

```
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
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
```

```
In [2]: 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 [3]: 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 [4]: 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 [5]: df2=df2.reset_index(drop=True)
df2['Date_created'] = pd.to_datetime(df2['Date_created'], errors='coerce')
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 [6]: df2[['sensor1','sensor2','sensor3','sensor4']] = minmax_scale(df2[['sensor1','sensor2','sensor3','sensor4']])
```

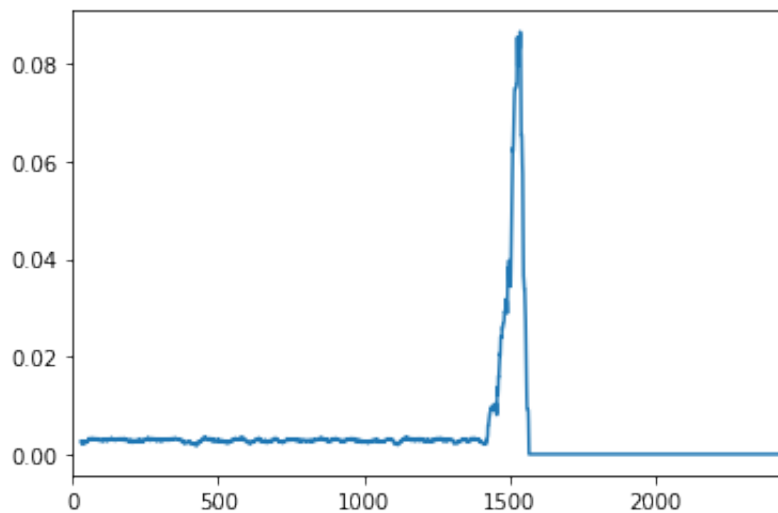
we use the moving variance to approximate the variation in the series at a point in time

```
In [7]: 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 [9]: df2['sensor2'].plot()
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1e353071e48>
```



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

```
In [10]: df2=df2[:1600]
df2=df2.fillna(df2.mean())
df2_sensor1=pd.DataFrame(data=df2['sensor2'])
```

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 [12]: tukey_hinge=df2_sensor1.quantile(0.75)
df2_sensor1['labels']=df2_sensor1.apply(lambda row:1 if row.sensor2>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 [13]: def create_dataset(df2_sensor1, look_back=10):
          dataX, dataY = [], []
          for i in range(len(df2_sensor1)-look_back-1):
              a = df2_sensor1['sensor2'][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 [14]: 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[14]:
```

	0	1	2	3	4	5	6	7	8	
0	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	C
1	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	C
2	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	C
3	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	C
4	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	0.005648	C

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

```

In [15]: 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='coerce')
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[['sensor1', 'sensor2', 'sensor3', 'sensor4']])
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()

```

```

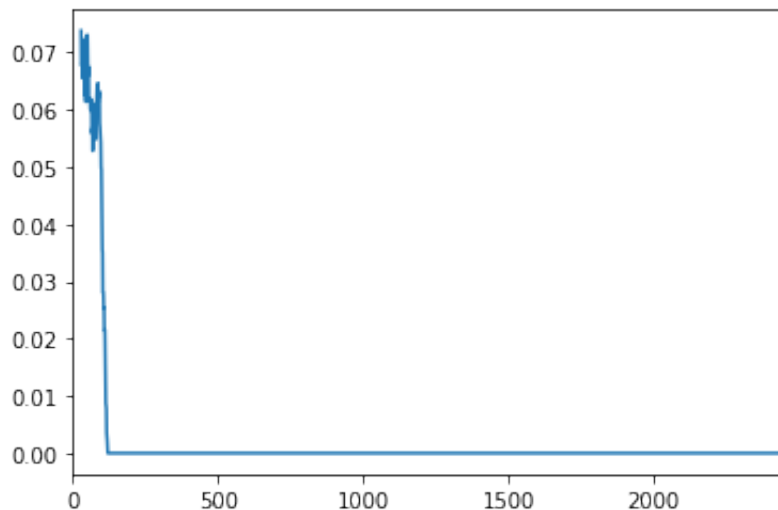
In [16]: df3['sensor2'].plot()

```

```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1e355639400>

```



```

In [18]: df3=df3[:130]
df3=df3.fillna(df3.mean())
df3_sensor1=pd.DataFrame(data=df3['sensor2'])
tukey_hinge=df3_sensor1.quantile(0.75)
df3_sensor1['labels']=df3_sensor1.apply(lambda row:1 if row.sensor2>tukey_hinge else 0,axis=1)
def create_dataset(df3_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df3_sensor1)-look_back-1):
        a = df3_sensor1['sensor2'][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[18]:

```

	0	1	2	3	4	5	6	7	8	
0	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	C
1	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	C
2	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	C
3	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	C
4	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	0.048019	C

```

In [19]: 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='coerce')
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[['sensor1', 'sensor2', 'sensor3', 'sensor4']])
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()

```

```

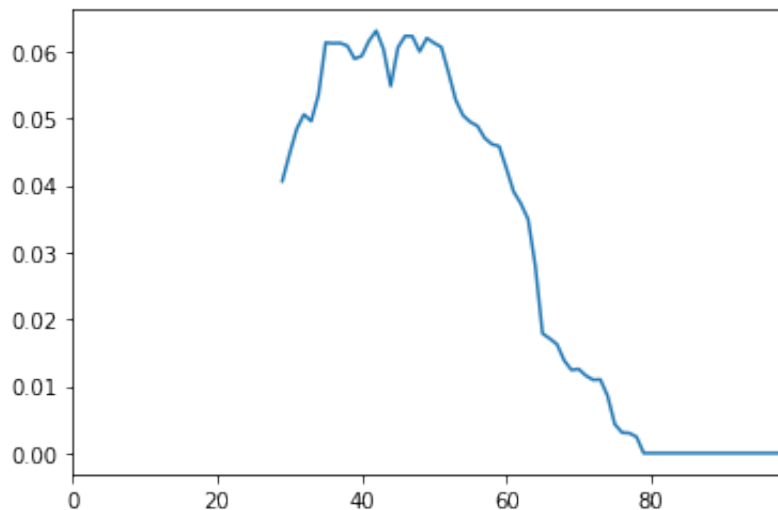
In [20]: df4['sensor2'][:100].plot()

```

