# ADS_Capstone.model_deployment.V1.00

Carlos Granados

## Advanced Data Science - Capstone Project

### Model Deployment - V.1.0

For this project I decided to use solar power generation data uploaded by Ani Kannal in kaggle.com website. The idea is to predict the power generation given different weather conditions, as temperature and irradiation, and check the health of the devices.

#### 0. Install TensorFlow and Keras:

Usually `TensorFlow` and `keras` are not installed by default, so we need to install them first…

[1]:
```
!pip install tensorflow==2.2.0rc0
```

[2]:
```
# On labs.cognitiveclass.ai the V 2.4.3 must be installed
# On IBM Watson it is not necessary to specify
!pip install keras==2.4.3
```

#### 1. Load Libraries:

[3]:
```
# Standard python libraries
import numpy as np
import types
import datetime as dt
from scipy import stats
from numpy.random import seed
from math import ceil

# pandas
import pandas as pd

# Libraries to make plots and related
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

# For Anomaly Detection, using keras
from keras.models import load_model
from keras.losses import MSE, MSLE
from keras.callbacks import Callback
import tensorflow as tf
```

```python
import sklearn
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from joblib import dump, load

# Custom functions used through this notebook:
from ADS_Capstone_model_evaluation_Aux import *

# Custom functions defined in other part of the project and needed here:
from ADS_Capstone_model_train_Aux import divideTimeSteps, dataProcess
from ADS_Capstone_model_train_Aux import customNorm, create_trimmed_data_norm

# Others, required by IBM Watson
#import ibm_boto3
#from botocore.client import Config
```

```
Intel(R) Data Analytics Acceleration Library (Intel(R) DAAL) solvers for sklearn
enabled: https://intelpython.github.io/daal4py/sklearn.html
```

### 2. Re-Read and Check Data Frames

Before to start, lets repeat a couple of steps performed in the *Data Exploration* step.

Some cells are hidden, because they contain sensitive information, as some keys and passwords. The original data files, in CSV format, are already uploaded to IBM Watson project, and they are called from the notebook directly.

**Note:** When using IBM Watson, the following cells can be (uncommented and) executed in order to read de CSV files directly from the cloud. Otherwise the files must be loaded more "traditionally" (see below)

**Note:** Execute the following lines when running on cognitiveclass.ai or another platform…

[4]:
```python
# Path to CSV data files
#path_data = '/resources/Projects/AnomalyDect/SolarPanels/Data/'
path_data = './Data/'

# Read generation data for Plant 1
df_plant1_gen = pd.read_csv(path_data+'Plant_1_Generation_Data.csv')

# Read sensor data for Plant 1
df_plant1_sen = pd.read_csv(path_data+'Plant_1_Weather_Sensor_Data.csv')
```

[5]:
```python
# Read generation data for Plant 2
df_plant2_gen = pd.read_csv(path_data+'Plant_2_Generation_Data.csv')

# Read sensor data for Plant 2
df_plant2_sen = pd.read_csv(path_data+'Plant_2_Weather_Sensor_Data.csv')
```

We create now a huge function that makes all the ETL, data cleasing and feature engineering for us!

[6]:
```python
# Create DF with all times...
t = np.arange(0, 24, 0.25)
times = []
headers = ['TIME', 'DAY', 'MONTH']

# Add days for May
for day in range(15, 32):
    for i in range(96):
        times.append([t[i], day, 5])

# Add days for June
for day in range(1, 18):
    for i in range(96):
        times.append([t[i], day, 6])

times = np.array(times)
df_times = pd.DataFrame(data=times, columns=headers)
```

Let's extract the 7 first sources for the plant 1 (used for training):

[7]:
```python
nDay_p1 = 10
nMonth_p1 = 5
df1_all = dataProcess(df_plant1_gen, df_plant1_sen, df_times, 7, nDay=nDay_p1,␣
 ↪nMonth=nMonth_p1)
```

```
Extracting data for Source Key : bvBOhCH3iADSZry
Extracting data for Source Key : 1BY6WEcLGh8j5v7
Extracting data for Source Key : VHMLBKoKgIrUVDU
Extracting data for Source Key : 7JYdWkrLSPkdwr4
Extracting data for Source Key : ih0vzX44oOqAx2f
Extracting data for Source Key : ZnxXDlPa8U1GXgE
Extracting data for Source Key : z9Y9gH1T5YWrNuG
```

And now some additional data sources, for evaluate the model…

[8]:
```python
df1_new = dataProcess(df_plant1_gen, df_plant1_sen, df_times, 15, nDay=nDay_p1,␣
 ↪nMonth=nMonth_p1)
```

```
Extracting data for Source Key : bvBOhCH3iADSZry
Extracting data for Source Key : 1BY6WEcLGh8j5v7
Extracting data for Source Key : VHMLBKoKgIrUVDU
Extracting data for Source Key : 7JYdWkrLSPkdwr4
Extracting data for Source Key : ih0vzX44oOqAx2f
Extracting data for Source Key : ZnxXDlPa8U1GXgE
Extracting data for Source Key : z9Y9gH1T5YWrNuG
Extracting data for Source Key : wCURE6d3bPkepu2
Extracting data for Source Key : iCRJl6heRkivqQ3
Extracting data for Source Key : uHbuxQJl8lW7ozc
Extracting data for Source Key : pkci93gMrogZuBj
Extracting data for Source Key : rGa61gmuvPhdLxV
Extracting data for Source Key : sjndEbLyjtCKgGv
Extracting data for Source Key : zVJPv84UY57bAof
Extracting data for Source Key : McdE0feGgRqW7Ca
```

And for the 7 first sources for the plant 2:

[9]:
```python
df1_eval = genTest(df1_all, df1_new, 8.00, nDay_p1, nMonth_p1,
                   n0=5, n1=5, x0=0.65, verbose=0)
```

[10]:
```
df1_eval[0].describe()
```

[10]:

| | TIME | DAY | MONTH | AMB_TEMP | MOD_TEMP | IRRADIATION | AC_POWER | DC_POWER |
|---|---|---|---|---|---|---|---|---|
| count | 16.000000 | 16.0 | 16.0 | 16.000000 | 16.000000 | 16.000000 | 16.000000 | 16.000000 |
| mean | 9.875000 | 17.0 | 5.0 | 29.142192 | 49.687791 | 0.676743 | 809.268304 | 8274.642857 |
| std | 1.190238 | 0.0 | 0.0 | 2.364151 | 8.262481 | 0.203506 | 218.635341 | 2246.655085 |
| min | 8.000000 | 17.0 | 5.0 | 25.061761 | 34.632715 | 0.378702 | 473.971429 | 4832.000000 |
| 25% | 8.937500 | 17.0 | 5.0 | 27.520126 | 44.952106 | 0.531785 | 642.169643 | 6555.700893 |
| 50% | 9.875000 | 17.0 | 5.0 | 29.630904 | 49.436558 | 0.701713 | 880.883036 | 9004.598215 |
| 75% | 10.812500 | 17.0 | 5.0 | 31.091784 | 55.328591 | 0.823069 | 1003.557143 | 10275.142855 |
| max | 11.750000 | 17.0 | 5.0 | 32.527864 | 63.145582 | 0.997904 | 1093.014286 | 11185.428570 |

[11]:
```
df1_eval[1].describe()
```

[11]:

| | TIME | DAY | MONTH | AMB_TEMP | MOD_TEMP | IRRADIATION | AC_POWER | DC_POWER |
|---|---|---|---|---|---|---|---|---|
| count | 16.000000 | 16.0 | 16.0 | 16.000000 | 16.000000 | 16.000000 | 16.000000 | 16.000000 |
| mean | 9.875000 | 17.0 | 5.0 | 29.142192 | 49.687791 | 0.676743 | 526.024397 | 5378.517857 |
| std | 1.190238 | 0.0 | 0.0 | 2.364151 | 8.262481 | 0.203506 | 142.112972 | 1460.325805 |
| min | 8.000000 | 17.0 | 5.0 | 25.061761 | 34.632715 | 0.378702 | 308.081429 | 3140.800000 |
| 25% | 8.937500 | 17.0 | 5.0 | 27.520126 | 44.952106 | 0.531785 | 417.410268 | 4261.205580 |
| 50% | 9.875000 | 17.0 | 5.0 | 29.630904 | 49.436558 | 0.701713 | 572.573973 | 5852.988839 |
| 75% | 10.812500 | 17.0 | 5.0 | 31.091784 | 55.328591 | 0.823069 | 652.312143 | 6678.842856 |
| max | 11.750000 | 17.0 | 5.0 | 32.527864 | 63.145582 | 0.997904 | 710.459286 | 7270.528571 |

### 3. Deep Learning Model

In the model definition phase we explored different deep learning models, and the one that had the best behavior were a sequential one, with several layers. In the training phase such model were used and saved. Here we load such model and use it in two stages: 1.) to evaluate how good the model is and 2.) using it when "broken" data is used to verify how the losses change and then to identify a possible problem in the solar panels.

