In [1]:

```
# To help you get started...
from IPython.display import display
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import LSTM, SimpleRNN, GRU, Bidirectiona
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import datetime as dt
from scipy import stats
from scipy.stats import scoreatpercentile
import math
from sklearn.preprocessing import minmax_scale
%matplotlib inline
```

Initializing all the data into dataframes

In [432]:

```
df0 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df1 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df2 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df3 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df4 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df5 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df6 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df7 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df8 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df9 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\exa
df10 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df11 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df12 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df13 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df14 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df15 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df16 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df17 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df18 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
df19 = pd.read_csv(r'C:\Users\srika\OneDrive\Desktop\Spring20\Tagup\ex
```

Reframing all the dataframes in an understandable way

In [433]:
```
df2['Date_created']=df2.index
df2=df2.reset_index(drop=True)
df2.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df2.columns)
cols = [cols[-1]] + cols[:-1]
df2 = df2[cols]
```

Outlier Removal using standard deviation

In [434]:
```
df2=df2[abs(df2.sensor1-df2.sensor1.mean()) <= (3*df2.sensor1.std())]
df2=df2[abs(df2.sensor2-df2.sensor2.mean()) <= (3*df2.sensor2.std())]
df2=df2[abs(df2.sensor3-df2.sensor3.mean()) <= (3*df2.sensor3.std())]
df2=df2[abs(df2.sensor4-df2.sensor4.mean()) <= (3*df2.sensor4.std())]
```

In [435]:
```
df2=df2.reset_index(drop=True)
df2['Date_created'] = pd.to_datetime(df2['Date_created'], errors='coer
df2['day_of_week'] = df2['Date_created'].dt.dayofweek
df2['month'] = pd.DatetimeIndex(df2['Date_created']).month
df2['hour'] = pd.DatetimeIndex(df2['Date_created']).hour
```

Standardizing the data using minmax_scale

In [436]:
```
df2[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df2[['se
```

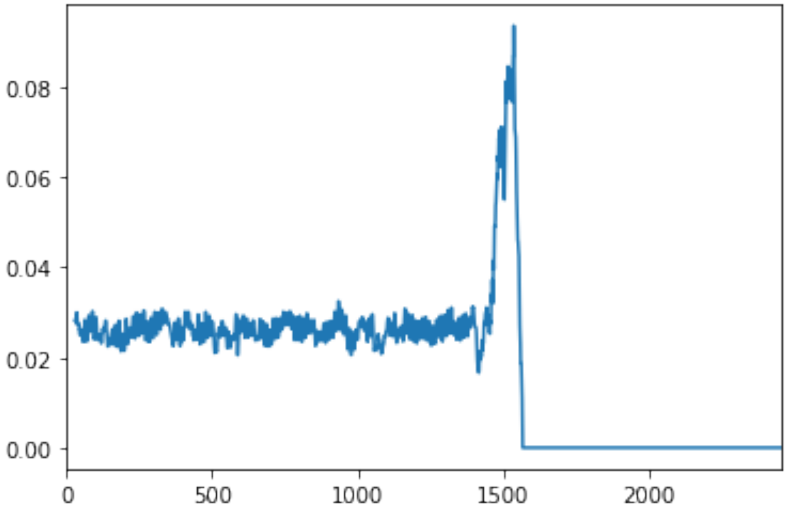
we use the moving variance to approximate the variation in the series at a point in time

In [437]:
```
df2['sensor1']=df2['sensor1'].rolling(window=30).var()
df2['sensor2']=df2['sensor2'].rolling(window=30).var()
df2['sensor3']=df2['sensor3'].rolling(window=30).var()
df2['sensor4']=df2['sensor4'].rolling(window=30).var()
```

Check when the machine is getting failed and remove the extra data

In [438]:
```
df2['sensor1'].plot()
```

Out[438]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x226091cb4e0>
```



let us take the data till 1600 so that we can think after the machine is completely dead

In [439]:
```
df2=df2[:1600]
df2=df2.fillna(df2.mean())
df2_sensor1=pd.DataFrame(data=df2['sensor1'])
```

To use supervised learning methods, we define an anomaly flag, to equal True if an observation lies outside of Tukey’s hinges across the sensor values.The anomaly flag is used to flag abnormal behaviour in the sensors

In [440]:
```
tukey_hinge=df2_sensor1.quantile(0.75)
df2_sensor1['labels']=df2_sensor1.apply(lambda row:1 if row.sensor1>tu
```

We fit a classification model to classify labels as predictors. Given that the predictors are sequence data, we consider the use of recurrent neural network (RNN) models for classifying anomalies. Traditional RNN units are unable to remember long-term dependencies and susceptible to the vanishing gradient problem, and for this purpose LSTM units may be more suitable

In [441]:

```python
def create_dataset(df2_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df2_sensor1)-look_back-1):
        a = df2_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df2_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
```

Constructig Dataframe

In [442]:

```python
values=create_dataset(df2_sensor1,look_back=10)
df_sensor1=pd.DataFrame(data=values[0])
df_sensor1['labels']=pd.DataFrame(data=values[1])
df2_sensor1_final=df_sensor1
df2_sensor1_final.head()
```

Out[442]:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908
1	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908
2	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908
3	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908
4	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908	0.027908

The exact same procedure follows for all the remaining 18 machines and 1st machine dataset can be verified for test set

In [443]:

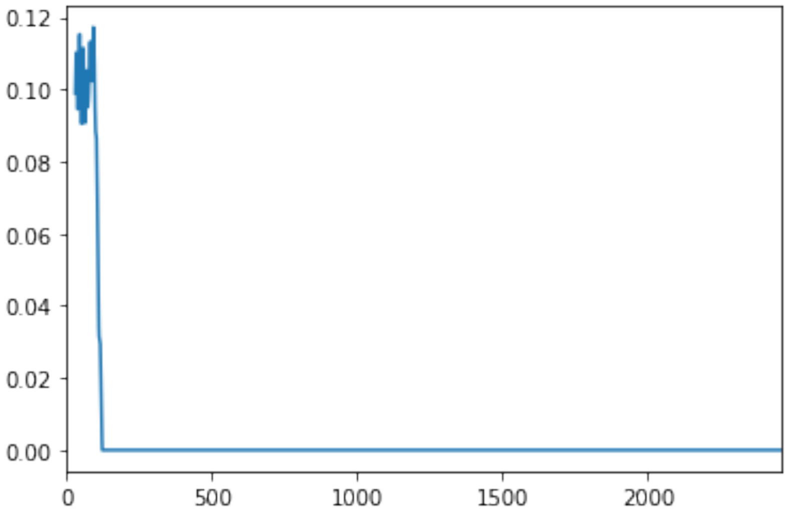
```
df3['Date_created']=df3.index
df3=df3.reset_index(drop=True)
df3.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df3.columns)
cols = [cols[-1]] + cols[:-1]
df3 = df3[cols]
df3=df3[abs(df3.sensor1-df3.sensor1.mean()) <= (3*df3.sensor1.std())]
df3=df3[abs(df3.sensor2-df3.sensor2.mean()) <= (3*df3.sensor2.std())]
df3=df3[abs(df3.sensor3-df3.sensor3.mean()) <= (3*df3.sensor3.std())]
df3=df3[abs(df3.sensor4-df3.sensor4.mean()) <= (3*df3.sensor4.std())]
df3=df3.reset_index(drop=True)
df3['Date_created'] = pd.to_datetime(df3['Date_created'], errors='coer
df3['day_of_week'] = df3['Date_created'].dt.dayofweek
df3['month'] = pd.DatetimeIndex(df3['Date_created']).month
df3['hour'] = pd.DatetimeIndex(df3['Date_created']).hour
df3[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df3[['se
df3['sensor1']=df3['sensor1'].rolling(window=30).var()
df3['sensor2']=df3['sensor2'].rolling(window=30).var()
df3['sensor3']=df3['sensor3'].rolling(window=30).var()
df3['sensor4']=df3['sensor4'].rolling(window=30).var()
```

In [444]:

```
df3['sensor1'].plot()
```

Out[444]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a50e4a8>
```



In [445]:

```
df3=df3[:130]
df3=df3.fillna(df3.mean())
df3_sensor1=pd.DataFrame(data=df3['sensor1'])
tukey_hinge=df3_sensor1.quantile(0.75)
df3_sensor1['labels']=df3_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df3_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df3_sensor1)-look_back-1):
        a = df3_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df3_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df3_sensor1,look_back=10)
df3_sensor1_final=pd.DataFrame(data=values[0])
df3_sensor1_final['labels']=pd.DataFrame(data=values[1])
df3_sensor1_final.head()
```

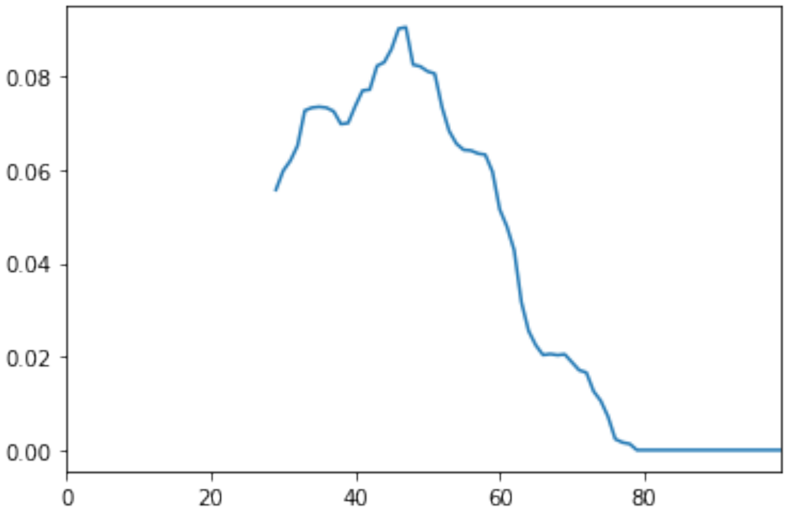
Out[445]:

	0	1	2	3	4	5	6	7	8	
0	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	C
1	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	C
2	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	C
3	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	C
4	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	0.081999	C