```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1e35582c780>

```



```

In [21]: df4=df4[:80]
df4=df4.fillna(df4.mean())
df4_sensor1=pd.DataFrame(data=df4['sensor2'])
tukey_hinge=df4_sensor1.quantile(0.75)
df4_sensor1['labels']=df4_sensor1.apply(lambda row:1 if row.sensor2>tukey_hinge else 0,axis=1)
def create_dataset(df4_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df4_sensor1)-look_back-1):
        a = df4_sensor1['sensor2'][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[21]:

```

	0	1	2	3	4	5	6	7	8	
0	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	C
1	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	C
2	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	C
3	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	C
4	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	0.040041	C


```

In [22]: 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='coerce')
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[['sensor1', 'sensor2', 'sensor3', 'sensor4']])
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()

```

```

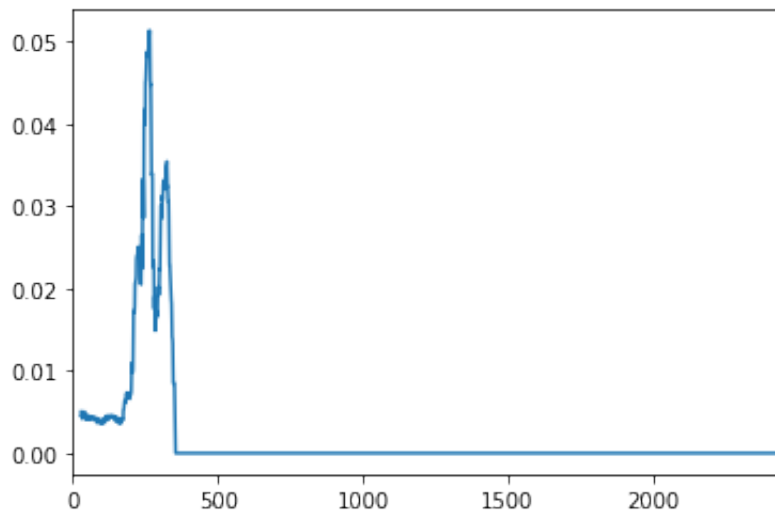
In [23]: df5['sensor2'].plot()

```

```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1e352f09278>

```



```

In [24]: df5=df5[:370]
df5=df5.fillna(df5.mean())
df5_sensor1=pd.DataFrame(data=df5['sensor2'])
tukey_hinge=df5_sensor1.quantile(0.75)
df5_sensor1['labels']=df5_sensor1.apply(lambda row:1 if row.sensor2>tukey_hinge else 0,axis=1)
def create_dataset(df5_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df5_sensor1)-look_back-1):
        a = df5_sensor1['sensor2'][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[24]:

```

	0	1	2	3	4	5	6	7	8	9	la
0	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	
1	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	
2	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	
3	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	
4	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	0.01421	

```

In [25]: 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='coerce')
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[['sensor1', 'sensor2', 'sensor3', 'sensor4']])
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()

```

```

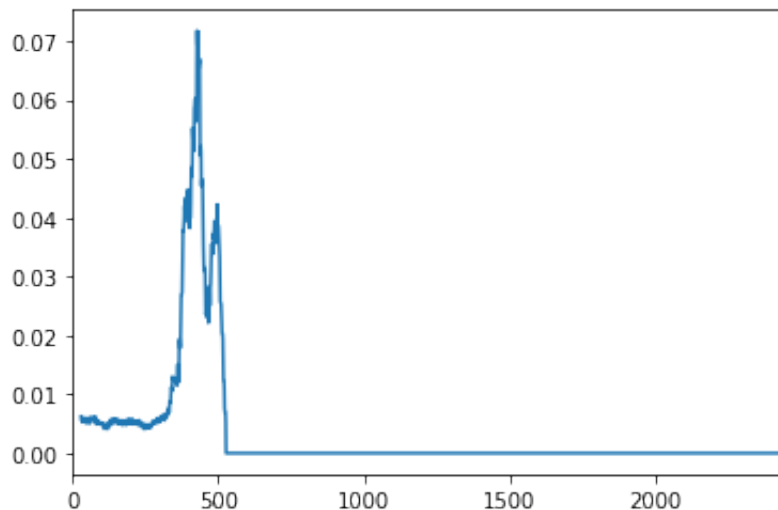
In [26]: df6['sensor2'].plot()

```

```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1e353037400>

```



```

In [27]: df6=df6[:550]
df6=df6.fillna(df6.mean())
df6_sensor1=pd.DataFrame(data=df6['sensor2'])
tukey_hinge=df6_sensor1.quantile(0.75)
df6_sensor1['labels']=df6_sensor1.apply(lambda row:1 if row.sensor2>tukey_hinge else 0,axis=1)
def create_dataset(df6_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df6_sensor1)-look_back-1):
        a = df6_sensor1['sensor2'][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[27]:

```

	0	1	2	3	4	5	6	7	8	
0	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	C
1	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	C
2	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	C
3	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	C
4	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	0.015426	C

```

In [28]: 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='coerce')
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[['sensor1', 'sensor2', 'sensor3', 'sensor4']])
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()

```

```

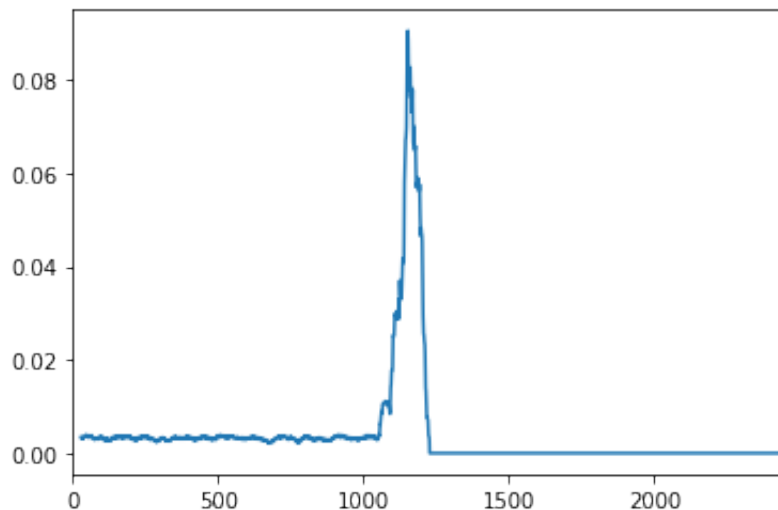
In [29]: df7['sensor2'].plot()

```

```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1e352f78400>

```



```

In [30]: df7=df7[:1300]
df7=df7.fillna(df7.mean())
df7_sensor1=pd.DataFrame(data=df7['sensor2'])
tukey_hinge=df7_sensor1.quantile(0.75)
df7_sensor1['labels']=df7_sensor1.apply(lambda row:1 if row.sensor2>tukey_hinge else 0,axis=1)
def create_dataset(df7_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df7_sensor1)-look_back-1):
        a = df7_sensor1['sensor2'][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[30]:

```

	0	1	2	3	4	5	6	7	8	
0	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	C
1	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	C
2	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	C
3	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	C
4	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	0.007551	C

```

In [31]: 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='coerce')
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[['sensor1', 'sensor2', 'sensor3', 'sensor4']])
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()

```

```

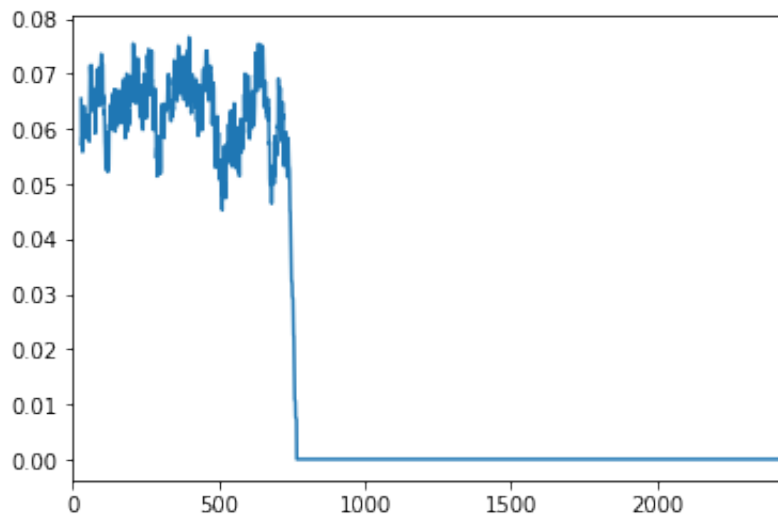
In [33]: df8['sensor2'].plot()

```

```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1e35563c208>

```



```

In [34]: df8=df8[:900]
df8=df8.fillna(df8.mean())
df8_sensor1=pd.DataFrame(data=df8['sensor2'])
tukey_hinge=df8_sensor1.quantile(0.75)
df8_sensor1['labels']=df8_sensor1.apply(lambda row:1 if row.sensor2>tu
def create_dataset(df8_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df8_sensor1)-look_back-1):
        a = df8_sensor1['sensor2'][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[34]:

```

	0	1	2	3	4	5	6	7	8	
0	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	C
1	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	C
2	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	C
3	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	C
4	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	0.052667	C


```

In [35]: 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='coerce')
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[['sensor1', 'sensor2', 'sensor3', 'sensor4']])
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()

```

```

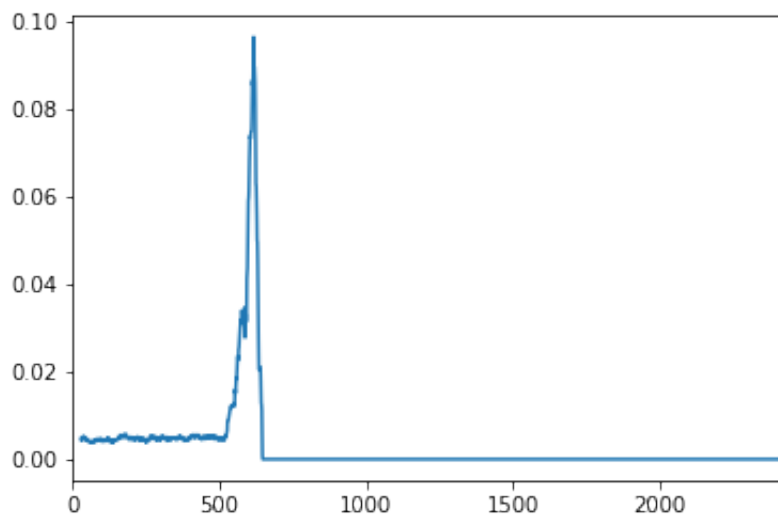
In [36]: df9['sensor2'].plot()

```

```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1e3558e95f8>

```



```

In [37]: df9=df9[:750]
df9=df9.fillna(df9.mean())
df9_sensor1=pd.DataFrame(data=df9['sensor2'])
tukey_hinge=df9_sensor1.quantile(0.75)
df9_sensor1['labels']=df9_sensor1.apply(lambda row:1 if row.sensor2>tukey_hinge else 0,axis=1)
def create_dataset(df9_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df9_sensor1)-look_back-1):
        a = df9_sensor1['sensor2'][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[37]:

```

	0	1	2	3	4	5	6	7	8	
0	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	C
1	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	C
2	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	C
3	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	C
4	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	0.009324	C

```

In [38]: 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.stc
df10=df10[abs(df10.sensor2-df10.sensor2.mean()) <= (3*df10.sensor2.stc
df10=df10[abs(df10.sensor3-df10.sensor3.mean()) <= (3*df10.sensor3.stc
df10=df10[abs(df10.sensor4-df10.sensor4.mean()) <= (3*df10.sensor4.stc
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()

```

```

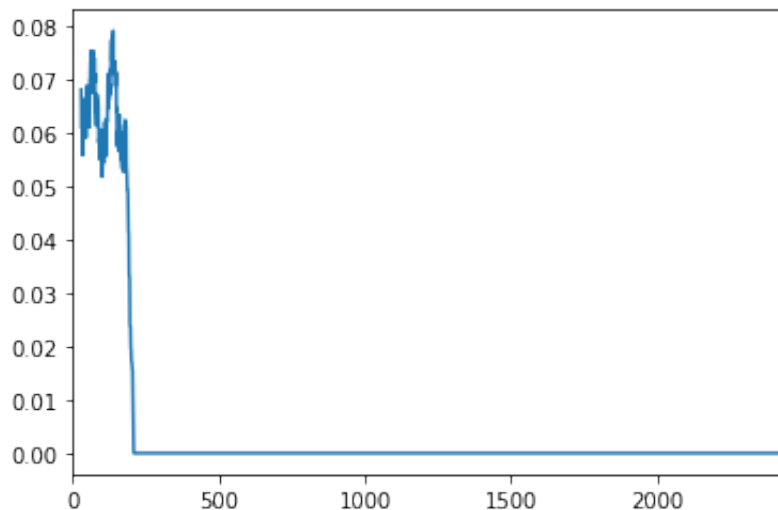
In [40]: df10['sensor2'].plot()

```

```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1e352d6cc18>

```



```

In [41]: df10=df10[:750]
df10=df10.fillna(df10.mean())
df10_sensor1=pd.DataFrame(data=df10['sensor2'])
tukey_hinge=df10_sensor1.quantile(0.75)
df10_sensor1['labels']=df10_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df10_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df10_sensor1)-look_back-1):
        a = df10_sensor1['sensor2'][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[41]:

```

	0	1	2	3	4	5	6	7	8	
0	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	C
1	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	C
2	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	C
3	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	C
4	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	0.014799	C

```

In [42]: 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.stc
df11=df11[abs(df11.sensor2-df11.sensor2.mean()) <= (3*df11.sensor2.stc
df11=df11[abs(df11.sensor3-df11.sensor3.mean()) <= (3*df11.sensor3.stc
df11=df11[abs(df11.sensor4-df11.sensor4.mean()) <= (3*df11.sensor4.stc
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()

```

```

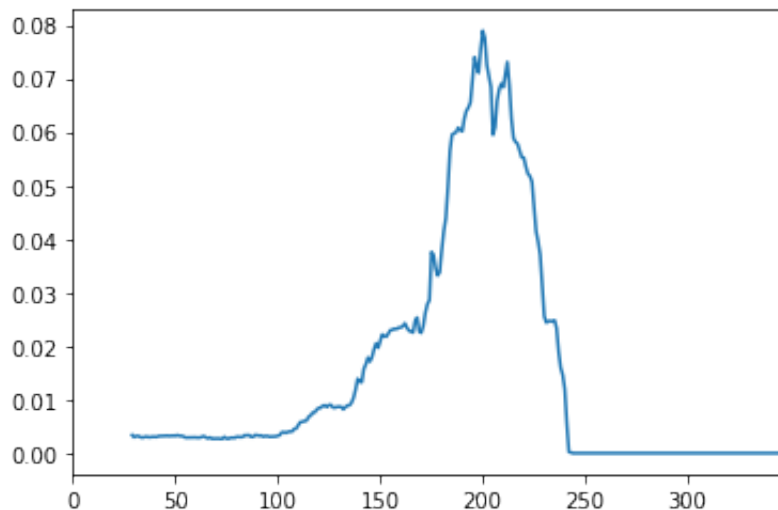
In [43]: df11['sensor2'][:350].plot()

```

```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x1e355893128>

```



```

In [44]: df11=df11[:250]
df11=df11.fillna(df11.mean())
df11_sensor1=pd.DataFrame(data=df11['sensor2'])
tukey_hinge=df11_sensor1.quantile(0.75)
df11_sensor1['labels']=df11_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df11_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df11_sensor1)-look_back-1):
        a = df11_sensor1['sensor2'][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[44]:

```

	0	1	2	3	4	5	6	7	8	9	la
0	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	
1	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	
2	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	
3	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	
4	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	0.02114	

```

In [45]: 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.stc
df12=df12[abs(df12.sensor2-df12.sensor2.mean()) <= (3*df12.sensor2.stc
df12=df12[abs(df12.sensor3-df12.sensor3.mean()) <= (3*df12.sensor3.stc
df12=df12[abs(df12.sensor4-df12.sensor4.mean()) <= (3*df12.sensor4.stc
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()

```

```

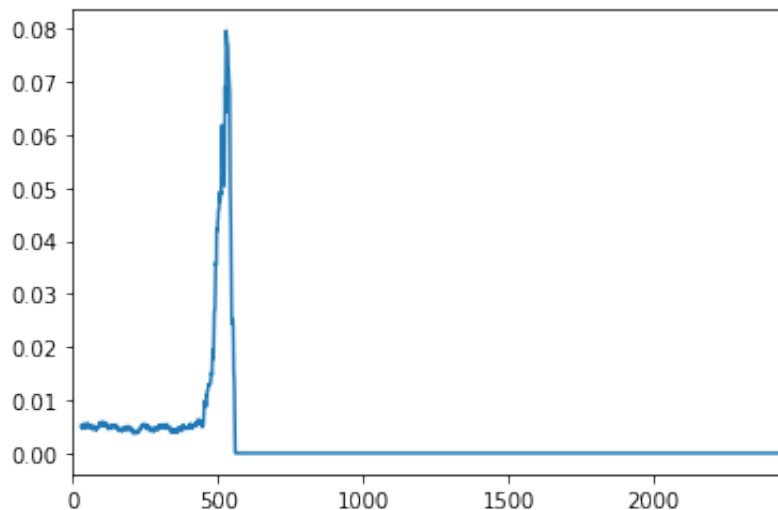
In [46]: df12['sensor2'].plot()

```

```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x1e352da3208>

```



```

In [47]: df12=df12[:600]
df12=df12.fillna(df12.mean())
df12_sensor1=pd.DataFrame(data=df12['sensor2'])
tukey_hinge=df12_sensor1.quantile(0.75)
df12_sensor1['labels']=df12_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df12_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df12_sensor1)-look_back-1):
        a = df12_sensor1['sensor2'][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[47]:

```

	0	1	2	3	4	5	6	7	8	
0	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	C
1	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	C
2	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	C
3	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	C
4	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	0.010461	C


```

In [48]: 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.sto
df13=df13[abs(df13.sensor2-df13.sensor2.mean()) <= (3*df13.sensor2.sto
df13=df13[abs(df13.sensor3-df13.sensor3.mean()) <= (3*df13.sensor3.sto
df13=df13[abs(df13.sensor4-df13.sensor4.mean()) <= (3*df13.sensor4.sto
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()

```

```

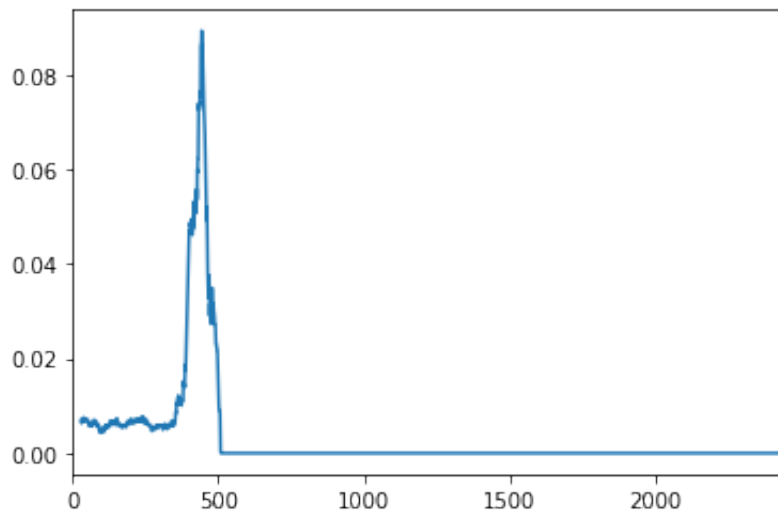
In [49]: df13['sensor2'].plot()

```

```

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x1e3530f0438>

```



```

In [50]: df13=df13[:500]
df13=df13.fillna(df13.mean())
df13_sensor1=pd.DataFrame(data=df13['sensor2'])
tukey_hinge=df13_sensor1.quantile(0.75)
df13_sensor1['labels']=df13_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df13_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df13_sensor1)-look_back-1):
        a = df13_sensor1['sensor2'][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[50]:

```

	0	1	2	3	4	5	6	7	8	
0	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	C
1	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	C
2	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	C
3	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	C
4	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	0.016397	C

```

In [51]: 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.stc
df14=df14[abs(df14.sensor2-df14.sensor2.mean()) <= (3*df14.sensor2.stc
df14=df14[abs(df14.sensor3-df14.sensor3.mean()) <= (3*df14.sensor3.stc
df14=df14[abs(df14.sensor4-df14.sensor4.mean()) <= (3*df14.sensor4.stc
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()

```

```

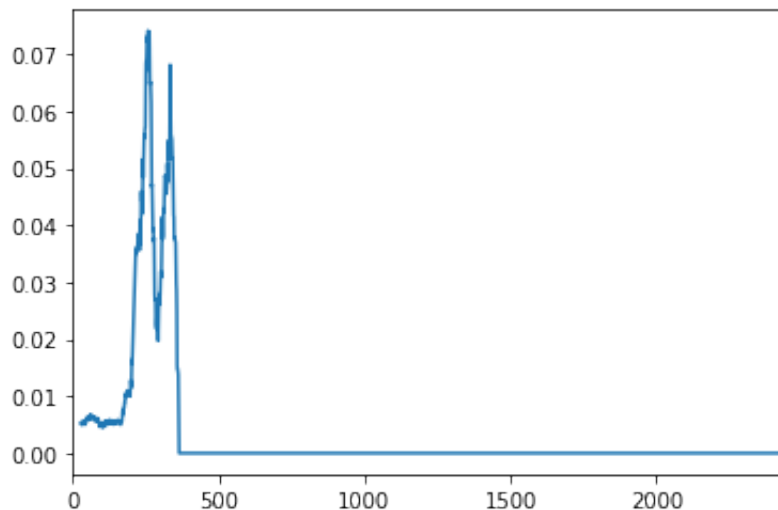
In [52]: df14['sensor2'].plot()

```

```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1e35311b828>

```



```

In [53]: df14=df14[:500]
df14=df14.fillna(df14.mean())
df14_sensor1=pd.DataFrame(data=df14['sensor2'])
tukey_hinge=df14_sensor1.quantile(0.75)
df14_sensor1['labels']=df14_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df14_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df14_sensor1)-look_back-1):
        a = df14_sensor1['sensor2'][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[53]:

```

	0	1	2	3	4	5	6	7	8	9	la
0	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	
1	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	
2	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	
3	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	
4	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	0.01669	

```

In [54]: 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.sto
df15=df15[abs(df15.sensor2-df15.sensor2.mean()) <= (3*df15.sensor2.sto
df15=df15[abs(df15.sensor3-df15.sensor3.mean()) <= (3*df15.sensor3.sto
df15=df15[abs(df15.sensor4-df15.sensor4.mean()) <= (3*df15.sensor4.sto
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()

```

```

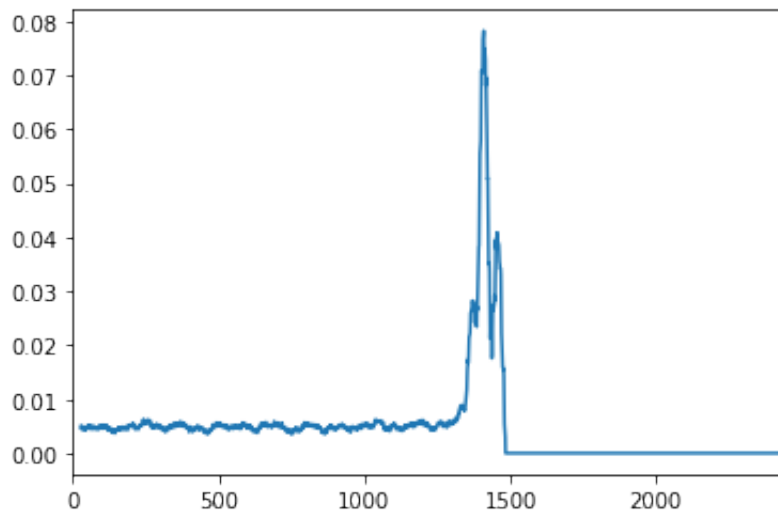
In [55]: df15['sensor2'].plot()

```

```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x1e355773240>

```



```

In [56]: df15=df15[:1450]
df15=df15.fillna(df15.mean())
df15_sensor1=pd.DataFrame(data=df15['sensor2'])
tukey_hinge=df15_sensor1.quantile(0.75)
df15_sensor1['labels']=df15_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df15_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df15_sensor1)-look_back-1):
        a = df15_sensor1['sensor2'][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[56]:

```

	0	1	2	3	4	5	6	7	8	
0	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	C
1	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	C
2	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	C
3	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	C
4	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	0.007309	C

```

In [57]: 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='co
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()

```

```

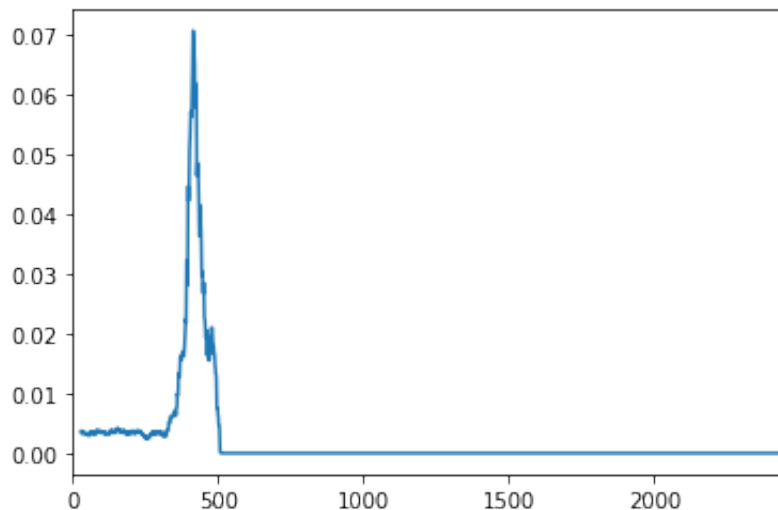
In [58]: df16['sensor2'].plot()

```

```

Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x1e355768b00>

```



```

In [59]: df16=df16[:500]
df16=df16.fillna(df16.mean())
df16_sensor1=pd.DataFrame(data=df16['sensor2'])
tukey_hinge=df16_sensor1.quantile(0.75)
df16_sensor1['labels']=df16_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df16_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df16_sensor1)-look_back-1):
        a = df16_sensor1['sensor2'][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[59]:

```

	0	1	2	3	4	5	6	7	8	9	la
0	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	
1	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	
2	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	
3	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	
4	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	0.01142	


```

In [60]: 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='co
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()

```

```

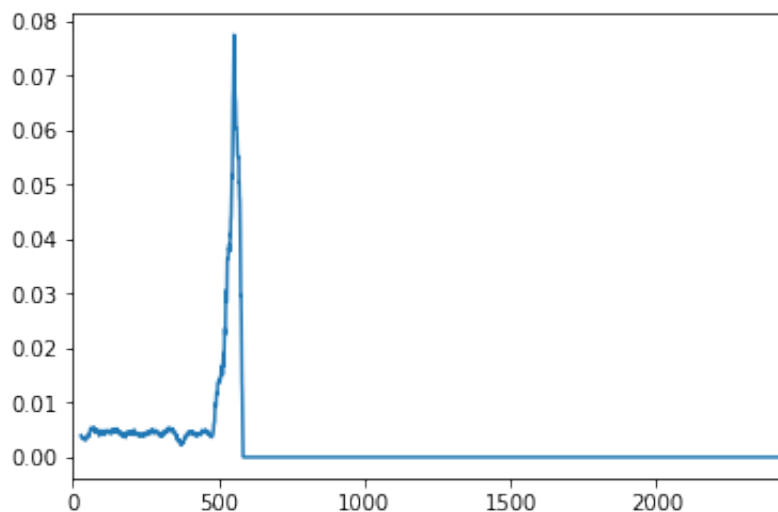
In [61]: df17['sensor2'].plot()

```

```

Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x1e356a564e0>

```



```

In [62]: df17=df17[:550]
df17=df17.fillna(df17.mean())
df17_sensor1=pd.DataFrame(data=df17['sensor2'])
tukey_hinge=df17_sensor1.quantile(0.75)
df17_sensor1['labels']=df17_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df17_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df17_sensor1)-look_back-1):
        a = df17_sensor1['sensor2'][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[62]:

```

	0	1	2	3	4	5	6	7	8	
0	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	C
1	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	C
2	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	C
3	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	C
4	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	0.006899	C

```

In [63]: 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.stc
df18=df18[abs(df18.sensor2-df18.sensor2.mean()) <= (3*df18.sensor2.stc
df18=df18[abs(df18.sensor3-df18.sensor3.mean()) <= (3*df18.sensor3.stc
df18=df18[abs(df18.sensor4-df18.sensor4.mean()) <= (3*df18.sensor4.stc
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()

```

```

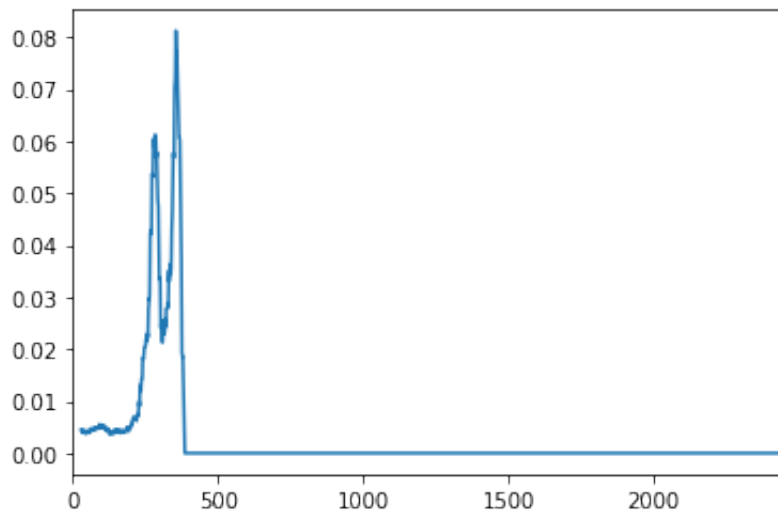
In [64]: df18['sensor2'].plot()

```

```

Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x1e353210278>

```



```

In [65]: df18=df18[:450]
df18=df18.fillna(df18.mean())
df18_sensor1=pd.DataFrame(data=df18['sensor2'])
tukey_hinge=df18_sensor1.quantile(0.75)
df18_sensor1['labels']=df18_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df18_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df18_sensor1)-look_back-1):
        a = df18_sensor1['sensor2'][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[65]:

```

	0	1	2	3	4	5	6	7	8	9	la
0	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	
1	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	
2	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	
3	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	
4	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	0.01619	

```

In [66]: 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='co
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()

```

```

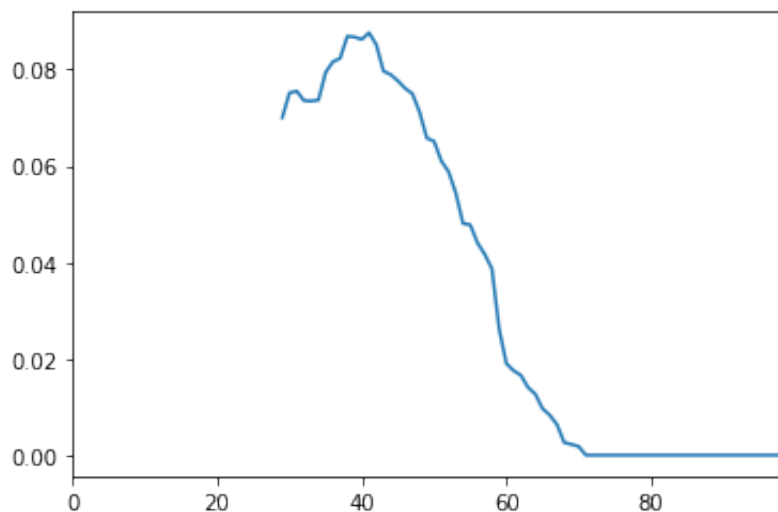
In [67]: df19['sensor2'][:100].plot()

```

```

Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x1e35694c9b0>

```



```

In [68]: df19=df19[:70]
df19=df19.fillna(df19.mean())
df19_sensor1=pd.DataFrame(data=df19['sensor2'])
tukey_hinge=df19_sensor1.quantile(0.75)
df19_sensor1['labels']=df19_sensor1.apply(lambda row:1 if row.sensor2>
def create_dataset(df19_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df19_sensor1)-look_back-1):
        a = df19_sensor1['sensor2'][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[68]:

```

	0	1	2	3	4	5	6	7	8	
0	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	C
1	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	C
2	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	C
3	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	C
4	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	0.054539	C

```

In [69]: 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='coerce')
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[['sensor1', 'sensor2', 'sensor3', 'sensor4']])
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()

```

```

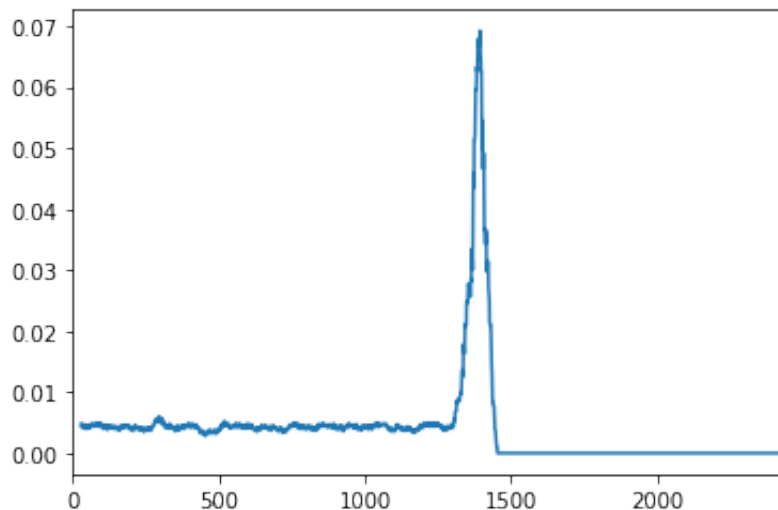
In [70]: df1['sensor2'].plot()

```

```

Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x1e356afeb70>

```



```

In [71]: df1=df1[:1500]
df1=df1.fillna(df1.mean())
df1_sensor1=pd.DataFrame(data=df1['sensor2'])
tukey_hinge=df1_sensor1.quantile(0.75)
df1_sensor1['labels']=df1_sensor1.apply(lambda row:1 if row.sensor2>tukey_hinge else 0,axis=1)
def create_dataset(df1_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df1_sensor1)-look_back-1):
        a = df1_sensor1['sensor2'][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[71]:

```

	0	1	2	3	4	5	6	7	8	
0	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	C
1	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	C
2	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	C
3	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	C
4	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	0.006535	C

Preparing test data


```

In [72]: 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='coer
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()

```

```

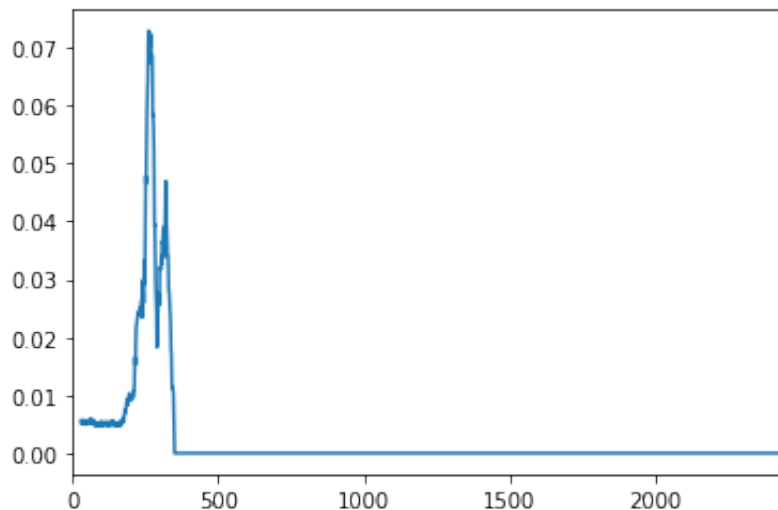
In [73]: df0['sensor2'].plot()

```

```

Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x1e353299be0>

```



```
In [74]: df0=df0[:400]
df0=df0.fillna(df0.mean())
df0_sensor1=pd.DataFrame(data=df0['sensor2'])
tukey_hinge=df0_sensor1.quantile(0.75)
df0_sensor1['labels']=df0_sensor1.apply(lambda row:1 if row.sensor2>tukey_hinge else 0,axis=1)
def create_dataset(df0_sensor1, look_back=10):
    dataX, dataY = [], []
    for i in range(len(df0_sensor1)-look_back-1):
        a = df0_sensor1['sensor2'][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[74]:
```

	0	1	2	3	4	5	6	7	8	
0	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	C
1	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	C
2	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	C
3	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	C
4	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	0.015923	C

Generating the data that needs to be sent for LSTM network

```
In [75]: 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_output.shape[1])
```

Sensor1 values of all the dataframes will be used as trainset

```
In [76]: x_train, y_train = generate_data([df1_sensor1_final, df2_sensor1_final,
```

Model building for LSTM network

```
In [77]: 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"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100)	40800
dense (Dense)	(None, 10)	1010
dense_1 (Dense)	(None, 10)	110
dense_2 (Dense)	(None, 1)	11
Total params: 41,931		
Trainable params: 41,931		
Non-trainable params: 0		
None		

```
In [78]: 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_1"

Layer (type)	Output Shape	Param #
=====		
lstm_1 (LSTM)	(None, 100)	40800
dense_3 (Dense)	(None, 10)	1010
dense_4 (Dense)	(None, 10)	110
dense_5 (Dense)	(None, 1)	11
=====		
Total params: 41,931		
Trainable params: 41,931		
Non-trainable params: 0		

None

Fitting the model

```
In [79]: model.fit(x_train, y_train, epochs=20)
```

Train on 12591 samples

Epoch 1/20

12591/12591 [=====] - 4s 321us/sample - loss : 0.4917

Epoch 2/20

12591/12591 [=====] - 2s 184us/sample - loss : 0.4422

Epoch 3/20

12591/12591 [=====] - 2s 183us/sample - loss : 0.4353

Epoch 4/20

12591/12591 [=====] - 2s 182us/sample - loss : 0.4347

Epoch 5/20

12591/12591 [=====] - 2s 187us/sample - loss : 0.4333

Epoch 6/20

```
12591/12591 [=====] - 2s 185us/sample - loss
: 0.4306
Epoch 7/20
12591/12591 [=====] - 2s 184us/sample - loss
: 0.4323
Epoch 8/20
12591/12591 [=====] - 2s 184us/sample - loss
: 0.4275
Epoch 9/20
12591/12591 [=====] - 2s 187us/sample - loss
: 0.4238
Epoch 10/20
12591/12591 [=====] - 2s 186us/sample - loss
: 0.4238
Epoch 11/20
12591/12591 [=====] - 2s 191us/sample - loss
: 0.4210
Epoch 12/20
12591/12591 [=====] - 2s 185us/sample - loss
: 0.4180
Epoch 13/20
12591/12591 [=====] - 2s 187us/sample - loss
: 0.4174
Epoch 14/20
12591/12591 [=====] - 2s 186us/sample - loss
: 0.4144
Epoch 15/20
12591/12591 [=====] - 2s 192us/sample - loss
: 0.4123
Epoch 16/20
12591/12591 [=====] - 2s 188us/sample - loss
: 0.4107
Epoch 17/20
12591/12591 [=====] - 2s 196us/sample - loss
: 0.4092
Epoch 18/20
12591/12591 [=====] - 3s 207us/sample - loss
: 0.4066
Epoch 19/20
12591/12591 [=====] - 2s 197us/sample - loss
: 0.4094
Epoch 20/20
12591/12591 [=====] - 2s 198us/sample - loss
: 0.4093
```

Out[79]: <tensorflow.python.keras.callbacks.History at 0x1e358248160>

In [80]: `model_GRU.fit(x_train, y_train, epochs=20)`

Train on 12591 samples

```
Epoch 1/20
12591/12591 [=====] - 4s 302us/sample - loss
: 0.4890
Epoch 2/20
12591/12591 [=====] - 2s 189us/sample - loss
: 0.4384
Epoch 3/20
12591/12591 [=====] - 2s 188us/sample - loss
: 0.4346
Epoch 4/20
12591/12591 [=====] - 2s 185us/sample - loss
: 0.4321
Epoch 5/20
12591/12591 [=====] - 2s 190us/sample - loss
: 0.4306
Epoch 6/20
12591/12591 [=====] - ETA: 0s - loss: 0.432
- 2s 196us/sample - loss: 0.4326
Epoch 7/20
12591/12591 [=====] - 2s 197us/sample - loss
: 0.4283
Epoch 8/20
12591/12591 [=====] - 2s 197us/sample - loss
: 0.4258
Epoch 9/20
12591/12591 [=====] - 2s 196us/sample - loss
: 0.4237
Epoch 10/20
12591/12591 [=====] - 2s 197us/sample - loss
: 0.4234
Epoch 11/20
12591/12591 [=====] - 2s 195us/sample - loss
: 0.4186
Epoch 12/20
12591/12591 [=====] - 2s 196us/sample - loss
: 0.4200
Epoch 13/20
12591/12591 [=====] - 2s 198us/sample - loss
: 0.4193
Epoch 14/20
12591/12591 [=====] - 3s 206us/sample - loss
: 0.4166
Epoch 15/20
12591/12591 [=====] - 3s 231us/sample - loss
: 0.4156
Epoch 16/20
12591/12591 [=====] - 3s 213us/sample - loss
: 0.4130
Epoch 17/20
12591/12591 [=====] - 3s 215us/sample - loss
```

```

: 0.4120
Epoch 18/20
12591/12591 [=====] - 3s 208us/sample - loss
: 0.4092
Epoch 19/20
12591/12591 [=====] - 2s 198us/sample - loss
: 0.4076
Epoch 20/20
12591/12591 [=====] - 2s 198us/sample - loss
: 0.4063

```

Out[80]: <tensorflow.python.keras.callbacks.History at 0x1e35c9bd940>

Creating the test_data

```
In [82]: test_data, test_label = generate_data(df0_sensor1_final)
```

Checking for the accuracy

```
In [83]: from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
```

```
In [84]: 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_label)
    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 [85]: predict_values_LSTM = predict(model, test_data, test_label)
```

```
predict_values_GRU=predict(model_GRU,test_data,test_label)
```

```
print(predict_values_LSTM)
```

[illegible]


```
In [88]: print(predict_values_GRU)
```

[illegible]

```
0., 0., 0., 0., 0., 0., 0., 0.]}, 'conf_mat': (259, 0, 30, 100
), 'acc': 0.922879177377892}
```

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($s(t) > \text{threshold}$) values then the next step is to make X_values as $s(t-1), s(t-2), \dots, s(t-10)$
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: (246, 2, 43, 98)

Accuracy: 88.43

GRU:

Confusion Matrix: (259, 0, 30, 100)

Accuracy: 92.28