The FFT of the data is not considered, since several of the time parameters (such as time and day and month) are needed to build up the model.

Function to detect an anomaly, based on the changes of the loss function. A warning is issued if a change larger than the `l_min` value is detected. In the training phase we see that such values are arond `l_min = 0.075`.

[33]:
```
def checkVal(score, indx=0):
    """
    Function to check the Anomaly boolean value in score DF
    """
    s0 = score['Anomaly'].unique()
    if True in s0:
        print('Warning! Possible anomaly detected in entry {}'.format(indx))

def checkPanel(df_test, df_eval, time_steps, batch_size, model, scaler,
               lossFun='mse', l_min=0.1):
    """
    Function designed to give warnings, when the loss function is larger than a␣
 ↪given value
    df_test : DF, using for the training of model
    df_eval : DF, used to test the validity of model
    model   : previusly trained model, using df_test
    loss    : (str) loss function to use. 'MSE' by default
    """
    # Check loss with data used during the training phase
    score_all = {}
    icount = 0
    keys = list(df_test.keys())
    i_min = min(keys)
    for i in df_test:
        df = df_test[i]
        if i == i_min:
            score = predictModel(df, model, create_trimmed_data_norm,
                                 scaler, time_steps, batch_size,
                                 lossFun=lossFun, l_min=l_min)
            checkVal(score)
        else:
            score_loop = predictModel(df, model, create_trimmed_data_norm,
                                      scaler, time_steps, batch_size,
                                      lossFun=lossFun, l_min=l_min)
            checkVal(score_loop, indx=-i)
            score = pd.concat([score_loop, score])
    score_all[icount] = score
    icount += 1
    # Check the loss for each validation data
    loss_val = {}
    for i in df_eval:
        df = df_eval[i]
        score = predictModel(df, model, create_trimmed_data_norm,
                             scaler, time_steps, batch_size,
                             lossFun=lossFun, l_min=l_min)
        score_all[icount] = score
        loss_val[i] = score
        icount += 1
        # Check if there is some anomalies (loss values over the threshold)
        checkVal(score, indx=i)
    return [score_all, loss_val]
```

**3.2 Deep Learning Model:** We start demonstraring how the deep learming model performs.

We load the model for the solar panel 1:

[13]:
```
model1 = load_model('./models/ADS_Capstone.solar_panel_1.model.h5')
```

```
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your
model is starting with a freshly initialized optimizer.
```

Let's check the summary:

[14]:
```
model1.summary()
```

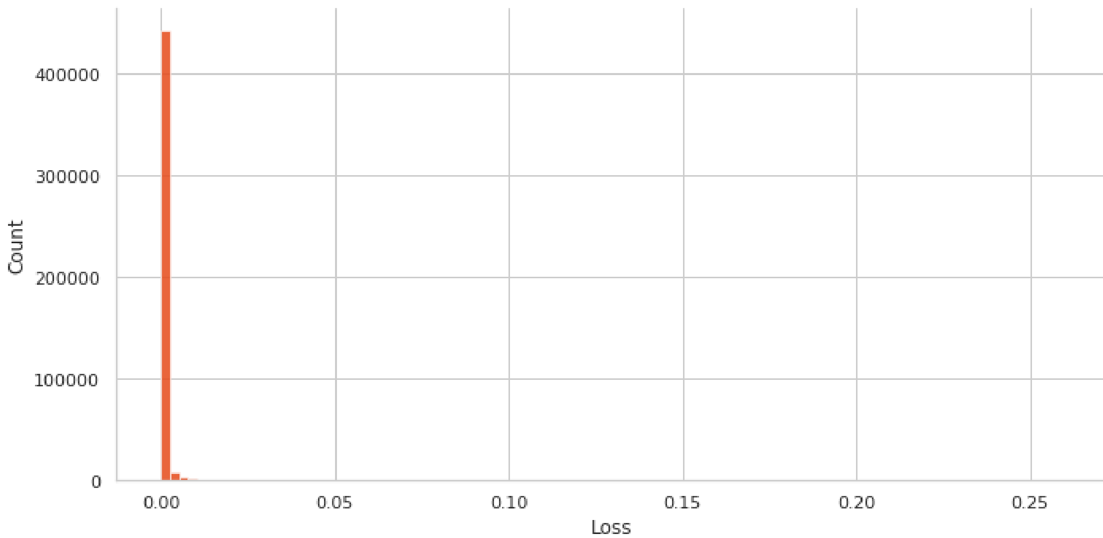
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 16, 20)            2320
_________________________________________________________________
lstm_1 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_2 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_3 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_4 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_5 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_6 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_7 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_8 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_9 (LSTM)                (None, 16, 20)            3280
_________________________________________________________________
lstm_10 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_11 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_12 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_13 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_14 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_15 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_16 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_17 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_18 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
lstm_19 (LSTM)               (None, 16, 20)            3280
_________________________________________________________________
dense (Dense)                (None, 16, 8)             168
=================================================================
Total params: 64,808
```

```
Trainable params: 64,808
Non-trainable params: 0
_________________________________________________________________
```

And the distribution of the losses during the training phase:

[159]:
```
# Distribution plot

distLoss(lossAll_m1, lossFun, optFun, bins=100)
```




For some of the data using during the training, we obtained the scores:

[21]:
```
time_steps = 16       # Equivalent to obtained observations in a 4 hour period
dim = 8               # Number of considered parameters
batch_size = 4       # Best batch:size obtained in evaluation of the model

# Initialize scaler
scaler1 = customNorm(df_plant1_gen, df_plant1_sen)

df1_check = {i:df1_all[i] for i in range(20, 30)}

# Check loss for data used in training
scoreAll(df1_check, time_steps, scaler1, model1)
```

```
df : 20
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean       0.060608
std        0.026950
min        0.040953
25%        0.042169
50%        0.043486
75%        0.085870
max        0.116305
Name: Loss, dtype: float64
```

```
--------------------------

df : 21
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.141175
std       0.058864
min       0.069155
25%       0.115781
50%       0.122548
75%       0.146420
max       0.301124
Name: Loss, dtype: float64
--------------------------

df : 22
1/1 [==============================] - 0s 875us/step
Loss : count    16.000000
mean      0.018987
std       0.016057
min       0.010190
25%       0.010383
50%       0.011392
75%       0.018661
max       0.069095
Name: Loss, dtype: float64
--------------------------

df : 23
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.006671
std       0.000390
min       0.006379
25%       0.006409
50%       0.006445
75%       0.006757
max       0.007433
Name: Loss, dtype: float64
--------------------------

df : 24
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.092147
std       0.000973
min       0.089636
25%       0.091836
50%       0.092483
75%       0.092690
max       0.093382
Name: Loss, dtype: float64
--------------------------

df : 25
1/1 [==============================] - 0s 897us/step
Loss : count    16.000000
mean      0.064032
std       0.005408
```

```
min       0.060496
25%       0.061086
50%       0.061168
75%       0.063725
max       0.077745
Name: Loss, dtype: float64
---------------------------

df : 26
1/1 [==============================] - 0s 991us/step
Loss : count    16.000000
mean      0.104553
std       0.046733
min       0.042272
25%       0.070122
50%       0.093347
75%       0.138420
max       0.217617
Name: Loss, dtype: float64
---------------------------

df : 27
1/1 [==============================] - 0s 2ms/step
Loss : count    16.000000
mean      0.143048
std       0.050270
min       0.076619
25%       0.092595
50%       0.139375
75%       0.186537
max       0.230456
Name: Loss, dtype: float64
---------------------------

df : 28
1/1 [==============================] - 0s 892us/step
Loss : count    16.000000
mean      0.021309
std       0.021769
min       0.007801
25%       0.008299
50%       0.010020
75%       0.023557
max       0.078250
Name: Loss, dtype: float64
---------------------------

df : 29
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.005114
std       0.000474
min       0.004705
25%       0.004732
50%       0.004813
75%       0.005588
max       0.005859
Name: Loss, dtype: float64
---------------------------
```

And for the new data sets:

[22]:
```
scoreAll(df1_eval, time_steps, scaler1, model1)
```

```
df : 0
1/1 [==============================] - 0s 857us/step
Loss : count    16.000000
mean      0.193208
std       0.080922
min       0.083423
25%       0.127732
50%       0.208926
75%       0.266618
max       0.314830
Name: Loss, dtype: float64
--------------------------

df : 1
1/1 [==============================] - 0s 2ms/step
Loss : count    16.000000
mean      0.142593
std       0.056866
min       0.067230
25%       0.095852
50%       0.150062
75%       0.189677
max       0.234648
Name: Loss, dtype: float64
--------------------------

df : 2
1/1 [==============================] - 0s 989us/step
Loss : count    16.000000
mean      0.111243
std       0.032005
min       0.061529
25%       0.081044
50%       0.113519
75%       0.134990
max       0.165394
Name: Loss, dtype: float64
--------------------------

df : 3
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.024283
std       0.019957
min       0.012876
25%       0.013078
50%       0.014329
75%       0.023808
max       0.081600
Name: Loss, dtype: float64
--------------------------

df : 4
```

```
1/1 [==============================] - 0s 2ms/step
Loss : count    16.000000
mean      0.131860
std       0.040735
min       0.069823
25%       0.098656
50%       0.135372
75%       0.157068
max       0.205550
Name: Loss, dtype: float64
---------------------------

df : 5
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.099490
std       0.026498
min       0.059131
25%       0.077830
50%       0.101716
75%       0.117040
max       0.144756
Name: Loss, dtype: float64
---------------------------

df : 6
1/1 [==============================] - 0s 942us/step
Loss : count    16.000000
mean      0.102793
std       0.029551
min       0.053753
25%       0.084892
50%       0.096269
75%       0.116759
max       0.163376
Name: Loss, dtype: float64
---------------------------

df : 7
1/1 [==============================] - 0s 952us/step
Loss : count    16.000000
mean      0.021995
std       0.006756
min       0.016505
25%       0.017261
50%       0.020002
75%       0.024241
max       0.042670
Name: Loss, dtype: float64
---------------------------

df : 8
1/1 [==============================] - 0s 890us/step
Loss : count    16.000000
mean      0.115856
std       0.037331
min       0.059953
25%       0.082463
50%       0.118387
```

```
75%       0.143799
max       0.170699
Name: Loss, dtype: float64
---------------------------

df : 9
1/1 [==============================] - 0s 954us/step
Loss : count    16.000000
mean      0.094470
std       0.026085
min       0.054029
25%       0.071414
50%       0.097763
75%       0.118694
max       0.127197
Name: Loss, dtype: float64
---------------------------

df : 10
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.109867
std       0.031200
min       0.072631
25%       0.086699
50%       0.102257
75%       0.126653
max       0.189662
Name: Loss, dtype: float64
---------------------------

df : 11
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.031266
std       0.016252
min       0.016538
25%       0.019139
50%       0.024422
75%       0.035788
max       0.063860
Name: Loss, dtype: float64
---------------------------

df : 12
1/1 [==============================] - 0s 903us/step
Loss : count    16.000000
mean      0.204474
std       0.082771
min       0.056451
25%       0.138548
50%       0.219305
75%       0.276737
max       0.307463
Name: Loss, dtype: float64
---------------------------

df : 13
1/1 [==============================] - 0s 2ms/step
```

```
Loss : count    16.000000
mean      0.146309
std       0.057377
min       0.046064
25%       0.099332
50%       0.155576
75%       0.197119
max       0.219728
Name: Loss, dtype: float64
---------------------------

df : 14
1/1 [==============================] - 0s 906us/step
Loss : count    16.000000
mean      0.162373
std       0.030192
min       0.100857
25%       0.146161
50%       0.172883
75%       0.186585
max       0.194846
Name: Loss, dtype: float64
---------------------------

df : 15
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.017840
std       0.029840
min       0.004762
25%       0.005470
50%       0.008568
75%       0.010103
max       0.122665
Name: Loss, dtype: float64
---------------------------

df : 16
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.191893
std       0.075415
min       0.077771
25%       0.126650
50%       0.199150
75%       0.245886
max       0.302711
Name: Loss, dtype: float64
---------------------------

df : 17
1/1 [==============================] - 0s 963us/step
Loss : count    16.000000
mean      0.136372
std       0.050499
min       0.061568
25%       0.091799
50%       0.140846
75%       0.172534
```

```
max       0.211020
Name: Loss, dtype: float64
---------------------------

df : 18
1/1 [==============================] - 0s 2ms/step
Loss : count    16.000000
mean      0.114602
std       0.060317
min       0.063965
25%       0.067991
50%       0.072852
75%       0.178306
max       0.201588
Name: Loss, dtype: float64
---------------------------

df : 19
1/1 [==============================] - 0s 922us/step
Loss : count    16.000000
mean      0.034999
std       0.032958
min       0.007795
25%       0.008557
50%       0.013427
75%       0.073418
max       0.078806
Name: Loss, dtype: float64
---------------------------

df : 20
1/1 [==============================] - 0s 877us/step
Loss : count    16.000000
mean      0.193208
std       0.080922
min       0.083423
25%       0.127732
50%       0.208926
75%       0.266618
max       0.314830
Name: Loss, dtype: float64
---------------------------

df : 21
1/1 [==============================] - 0s 971us/step
Loss : count    16.000000
mean      0.142593
std       0.056866
min       0.067230
25%       0.095852
50%       0.150062
75%       0.189677
max       0.234648
Name: Loss, dtype: float64
---------------------------

df : 22
1/1 [==============================] - 0s 961us/step
Loss : count    16.000000
```

```
mean      0.111243
std       0.032005
min       0.061529
25%       0.081044
50%       0.113519
75%       0.134990
max       0.165394
Name: Loss, dtype: float64
---------------------------

df : 23
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.022719
std       0.017491
min       0.012876
25%       0.013078
50%       0.014372
75%       0.023116
max       0.078030
Name: Loss, dtype: float64
---------------------------

df : 24
1/1 [==============================] - 0s 2ms/step
Loss : count    16.000000
mean      0.186726
std       0.085300
min       0.058819
25%       0.122544
50%       0.175410
75%       0.230584
max       0.340056
Name: Loss, dtype: float64
---------------------------

df : 25
1/1 [==============================] - 0s 2ms/step
Loss : count    16.000000
mean      0.135607
std       0.057137
min       0.052367
25%       0.091010
50%       0.129641
75%       0.159996
max       0.239053
Name: Loss, dtype: float64
---------------------------

df : 26
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.122260
std       0.039341
min       0.061871
25%       0.087704
50%       0.129108
75%       0.152790
max       0.189020
```

```
Name: Loss, dtype: float64
---------------------------

df : 27
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.013561
std       0.011765
min       0.006658
25%       0.007280
50%       0.008386
75%       0.010391
max       0.043468
Name: Loss, dtype: float64
---------------------------

df : 28
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.060608
std       0.026950
min       0.040953
25%       0.042169
50%       0.043486
75%       0.085870
max       0.116305
Name: Loss, dtype: float64
---------------------------

df : 29
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.052207
std       0.017072
min       0.039904
25%       0.040371
50%       0.041379
75%       0.068445
max       0.087815
Name: Loss, dtype: float64
---------------------------

df : 30
1/1 [==============================] - 0s 7ms/step
Loss : count    16.000000
mean      0.097512
std       0.036955
min       0.054596
25%       0.080625
50%       0.085406
75%       0.100189
max       0.199640
Name: Loss, dtype: float64
---------------------------

df : 31
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.018600
```

```
std       0.015890
min       0.010190
25%       0.010383
50%       0.011252
75%       0.017349
max       0.069095
Name: Loss, dtype: float64
---------------------------

df : 32
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.186726
std       0.085300
min       0.058819
25%       0.122544
50%       0.175410
75%       0.230584
max       0.340056
Name: Loss, dtype: float64
---------------------------

df : 33
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.135607
std       0.057137
min       0.052367
25%       0.091010
50%       0.129641
75%       0.159996
max       0.239053
Name: Loss, dtype: float64
---------------------------

df : 34
1/1 [==============================] - 0s 3ms/step
Loss : count    16.000000
mean      0.122260
std       0.039341
min       0.061871
25%       0.087704
50%       0.129108
75%       0.152790
max       0.189020
Name: Loss, dtype: float64
---------------------------

df : 35
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.013846
std       0.012041
min       0.006658
25%       0.007280
50%       0.008398
75%       0.010958
max       0.043468
Name: Loss, dtype: float64
```

```
---------------------------

df : 36
1/1 [==============================] - 0s 966us/step
Loss : count    16.000000
mean      0.204474
std       0.082771
min       0.056451
25%       0.138548
50%       0.219305
75%       0.276737
max       0.307463
Name: Loss, dtype: float64
---------------------------

df : 37
1/1 [==============================] - 0s 945us/step
Loss : count    16.000000
mean      0.146309
std       0.057377
min       0.046064
25%       0.099332
50%       0.155576
75%       0.197119
max       0.219728
Name: Loss, dtype: float64
---------------------------

df : 38
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.162373
std       0.030192
min       0.100857
25%       0.146161
50%       0.172883
75%       0.186585
max       0.194846
Name: Loss, dtype: float64
---------------------------

df : 39
1/1 [==============================] - 0s 1ms/step
Loss : count    16.000000
mean      0.017861
std       0.029719
min       0.004762
25%       0.005470
50%       0.008502
75%       0.010299
max       0.122665
Name: Loss, dtype: float64
---------------------------
```

Let's check if some issues are printed!

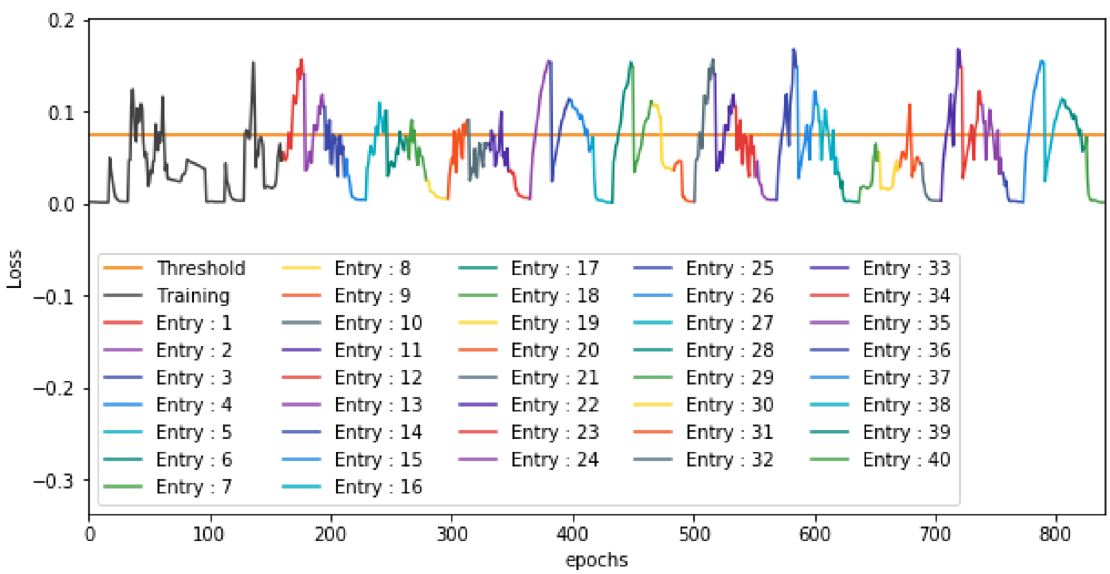
[34]:

```python
[score, loss_eval] = checkPanel(df1_check, df1_eval, time_steps,
                                batch_size, model1, scaler1,
                                lossFun='msle', l_min=0.075)
```

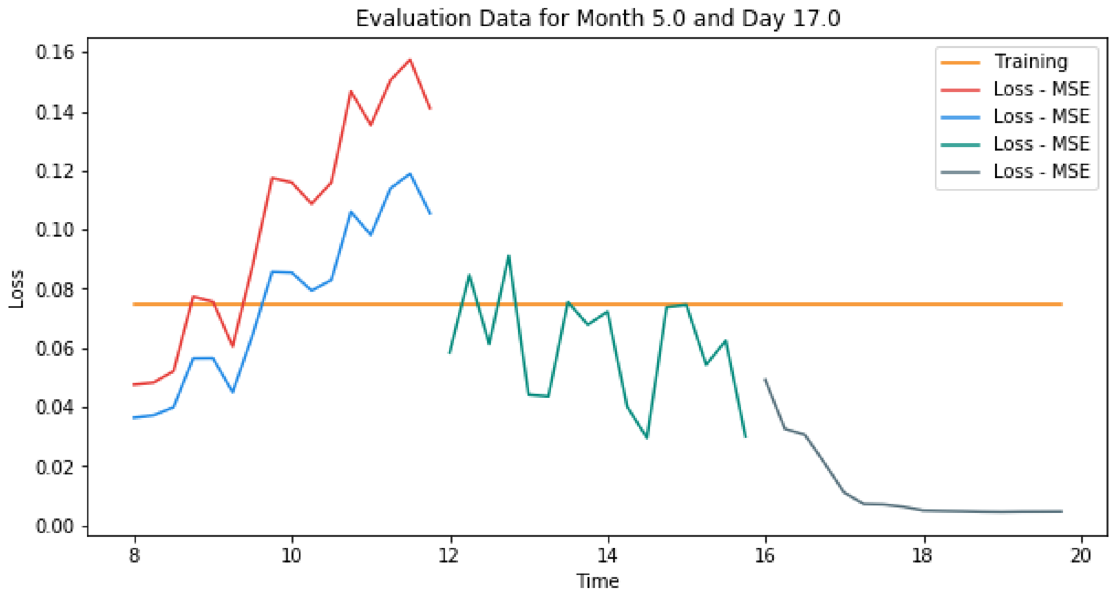
```
1/1 [==============================] - 0s 6ms/step
1/1 [==============================] - 0s 957us/step
Warning! Possible anomaly detected in entry -21
1/1 [==============================] - 0s 887us/step
1/1 [==============================] - 0s 989us/step
1/1 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry -26
1/1 [==============================] - 0s 964us/step
Warning! Possible anomaly detected in entry -27
1/1 [==============================] - 0s 949us/step
1/1 [==============================] - 0s 939us/step
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 0
1/1 [==============================] - 0s 2ms/step
Warning! Possible anomaly detected in entry 1
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 2
1/1 [==============================] - 0s 2ms/step
1/1 [==============================] - 0s 2ms/step
Warning! Possible anomaly detected in entry 4
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 5
1/1 [==============================] - 0s 5ms/step
Warning! Possible anomaly detected in entry 6
1/1 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 8
1/1 [==============================] - 0s 5ms/step
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 10
1/1 [==============================] - 0s 990us/step
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 12
1/1 [==============================] - 0s 877us/step
Warning! Possible anomaly detected in entry 13
1/1 [==============================] - 0s 2ms/step
Warning! Possible anomaly detected in entry 14
1/1 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 16
1/1 [==============================] - 0s 2ms/step
Warning! Possible anomaly detected in entry 17
1/1 [==============================] - 0s 3ms/step
Warning! Possible anomaly detected in entry 18
1/1 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 950us/step
Warning! Possible anomaly detected in entry 20
1/1 [==============================] - 0s 2ms/step
Warning! Possible anomaly detected in entry 21
1/1 [==============================] - 0s 915us/step
Warning! Possible anomaly detected in entry 22
1/1 [==============================] - 0s 4ms/step
```

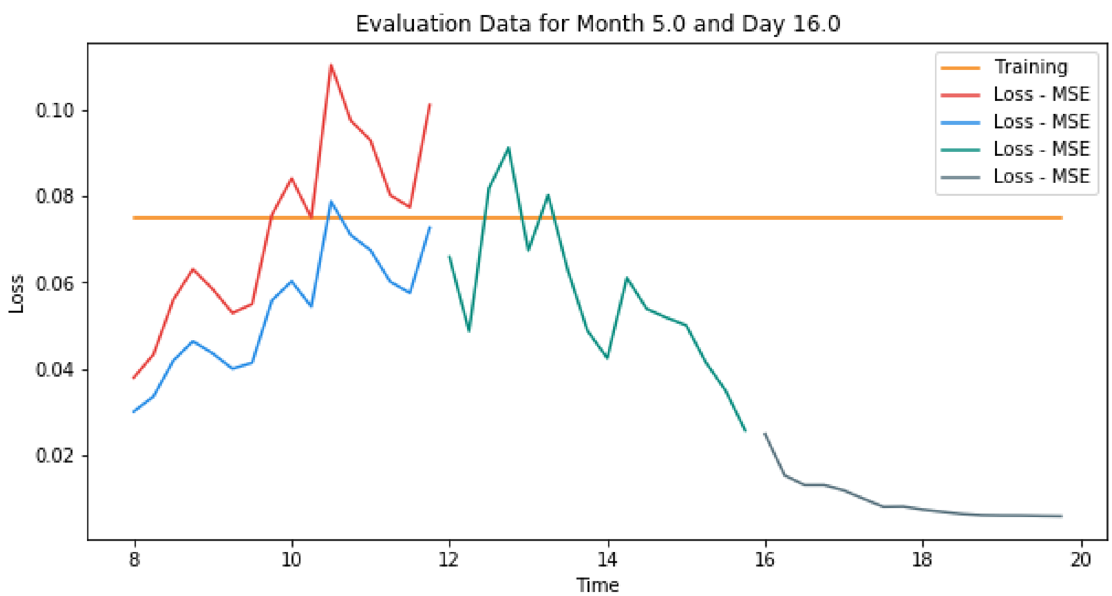
```
1/1 [==============================] - 0s 4ms/step
Warning! Possible anomaly detected in entry 24
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 25
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 26
1/1 [==============================] - 0s 970us/step
1/1 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 984us/step
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 30
1/1 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 32
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 33
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 34
1/1 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 923us/step
Warning! Possible anomaly detected in entry 36
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 37
1/1 [==============================] - 0s 1ms/step
Warning! Possible anomaly detected in entry 38
1/1 [==============================] - 0s 1ms/step
```

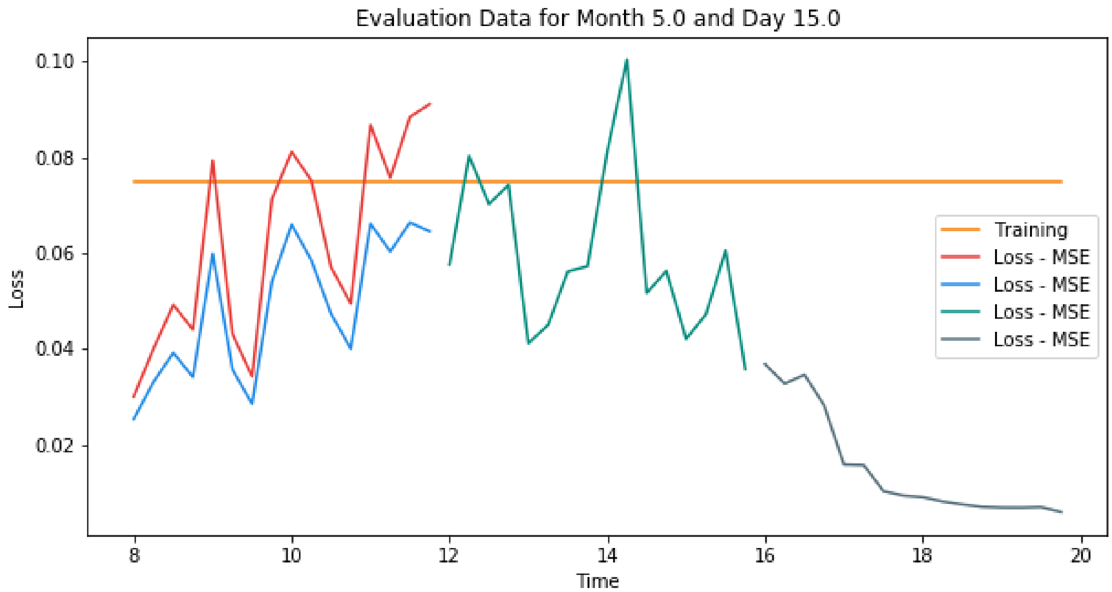
[36]:
```
plotLosses(score, c_xmin=0.0, c_ymin=-2.0, c_ymax=1.2)
```

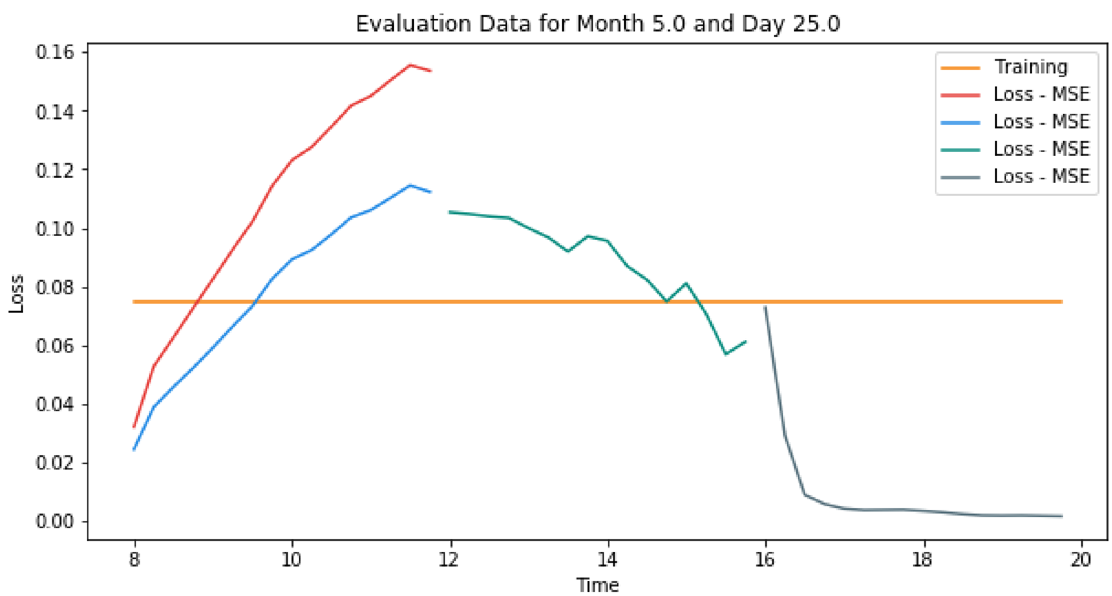


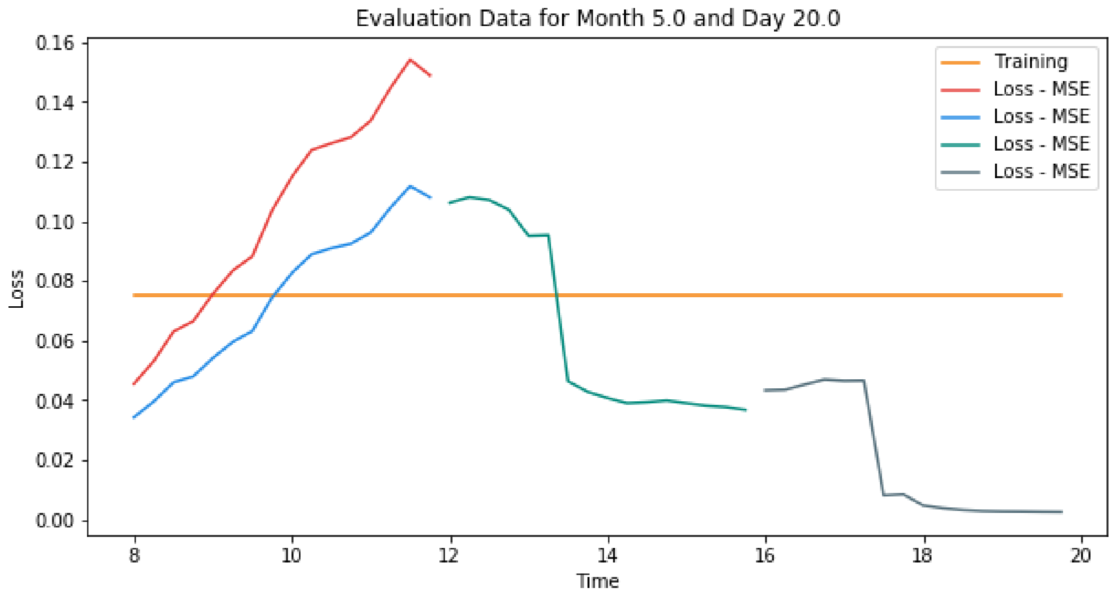

[37]:
```
plotTimeLosses(df1_eval, loss_eval)
```

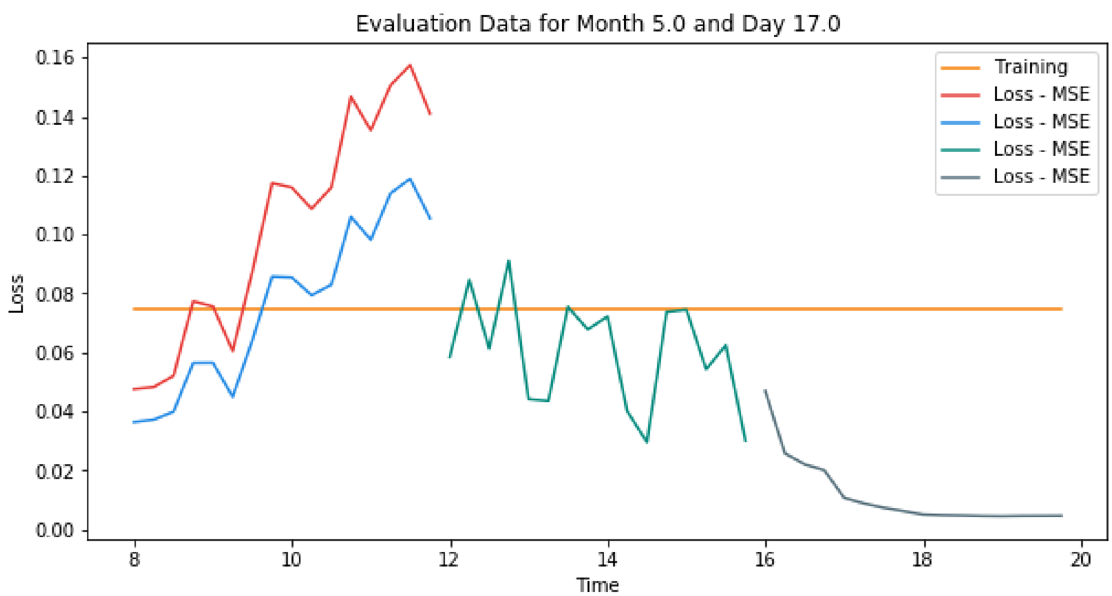

Evaluation Data for Month 5.0 and Day 17.0
Training
Loss - MSE
Loss - MSE
Loss - MSE
Loss - MSE
Loss
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
8
10
12
14
16
18
20
Time

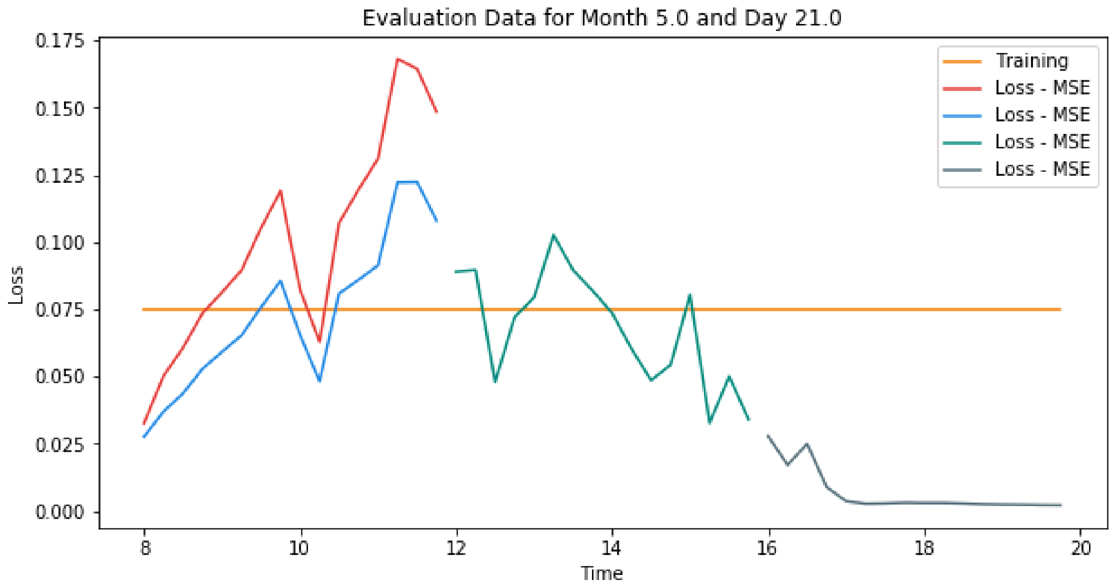

Evaluation Data for Month 5.0 and Day 16.0
Training
Loss - MSE
Loss - MSE
Loss - MSE
Loss - MSE
Loss
0.10
0.08
0.06
0.04
0.02
8
10
12
14
16
18
20
Time

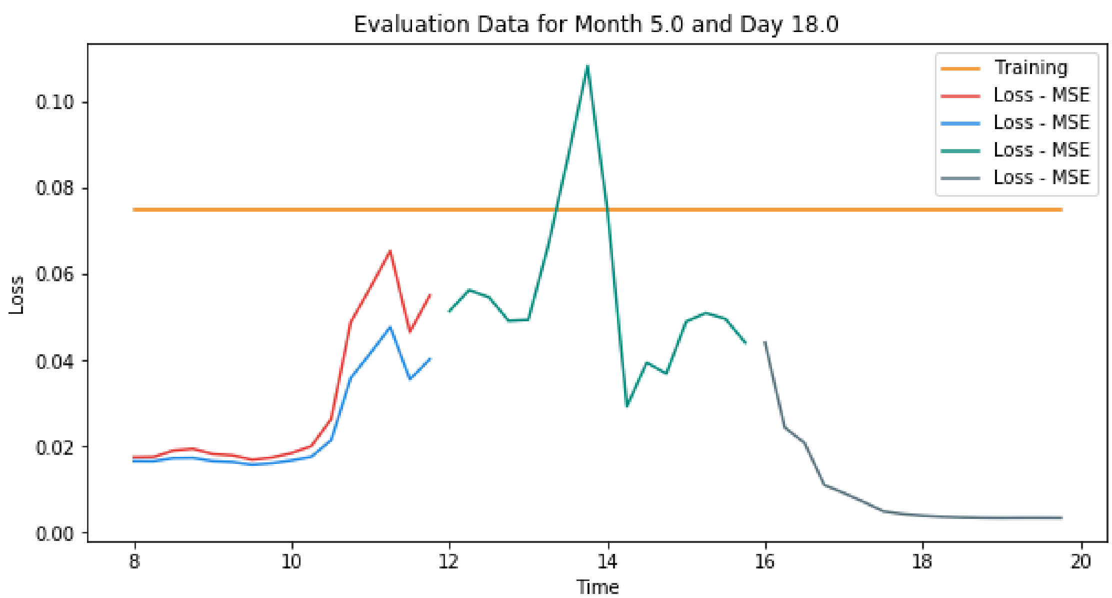

Evaluation Data for Month 5.0 and Day 15.0
Training
Loss - MSE
Loss - MSE
Loss - MSE
Loss - MSE
Loss
Time
0.10
0.08
0.06
0.04
0.02
8
10
12
14
16
18
20

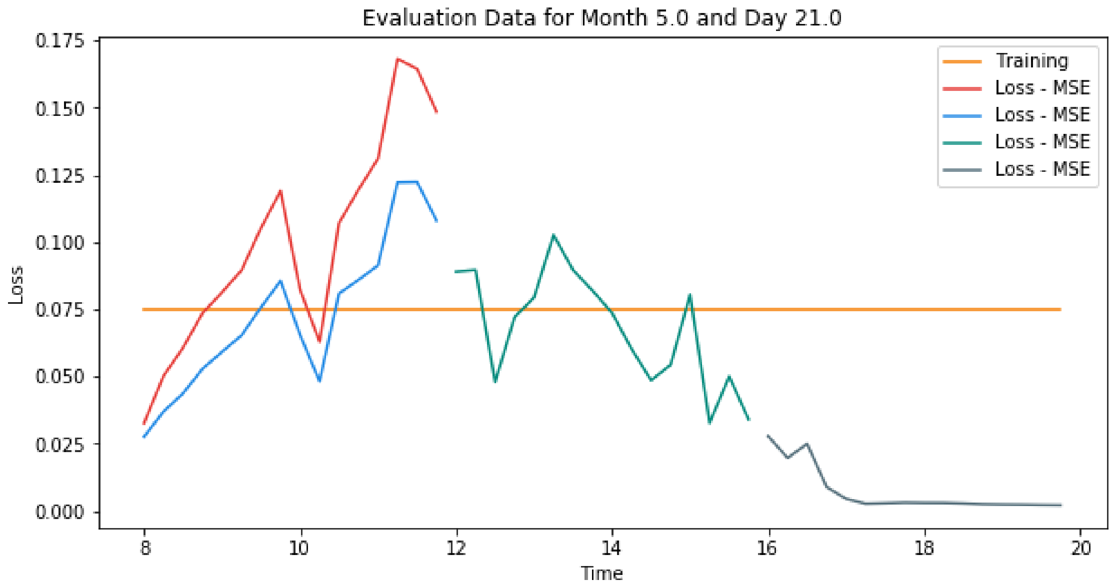

Evaluation Data for Month 5.0 and Day 25.0
Training
Loss - MSE
Loss - MSE
Loss - MSE
Loss - MSE
Loss
Time
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
8
10
12
14
16
18
20

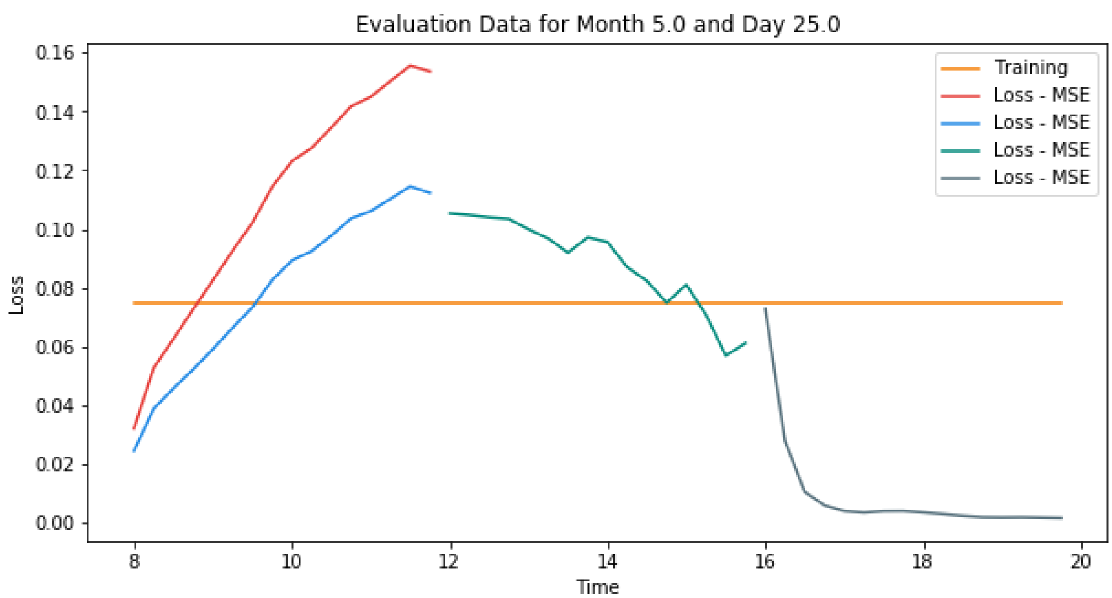

Evaluation Data for Month 5.0 and Day 20.0
Training
Loss - MSE
Loss - MSE
Loss - MSE
Loss - MSE
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Loss
8
10
12
14
16
18
20
Time


Evaluation Data for Month 5.0 and Day 17.0
Training
Loss - MSE
Loss - MSE
Loss - MSE
Loss - MSE
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Loss
8
10
12
14
16
18
20
Time


Evaluation Data for Month 5.0 and Day 21.0
Training
Loss - MSE
Loss - MSE
Loss - MSE
Loss - MSE
Loss
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
8
10
12
14
16
18
20
Time


Evaluation Data for Month 5.0 and Day 18.0
Training
Loss - MSE
Loss - MSE
Loss - MSE
Loss - MSE
Loss
0.10
0.08
0.06
0.04
0.02
0.00
8
10
12
14
16
18
20
Time




[39]:
```
score[0]
```

[39]:
```
        Loss  Threshold  Anomaly
0   0.002094      0.075    False
1   0.002006      0.075    False
2   0.002072      0.075    False
3   0.002149      0.075    False
4   0.002010      0.075    False
..       …          …        …
```

```
11  0.048841       0.075    False
12  0.056977       0.075    False
13  0.065382       0.075    False
14  0.046585       0.075    False
15  0.055036       0.075    False

[160 rows x 3 columns]
```

### 4. Machine Learning Model

In the model definition phase we trained a linear regression machine learning model, usign polynomial features, in order create a model that helps us to make an anomaly detector. We will use the same evaluation data as for the deep learning model.

The FFT of the data is not considered, since several of the time parameters (such as time and day and month) are needed to build up the model.

[40]:
```
def checkVal_LR(score, indx=0):
    """
    Function to check the Anomaly boolean value in score DF
    """
    if score <= 0.75 and score != 0.0:
        print('Warning! Possible anomaly detected in entry {}'.format(indx))

def checkPanel_LR(df_test, df_eval, pipe):
    """
    Function designed to give warnings, when the loss function is larger than a␣
 ↪given value
    df_test : DF, using for the training of model
    df_eval : DF, used to test the validity of model
    model   : previusly trained model, using df_test
    loss    : (str) loss function to use. 'MSE' by default
    """
    # Check loss with data used during the training phase
    score_all = {}
    score = []
    icount = 0
    for i in df_test:
        df = df_test[i]
        r2 = predictModel_LR(df, pipe)
        checkVal_LR(r2, indx=-i)
        score.append(r2)
    score_all[icount] = score
    icount += 1
    # Check the loss for each validation data
    r2_val = {}
    for i in df_eval:
        df = df_eval[i]
        score = predictModel_LR(df, pipe)
        score_all[icount] = score
        r2_val[i] = score
        icount += 1
        # Check if there is some anomalies (loss values over the threshold)
        checkVal_LR(score, indx=i)
    return [score_all, r2_val]
```

Load the trained model:

[41]:
```
pipe1 = load('./models/ADS_Capstone.solar_panel_1.ml-deg3.joblib')
print('Model loaded')
```

```
Model loaded
```

[42]:
```
scoreAll_LR(df1_check, pipe1)
```

```
df : 20
R2 : 0.9925075863949874
MSE : 26533.26069363776
MAE : 103.67187500000003
---------------------------

df : 21
R2 : 0.917262162991984
MSE : 270574.42934320285
MAE : 478.30468768750006
---------------------------

df : 22
R2 : 0.9911035276704409
MSE : 24567.090875077043
MAE : 93.07031245625001
---------------------------

df : 23
R2 : 0.0
MSE : 800.0
MAE : 22.0
---------------------------

df : 24
R2 : 0.0
MSE : 688.0
MAE : 23.0
---------------------------

df : 25
R2 : 0.994931795243462
MSE : 5899.32222987755
MAE : 56.459821443749995
---------------------------

df : 26
R2 : 0.8825366968293693
MSE : 665240.3863729553
MAE : 662.5703125625
---------------------------

df : 27
R2 : 0.9746959279740737
MSE : 237821.35219119125
MAE : 430.2098214375
---------------------------

df : 28
```

```
R2 : 0.9939307017963517
MSE : 24299.979147639035
MAE : 114.05022325
---------------------------

df : 29
R2 : 0.0
MSE : 752.0
MAE : 23.0
---------------------------

Mean R2 : 0.6746968398900668
Mean MSE : 125717.58208535807
MeanMAE : 200.63370538375003
---------------------------
```

[43]:

```
scoreAll_LR(df1_eval, pipe1)
```

```
df : 0
R2 : 0.8782421926141826
MSE : 576157.077562375
MAE : 527.0825891874999
---------------------------

df : 1
R2 : -5.12289851398833
MSE : 12241308.89163424
MAE : 3315.4821429093745
---------------------------

df : 2
R2 : -5.197471495350931
MSE : 10247289.220212938
MAE : 3060.7928011187496
---------------------------

df : 3
R2 : 0.9690144172238742
MSE : 103367.68077118302
MAE : 194.69832587937498
---------------------------

df : 4
R2 : 0.9018585011958502
MSE : 220505.9248849285
MAE : 366.06696424999996
---------------------------

df : 5
R2 : -4.868477763083686
MSE : 5570828.049463117
MAE : 2307.16629463125
---------------------------

df : 6
R2 : -4.365455457961028
MSE : 9344442.684675826
```

```
MAE : 2985.041964221875
---------------------------

df : 7
R2 : 0.9388639982412553
MSE : 70201.82166045008
MAE : 171.47081478437494
---------------------------

df : 8
R2 : 0.3407788698311487
MSE : 1611679.9845634499
MAE : 1053.0904019374998
---------------------------

df : 9
R2 : -7.053285995103613
MSE : 8318548.0333298985
MAE : 2648.9642300093747
---------------------------

df : 10
R2 : -8.061254212616694
MSE : 10002678.06686575
MAE : 3014.3258370187496
---------------------------

df : 11
R2 : 0.9669625909037576
MSE : 90386.20476796008
MAE : 205.91982887909373
---------------------------

df : 12
R2 : 0.9383064683248795
MSE : 299571.9805364386
MAE : 525.1450896874999
---------------------------

df : 13
R2 : -5.795681945236824
MSE : 13941879.223882642
MAE : 3597.749665428125
---------------------------

df : 14
R2 : -20.38199955503949
MSE : 16816524.470018573
MAE : 4026.006529128125
---------------------------

df : 15
R2 : 0.9884916048261062
MSE : 40934.7982459232
MAE : 146.74296871718747
---------------------------

df : 16
R2 : 0.9563174627472051
```

```
MSE : 190863.98616085586
MAE : 396.65066949999994
---------------------------

df : 17
R2 : -5.873955472999953
MSE : 12689647.6022163
MAE : 3446.672935175
---------------------------

df : 18
R2 : 0.32442463240858244
MSE : 8887343.447563428
MAE : 1840.9264878437498
---------------------------

df : 19
R2 : 0.9655696953765198
MSE : 12106.7747585874
MAE : 75.98376112468752
---------------------------

df : 20
R2 : 0.8782421926141826
MSE : 576157.077562375
MAE : 527.0825891874999
---------------------------

df : 21
R2 : -5.12289851398833
MSE : 12241308.89163424
MAE : 3315.4821429093745
---------------------------

df : 22
R2 : -5.197471495350931
MSE : 10247289.220212938
MAE : 3060.7928011187496
---------------------------

df : 23
R2 : 0.8728993250861343
MSE : 306680.4650857449
MAE : 291.4965401484374
---------------------------

df : 24
R2 : 0.7765122338674139
MSE : 1404696.6426251729
MAE : 895.3880205625001
---------------------------

df : 25
R2 : -4.013170431528515
MSE : 13312756.012478903
MAE : 3468.5055619281247
---------------------------

df : 26
```

```
R2 : -7.586046777415865
MSE : 15152217.17940004
MAE : 3715.87410696875
---------------------------

df : 27
R2 : 0.9727586723225563
MSE : 52640.81928271962
MAE : 154.87533485812503
---------------------------

df : 28
R2 : 0.9925075863949874
MSE : 26533.26069363776
MAE : 103.67187500000003
---------------------------

df : 29
R2 : -0.23783156077876977
MSE : 1852068.925850547
MAE : 1116.29174110625
---------------------------

df : 30
R2 : -6.665039730028677
MSE : 10590676.725659676
MAE : 3178.5082032468745
---------------------------

df : 31
R2 : 0.9687153960344067
MSE : 82520.84195385268
MAE : 169.81372764312502
---------------------------

df : 32
R2 : 0.7765122338674139
MSE : 1404696.6426251729
MAE : 895.3880205625001
---------------------------

df : 33
R2 : -4.013170431528515
MSE : 13312756.012478903
MAE : 3468.5055619281247
---------------------------

df : 34
R2 : -7.586046777415865
MSE : 15152217.17940004
MAE : 3715.87410696875
---------------------------

df : 35
R2 : 0.9791717425792086
MSE : 42773.89931572313
MAE : 143.46350450500006
---------------------------
```

```
df : 36
R2 : 0.9383064683248795
MSE : 299571.9805364386
MAE : 525.1450896874999
---------------------------

df : 37
R2 : -5.795681945236824
MSE : 13941879.223882642
MAE : 3597.749665428125
---------------------------

df : 38
R2 : -20.38199955503949
MSE : 16816524.470018573
MAE : 4026.006529128125
---------------------------

df : 39
R2 : 0.986800838914889
MSE : 46263.33131774393
MAE : 123.26713166593746
---------------------------

Mean R2 : -2.8752145126498223
Mean MSE : 5953462.368144747
MeanMAE : 1759.9790638995867
---------------------------
```

[44]:

```
[r2_all, r2_val] = checkPanel_LR(df1_all, df1_eval, pipe1)
```

```
Warning! Possible anomaly detected in entry -2
Warning! Possible anomaly detected in entry -39
Warning! Possible anomaly detected in entry -57
Warning! Possible anomaly detected in entry -63
Warning! Possible anomaly detected in entry -68
Warning! Possible anomaly detected in entry -104
Warning! Possible anomaly detected in entry -105
Warning! Possible anomaly detected in entry -111
Warning! Possible anomaly detected in entry -123
Warning! Possible anomaly detected in entry -134
Warning! Possible anomaly detected in entry -170
Warning! Possible anomaly detected in entry -236
Warning! Possible anomaly detected in entry -326
Warning! Possible anomaly detected in entry -398
Warning! Possible anomaly detected in entry -399
Warning! Possible anomaly detected in entry -411
Warning! Possible anomaly detected in entry 1
Warning! Possible anomaly detected in entry 2
Warning! Possible anomaly detected in entry 5
Warning! Possible anomaly detected in entry 6
Warning! Possible anomaly detected in entry 8
Warning! Possible anomaly detected in entry 9
Warning! Possible anomaly detected in entry 10
Warning! Possible anomaly detected in entry 13
Warning! Possible anomaly detected in entry 14
Warning! Possible anomaly detected in entry 17
```

```
Warning! Possible anomaly detected in entry 18
Warning! Possible anomaly detected in entry 21
Warning! Possible anomaly detected in entry 22
Warning! Possible anomaly detected in entry 25
Warning! Possible anomaly detected in entry 26
Warning! Possible anomaly detected in entry 29
Warning! Possible anomaly detected in entry 30
Warning! Possible anomaly detected in entry 33
Warning! Possible anomaly detected in entry 34
Warning! Possible anomaly detected in entry 37
Warning! Possible anomaly detected in entry 38
```

[45]:

```
plotR2(r2_all)
```