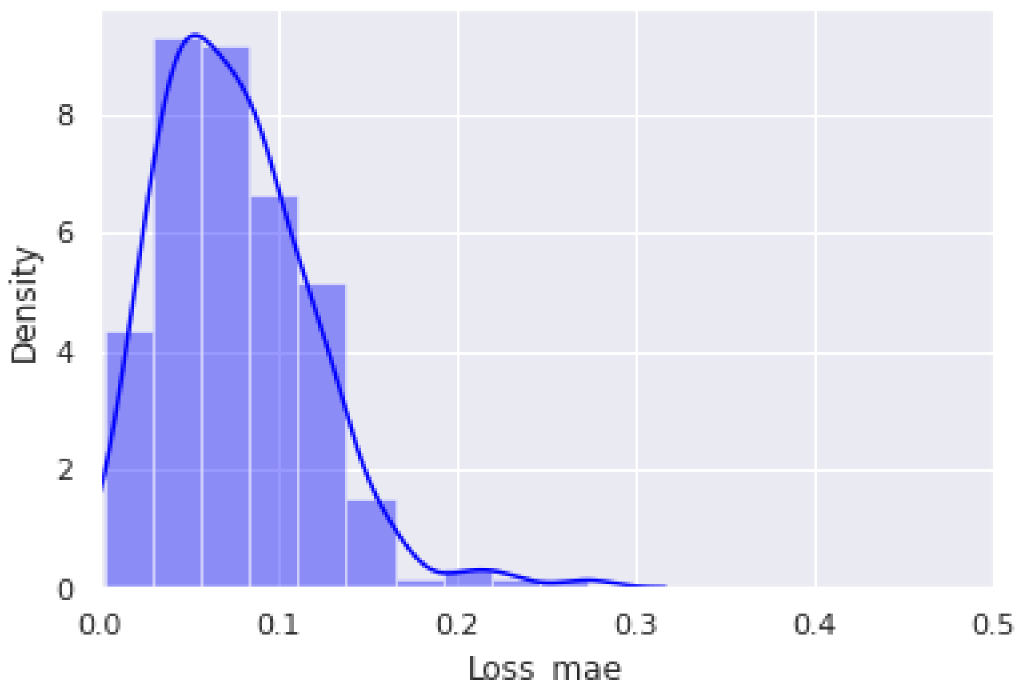
[45]:

```
X_pred = model.predict(np.array(X_train))
X_pred = pd.DataFrame(X_pred,
                      columns=X_train.columns)
X_pred.index = X_train.index

scored = pd.DataFrame(index=X_train.index)
scored['Loss_mae'] = np.mean(np.abs(X_pred-X_train), axis = 1)
plt.figure()
```

```python
sns.distplot(scored['Loss_mae'],
             bins = 10,
             kde= True,
             color = 'blue');
plt.xlim([0.0,.5])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[45]: (0.0, 0.5)



From the above loss distribution, let us try a threshold of 0.3 for flagging an anomaly. We can then calculate the loss in the test set, to check when the output crosses the anomaly threshold.

[46]:
```python
X_pred = model.predict(np.array(X_test))
X_pred = pd.DataFrame(X_pred,
                      columns=X_test.columns)
X_pred.index = X_test.index

scored = pd.DataFrame(index=X_test.index)
scored['Loss_mae'] = np.mean(np.abs(X_pred-X_test), axis = 1)
```

```
scored['Threshold'] = 0.3
scored['Anomaly'] = scored['Loss_mae'] > scored['Threshold']
scored.head()
```

[46]:

	Loss_mae	Threshold	Anomaly
2004-02-13 23:52:39	0.135230	0.3	False
2004-02-14 00:02:39	0.103883	0.3	False
2004-02-14 00:12:39	0.031544	0.3	False
2004-02-14 00:22:39	0.072124	0.3	False
2004-02-14 00:32:39	0.120766	0.3	False

We then calculate the same metrics also for the training set, and merge all data in a single dataframe:

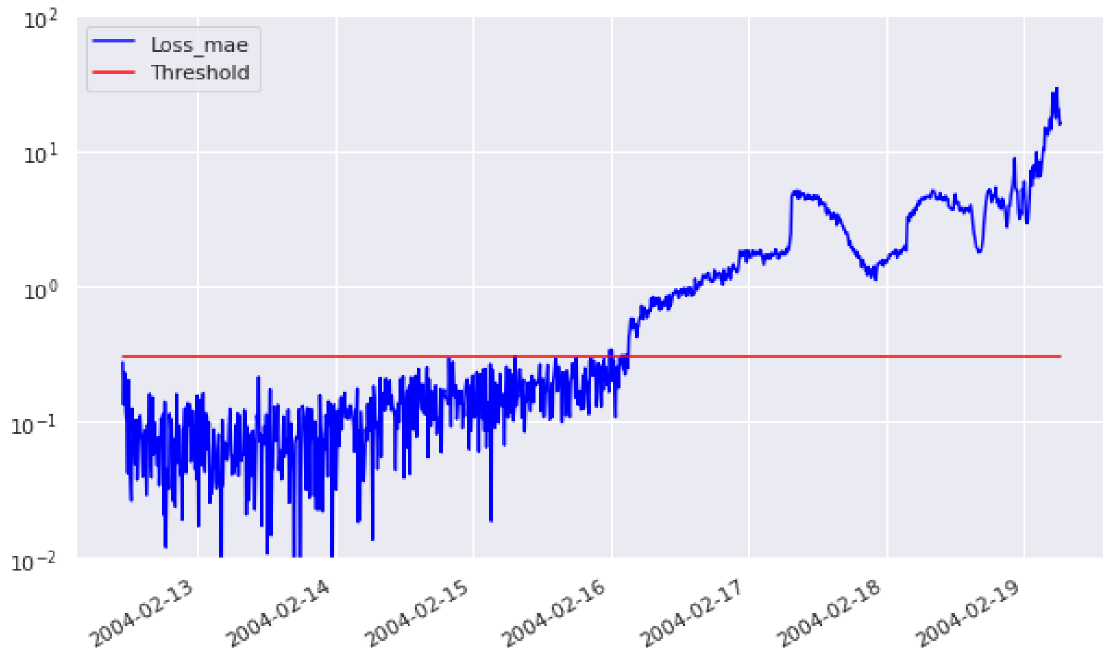
[47]:
```
X_pred_train = model.predict(np.array(X_train))
X_pred_train = pd.DataFrame(X_pred_train,
                            columns=X_train.columns)

X_pred_train.index = X_train.index
scored_train = pd.DataFrame(index=X_train.index)
scored_train['Loss_mae'] = np.mean(np.abs(X_pred_train-X_train),axis = 1)
scored_train['Threshold'] = 0.3
scored_train['Anomaly'] = scored_train['Loss_mae'] > scored_train['Threshold']

scored = pd.concat([scored_train, scored])
```

5 Results from Autoencoder model:

Having calculated the loss distribution and the anomaly threshold, we can visualize the model output in the time leading up to the bearing failure:

[48]:
```
scored.plot(logy=True,figsize = (10,6), ylim = [1e-2,1e2], color =␣
↪['blue','red'])
```

[48]: <matplotlib.axes._subplots.AxesSubplot at 0x7fab7c665110>



[49]:
```
scored
```

[49]:
```
                      Loss_mae  Threshold  Anomaly
2004-02-12 11:02:39   0.273986        0.3    False
2004-02-12 11:12:39   0.134584        0.3    False
2004-02-12 11:22:39   0.229360        0.3    False
2004-02-12 11:32:39   0.119207        0.3    False
2004-02-12 11:42:39   0.099732        0.3    False
…                          …          …        …
2004-02-19 05:42:39  29.001036        0.3     True
2004-02-19 05:52:39  20.075878        0.3     True
2004-02-19 06:02:39  20.135506        0.3     True
2004-02-19 06:12:39  15.617402        0.3     True
2004-02-19 06:22:39  16.083589        0.3     True

[982 rows x 3 columns]
```