In [446]:
```
df4['Date_created']=df4.index
df4=df4.reset_index(drop=True)
df4.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df4.columns)
cols = [cols[-1]] + cols[:-1]
df4 = df4[cols]
df4=df4[abs(df4.sensor1-df4.sensor1.mean()) <= (3*df4.sensor1.std())]
df4=df4[abs(df4.sensor2-df4.sensor2.mean()) <= (3*df4.sensor2.std())]
df4=df4[abs(df4.sensor3-df4.sensor3.mean()) <= (3*df4.sensor3.std())]
df4=df4[abs(df4.sensor4-df4.sensor4.mean()) <= (3*df4.sensor4.std())]
df4=df4.reset_index(drop=True)
df4['Date_created'] = pd.to_datetime(df4['Date_created'], errors='coe
df4['day_of_week'] = df4['Date_created'].dt.dayofweek
df4['month'] = pd.DatetimeIndex(df4['Date_created']).month
df4['hour'] = pd.DatetimeIndex(df4['Date_created']).hour
df4[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df4[['se
df4['sensor1']=df4['sensor1'].rolling(window=30).var()
df4['sensor2']=df4['sensor2'].rolling(window=30).var()
df4['sensor3']=df4['sensor3'].rolling(window=30).var()
df4['sensor4']=df4['sensor4'].rolling(window=30).var()
```

In [447]:
```
df4['sensor1'][:100].plot()
```

Out[447]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a866400>
```



In [448]:

```
df4=df4[:80]
df4=df4.fillna(df4.mean())
df4_sensor1=pd.DataFrame(data=df4['sensor1'])
tukey_hinge=df4_sensor1.quantile(0.75)
df4_sensor1['labels']=df4_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df4_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df4_sensor1)-look_back-1):
        a = df4_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df4_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df4_sensor1,look_back=10)
df4_sensor1_final=pd.DataFrame(data=values[0])
df4_sensor1_final['labels']=pd.DataFrame(data=values[1])
df4_sensor1_final.head()
```

Out[448]:

	0	1	2	3	4	5	6	7	8	
0	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	C
1	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	C
2	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	C
3	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	C
4	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	0.051925	C

In [449]:

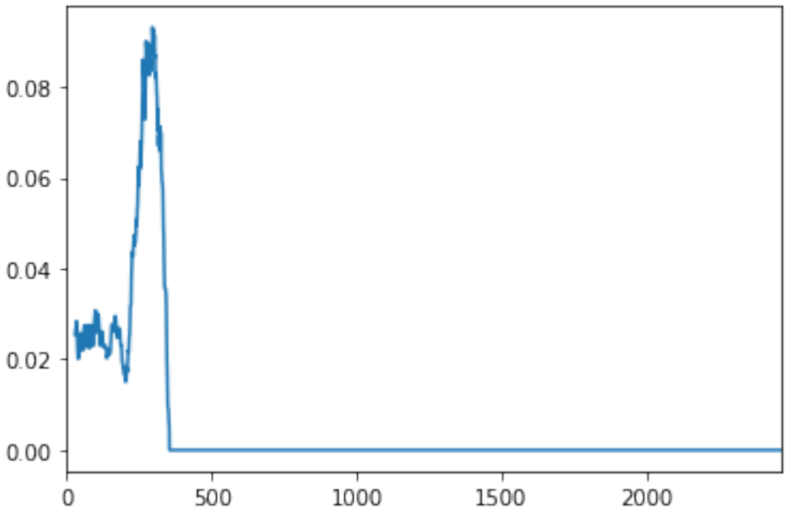
```
df5['Date_created']=df5.index
df5=df5.reset_index(drop=True)
df5.rename(columns={'0':'sensor1',
                    '1':'sensor2',
                    '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df5.columns)
cols = [cols[-1]] + cols[:-1]
df5 = df5[cols]
df5=df5[abs(df5.sensor1-df5.sensor1.mean()) <= (3*df5.sensor1.std())]
df5=df5[abs(df5.sensor2-df5.sensor2.mean()) <= (3*df5.sensor2.std())]
df5=df5[abs(df5.sensor3-df5.sensor3.mean()) <= (3*df5.sensor3.std())]
df5=df5[abs(df5.sensor4-df5.sensor4.mean()) <= (3*df5.sensor4.std())]
df5=df5.reset_index(drop=True)
df5['Date_created'] = pd.to_datetime(df5['Date_created'], errors='coe
df5['day_of_week'] = df5['Date_created'].dt.dayofweek
df5['month'] = pd.DatetimeIndex(df5['Date_created']).month
df5['hour'] = pd.DatetimeIndex(df5['Date_created']).hour
df5[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df5[['se
df5['sensor1']=df5['sensor1'].rolling(window=30).var()
df5['sensor2']=df5['sensor2'].rolling(window=30).var()
df5['sensor3']=df5['sensor3'].rolling(window=30).var()
df5['sensor4']=df5['sensor4'].rolling(window=30).var()
```

In [450]:

```
df5['sensor1'].plot()
```

Out[450]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a8c43c8>
```



In [451]:

```
df5=df5[:370]
df5=df5.fillna(df5.mean())
df5_sensor1=pd.DataFrame(data=df5['sensor1'])
tukey_hinge=df5_sensor1.quantile(0.75)
df5_sensor1['labels']=df5_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df5_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df5_sensor1)-look_back-1):
        a = df5_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df5_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df5_sensor1,look_back=10)
df5_sensor1_final=pd.DataFrame(data=values[0])
df5_sensor1_final['labels']=pd.DataFrame(data=values[1])
df5_sensor1_final.head()
```

Out[451]:

	0	1	2	3	4	5	6	7	8	9	la
0	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	
1	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	
2	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	
3	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	
4	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	0.03834	

In [452]:

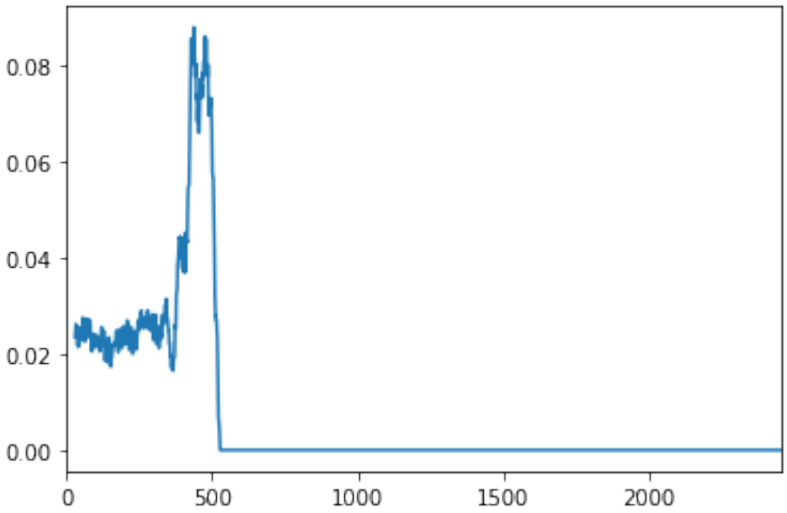
```
df6['Date_created']=df6.index
df6=df6.reset_index(drop=True)
df6.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df6.columns)
cols = [cols[-1]] + cols[:-1]
df6 = df6[cols]
df6=df6[abs(df6.sensor1-df6.sensor1.mean()) <= (3*df6.sensor1.std())]
df6=df6[abs(df6.sensor2-df6.sensor2.mean()) <= (3*df6.sensor2.std())]
df6=df6[abs(df6.sensor3-df6.sensor3.mean()) <= (3*df6.sensor3.std())]
df6=df6[abs(df6.sensor4-df6.sensor4.mean()) <= (3*df6.sensor4.std())]
df6=df6.reset_index(drop=True)
df6['Date_created'] = pd.to_datetime(df6['Date_created'], errors='coe
df6['day_of_week'] = df6['Date_created'].dt.dayofweek
df6['month'] = pd.DatetimeIndex(df6['Date_created']).month
df6['hour'] = pd.DatetimeIndex(df6['Date_created']).hour
df6[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df6[['se
df6['sensor1']=df6['sensor1'].rolling(window=30).var()
df6['sensor2']=df6['sensor2'].rolling(window=30).var()
df6['sensor3']=df6['sensor3'].rolling(window=30).var()
df6['sensor4']=df6['sensor4'].rolling(window=30).var()
```

In [453]:

```
df6['sensor1'].plot()
```

Out[453]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a8f78d0>
```



In [454]:

```
df6=df6[:550]
df6=df6.fillna(df6.mean())
df6_sensor1=pd.DataFrame(data=df6['sensor1'])
tukey_hinge=df6_sensor1.quantile(0.75)
df6_sensor1['labels']=df6_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df6_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df6_sensor1)-look_back-1):
        a = df6_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df6_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df6_sensor1,look_back=10)
df6_sensor1_final=pd.DataFrame(data=values[0])
df6_sensor1_final['labels']=pd.DataFrame(data=values[1])
df6_sensor1_final.head()
```

Out[454]:

	0	1	2	3	4	5	6	7	8	
0	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	C
1	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	C
2	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	C
3	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	C
4	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	0.032603	C

In [455]:

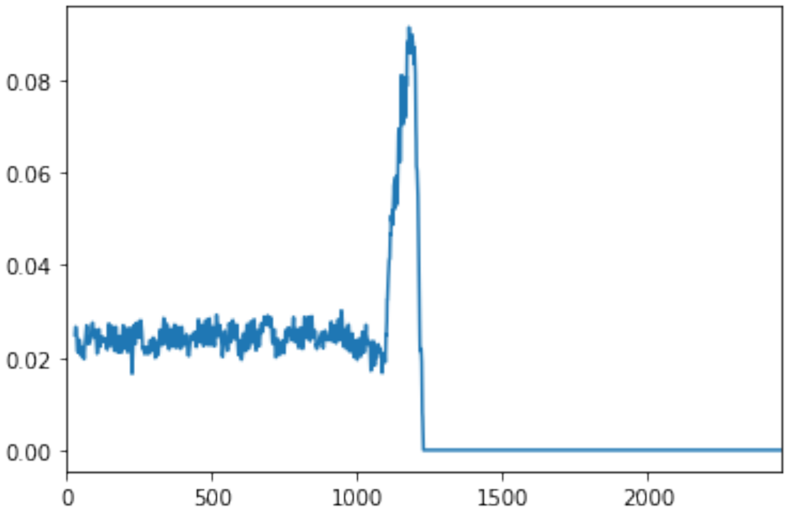
```
df7['Date_created']=df7.index
df7=df7.reset_index(drop=True)
df7.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df7.columns)
cols = [cols[-1]] + cols[:-1]
df7 = df7[cols]
df7=df7[abs(df7.sensor1-df7.sensor1.mean()) <= (3*df7.sensor1.std())]
df7=df7[abs(df7.sensor2-df7.sensor2.mean()) <= (3*df7.sensor2.std())]
df7=df7[abs(df7.sensor3-df7.sensor3.mean()) <= (3*df7.sensor3.std())]
df7=df7[abs(df7.sensor4-df7.sensor4.mean()) <= (3*df7.sensor4.std())]
df7=df7.reset_index(drop=True)
df7['Date_created'] = pd.to_datetime(df7['Date_created'], errors='coer
df7['day_of_week'] = df7['Date_created'].dt.dayofweek
df7['month'] = pd.DatetimeIndex(df7['Date_created']).month
df7['hour'] = pd.DatetimeIndex(df7['Date_created']).hour
df7[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df7[['se
df7['sensor1']=df7['sensor1'].rolling(window=30).var()
df7['sensor2']=df7['sensor2'].rolling(window=30).var()
df7['sensor3']=df7['sensor3'].rolling(window=30).var()
df7['sensor4']=df7['sensor4'].rolling(window=30).var()
```

In [456]:

```
df7['sensor1'].plot()
```

Out[456]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a92dc88>
```



In [457]:
```
df7=df7[:1300]
df7=df7.fillna(df7.mean())
df7_sensor1=pd.DataFrame(data=df7['sensor1'])
tukey_hinge=df7_sensor1.quantile(0.75)
df7_sensor1['labels']=df7_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df7_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df7_sensor1)-look_back-1):
        a = df7_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df7_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df7_sensor1,look_back=10)
df7_sensor1_final=pd.DataFrame(data=values[0])
df7_sensor1_final['labels']=pd.DataFrame(data=values[1])
df7_sensor1_final.head()
```

Out[457]:

	0	1	2	3	4	5	6	7	8	
0	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	C
1	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	C
2	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	C
3	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	C
4	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	0.026378	C

In [458]:

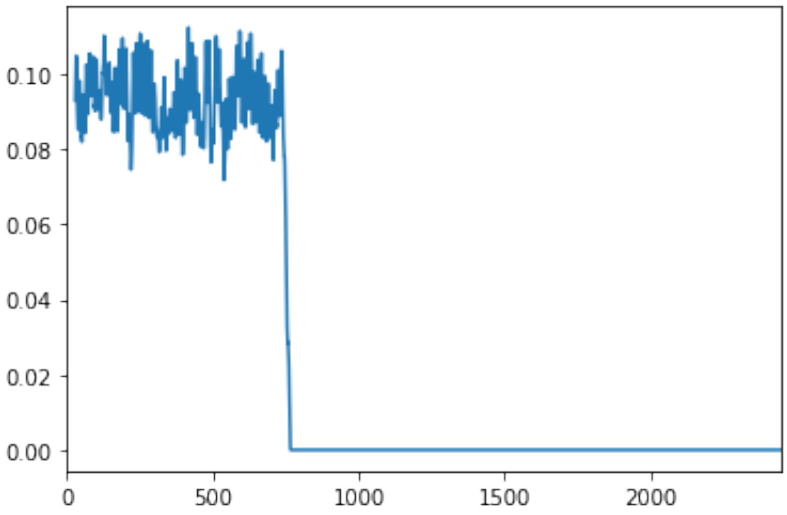
```
df8['Date_created']=df8.index
df8=df8.reset_index(drop=True)
df8.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df8.columns)
cols = [cols[-1]] + cols[:-1]
df8 = df8[cols]
df8=df8[abs(df8.sensor1-df8.sensor1.mean()) <= (3*df8.sensor1.std())]
df8=df8[abs(df8.sensor2-df8.sensor2.mean()) <= (3*df8.sensor2.std())]
df8=df8[abs(df8.sensor3-df8.sensor3.mean()) <= (3*df8.sensor3.std())]
df8=df8[abs(df8.sensor4-df8.sensor4.mean()) <= (3*df8.sensor4.std())]
df8=df8.reset_index(drop=True)
df8['Date_created'] = pd.to_datetime(df8['Date_created'], errors='coe
df8['day_of_week'] = df8['Date_created'].dt.dayofweek
df8['month'] = pd.DatetimeIndex(df8['Date_created']).month
df8['hour'] = pd.DatetimeIndex(df8['Date_created']).hour
df8[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df8[['se
df8['sensor1']=df8['sensor1'].rolling(window=30).var()
df8['sensor2']=df8['sensor2'].rolling(window=30).var()
df8['sensor3']=df8['sensor3'].rolling(window=30).var()
df8['sensor4']=df8['sensor4'].rolling(window=30).var()
```

In [459]:

```
df8['sensor1'].plot()
```

Out[459]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a96c780>
```



In [460]:

```
df8=df8[:900]
df8=df8.fillna(df8.mean())
df8_sensor1=pd.DataFrame(data=df8['sensor1'])
tukey_hinge=df8_sensor1.quantile(0.75)
df8_sensor1['labels']=df8_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df8_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df8_sensor1)-look_back-1):
        a = df8_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df8_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df8_sensor1,look_back=10)
df8_sensor1_final=pd.DataFrame(data=values[0])
df8_sensor1_final['labels']=pd.DataFrame(data=values[1])
df8_sensor1_final.head()
```

Out[460]:

	0	1	2	3	4	5	6	7	8	
0	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	C
1	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	C
2	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	C
3	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	C
4	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	0.077978	C

In [461]:

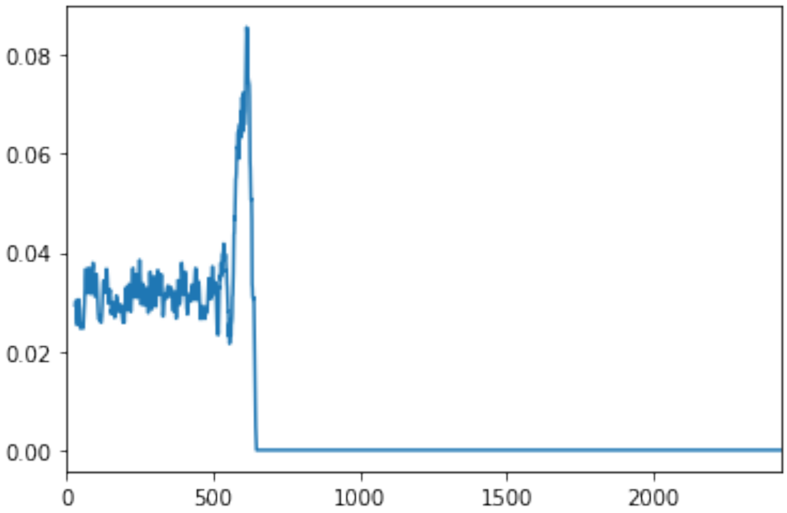
```
df9['Date_created']=df9.index
df9=df9.reset_index(drop=True)
df9.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df9.columns)
cols = [cols[-1]] + cols[:-1]
df9 = df9[cols]
df9=df9[abs(df9.sensor1-df9.sensor1.mean()) <= (3*df9.sensor1.std())]
df9=df9[abs(df9.sensor2-df9.sensor2.mean()) <= (3*df9.sensor2.std())]
df9=df9[abs(df9.sensor3-df9.sensor3.mean()) <= (3*df9.sensor3.std())]
df9=df9[abs(df9.sensor4-df9.sensor4.mean()) <= (3*df9.sensor4.std())]
df9=df9.reset_index(drop=True)
df9['Date_created'] = pd.to_datetime(df9['Date_created'], errors='coe
df9['day_of_week'] = df9['Date_created'].dt.dayofweek
df9['month'] = pd.DatetimeIndex(df9['Date_created']).month
df9['hour'] = pd.DatetimeIndex(df9['Date_created']).hour
df9[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df9[['se
df9['sensor1']=df9['sensor1'].rolling(window=30).var()
df9['sensor2']=df9['sensor2'].rolling(window=30).var()
df9['sensor3']=df9['sensor3'].rolling(window=30).var()
df9['sensor4']=df9['sensor4'].rolling(window=30).var()
```

In [462]:

```
df9['sensor1'].plot()
```

Out[462]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a93d898>
```



In [463]:

```
df9=df9[:750]
df9=df9.fillna(df9.mean())
df9_sensor1=pd.DataFrame(data=df9['sensor1'])
tukey_hinge=df9_sensor1.quantile(0.75)
df9_sensor1['labels']=df9_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df9_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df9_sensor1)-look_back-1):
        a = df9_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df9_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df9_sensor1,look_back=10)
df9_sensor1_final=pd.DataFrame(data=values[0])
df9_sensor1_final['labels']=pd.DataFrame(data=values[1])
df9_sensor1_final.head()
```

Out[463]:

```
          0         1         2         3         4         5         6         7         8
0  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613
1  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613
2  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613
3  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613
4  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613  0.029613
```

In [464]:

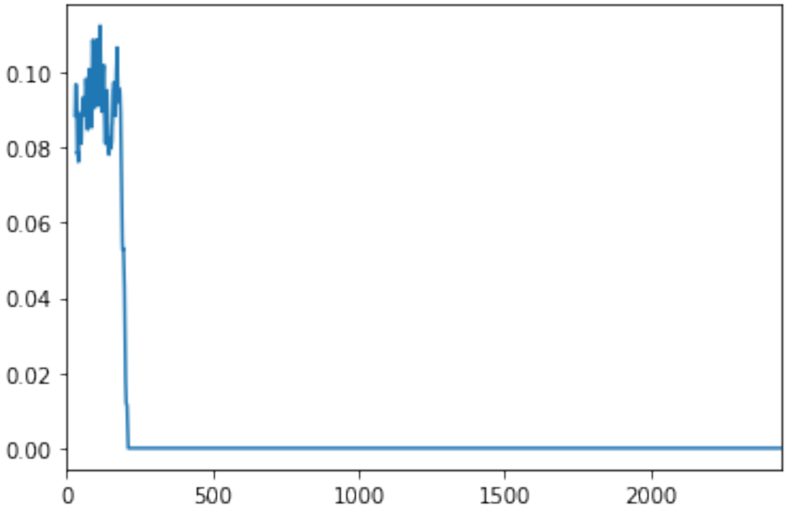
```
df10['Date_created']=df10.index
df10=df10.reset_index(drop=True)
df10.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df10.columns)
cols = [cols[-1]] + cols[:-1]
df10 = df10[cols]
df10=df10[abs(df10.sensor1-df10.sensor1.mean()) <= (3*df10.sensor1.std
df10=df10[abs(df10.sensor2-df10.sensor2.mean()) <= (3*df10.sensor2.std
df10=df10[abs(df10.sensor3-df10.sensor3.mean()) <= (3*df10.sensor3.std
df10=df10[abs(df10.sensor4-df10.sensor4.mean()) <= (3*df10.sensor4.std
df10=df10.reset_index(drop=True)
df10['Date_created'] = pd.to_datetime(df10['Date_created'], errors='co
df10['day_of_week'] = df10['Date_created'].dt.dayofweek
df10['month'] = pd.DatetimeIndex(df10['Date_created']).month
df10['hour'] = pd.DatetimeIndex(df10['Date_created']).hour
df10[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df10[['
df10['sensor1']=df10['sensor1'].rolling(window=30).var()
df10['sensor2']=df10['sensor2'].rolling(window=30).var()
df10['sensor3']=df10['sensor3'].rolling(window=30).var()
df10['sensor4']=df10['sensor4'].rolling(window=30).var()
```

In [465]:

```
df10['sensor1'].plot()
```

Out[465]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a9e6be0>
```



In [466]:

```
df10=df10[:750]
df10=df10.fillna(df10.mean())
df10_sensor1=pd.DataFrame(data=df10['sensor1'])
tukey_hinge=df10_sensor1.quantile(0.75)
df10_sensor1['labels']=df10_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df10_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df10_sensor1)-look_back-1):
        a = df10_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df10_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df10_sensor1,look_back=10)
df10_sensor1_final=pd.DataFrame(data=values[0])
df10_sensor1_final['labels']=pd.DataFrame(data=values[1])
df10_sensor1_final.head()
```

Out[466]:

	0	1	2	3	4	5	6	7	8	9	la
0	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	
1	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	
2	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	
3	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	
4	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	0.02146	

In [467]:

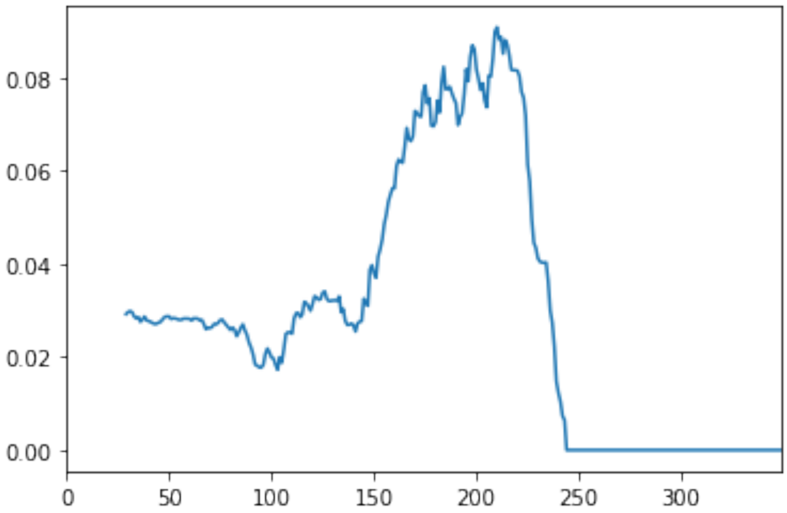
```
df11['Date_created']=df11.index
df11=df11.reset_index(drop=True)
df11.rename(columns={'0':'sensor1',
                      '1':'sensor2',
                      '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df11.columns)
cols = [cols[-1]] + cols[:-1]
df11 = df11[cols]
df11=df11[abs(df11.sensor1-df11.sensor1.mean()) <= (3*df11.sensor1.std
df11=df11[abs(df11.sensor2-df11.sensor2.mean()) <= (3*df11.sensor2.std
df11=df11[abs(df11.sensor3-df11.sensor3.mean()) <= (3*df11.sensor3.std
df11=df11[abs(df11.sensor4-df11.sensor4.mean()) <= (3*df11.sensor4.std
df11=df11.reset_index(drop=True)
df11['Date_created'] = pd.to_datetime(df11['Date_created'], errors='co
df11['day_of_week'] = df11['Date_created'].dt.dayofweek
df11['month'] = pd.DatetimeIndex(df11['Date_created']).month
df11['hour'] = pd.DatetimeIndex(df11['Date_created']).hour
df11[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df11[['
df11['sensor1']=df11['sensor1'].rolling(window=30).var()
df11['sensor2']=df11['sensor2'].rolling(window=30).var()
df11['sensor3']=df11['sensor3'].rolling(window=30).var()
df11['sensor4']=df11['sensor4'].rolling(window=30).var()
```

In [468]:

```
df11['sensor1'][:350].plot()
```

Out[468]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a7dd048>
```



In [469]:

```
df11=df11[:250]
df11=df11.fillna(df11.mean())
df11_sensor1=pd.DataFrame(data=df11['sensor1'])
tukey_hinge=df11_sensor1.quantile(0.75)
df11_sensor1['labels']=df11_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df11_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df11_sensor1)-look_back-1):
        a = df11_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df11_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df11_sensor1,look_back=10)
df11_sensor1_final=pd.DataFrame(data=values[0])
df11_sensor1_final['labels']=pd.DataFrame(data=values[1])
df11_sensor1_final.head()
```

Out[469]:

	0	1	2	3	4	5	6	7	8
0	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398
1	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398
2	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398
3	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398
4	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398	0.042398

In [470]:

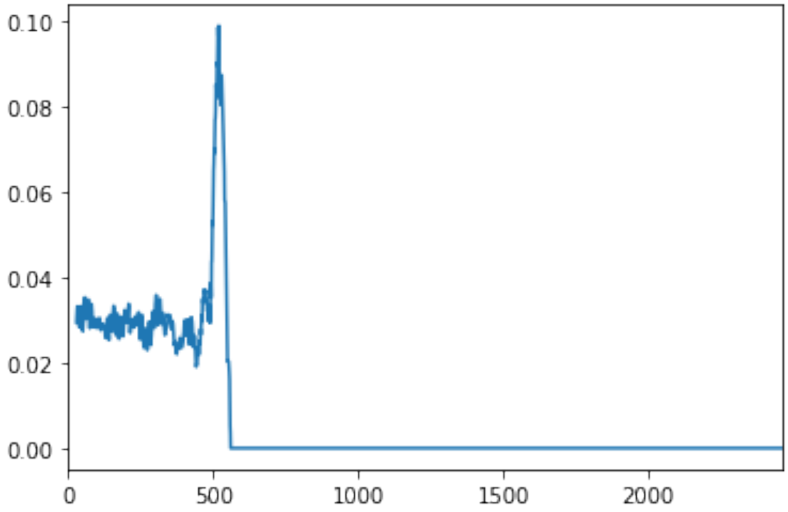
```
df12['Date_created']=df12.index
df12=df12.reset_index(drop=True)
df12.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df12.columns)
cols = [cols[-1]] + cols[:-1]
df12 = df12[cols]
df12=df12[abs(df12.sensor1-df12.sensor1.mean()) <= (3*df12.sensor1.std
df12=df12[abs(df12.sensor2-df12.sensor2.mean()) <= (3*df12.sensor2.std
df12=df12[abs(df12.sensor3-df12.sensor3.mean()) <= (3*df12.sensor3.std
df12=df12[abs(df12.sensor4-df12.sensor4.mean()) <= (3*df12.sensor4.std
df12=df12.reset_index(drop=True)
df12['Date_created'] = pd.to_datetime(df12['Date_created'], errors='co
df12['day_of_week'] = df12['Date_created'].dt.dayofweek
df12['month'] = pd.DatetimeIndex(df12['Date_created']).month
df12['hour'] = pd.DatetimeIndex(df12['Date_created']).hour
df12[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df12[['
df12['sensor1']=df12['sensor1'].rolling(window=30).var()
df12['sensor2']=df12['sensor2'].rolling(window=30).var()
df12['sensor3']=df12['sensor3'].rolling(window=30).var()
df12['sensor4']=df12['sensor4'].rolling(window=30).var()
```

In [471]:

```
df12['sensor1'].plot()
```

Out[471]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a7c35c0>
```



In [472]:

```
df12=df12[:600]
df12=df12.fillna(df12.mean())
df12_sensor1=pd.DataFrame(data=df12['sensor1'])
tukey_hinge=df12_sensor1.quantile(0.75)
df12_sensor1['labels']=df12_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df12_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df12_sensor1)-look_back-1):
        a = df12_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df12_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df12_sensor1,look_back=10)
df12_sensor1_final=pd.DataFrame(data=values[0])
df12_sensor1_final['labels']=pd.DataFrame(data=values[1])
df12_sensor1_final.head()
```

Out[472]:

	0	1	2	3	4	5	6	7	8	
0	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	C
1	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	C
2	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	C
3	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	C
4	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	0.030865	C

In [473]:

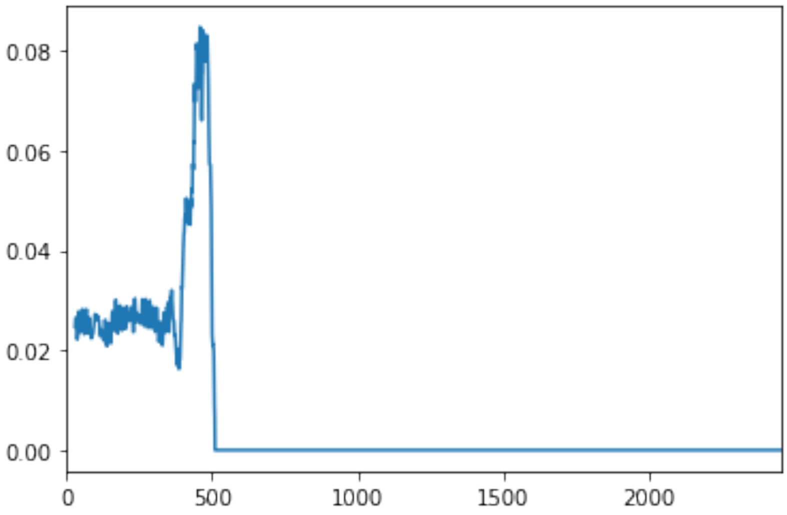
```
df13['Date_created']=df13.index
df13=df13.reset_index(drop=True)
df13.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df13.columns)
cols = [cols[-1]] + cols[:-1]
df13 = df13[cols]
df13=df13[abs(df13.sensor1-df13.sensor1.mean()) <= (3*df13.sensor1.std
df13=df13[abs(df13.sensor2-df13.sensor2.mean()) <= (3*df13.sensor2.std
df13=df13[abs(df13.sensor3-df13.sensor3.mean()) <= (3*df13.sensor3.std
df13=df13[abs(df13.sensor4-df13.sensor4.mean()) <= (3*df13.sensor4.std
df13=df13.reset_index(drop=True)
df13['Date_created'] = pd.to_datetime(df13['Date_created'], errors='co
df13['day_of_week'] = df13['Date_created'].dt.dayofweek
df13['month'] = pd.DatetimeIndex(df13['Date_created']).month
df13['hour'] = pd.DatetimeIndex(df13['Date_created']).hour
df13[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df13[['
df13['sensor1']=df13['sensor1'].rolling(window=30).var()
df13['sensor2']=df13['sensor2'].rolling(window=30).var()
df13['sensor3']=df13['sensor3'].rolling(window=30).var()
df13['sensor4']=df13['sensor4'].rolling(window=30).var()
```

In [474]:

```
df13['sensor1'].plot()
```

Out[474]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260bd69208>
```



In [475]:

```
df13=df13[:500]
df13=df13.fillna(df13.mean())
df13_sensor1=pd.DataFrame(data=df13['sensor1'])
tukey_hinge=df13_sensor1.quantile(0.75)
df13_sensor1['labels']=df13_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df13_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df13_sensor1)-look_back-1):
        a = df13_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df13_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df13_sensor1,look_back=10)
df13_sensor1_final=pd.DataFrame(data=values[0])
df13_sensor1_final['labels']=pd.DataFrame(data=values[1])
df13_sensor1_final.head()
```

Out[475]:

	0	1	2	3	4	5	6	7	8	
0	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	C
1	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	C
2	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	C
3	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	C
4	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	0.033691	C

In [476]:

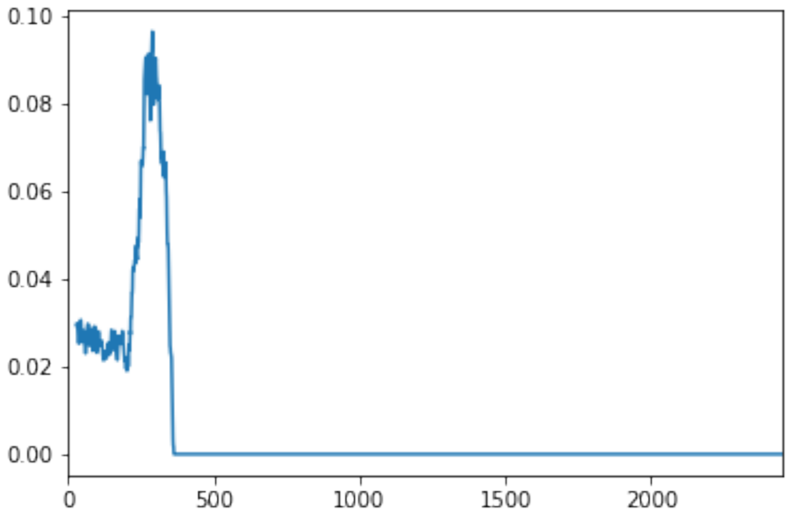
```
df14['Date_created']=df14.index
df14=df14.reset_index(drop=True)
df14.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df14.columns)
cols = [cols[-1]] + cols[:-1]
df14 = df14[cols]
df14=df14[abs(df14.sensor1-df14.sensor1.mean()) <= (3*df14.sensor1.std
df14=df14[abs(df14.sensor2-df14.sensor2.mean()) <= (3*df14.sensor2.std
df14=df14[abs(df14.sensor3-df14.sensor3.mean()) <= (3*df14.sensor3.std
df14=df14[abs(df14.sensor4-df14.sensor4.mean()) <= (3*df14.sensor4.std
df14=df14.reset_index(drop=True)
df14['Date_created'] = pd.to_datetime(df14['Date_created'], errors='co
df14['day_of_week'] = df14['Date_created'].dt.dayofweek
df14['month'] = pd.DatetimeIndex(df14['Date_created']).month
df14['hour'] = pd.DatetimeIndex(df14['Date_created']).hour
df14[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df14[['
df14['sensor1']=df14['sensor1'].rolling(window=30).var()
df14['sensor2']=df14['sensor2'].rolling(window=30).var()
df14['sensor3']=df14['sensor3'].rolling(window=30).var()
df14['sensor4']=df14['sensor4'].rolling(window=30).var()
```

In [477]:

```
df14['sensor1'].plot()
```

Out[477]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260aab1b00>
```



In [478]:

```
df14=df14[:500]
df14=df14.fillna(df14.mean())
df14_sensor1=pd.DataFrame(data=df14['sensor1'])
tukey_hinge=df14_sensor1.quantile(0.75)
df14_sensor1['labels']=df14_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df14_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df14_sensor1)-look_back-1):
        a = df14_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df14_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df14_sensor1,look_back=10)
df14_sensor1_final=pd.DataFrame(data=values[0])
df14_sensor1_final['labels']=pd.DataFrame(data=values[1])
df14_sensor1_final.head()
```

Out[478]:

	0	1	2	3	4	5	6	7	8
0	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594
1	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594
2	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594
3	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594
4	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594	0.029594

In [479]:

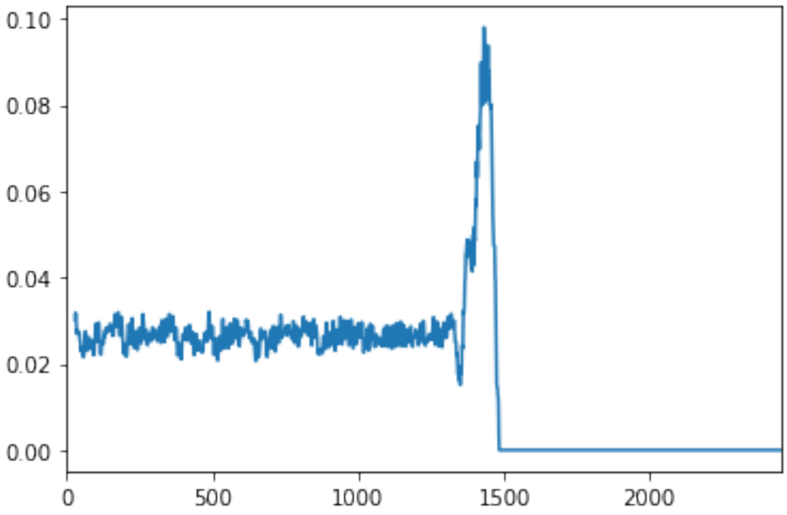
```
df15['Date_created']=df15.index
df15=df15.reset_index(drop=True)
df15.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df15.columns)
cols = [cols[-1]] + cols[:-1]
df15 = df15[cols]
df15=df15[abs(df15.sensor1-df15.sensor1.mean()) <= (3*df15.sensor1.std
df15=df15[abs(df15.sensor2-df15.sensor2.mean()) <= (3*df15.sensor2.std
df15=df15[abs(df15.sensor3-df15.sensor3.mean()) <= (3*df15.sensor3.std
df15=df15[abs(df15.sensor4-df15.sensor4.mean()) <= (3*df15.sensor4.std
df15=df15.reset_index(drop=True)
df15['Date_created'] = pd.to_datetime(df15['Date_created'], errors='co
df15['day_of_week'] = df15['Date_created'].dt.dayofweek
df15['month'] = pd.DatetimeIndex(df15['Date_created']).month
df15['hour'] = pd.DatetimeIndex(df15['Date_created']).hour
df15[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df15[['
df15['sensor1']=df15['sensor1'].rolling(window=30).var()
df15['sensor2']=df15['sensor2'].rolling(window=30).var()
df15['sensor3']=df15['sensor3'].rolling(window=30).var()
df15['sensor4']=df15['sensor4'].rolling(window=30).var()
```

In [480]:

```
df15['sensor1'].plot()
```

Out[480]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260a7f86d8>
```



In [481]:

```
df15=df15[:1450]
df15=df15.fillna(df15.mean())
df15_sensor1=pd.DataFrame(data=df15['sensor1'])
tukey_hinge=df15_sensor1.quantile(0.75)
df15_sensor1['labels']=df15_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df15_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df15_sensor1)-look_back-1):
        a = df15_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df15_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df15_sensor1,look_back=10)
df15_sensor1_final=pd.DataFrame(data=values[0])
df15_sensor1_final['labels']=pd.DataFrame(data=values[1])
df15_sensor1_final.head()
```

Out[481]:

	0	1	2	3	4	5	6	7	8	
0	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	C
1	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	C
2	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	C
3	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	C
4	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	0.028742	C

In [482]:

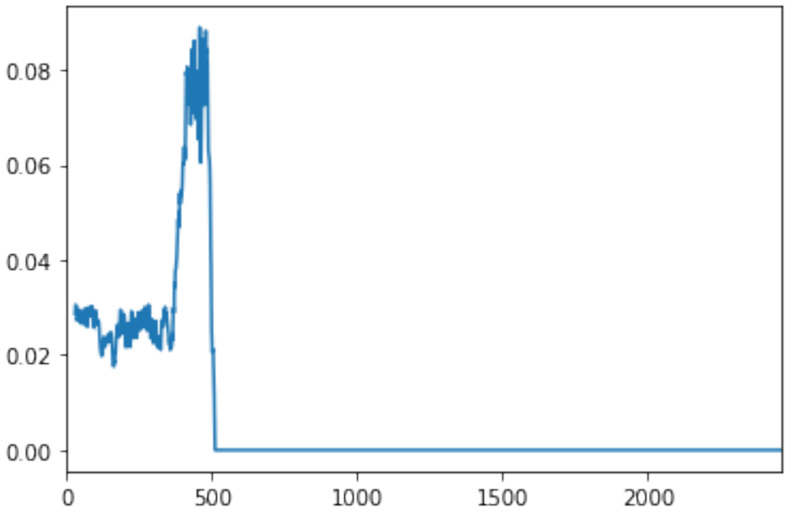
```
df16['Date_created']=df16.index
df16=df16.reset_index(drop=True)
df16.rename(columns={'0':'sensor1',
                     '1':'sensor2',
                     '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df16.columns)
cols = [cols[-1]] + cols[:-1]
df16 = df16[cols]
df16=df16[abs(df16.sensor1-df16.sensor1.mean()) <= (3*df16.sensor1.stc
df16=df16[abs(df16.sensor2-df16.sensor2.mean()) <= (3*df16.sensor2.stc
df16=df16[abs(df16.sensor3-df16.sensor3.mean()) <= (3*df16.sensor3.stc
df16=df16[abs(df16.sensor4-df16.sensor4.mean()) <= (3*df16.sensor4.stc
df16=df16.reset_index(drop=True)
df16['Date_created'] = pd.to_datetime(df16['Date_created'], errors='cc
df16['day_of_week'] = df16['Date_created'].dt.dayofweek
df16['month'] = pd.DatetimeIndex(df16['Date_created']).month
df16['hour'] = pd.DatetimeIndex(df16['Date_created']).hour
df16[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df16[['
df16['sensor1']=df16['sensor1'].rolling(window=30).var()
df16['sensor2']=df16['sensor2'].rolling(window=30).var()
df16['sensor3']=df16['sensor3'].rolling(window=30).var()
df16['sensor4']=df16['sensor4'].rolling(window=30).var()
```

In [483]:

```
df16['sensor1'].plot()
```

Out[483]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260ab29588>
```



In [484]:

```
df16=df16[:500]
df16=df16.fillna(df16.mean())
df16_sensor1=pd.DataFrame(data=df16['sensor1'])
tukey_hinge=df16_sensor1.quantile(0.75)
df16_sensor1['labels']=df16_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df16_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df16_sensor1)-look_back-1):
        a = df16_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df16_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df16_sensor1,look_back=10)
df16_sensor1_final=pd.DataFrame(data=values[0])
df16_sensor1_final['labels']=pd.DataFrame(data=values[1])
df16_sensor1_final.head()
```

Out[484]:

```
         0         1         2         3         4         5         6         7         8
0 0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  C
1 0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  C
2 0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  C
3 0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  C
4 0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  0.036846  C
```

In [485]:

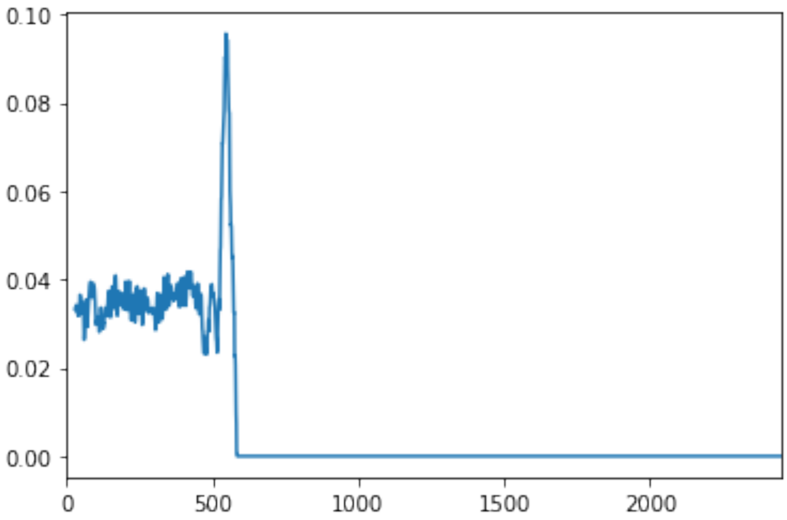
```
df17['Date_created']=df17.index
df17=df17.reset_index(drop=True)
df17.rename(columns={'0':'sensor1',
                     '1':'sensor2',
                     '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df17.columns)
cols = [cols[-1]] + cols[:-1]
df17 = df17[cols]
df17=df17[abs(df17.sensor1-df17.sensor1.mean()) <= (3*df17.sensor1.stc
df17=df17[abs(df17.sensor2-df17.sensor2.mean()) <= (3*df17.sensor2.stc
df17=df17[abs(df17.sensor3-df17.sensor3.mean()) <= (3*df17.sensor3.stc
df17=df17[abs(df17.sensor4-df17.sensor4.mean()) <= (3*df17.sensor4.stc
df17=df17.reset_index(drop=True)
df17['Date_created'] = pd.to_datetime(df17['Date_created'], errors='cc
df17['day_of_week'] = df17['Date_created'].dt.dayofweek
df17['month'] = pd.DatetimeIndex(df17['Date_created']).month
df17['hour'] = pd.DatetimeIndex(df17['Date_created']).hour
df17[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df17[['
df17['sensor1']=df17['sensor1'].rolling(window=30).var()
df17['sensor2']=df17['sensor2'].rolling(window=30).var()
df17['sensor3']=df17['sensor3'].rolling(window=30).var()
df17['sensor4']=df17['sensor4'].rolling(window=30).var()
```

In [486]:

```
df17['sensor1'].plot()
```

Out[486]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260beae5c0>
```



In [487]:

```
df17=df17[:550]
df17=df17.fillna(df17.mean())
df17_sensor1=pd.DataFrame(data=df17['sensor1'])
tukey_hinge=df17_sensor1.quantile(0.75)
df17_sensor1['labels']=df17_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df17_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df17_sensor1)-look_back-1):
        a = df17_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df17_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df17_sensor1,look_back=10)
df17_sensor1_final=pd.DataFrame(data=values[0])
df17_sensor1_final['labels']=pd.DataFrame(data=values[1])
df17_sensor1_final.head()
```

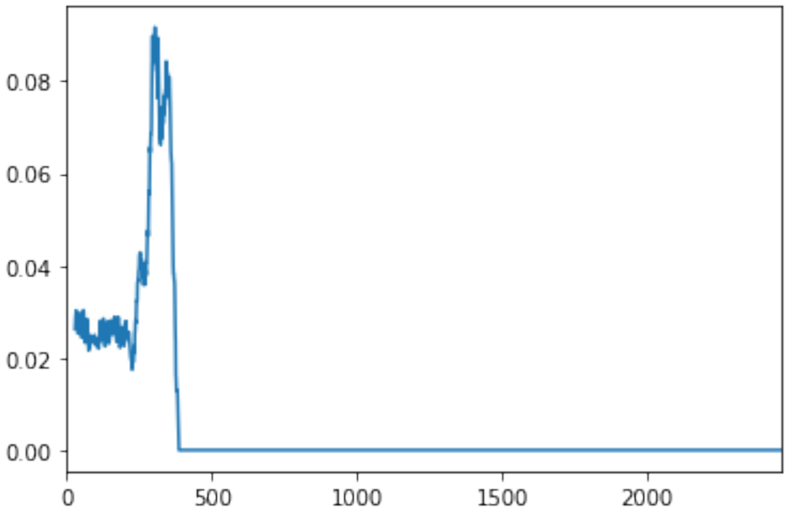
Out[487]:

	0	1	2	3	4	5	6	7	8
0	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243
1	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243
2	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243
3	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243
4	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243	0.036243

In [488]:
```
df18['Date_created']=df18.index
df18=df18.reset_index(drop=True)
df18.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df18.columns)
cols = [cols[-1]] + cols[:-1]
df18 = df18[cols]
df18=df18[abs(df18.sensor1-df18.sensor1.mean()) <= (3*df18.sensor1.std
df18=df18[abs(df18.sensor2-df18.sensor2.mean()) <= (3*df18.sensor2.std
df18=df18[abs(df18.sensor3-df18.sensor3.mean()) <= (3*df18.sensor3.std
df18=df18[abs(df18.sensor4-df18.sensor4.mean()) <= (3*df18.sensor4.std
df18=df18.reset_index(drop=True)
df18['Date_created'] = pd.to_datetime(df18['Date_created'], errors='co
df18['day_of_week'] = df18['Date_created'].dt.dayofweek
df18['month'] = pd.DatetimeIndex(df18['Date_created']).month
df18['hour'] = pd.DatetimeIndex(df18['Date_created']).hour
df18[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df18[['
df18['sensor1']=df18['sensor1'].rolling(window=30).var()
df18['sensor2']=df18['sensor2'].rolling(window=30).var()
df18['sensor3']=df18['sensor3'].rolling(window=30).var()
df18['sensor4']=df18['sensor4'].rolling(window=30).var()
```

In [489]:
```
df18['sensor1'].plot()
```

Out[489]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2260ab4c240>
```



In [490]:

```
df18=df18[:450]
df18=df18.fillna(df18.mean())
df18_sensor1=pd.DataFrame(data=df18['sensor1'])
tukey_hinge=df18_sensor1.quantile(0.75)
df18_sensor1['labels']=df18_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df18_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df18_sensor1)-look_back-1):
        a = df18_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df18_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df18_sensor1,look_back=10)
df18_sensor1_final=pd.DataFrame(data=values[0])
df18_sensor1_final['labels']=pd.DataFrame(data=values[1])
df18_sensor1_final.head()
```

Out[490]:

	0	1	2	3	4	5	6	7	8
0	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535
1	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535
2	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535
3	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535
4	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535	0.032535

In [491]:

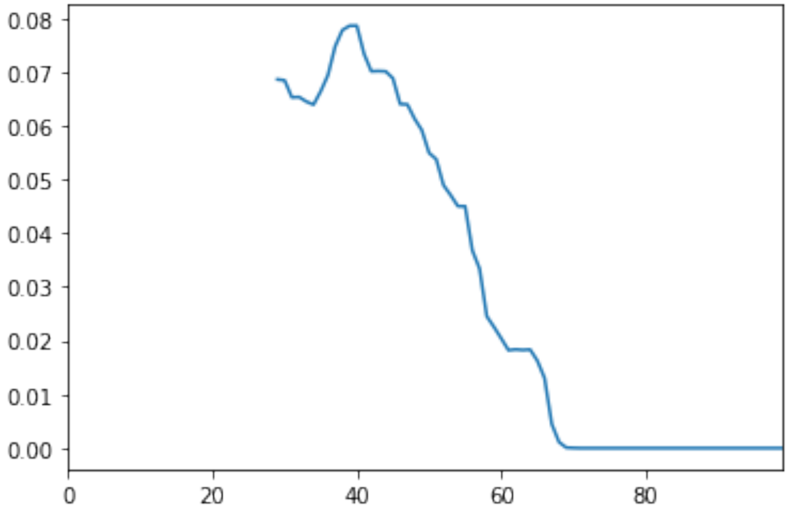
```
df19['Date_created']=df19.index
df19=df19.reset_index(drop=True)
df19.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df19.columns)
cols = [cols[-1]] + cols[:-1]
df19 = df19[cols]
df19=df19[abs(df19.sensor1-df19.sensor1.mean()) <= (3*df19.sensor1.stc
df19=df19[abs(df19.sensor2-df19.sensor2.mean()) <= (3*df19.sensor2.stc
df19=df19[abs(df19.sensor3-df19.sensor3.mean()) <= (3*df19.sensor3.stc
df19=df19[abs(df19.sensor4-df19.sensor4.mean()) <= (3*df19.sensor4.stc
df19=df19.reset_index(drop=True)
df19['Date_created'] = pd.to_datetime(df19['Date_created'], errors='cc
df19['day_of_week'] = df19['Date_created'].dt.dayofweek
df19['month'] = pd.DatetimeIndex(df19['Date_created']).month
df19['hour'] = pd.DatetimeIndex(df19['Date_created']).hour
df19[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df19[['
df19['sensor1']=df19['sensor1'].rolling(window=30).var()
df19['sensor2']=df19['sensor2'].rolling(window=30).var()
df19['sensor3']=df19['sensor3'].rolling(window=30).var()
df19['sensor4']=df19['sensor4'].rolling(window=30).var()
```

In [492]:

```
df19['sensor1'][:100].plot()
```

Out[492]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260be83518>
```



In [493]:

```
df19=df19[:70]
df19=df19.fillna(df19.mean())
df19_sensor1=pd.DataFrame(data=df19['sensor1'])
tukey_hinge=df19_sensor1.quantile(0.75)
df19_sensor1['labels']=df19_sensor1.apply(lambda row:1 if row.sensor1>
def create_dataset(df19_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df19_sensor1)-look_back-1):
        a = df19_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df19_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df19_sensor1,look_back=10)
df19_sensor1_final=pd.DataFrame(data=values[0])
df19_sensor1_final['labels']=pd.DataFrame(data=values[1])
df19_sensor1_final.head()
```

Out[493]:

	0	1	2	3	4	5	6	7	8	
0	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	C
1	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	C
2	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	C
3	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	C
4	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	0.048379	C

In [494]:

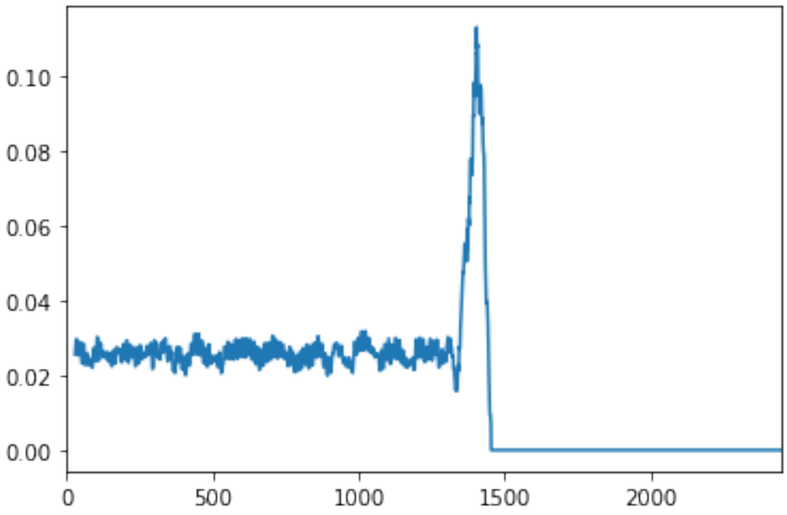
```
df1['Date_created']=df1.index
df1=df1.reset_index(drop=True)
df1.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df1.columns)
cols = [cols[-1]] + cols[:-1]
df1 = df1[cols]
df1=df1[abs(df1.sensor1-df1.sensor1.mean()) <= (3*df1.sensor1.std())]
df1=df1[abs(df1.sensor2-df1.sensor2.mean()) <= (3*df1.sensor2.std())]
df1=df1[abs(df1.sensor3-df1.sensor3.mean()) <= (3*df1.sensor3.std())]
df1=df1[abs(df1.sensor4-df1.sensor4.mean()) <= (3*df1.sensor4.std())]
df1=df1.reset_index(drop=True)
df1['Date_created'] = pd.to_datetime(df1['Date_created'], errors='coe
df1['day_of_week'] = df1['Date_created'].dt.dayofweek
df1['month'] = pd.DatetimeIndex(df1['Date_created']).month
df1['hour'] = pd.DatetimeIndex(df1['Date_created']).hour
df1[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df1[['se
df1['sensor1']=df1['sensor1'].rolling(window=30).var()
df1['sensor2']=df1['sensor2'].rolling(window=30).var()
df1['sensor3']=df1['sensor3'].rolling(window=30).var()
df1['sensor4']=df1['sensor4'].rolling(window=30).var()
```

In [495]:

```
df1['sensor1'].plot()
```

Out[495]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260bea6470>
```



In [496]:

```
df1=df1[:1500]
df1=df1.fillna(df1.mean())
df1_sensor1=pd.DataFrame(data=df1['sensor1'])
tukey_hinge=df1_sensor1.quantile(0.75)
df1_sensor1['labels']=df1_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df1_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df1_sensor1)-look_back-1):
        a = df1_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df1_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df1_sensor1,look_back=10)
df1_sensor1_final=pd.DataFrame(data=values[0])
df1_sensor1_final['labels']=pd.DataFrame(data=values[1])
df1_sensor1_final.head()
```

Out[496]:

```
         0         1         2         3         4         5         6         7         8
0 0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0
1 0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0
2 0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0
3 0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0
4 0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0.027712  0
```

Preparing test data

In [497]:

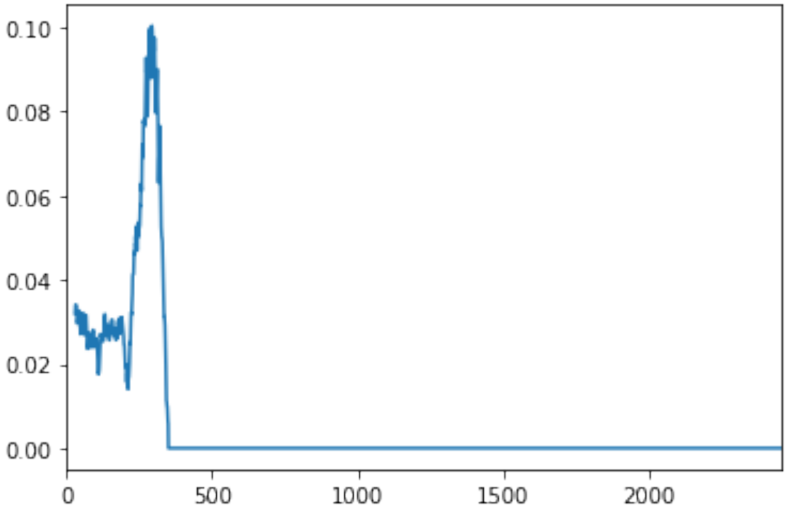
```
df0['Date_created']=df0.index
df0=df0.reset_index(drop=True)
df0.rename(columns={'0':'sensor1',
                          '1':'sensor2',
                          '2':'sensor3','3':'sensor4'},inplace=True)
cols = list(df0.columns)
cols = [cols[-1]] + cols[:-1]
df0 = df0[cols]
df0=df0[abs(df0.sensor1-df0.sensor1.mean()) <= (3*df0.sensor1.std())]
df0=df0[abs(df0.sensor2-df0.sensor2.mean()) <= (3*df0.sensor2.std())]
df0=df0[abs(df0.sensor3-df0.sensor3.mean()) <= (3*df0.sensor3.std())]
df0=df0[abs(df0.sensor4-df0.sensor4.mean()) <= (3*df0.sensor4.std())]
df0=df0.reset_index(drop=True)
df0['Date_created'] = pd.to_datetime(df0['Date_created'], errors='coe
df0['day_of_week'] = df0['Date_created'].dt.dayofweek
df0['month'] = pd.DatetimeIndex(df0['Date_created']).month
df0['hour'] = pd.DatetimeIndex(df0['Date_created']).hour
df0[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df0[['se
df0['sensor1']=df0['sensor1'].rolling(window=30).var()
df0['sensor2']=df0['sensor2'].rolling(window=30).var()
df0['sensor3']=df0['sensor3'].rolling(window=30).var()
df0['sensor4']=df0['sensor4'].rolling(window=30).var()
```

In [498]:

```
df0['sensor1'].plot()
```

Out[498]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2260bd69f28>
```



In [499]:

```
df0=df0[:400]
df0=df0.fillna(df0.mean())
df0_sensor1=pd.DataFrame(data=df0['sensor1'])
tukey_hinge=df0_sensor1.quantile(0.75)
df0_sensor1['labels']=df0_sensor1.apply(lambda row:1 if row.sensor1>tu
def create_dataset(df0_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df0_sensor1)-look_back-1):
        a = df0_sensor1['sensor1'][i:(i+look_back)]
        dataX.append(a)
        dataY.append(df0_sensor1['labels'][i + look_back])
    return np.array(dataX), np.array(dataY)
values=create_dataset(df0_sensor1,look_back=10)
df0_sensor1_final=pd.DataFrame(data=values[0])
df0_sensor1_final['labels']=pd.DataFrame(data=values[1])
df0_sensor1_final.head()
```

Out[499]:

	0	1	2	3	4	5	6	7	8	
0	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0
1	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0
2	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0
3	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0
4	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0.035645	0

Generating the data that needs to be sent for LSTM network

In [500]:

```
def generate_data(lst):
    features = []
    labels = []
    for df in lst:
        f = np.array(df.iloc[:,0:10])
        l = np.array(df.iloc[:,10].astype(int))
        features.append(f)
        labels.append(l)
        feature_output = np.concatenate(features)
    return feature_output.reshape(feature_output.shape[0], feature_out
```

Sensor1 values of all the dataframes will be used as trainset

In [501]:
```
x_train, y_train = generate_data([df1_sensor1_final,df2_sensor1_final,
```

Model building for LSTM network

In [539]:
```
model = Sequential()
model.add(LSTM(100, input_shape=(10,1)))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam')
print(model.summary())
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_11 (LSTM)               (None, 100)               40800     
_________________________________________________________________
dense_36 (Dense)             (None, 10)                1010      
_________________________________________________________________
dense_37 (Dense)             (None, 10)                110       
_________________________________________________________________
dense_38 (Dense)             (None, 1)                 11        
=================================================================
Total params: 41,931
Trainable params: 41,931
Non-trainable params: 0
_________________________________________________________________
None
```

Model building for GRU network

In [546]:

```
model_GRU = Sequential()
model_GRU.add(LSTM(100, input_shape=(10,1)))
model_GRU.add(Dense(10, activation='relu'))
model_GRU.add(Dense(10, activation='relu'))
model_GRU.add(Dense(1, activation='sigmoid'))
model_GRU.compile(loss='binary_crossentropy', optimizer='adam')
print(model_GRU.summary())
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_12 (LSTM)               (None, 100)               40800     
_________________________________________________________________
dense_39 (Dense)             (None, 10)                1010      
_________________________________________________________________
dense_40 (Dense)             (None, 10)                110       
_________________________________________________________________
dense_41 (Dense)             (None, 1)                 11        
=================================================================
Total params: 41,931
Trainable params: 41,931
Non-trainable params: 0
_________________________________________________________________
None
```

Fitting the model

In [540]:

```
model.fit(x_train, y_train, epochs=20)
```

```
Train on 12591 samples
Epoch 1/20
12591/12591 [==============================] - 8s 638us/sample - loss
: 0.5422
Epoch 2/20
12591/12591 [==============================] - 5s 370us/sample - loss
: 0.4513
Epoch 3/20
12591/12591 [==============================] - 4s 347us/sample - loss
: 0.4444s - lo
Epoch 4/20
12591/12591 [==============================] - 4s 348us/sample - loss
: 0.4420
Epoch 5/20
12591/12591 [==============================] - 4s 346us/sample - loss
: 0.4377
Epoch 6/20
```

```
12591/12591 [==============================] - 5s 358us/sample - loss
: 0.4345
Epoch 7/20
12591/12591 [==============================] - 4s 346us/sample - loss
: 0.4314
Epoch 8/20
12591/12591 [==============================] - 4s 341us/sample - loss
: 0.4326
Epoch 9/20
12591/12591 [==============================] - 5s 371us/sample - loss
: 0.4309
Epoch 10/20
12591/12591 [==============================] - 5s 373us/sample - loss
: 0.4301
Epoch 11/20
12591/12591 [==============================] - 5s 371us/sample - loss
: 0.4292
Epoch 12/20
12591/12591 [==============================] - 5s 364us/sample - loss
: 0.4275
Epoch 13/20
12591/12591 [==============================] - 4s 351us/sample - loss
: 0.4238
Epoch 14/20
12591/12591 [==============================] - 4s 318us/sample - loss
: 0.4198
Epoch 15/20
12591/12591 [==============================] - 5s 363us/sample - loss
: 0.4115s - loss: 0
Epoch 16/20
12591/12591 [==============================] - 4s 311us/sample - loss
: 0.3907
Epoch 17/20
12591/12591 [==============================] - 4s 325us/sample - loss
: 0.3626
Epoch 18/20
12591/12591 [==============================] - 4s 337us/sample - loss
: 0.3544
Epoch 19/20
12591/12591 [==============================] - 5s 369us/sample - loss
: 0.3443
Epoch 20/20
12591/12591 [==============================] - 4s 339us/sample - loss
: 0.3392
```

```
Out[540]: <tensorflow.python.keras.callbacks.History at 0x22622fcdcc0>
```

```
In [547]: model_GRU.fit(x_train, y_train, epochs=20)
```

```
Train on 12591 samples
```

```
Epoch 1/20
12591/12591 [==============================] - 7s 576us/sample - loss
: 0.4918
Epoch 2/20
12591/12591 [==============================] - 4s 346us/sample - loss
: 0.4488
Epoch 3/20
12591/12591 [==============================] - 4s 322us/sample - loss
: 0.4422
Epoch 4/20
12591/12591 [==============================] - 4s 350us/sample - loss
: 0.4357
Epoch 5/20
12591/12591 [==============================] - 4s 343us/sample - loss
: 0.4369
Epoch 6/20
12591/12591 [==============================] - 5s 361us/sample - loss
: 0.4349
Epoch 7/20
12591/12591 [==============================] - 4s 349us/sample - loss
: 0.4319
Epoch 8/20
12591/12591 [==============================] - 5s 368us/sample - loss
: 0.4280s - loss:
Epoch 9/20
12591/12591 [==============================] - 4s 356us/sample - loss
: 0.4243
Epoch 10/20
12591/12591 [==============================] - 5s 370us/sample - loss
: 0.4171
Epoch 11/20
12591/12591 [==============================] - 5s 378us/sample - loss
: 0.4148
Epoch 12/20
12591/12591 [==============================] - 5s 378us/sample - loss
: 0.4026
Epoch 13/20
12591/12591 [==============================] - 5s 365us/sample - loss
: 0.3730
Epoch 14/20
12591/12591 [==============================] - 5s 364us/sample - loss
: 0.3485
Epoch 15/20
12591/12591 [==============================] - 5s 370us/sample - loss
: 0.3391
Epoch 16/20
12591/12591 [==============================] - 5s 365us/sample - loss
: 0.3314
Epoch 17/20
12591/12591 [==============================] - 4s 349us/sample - loss
```

```
: 0.3300
Epoch 18/20
12591/12591 [==============================] - 4s 315us/sample - loss
: 0.3246
Epoch 19/20
12591/12591 [==============================] - 4s 344us/sample - loss
: 0.3208
Epoch 20/20
12591/12591 [==============================] - 5s 377us/sample - loss
: 0.3178
```

Out[547]:

```
<tensorflow.python.keras.callbacks.History at 0x22626ee83c8>
```

Creating the test_data

In [541]:

```
test_data,test_label=gen_data([df0_sensor1_final])
```

Checking for the accuracy

In [542]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
```

In [543]:

```
def predict(model, test_data, test_label):
    pred = model.predict(test_data)
    pred = pred.round()
    try:
        tn, fp, fn, tp = confusion_matrix(pred.flatten(), test_label).
    except:
        tn = 0
        fp = 0
        fn = 0
        tp = 0

    acc = (pred.flatten() == test_label.flatten()).sum()/ len(test_lab
    pred_conc = np.concatenate([np.zeros(10),pred.flatten()])
    return {
        'pred': pred.flatten(),
        'pred_conc':pred_conc,
        'conf_mat': (tn, fp, fn, tp),
        'acc': acc
    }
```

In [544]:

```
predict_values_LSTM=predict(model,test_data,test_label)
```

In [548]:
```
predict_values_GRU=predict(model_GRU,test_data,test_label)
```

In [545]:
```
print(predict_values_LSTM)
```

```
{'pred': array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0
., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1.
, 1.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
      dtype=float32), 'pred_conc': array([0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 1.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0.]), 'conf_mat': (276, 13, 13, 87
), 'acc': 0.9331619537275064}
```

In [549]:
```
print(predict_values_GRU)
```

```
{'pred': array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0
., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.
, 1.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
      dtype=float32), 'pred_conc': array([0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 1.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
, 1.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1.
, 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
, 0.,
       0., 0., 0., 0., 0., 0., 0., 0.]), 'conf_mat': (278, 21, 11, 79
), 'acc': 0.9177377892030848}
```

How the model works:

1.Removing the outliers or noise in the initial stages of Data Preparation

2.Applying Moving Variance for the sensor values in DataFrame

3.Considering Tukey hinge into account and labeling the values which are greater than 0.75 quantile(threshold)

4.If we have labeled Y_values according to threshold(s(t)>threshold) values then the next step is to make X_values as s(t-1),s(t-2),...s(t-10)

5.Passing all the values into different models like LSTM, GRU and predicting the values by the past 10 time steps

6.In this case, I have taken machine_0 values as testset and concatenated all the remaining machine values for each different sensor

Models used:

LSTM, GRU

Accuracies:

LSTM:

Confusion Matrix: (276, 13, 13, 87)

Accuracy:93.31

GRU:

Confusion Matrix: (278, 21, 11, 79)

Accuracy:91.77

In [ ]: