DetectRotationsV2

January 13, 2022

[22]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
from datetime import timedelta
from scipy import signal
```

[23]:
```
Player = 9
Game = 2
df_Player = pd.read_csv('matrix_Player_' + str(Player) + '_game_' + str(Game) +␣
↪'.csv')
df_Player.columns =␣
↪['frAcc','frRoAcc','frDispl','frRoAng','frSpeed','timeLine','frameRotationalSpeedX','frameR
df_Action = pd.read_csv('Ned_DUI_Game_' + str(Game) + '.csv')
df_Player
```

[23]:

	frAcc	frRoAcc	frDispl	frRoAng	frSpeed	timeLine	\
0	0.000000	0.0	0.0	0.00000	0.000000	0.01	
1	0.000000	0.0	0.0	0.00000	0.000000	0.02	
2	0.000000	0.0	0.0	0.00000	0.000000	0.03	
3	0.000000	0.0	0.0	0.00000	0.000000	0.04	
4	0.000000	0.0	0.0	0.00000	0.000000	0.05	
…	…	…	…	…	…	…	
754220	-0.030424	4981.9	6585.3	-0.11980	-0.001860	7542.20	
754221	0.062668	4981.9	6585.3	-0.11723	-0.002164	7542.20	
754222	0.019899	4981.9	6585.2	-0.13407	-0.001537	7542.20	
754223	NaN	NaN	NaN	NaN	NaN	NaN	
754224	NaN	NaN	NaN	NaN	NaN	NaN	

	frameRotationalSpeedX	frameRotationalSpeedY	frameRotationalSpeedZ	\
0	NaN	NaN	NaN	
1	NaN	NaN	NaN	
2	NaN	NaN	NaN	
3	NaN	NaN	NaN	
4	NaN	NaN	NaN	
…	…	…	…	
754220	0.97222	2.4578	-0.98778	

```
754221                0.91000                2.5480               -1.05000
754222                0.92400                2.6320               -1.05000
754223                0.96444                2.5200               -1.01890
754224                0.88375                2.5200               -0.93625

        wheelRotationalSpeedX  wheelRotationalSpeedY  wheelRotationalSpeedZ  \
0                         NaN                    NaN                    NaN
1                         NaN                    NaN                    NaN
2                         NaN                    NaN                    NaN
3                         NaN                    NaN                    NaN
4                         NaN                    NaN                    NaN
…                           …                      …                      …
754220              -0.000000                  4.690                  2.730
754221              -0.056000                  4.494                  2.597
754222               0.070000                  4.620                  2.660
754223               0.070000                  4.620                  2.660
754224               0.081667                  4.585                  2.625

        frRoSpeed
0         0.00000
1         0.00000
2         0.00000
3         0.00000
4         0.00000
…           …
754220    0.25656
754221   -1.68370
754222        NaN
754223        NaN
754224        NaN

[754225 rows x 13 columns]
```

[24]:
```
# ----------------------------------------------------------------------

#LowPass Filter


#Lowpass filter design for rotation and wheelspeed to improve accuracy of code␣
↪(Butterworth filter)
df_Filter = df_Player
df_Filter = df_Filter.fillna(0)

Order = 5
cutoff_freq = 1.5
sampling_freq = 100
sampling_duration = len(df_Filter.wheelRotationalSpeedX)/100
```

```
normalized_cutoff_freq = 2 * cutoff_freq / sampling_freq
numerator_coeffs, denominator_coeffs = signal.butter(Order,␣
↪normalized_cutoff_freq)

filtered_WheelRotationspeed = signal.lfilter(numerator_coeffs,␣
↪denominator_coeffs, df_Filter.wheelRotationalSpeedX)
filtered_FrameRotationspeed = signal.lfilter(numerator_coeffs,␣
↪denominator_coeffs, df_Filter.frameRotationalSpeedZ)

# ------------------------------------------------------------------------
        #Operations çy


#Play with different operations to see clearer patterns
Sub = filtered_WheelRotationspeed + filtered_FrameRotationspeed
#Old_Conv1 = filtered_FrameRotationspeed / filtered_WheelRotationspeed
Conv1 = (abs(filtered_FrameRotationspeed)+abs(filtered_WheelRotationspeed))/
↪filtered_WheelRotationspeed



#Diff = arr[i+1] - arr[i]
DiffFrame = np.diff(df_Filter.frameRotationalSpeedZ,n=1)
DiffFrame = np.insert(DiffFrame,0,0)

DiffWheel = np.diff(df_Filter.wheelRotationalSpeedX,n=1)
DiffWheel = np.insert(DiffWheel,0,0)

Multi = filtered_WheelRotationspeed + filtered_FrameRotationspeed
#This isn't used


# ------------------------------------------------------------------------


#Set all data into a datafram
Data = pd.DataFrame({'Time':df_Player.timeLine,'WheelRotationspeed':
↪filtered_WheelRotationspeed,
                     'FrameRotationspeed':filtered_FrameRotationspeed,
                     'Sub':Sub ,'Conv':Conv1, 'DiffFrame':DiffFrame,␣
↪'DiffWheel':DiffWheel},
                     columns=['Time',␣
↪'WheelRotationspeed','FrameRotationspeed','Sub','Conv','DiffFrame','DiffWheel'])

#Convert data into chunks of n/100 of a second
```

```
n = 50  #chunk row size
Data_chunks = [Data[i:i+n] for i in range(0,Data.shape[0],n)]



#Search for the sprints
Startsprint = []
Stopsprint = []
Sprinting = False
Stop = True

#Use Sub en Conv to detect sprints
for chunks in Data_chunks:
    if abs(chunks['Conv'].max()) < 3 and chunks['Sub'].mean() > 300 and␣
 ↪chunks['WheelRotationspeed'].min() > 200  and Sprinting == False:
        Startsprint.append(chunks['Time'].min())
        Sprinting = True
        Stop = False
    elif abs(chunks['Conv'].max() < 3) and chunks['Sub'].mean() > 300 and␣
 ↪chunks['WheelRotationspeed'].min() > 200:
        Sprinting = True
        Stop = False
    elif Stop == False:
        Stopsprint.append(chunks['Time'].min())
        Sprinting = False
        Stop = True

#Use wheelrotation, framerotation and conv to detect rotations
Startrotate = []
Stoprotate = []
Rotate = False
Stop = True

count = 0

for chunks in Data_chunks:
    if abs(chunks['FrameRotationspeed'].max()) > 75 and abs(chunks['Conv'].
 ↪mean()) > 1.15 and count!=29 and Rotate == False:
        #this condition count!=29 is to make sure that it doesn't start a␣
 ↪rotation in the last
        #point of the graph because that way it would never enc

        Startrotate.append(chunks['Time'].min())
        Rotate = True
        Stop = False
```

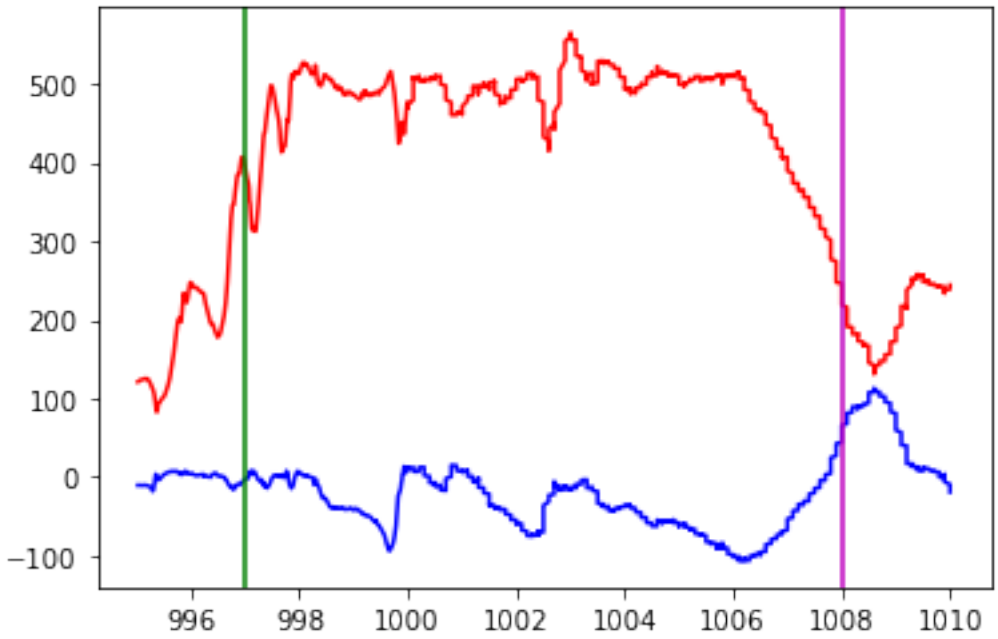
```
    elif abs(chunks['FrameRotationspeed'].max()) > 75 and abs(chunks['Conv'].
↪mean()) > 1.15 and count == 29 and Rotate == True:
        #this comes due to the need of matching sizes (between startsprint and␣
↪stopsprint), as in one fast defense
        #theres a rotation that never ends in the plot, so I'm forcing it (had␣
↪to add Rotate == True so that this condition only
        #happens when a rotation has started before the last point of the graph)


        Stoprotate.append(chunks['Time'].min())
        Rotate = False
        Stop = True
    elif abs(chunks['FrameRotationspeed'].max()) > 75 and abs(chunks['Conv'].
↪mean()) > 1.15:#si abs menor que num sigue siendo rotation
        #Rotate =True is implicit
        Rotate = True
        Stop = False
    elif Stop == False :#Rotate==True and the previous conditions aren´t met,␣
↪thats implicit
        Stoprotate.append(chunks['Time'].min())
        Rotate = False
        Stop = True
    count +=1

# ------------------------------------------------------------------------

    #Pop small sprints out



#Filter Sprints by lenght, if length is below 2 delete sprint
Deleted = 0

if len(Startsprint) > len(Stopsprint):
    Startsprint.pop(-1)

if len(Startsprint) == len(Stopsprint):
    for i in range(0,len(Startsprint)-1):
        if (Stopsprint[i-Deleted] - Startsprint[i-Deleted]) < 5:
            Startsprint.pop(i-Deleted)
            Stopsprint.pop(i-Deleted)
            Deleted = Deleted + 1

# ------------------------------------------------------------------------

print(Startsprint, Stopsprint)
print(len(Startsprint))
```

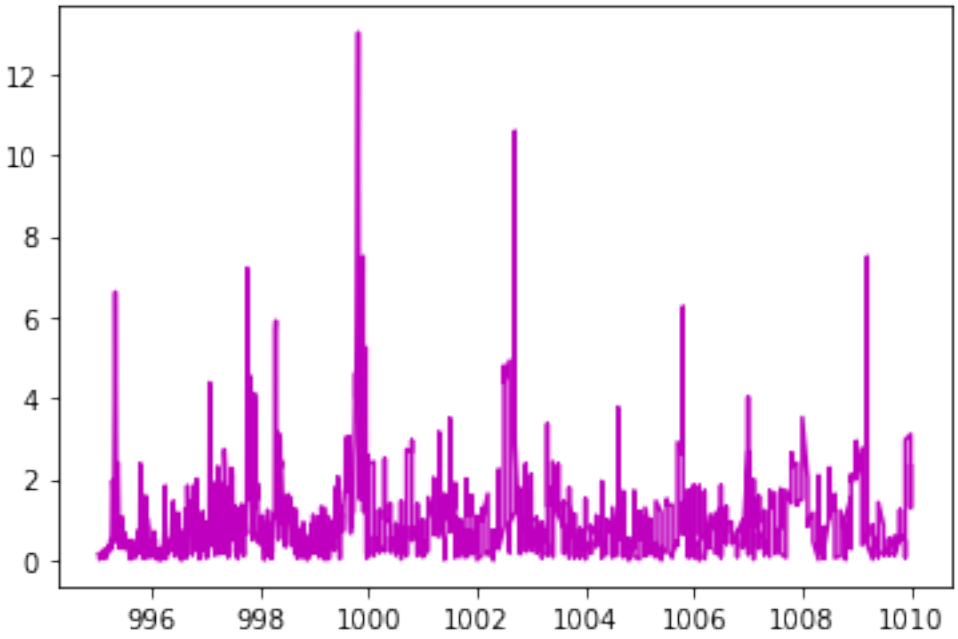
```
#Plot graph
for i in range(0,len(Startsprint),1):
    plt.plot(df_Player.timeLine.iloc[int(Startsprint[i]*100-200):
 ↪int(Stopsprint[i]*100+200)],
             df_Player.wheelRotationalSpeedX.iloc[int(Startsprint[i]*100-200):
 ↪int(Stopsprint[i]*100+200)],'r',
             df_Player.timeLine.iloc[int(Startsprint[i]*100-200):
 ↪int(Stopsprint[i]*100+200)],
             df_Player.frameRotationalSpeedZ.iloc[int(Startsprint[i]*100-200):
 ↪int(Stopsprint[i]*100+200)],'b'
            )
    plt.axvline(x=Startsprint[i],color = 'g')
    plt.axvline(x=Stopsprint[i], color = 'm')
    plt.show()
    plt.plot(df_Player.timeLine.iloc[int(Startsprint[i]*100-200):
 ↪int(Stopsprint[i]*100+200)],
             abs(Data.DiffFrame.iloc[int(Startsprint[i]*100-200):
 ↪int(Stopsprint[i]*100+200)]),'m'
            )
    plt.show()
```

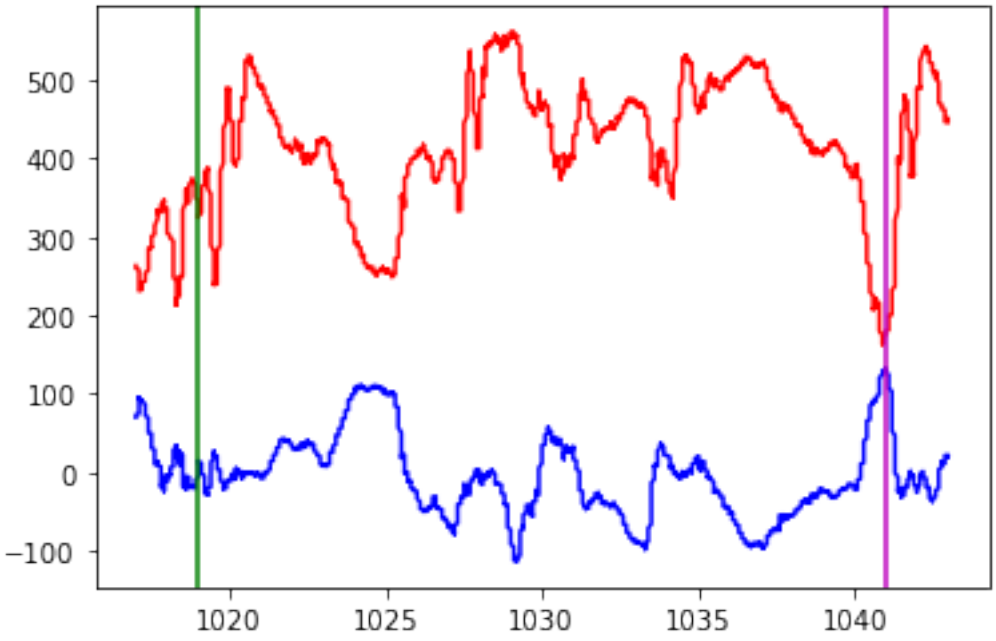
```
/tmp/ipykernel_48350/1580647309.py:28: RuntimeWarning: invalid value encountered
in true_divide
  Conv1 = (abs(filtered_FrameRotationspeed)+abs(filtered_WheelRotationspeed))/fi
ltered_WheelRotationspeed
```

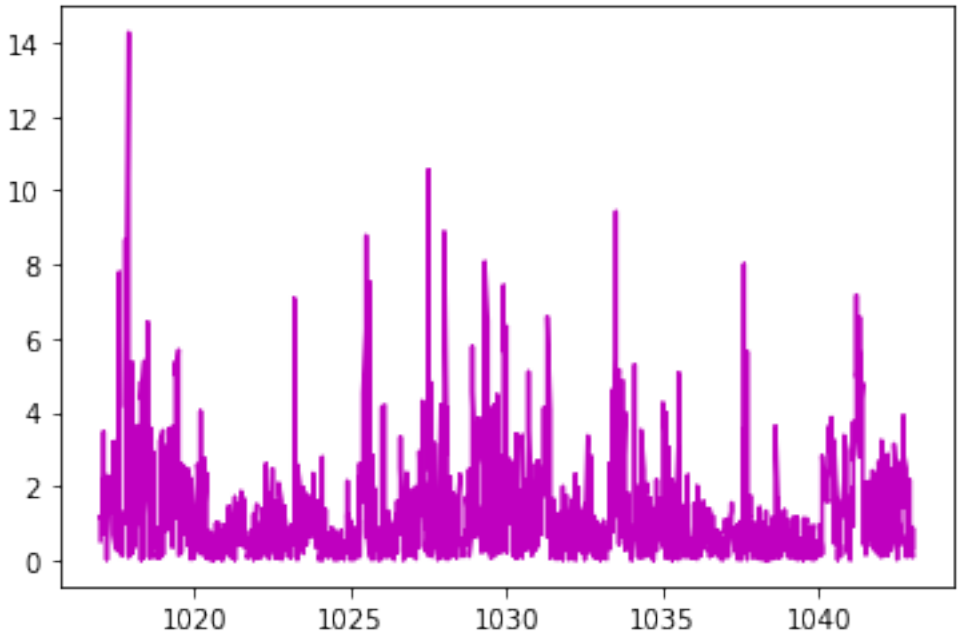
```
[997.01, 1019.0, 1041.5, 1062.5, 1078.0, 1086.0, 1093.5, 1105.5, 1138.0, 1161.0,
1181.5, 1201.0, 1232.5, 1248.5, 1257.0, 1427.5, 1485.5, 1511.5, 1538.0, 1566.0,
1593.0, 1626.5, 1659.5, 1676.0, 1734.0, 1832.0, 1912.0, 2216.5, 2224.5, 2235.5,
2433.5, 2465.5, 2490.0, 2558.0, 2598.0, 2631.0, 2647.0, 2671.5, 2684.0, 2786.0,
2800.0, 2850.0, 2873.0, 2888.5, 2940.0, 2956.5, 3004.5, 3025.5, 3049.0, 3074.0,
3093.5, 3118.5, 3137.5, 3156.0, 3256.0, 3276.0, 3329.5, 3372.0, 3506.5, 3530.0,
3555.5, 3602.5, 3628.0, 3654.0, 3685.0, 3703.5, 5040.0, 5053.5, 5080.5, 5095.0,
5135.5, 5146.5, 5177.0, 5207.0, 5324.0, 5345.5, 5369.0, 5400.5, 5411.0, 5434.0,
5462.0, 5510.5, 5594.5, 5621.0, 5704.0, 5745.5, 5842.5, 5895.0, 5927.5, 5955.5,
5974.5, 5993.0, 6044.5, 6059.5, 6076.5, 6897.0, 6912.5, 6923.0, 6950.5, 6963.0,
6981.0, 6998.0, 7051.0, 7186.5, 7206.0, 7225.5, 7249.0, 7281.0] [1008.0, 1041.0,
1047.0, 1074.0, 1083.5, 1093.0, 1103.0, 1121.0, 1148.5, 1181.0, 1186.5, 1219.5,
1246.0, 1254.0, 1262.0, 1433.5, 1490.5, 1516.5, 1543.5, 1571.0, 1598.0, 1632.0,
1665.5, 1681.5, 1739.0, 1837.0, 1917.5, 2221.5, 2231.5, 2241.5, 2441.0, 2471.0,
2495.0, 2566.5, 2607.5, 2640.0, 2654.5, 2680.0, 2689.5, 2795.0, 2807.0, 2856.0,
2879.0, 2894.0, 2946.5, 2963.5, 3011.5, 3031.0, 3057.0, 3081.0, 3100.0, 3126.0,
3144.0, 3162.5, 3264.5, 3283.0, 3335.0, 3378.0, 3513.5, 3536.0, 3563.5, 3611.5,
3634.5, 3661.0, 3691.5, 3708.5, 5046.0, 5079.0, 5092.5, 5132.5, 5146.0, 5172.0,
5201.0, 5227.5, 5333.0, 5353.0, 5377.0, 5405.5, 5419.5, 5440.0, 5470.5, 5519.0,
5604.5, 5626.5, 5710.5, 5755.5, 5848.5, 5903.5, 5934.0, 5962.0, 5981.0, 6001.0,
```

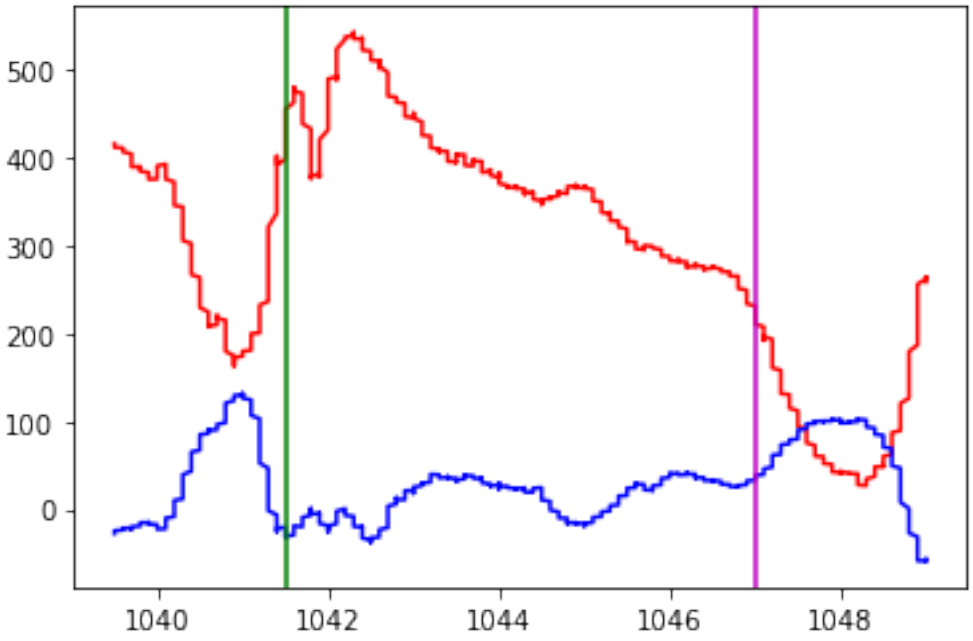
```
6051.0, 6067.0, 6089.0, 6903.0, 6918.5, 6928.0, 6955.5, 6977.0, 6989.5, 7008.0,
7057.5, 7193.0, 7211.5, 7234.5, 7254.5, 7285.0]
108
```

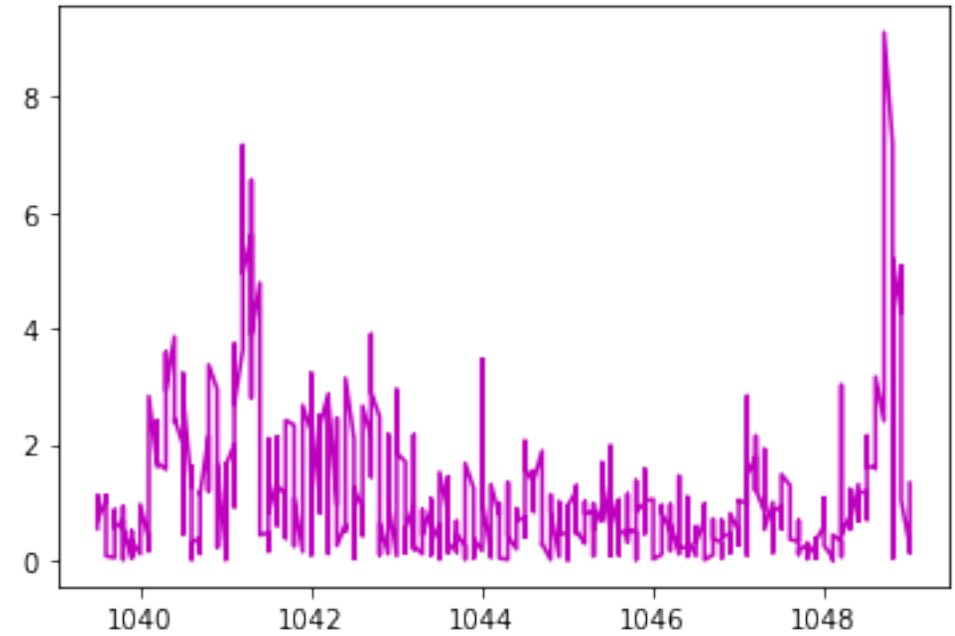


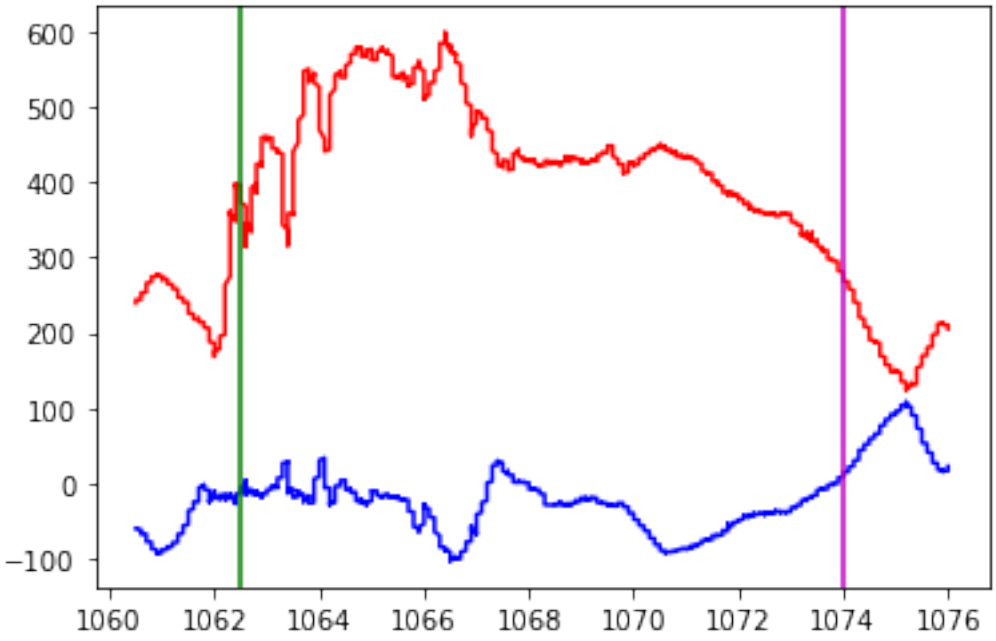


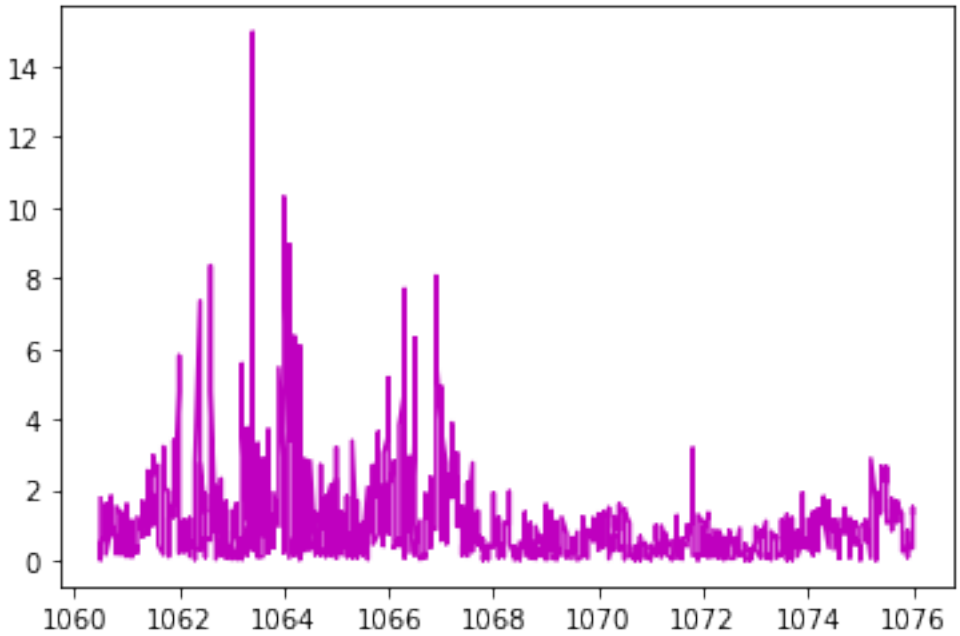

500
400
300
200
100
0
−100
1020
1025
1030
1035
1040

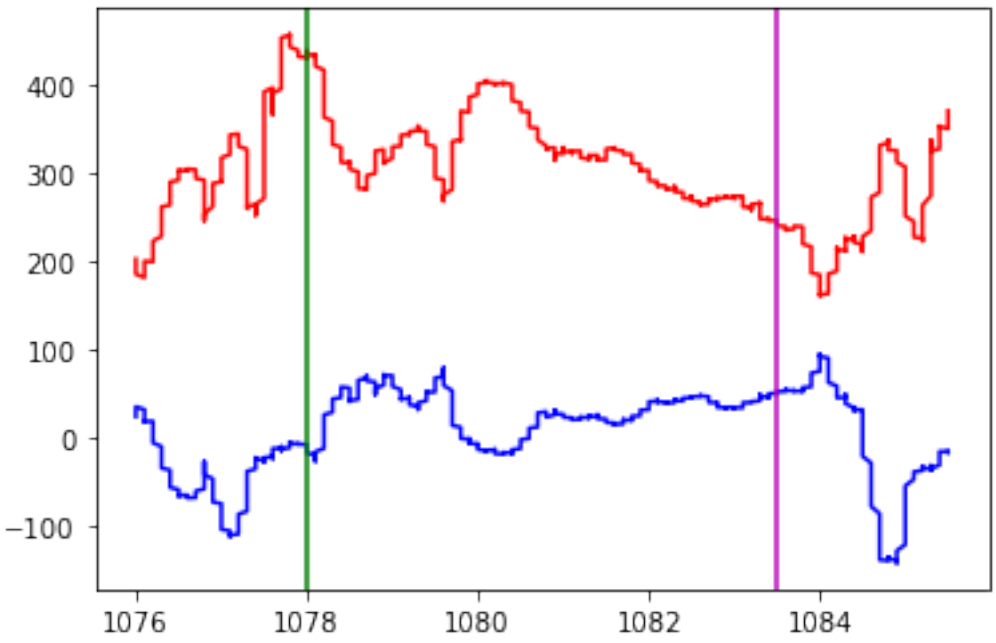

14
12
10
8
6
4
2
0
1020
1025
1030
1035
1040

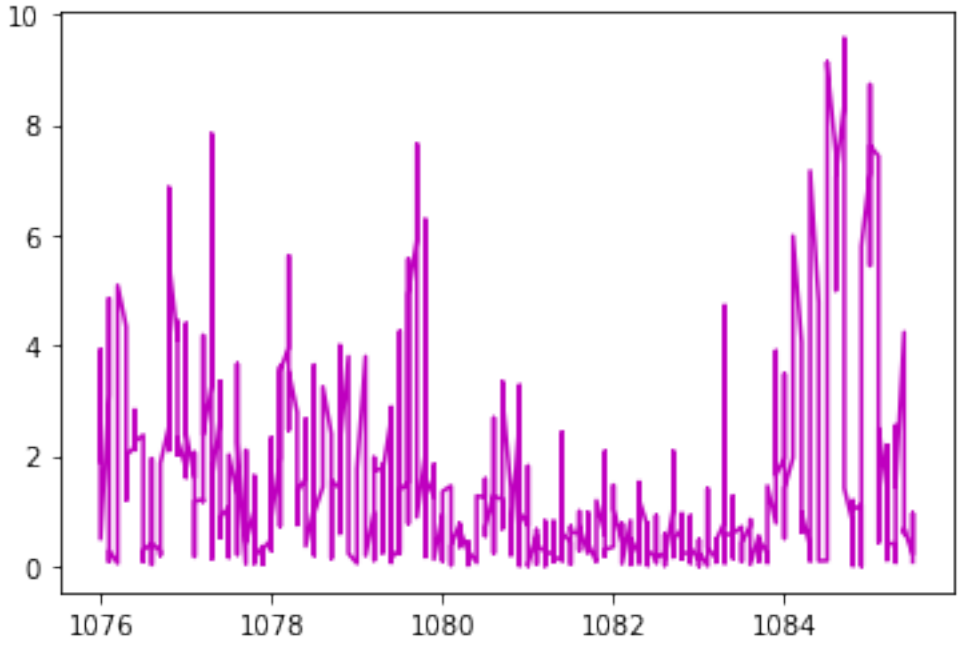

500
400
300
200
100
0
1040
1042
1044
1046
1048

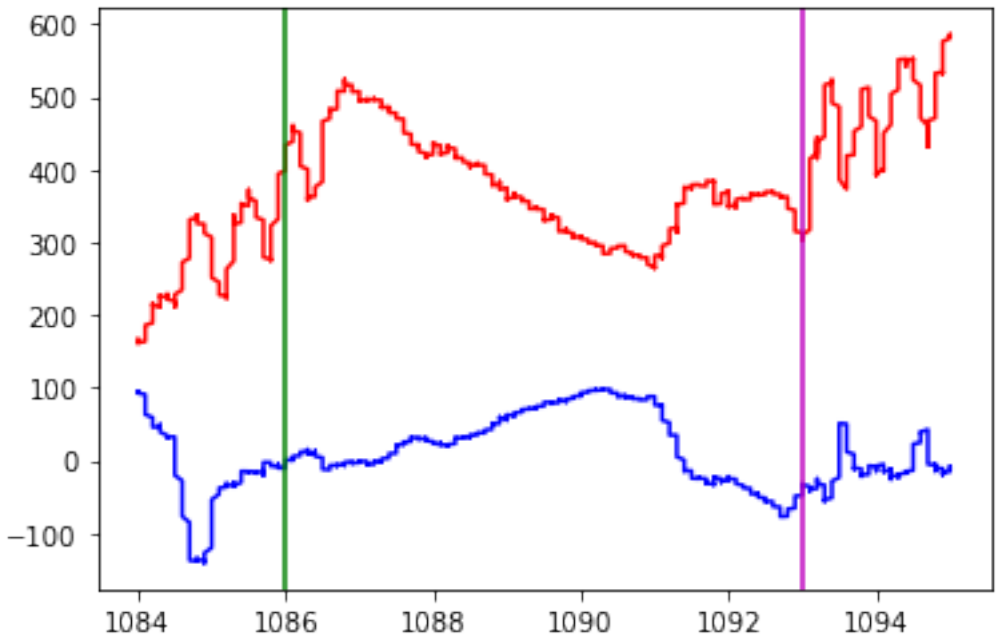

8
6
4
2
0
1040
1042
1044
1046
1048

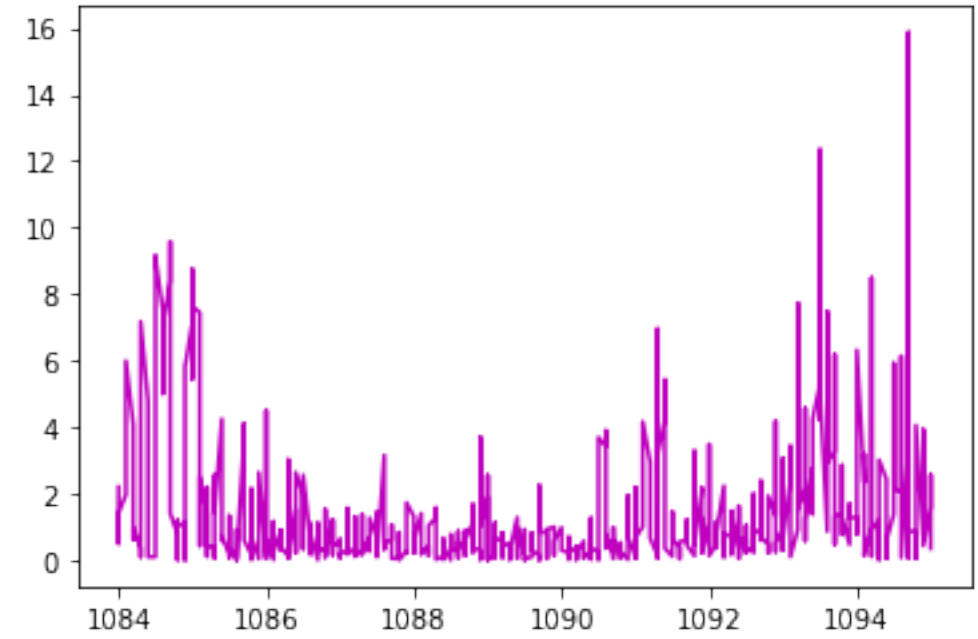

600
500
400
300
200
100
0
−100
1060
1062
1064
1066
1068
1070
1072
1074
1076

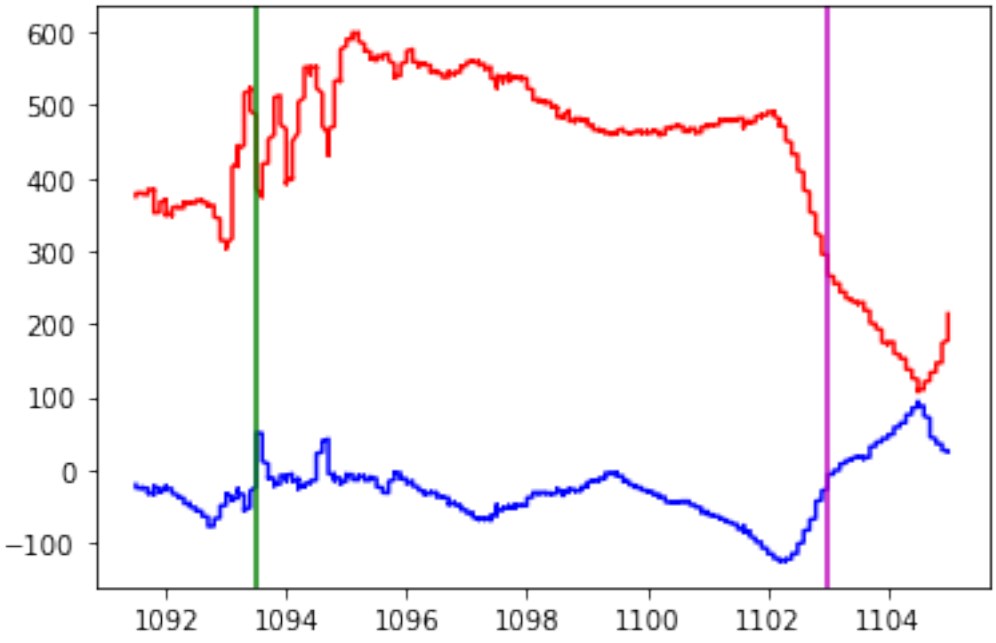

14
12
10
8
6
4
2
0
1060
1062
1064
1066
1068
1070
1072
1074
1076

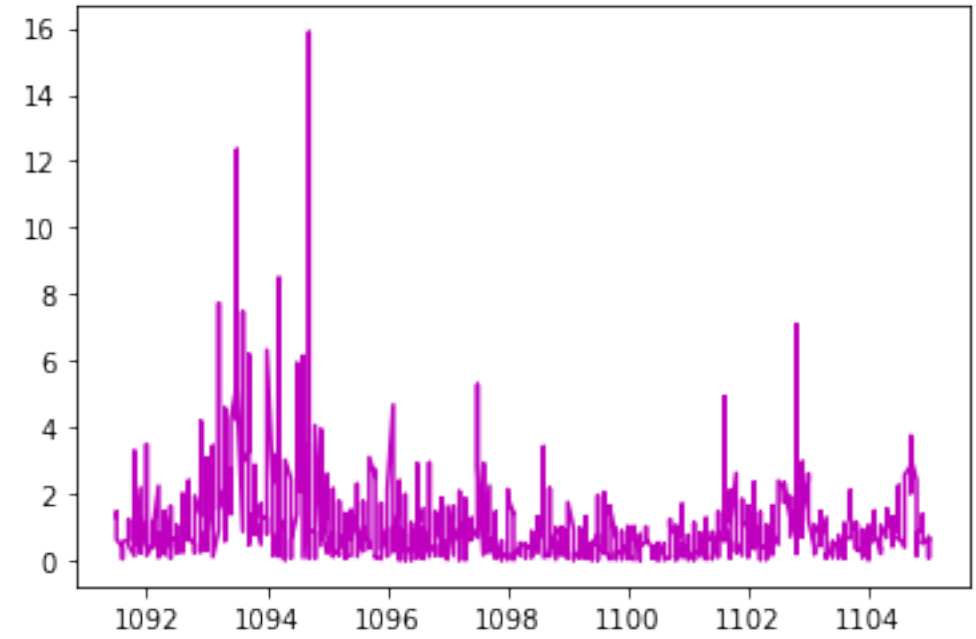

400
300
200
100
0
−100
1076
1078
1080
1082
1084

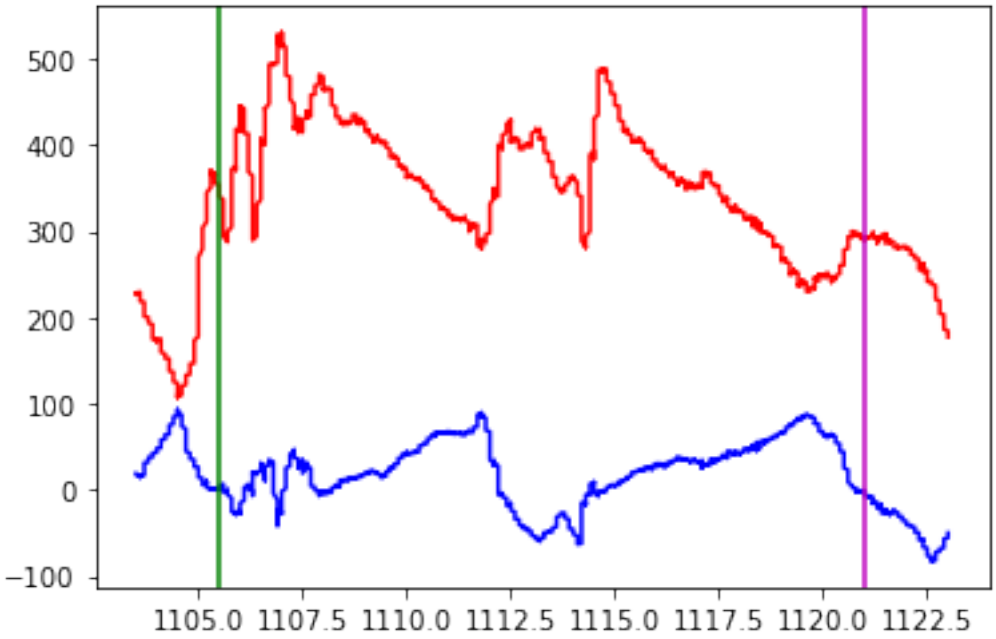

10
8
6
4
2
0
1076
1078
1080
1082
1084

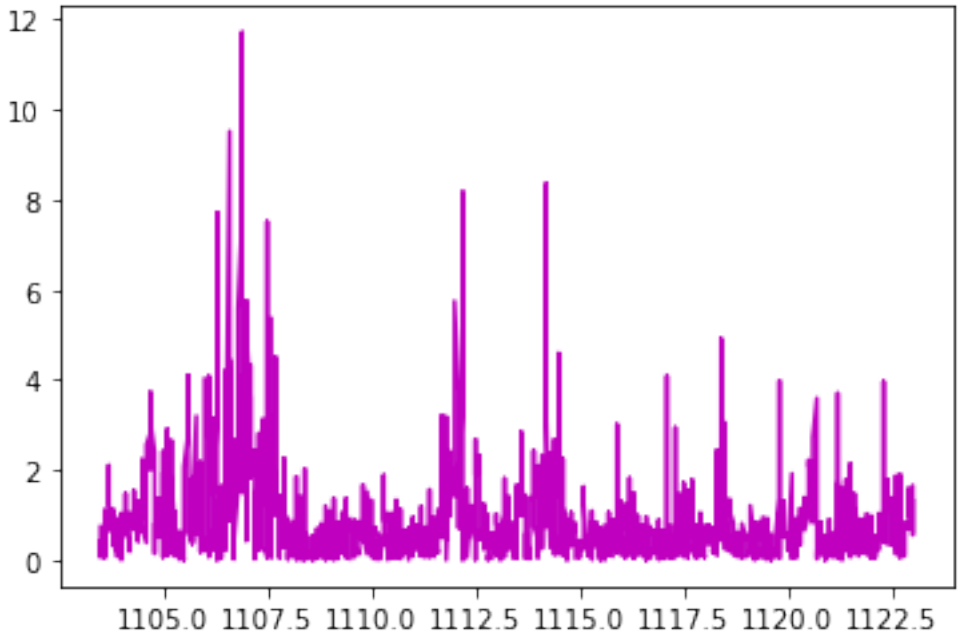

600
500
400
300
200
100
0
−100
1084
1086
1088
1090
1092
1094

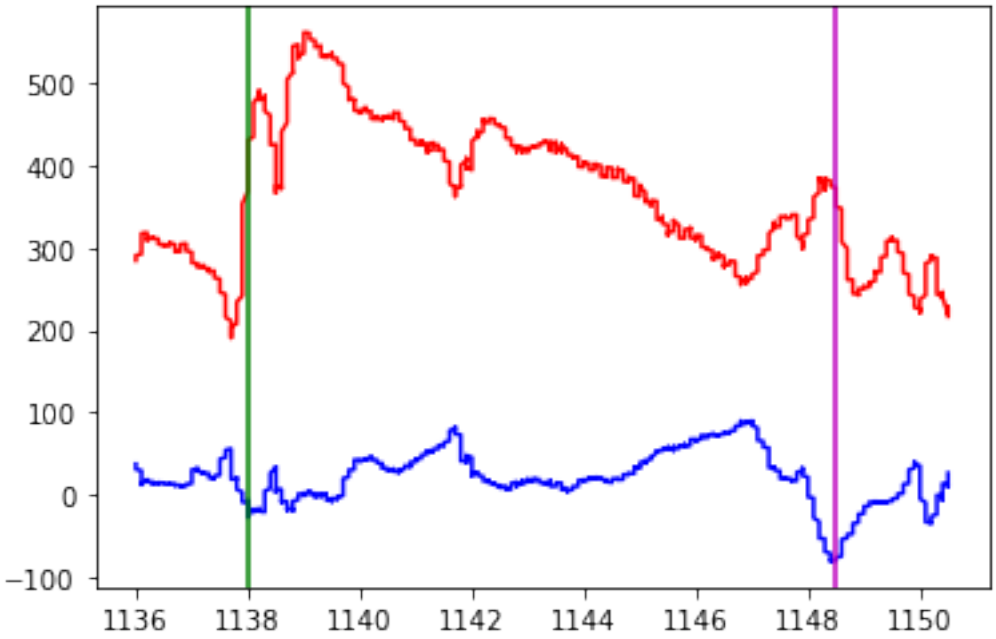

16
14
12
10
8
6
4
2
0
1084
1086
1088
1090
1092
1094

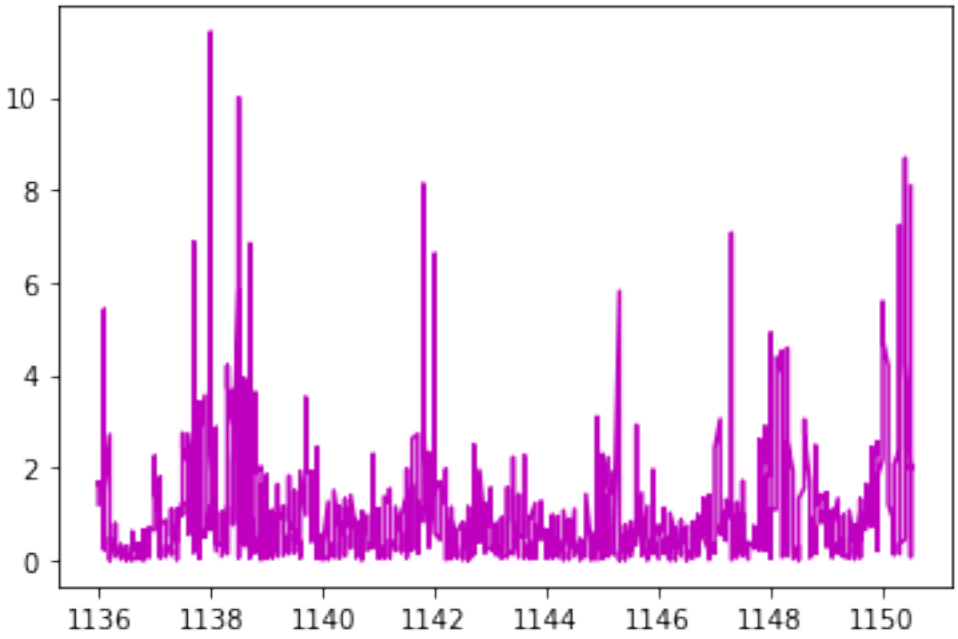

600
500
400
300
200
100
0
−100
1092
1094
1096
1098
1100
1102
1104

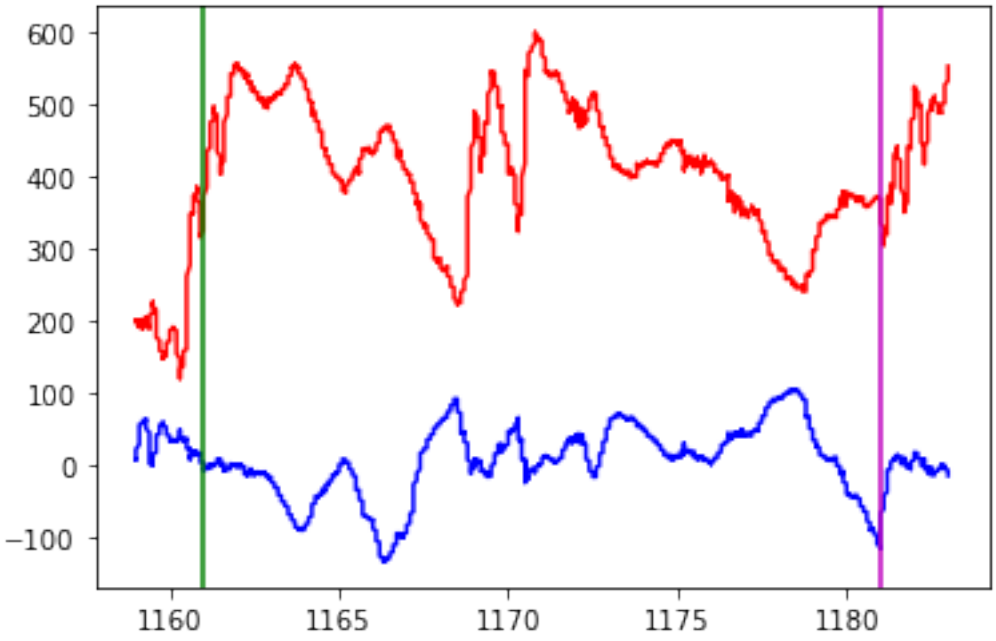

16
14
12
10
8
6
4
2
0
1092
1094
1096
1098
1100
1102
1104

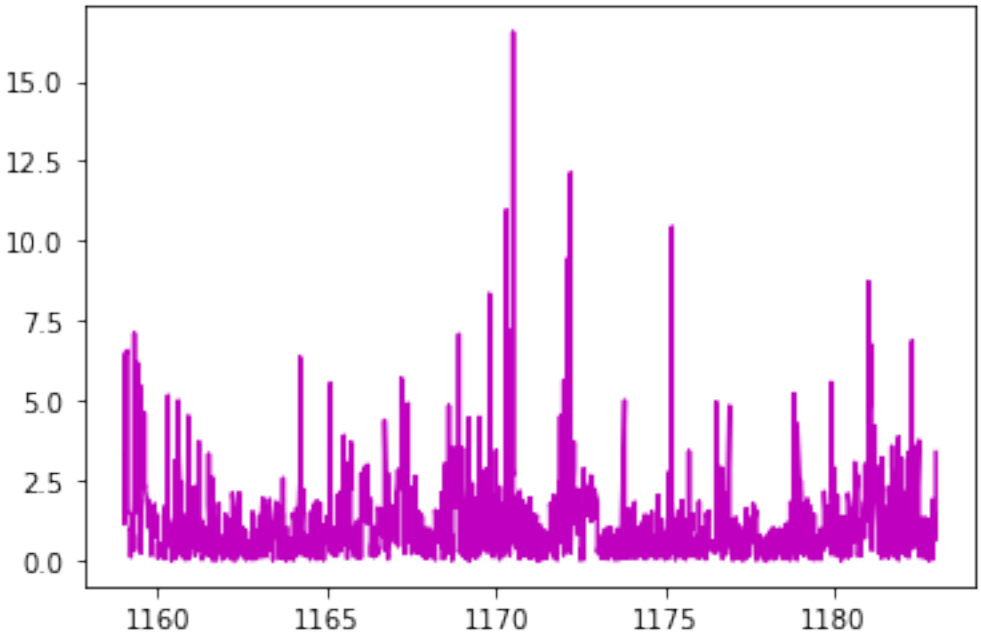

500
400
300
200
100
0
−100
1105.0
1107.5
1110.0
1112.5
1115.0
1117.5
1120.0
1122.5

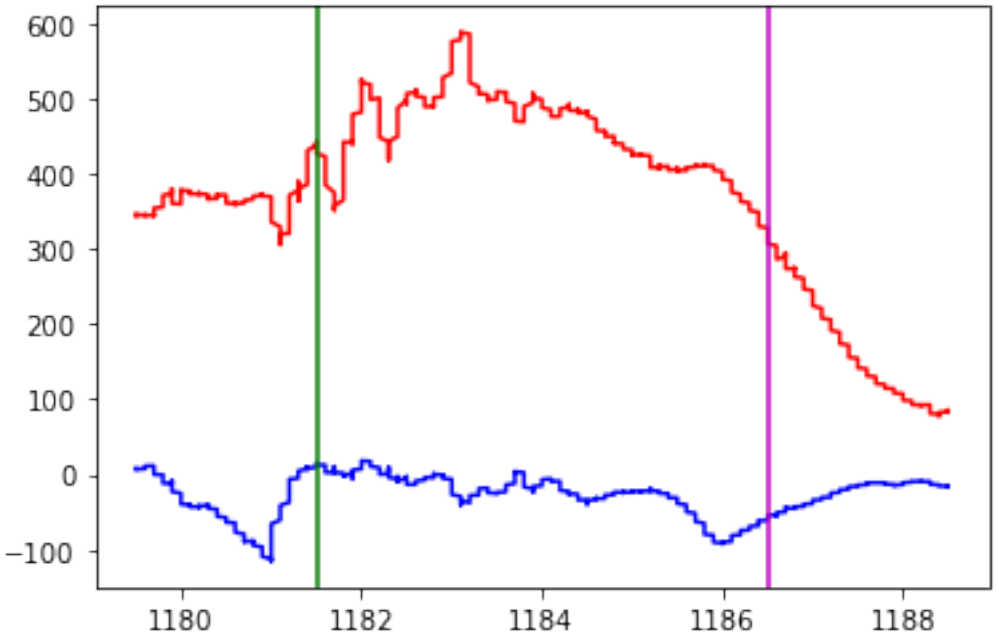

12
10
8
6
4
2
0
1105.0
1107.5
1110.0
1112.5
1115.0
1117.5
1120.0
1122.5

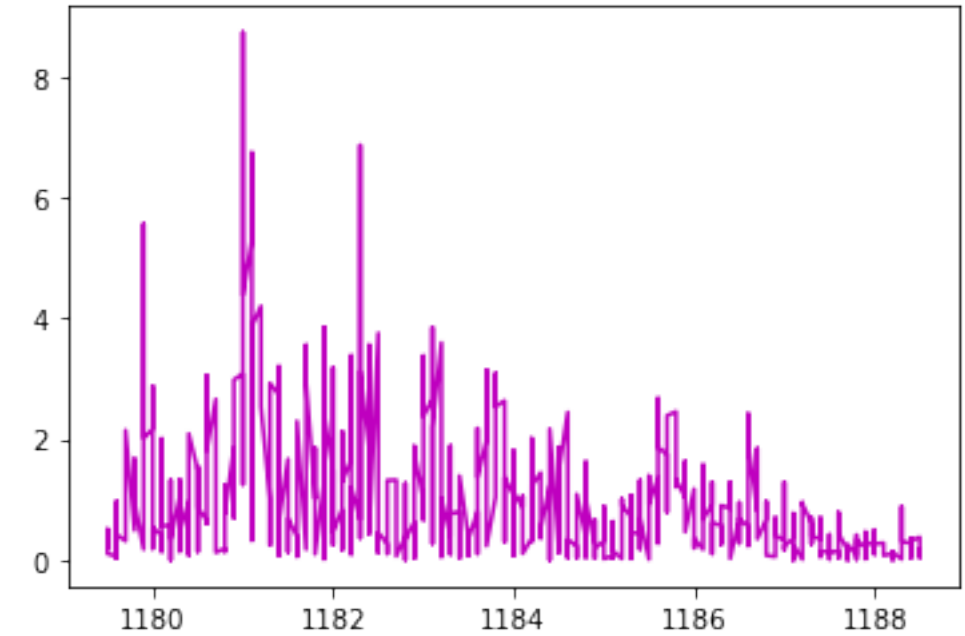

500
400
300
200
100
0
−100
1136
1138
1140
1142
1144
1146
1148
1150

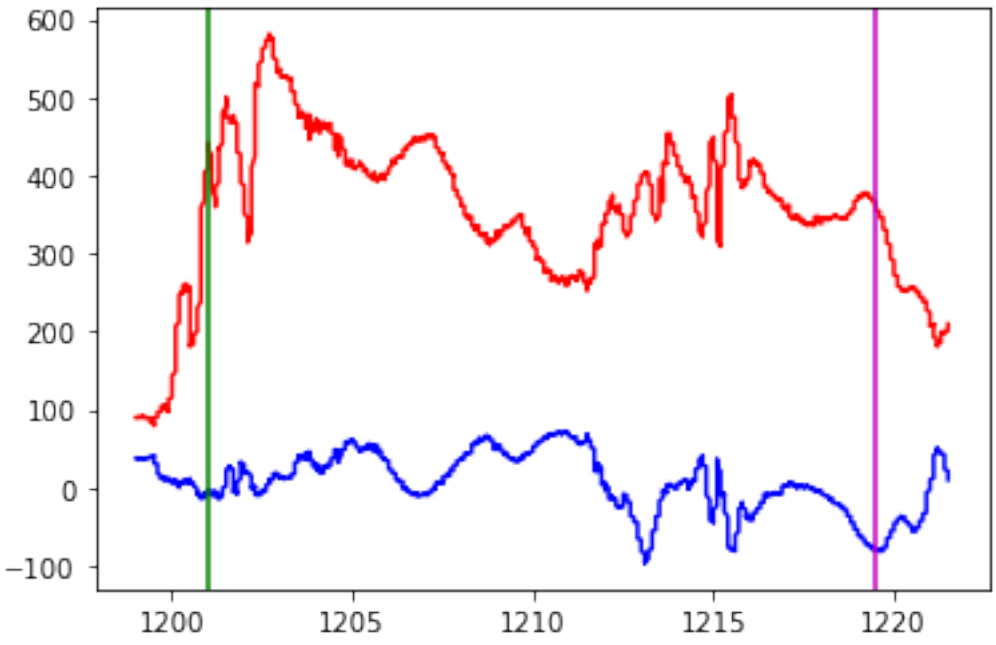

10
8
6
4
2
0
1136
1138
1140
1142
1144
1146
1148
1150

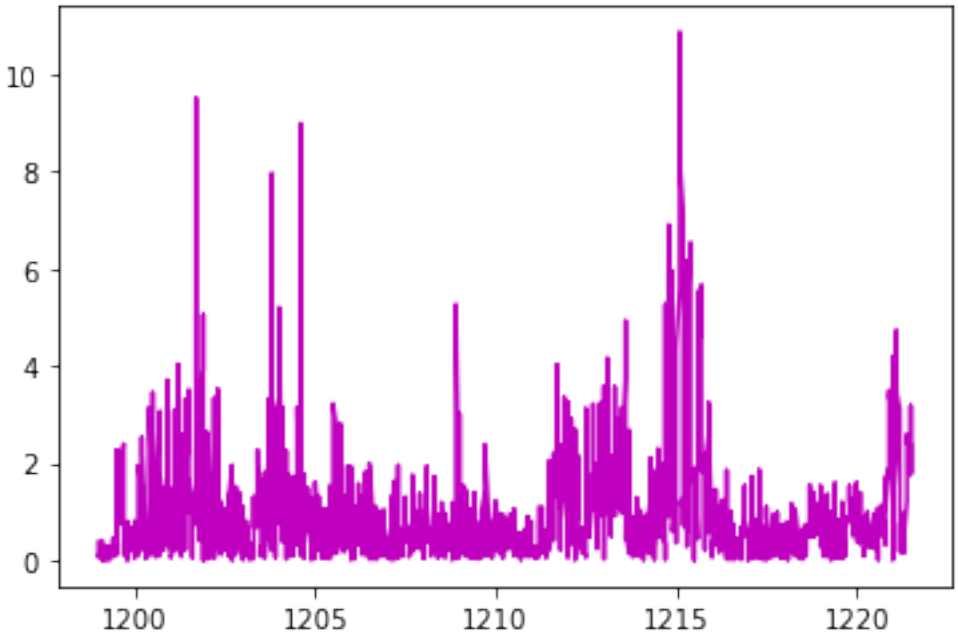

600
500
400
300
200
100
0
−100
1160
1165
1170
1175
1180

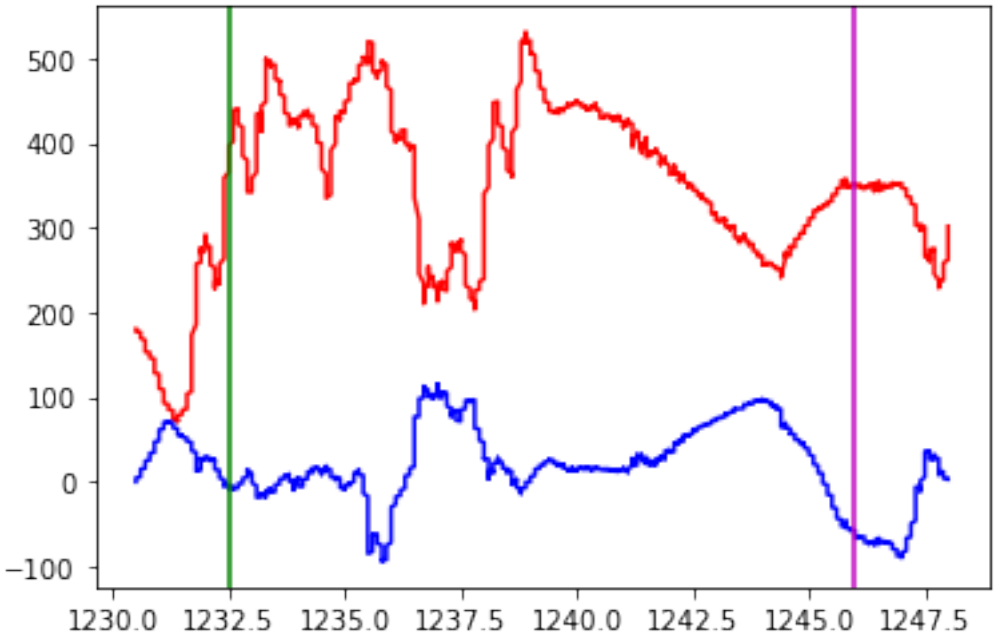

15.0
12.5
10.0
7.5
5.0
2.5
0.0
1160
1165
1170
1175
1180

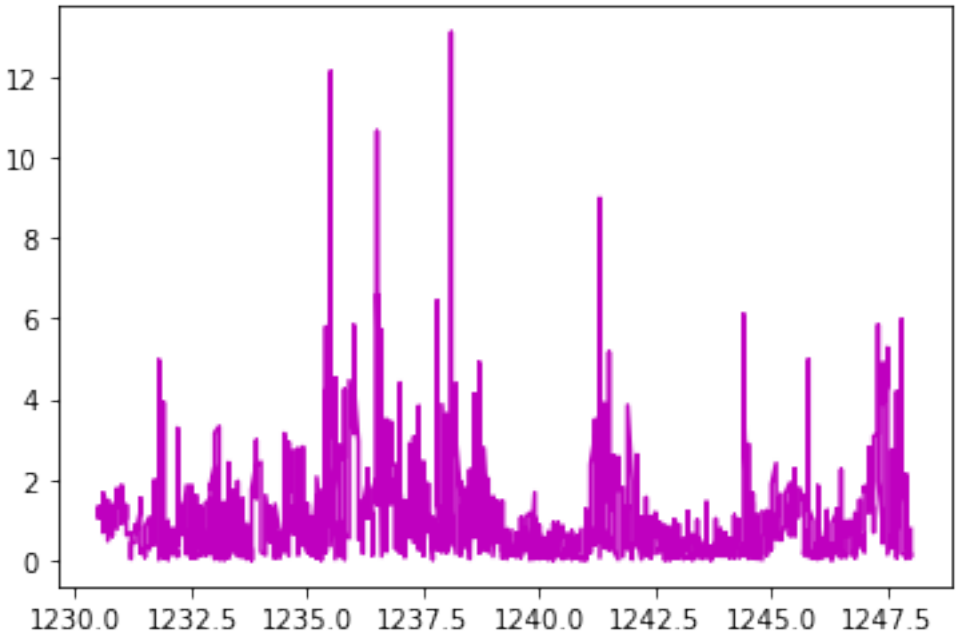

600
500
400
300
200
100
0
−100
1180
1182
1184
1186
1188

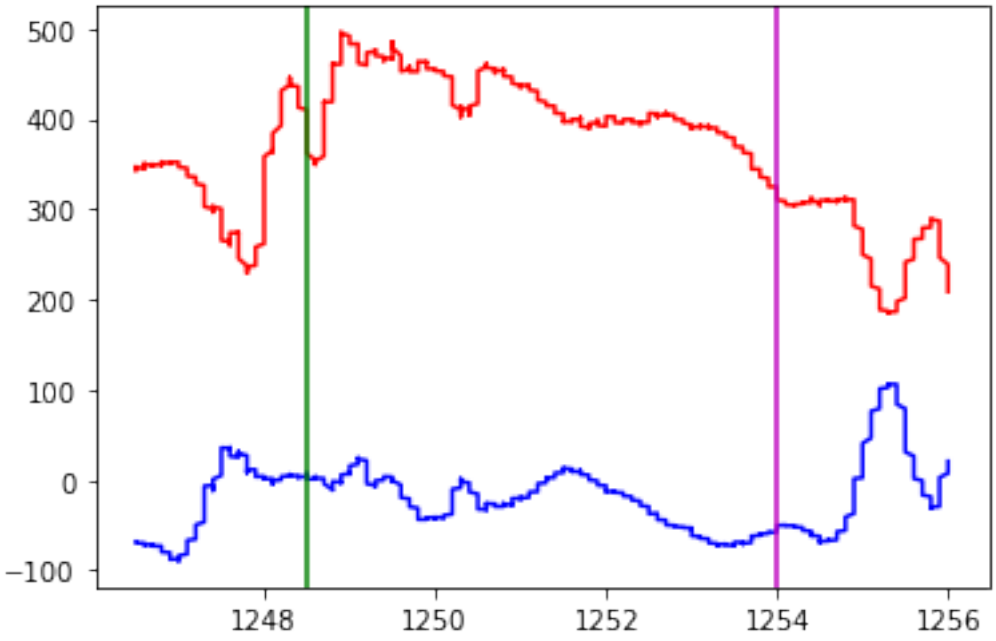

8
6
4
2
0
1180
1182
1184
1186
1188

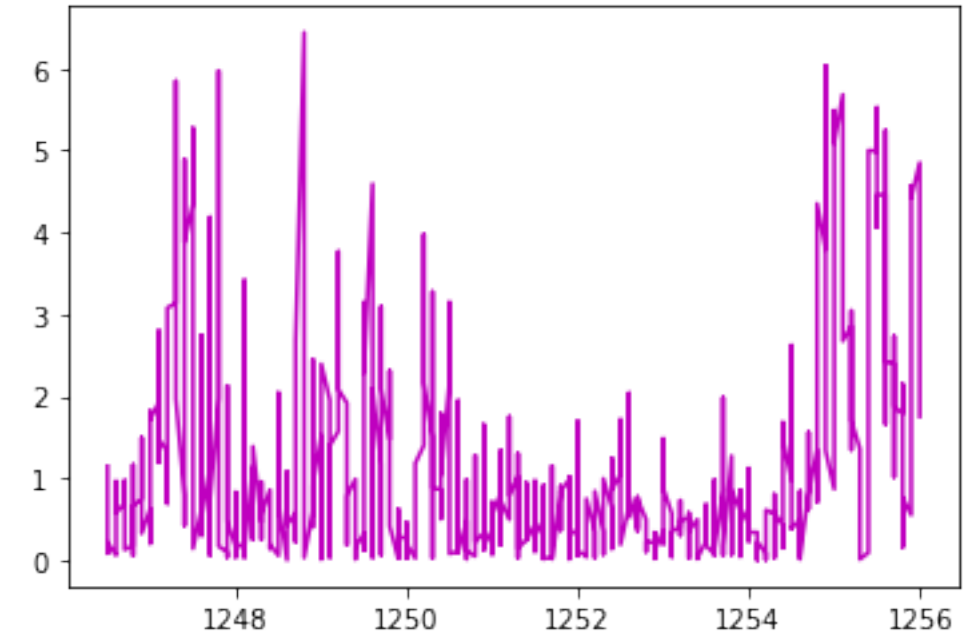

600
500
400
300
200
100
0
−100
1200
1205
1210
1215
1220

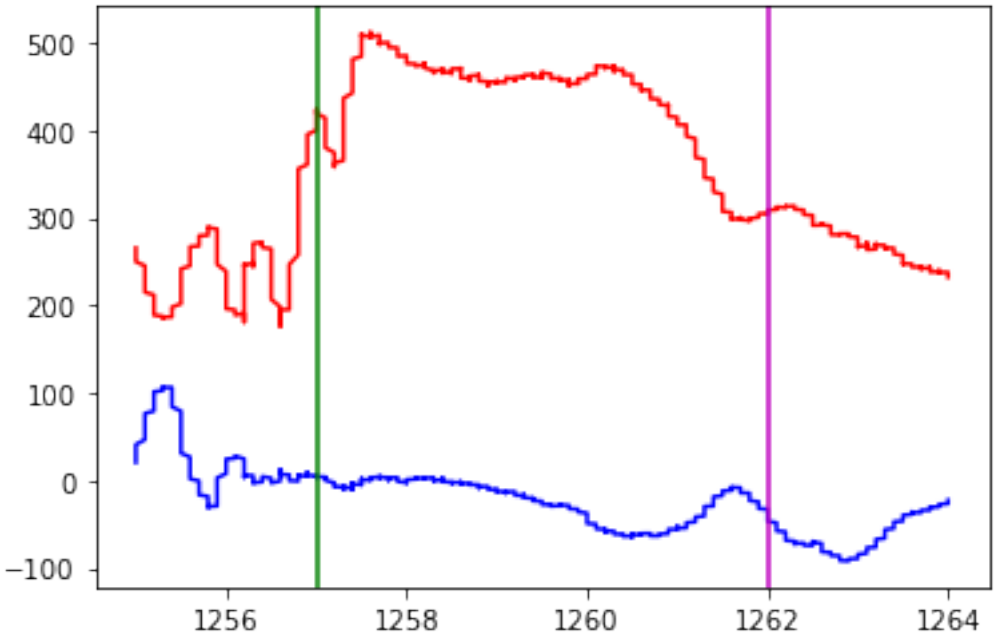

10
8
6
4
2
0
1200
1205
1210
1215
1220

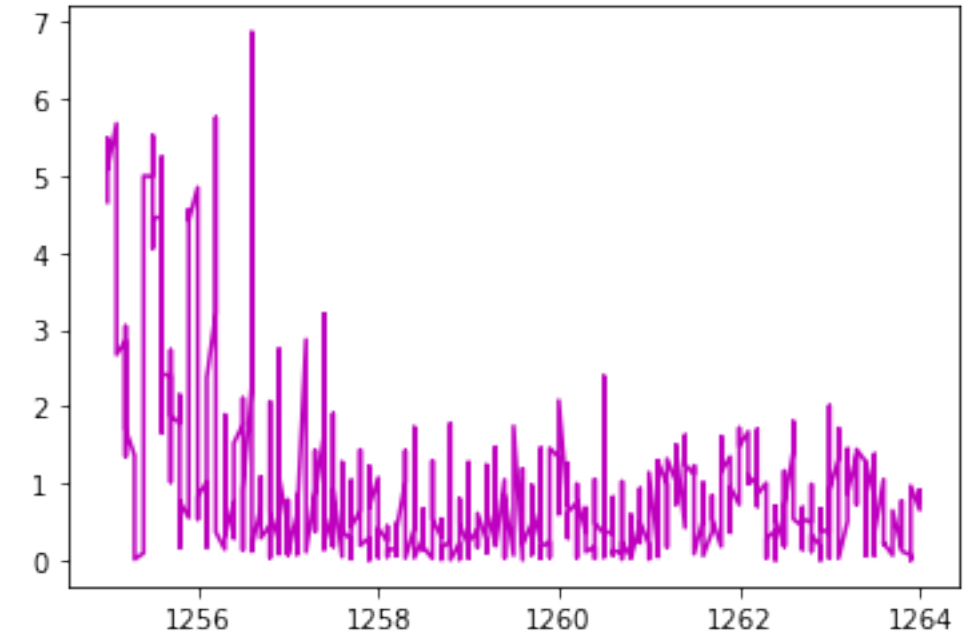

500
400
300
200
100
0
−100
1230.0
1232.5
1235.0
1237.5
1240.0
1242.5
1245.0
1247.5

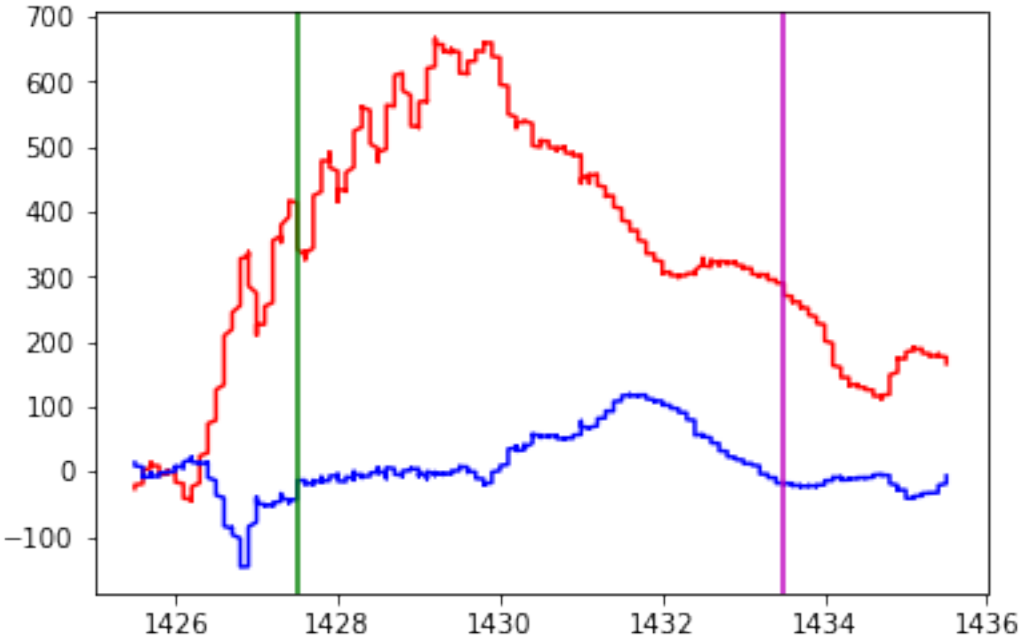

12
10
8
6
4
2
0
1230.0
1232.5
1235.0
1237.5
1240.0
1242.5
1245.0
1247.5

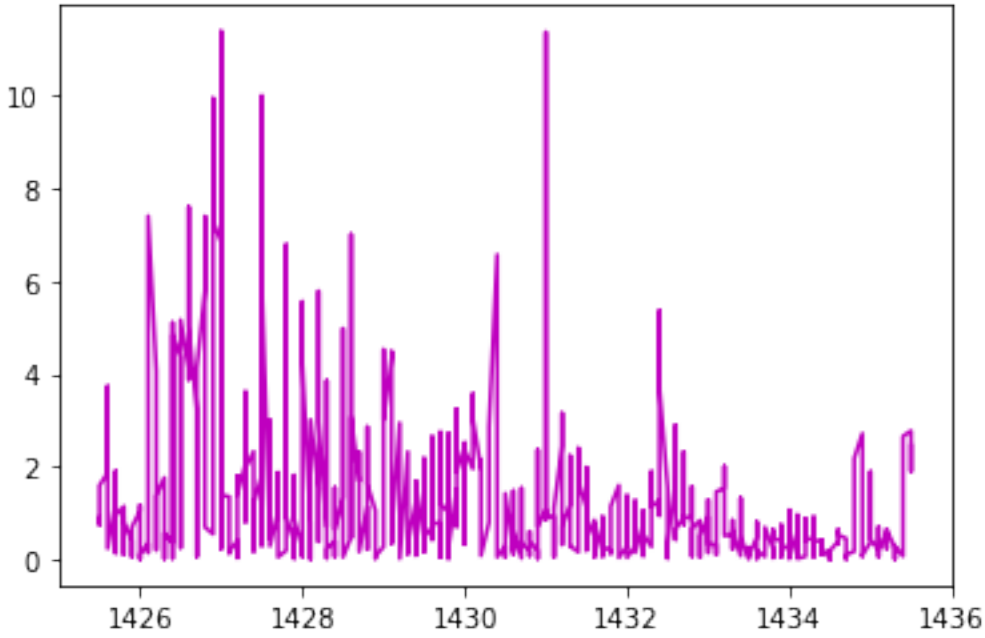

500
400
300
200
100
0
−100
1248
1250
1252
1254
1256

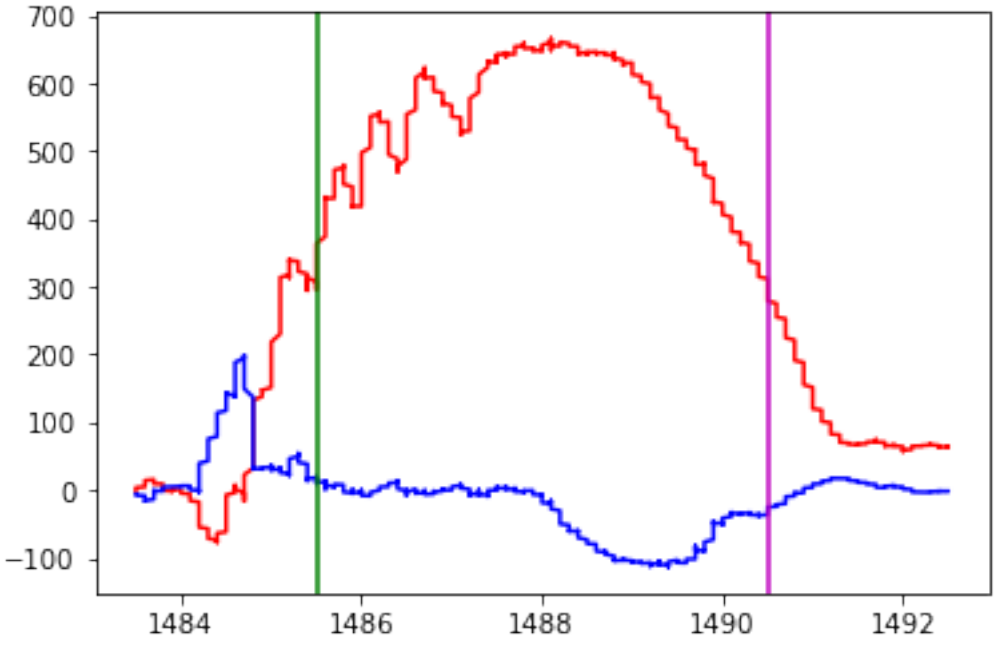

6
5
4
3
2
1
0
1248
1250
1252
1254
1256

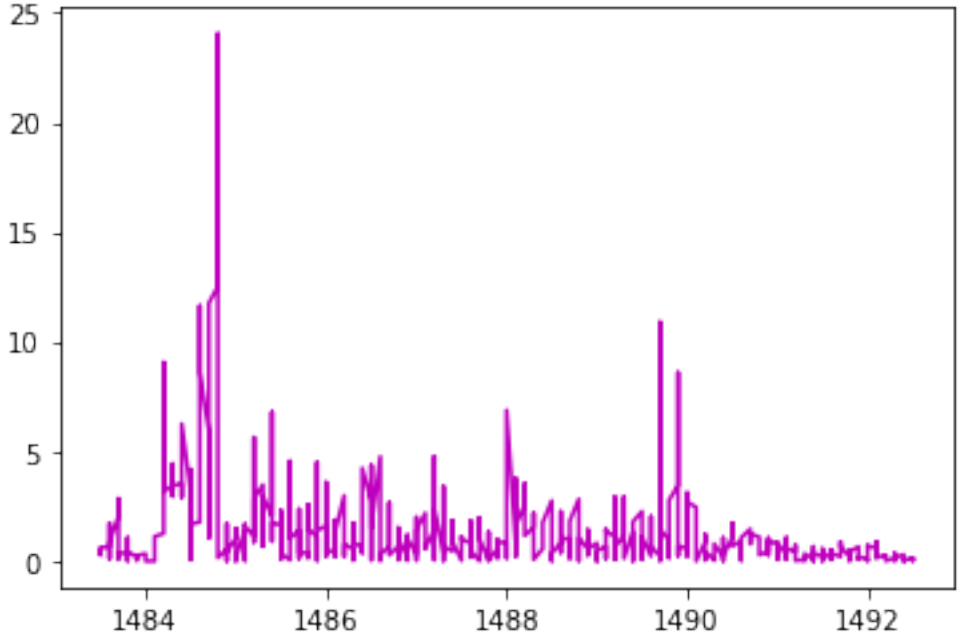

500
400
300
200
100
0
−100
1256
1258
1260
1262
1264

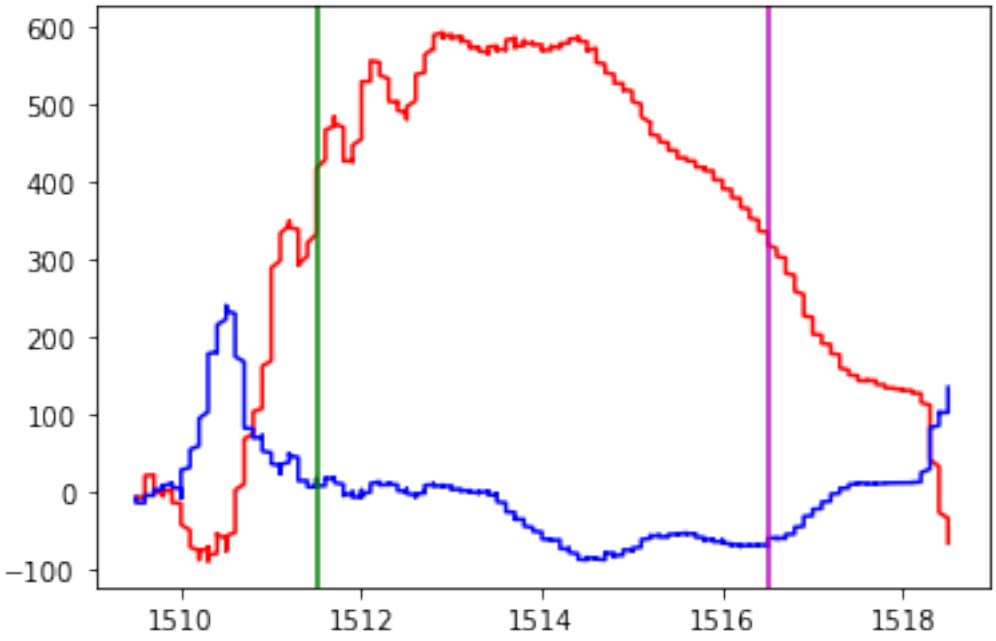

7
6
5
4
3
2
1
0
1256
1258
1260
1262
1264

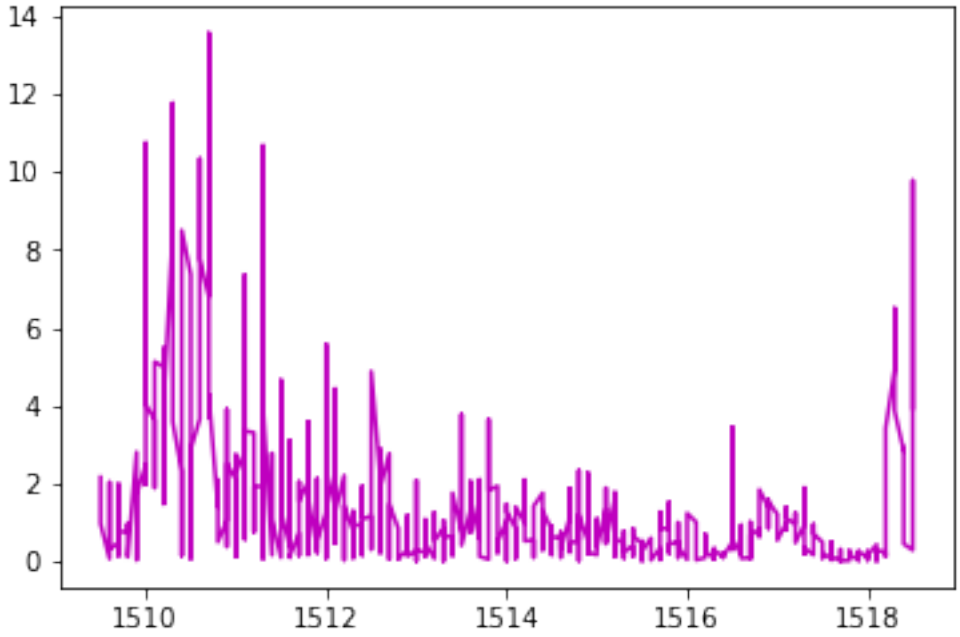

700
600
500
400
300
200
100
0
−100
1426
1428
1430
1432
1434
1436

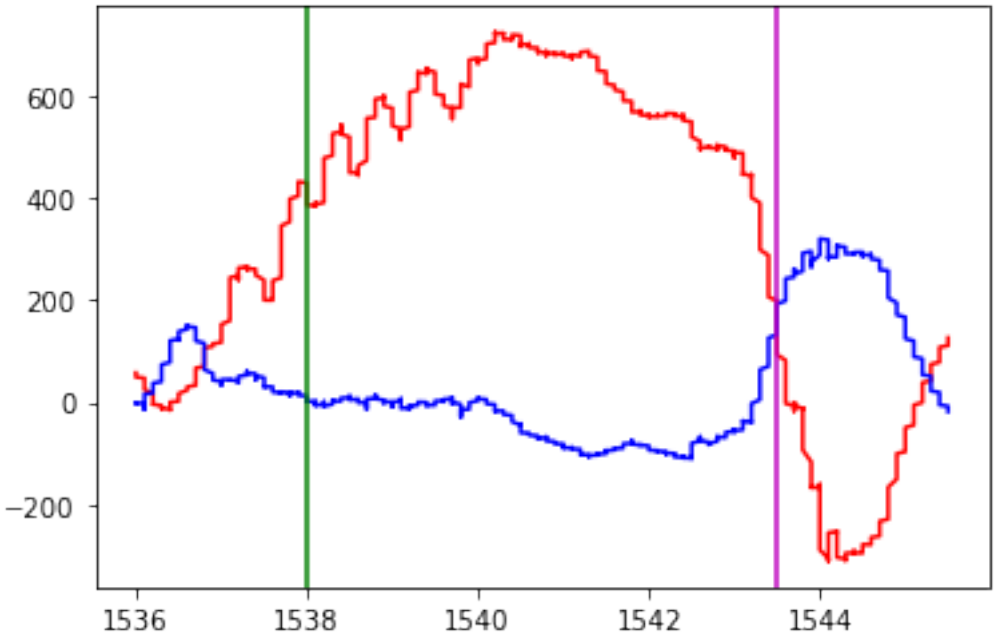

10
8
6
4
2
0
1426
1428
1430
1432
1434
1436

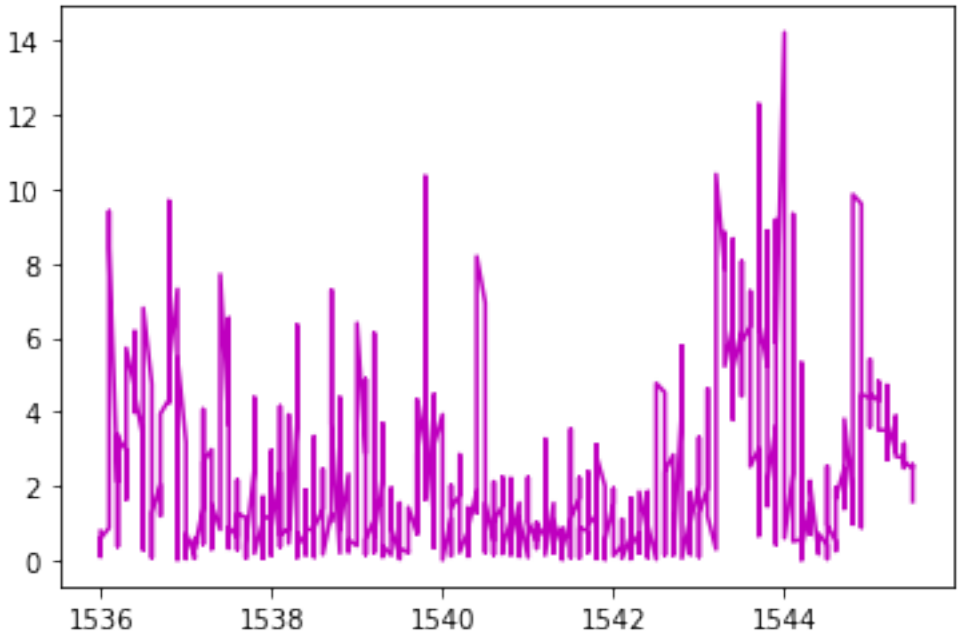

700
600
500
400
300
200
100
0
−100
1484
1486
1488
1490
1492

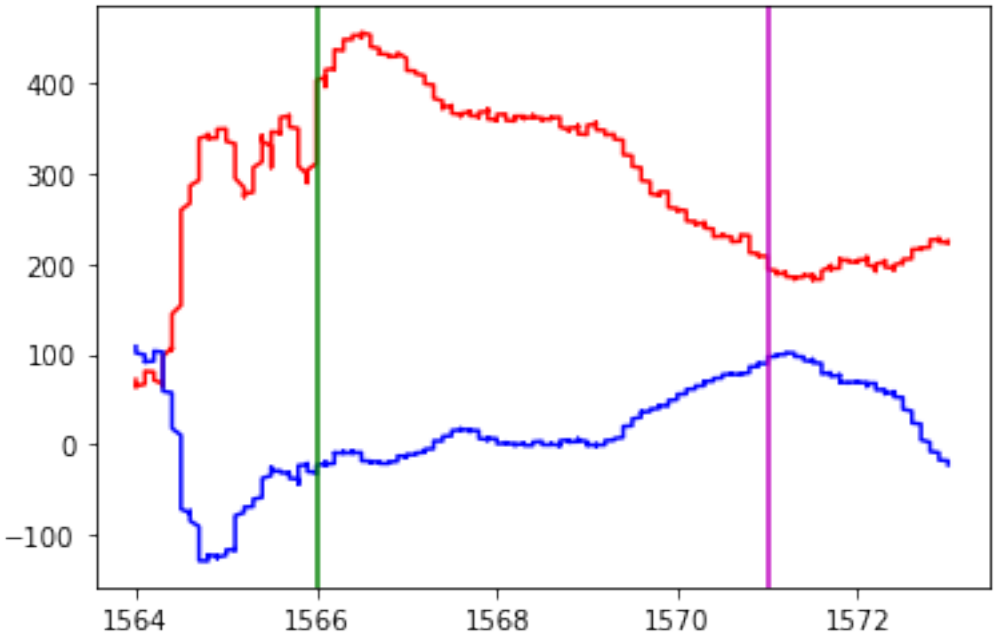

25
20
15
10
5
0
1484
1486
1488
1490
1492

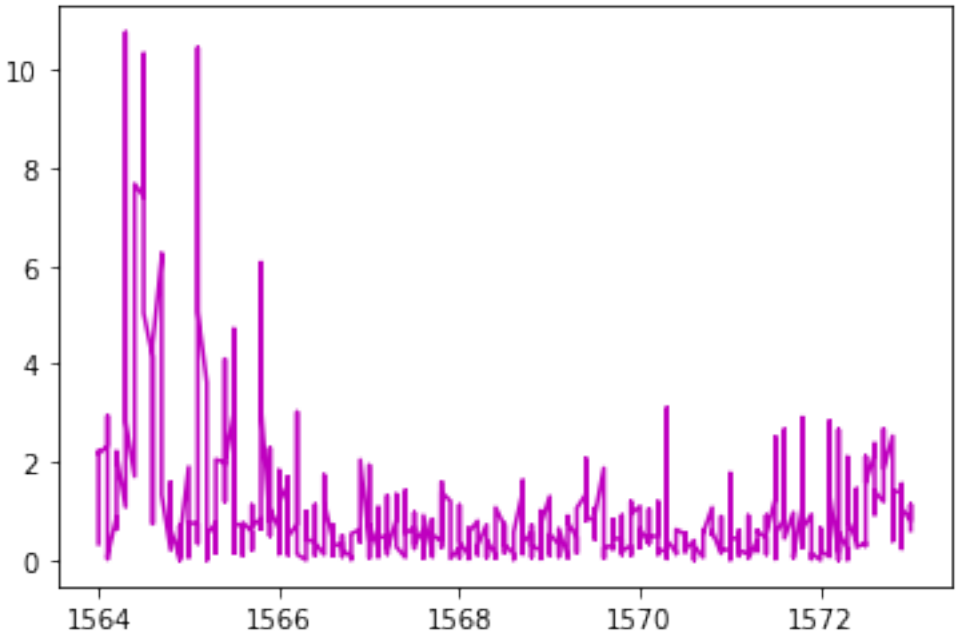

600
500
400
300
200
100
0
−100
1510
1512
1514
1516
1518

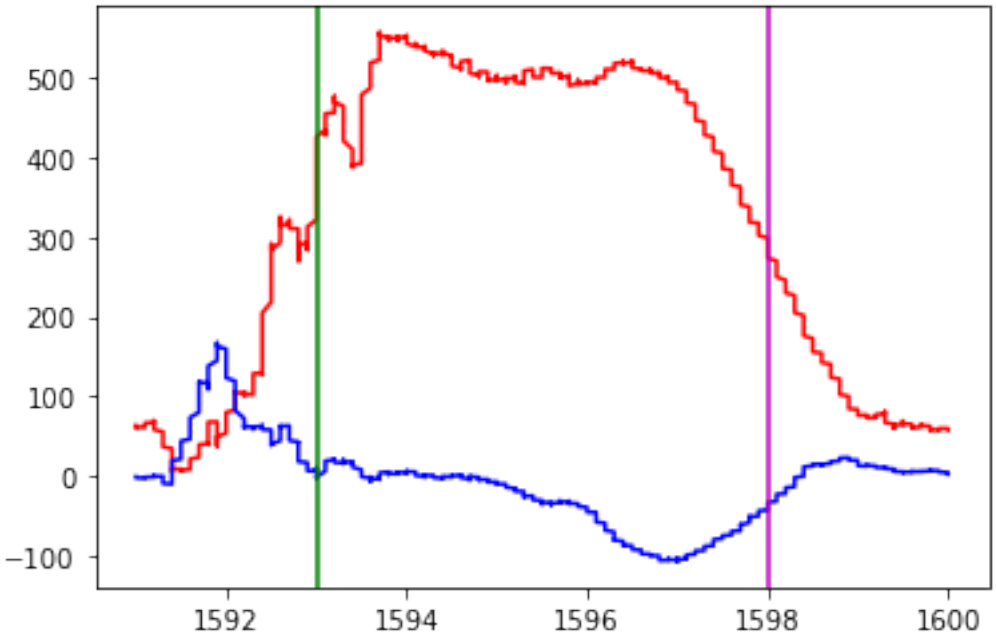

14
12
10
8
6
4
2
0
1510
1512
1514
1516
1518

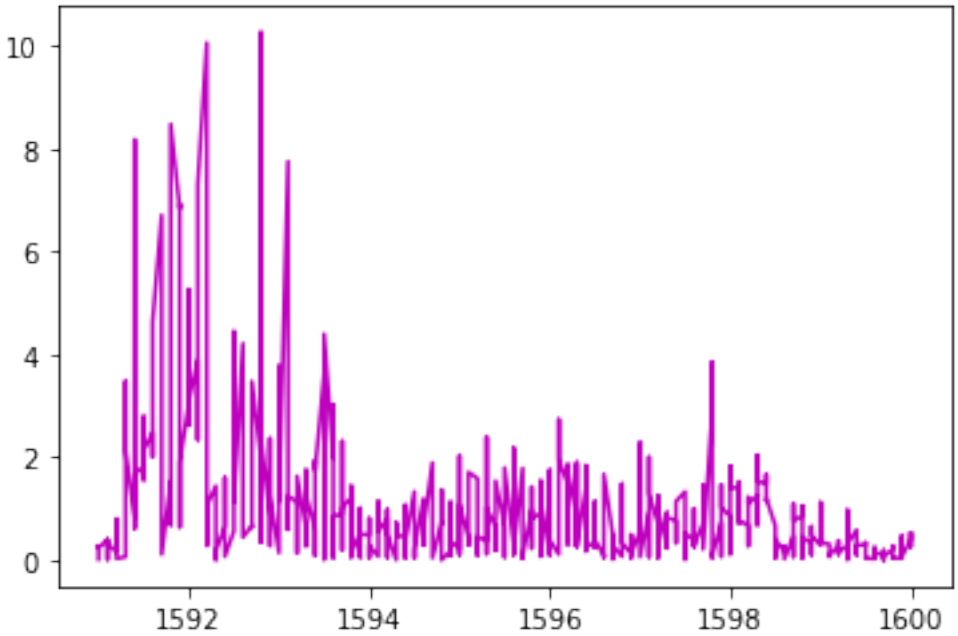

600
400
200
0
−200
1536
1538
1540
1542
1544

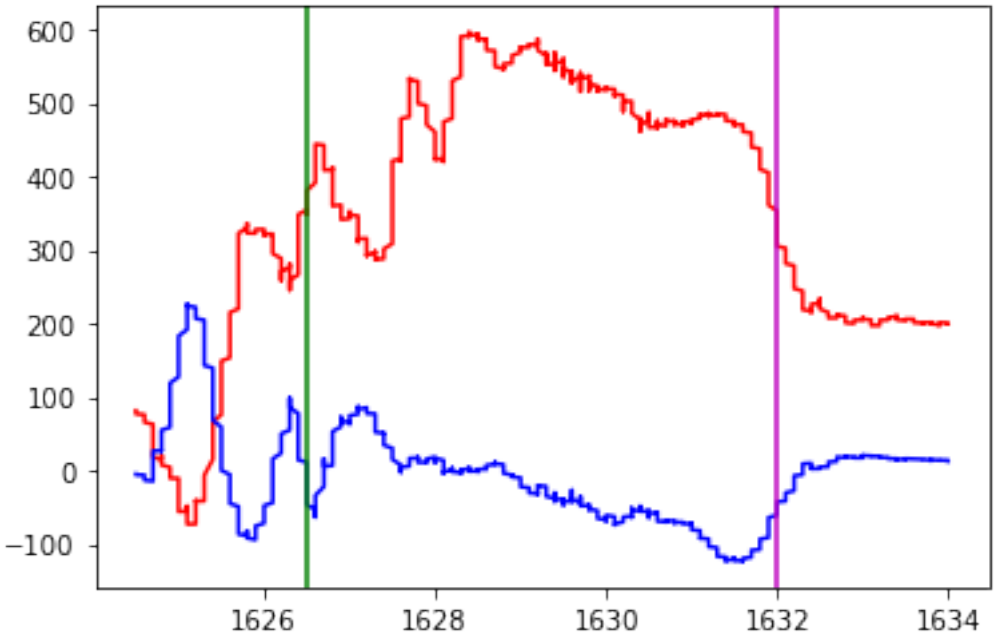

14
12
10
8
6
4
2
0
1536
1538
1540
1542
1544

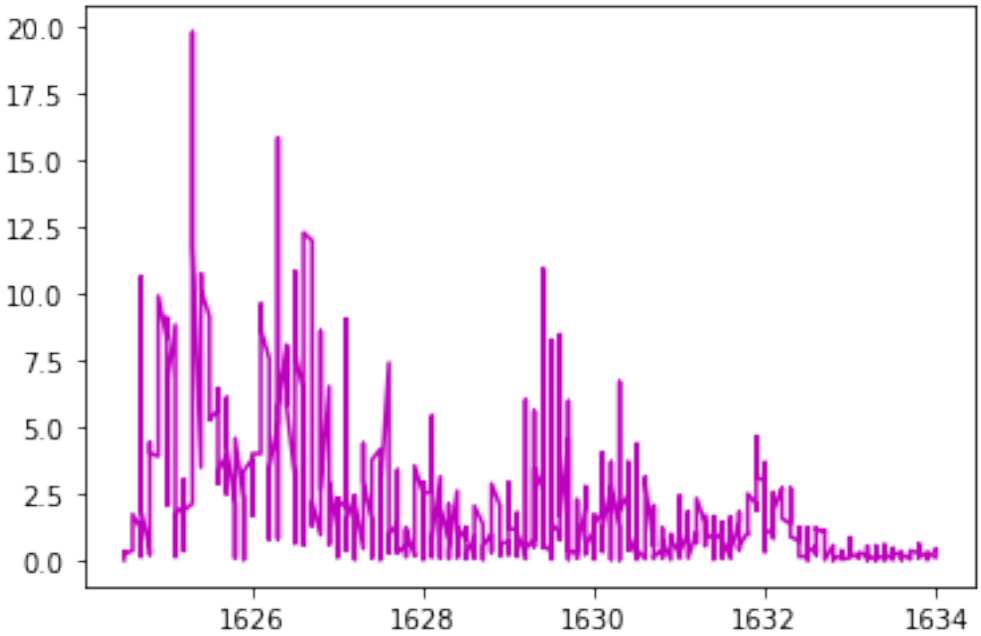

400
300
200
100
0
−100
1564
1566
1568
1570
1572

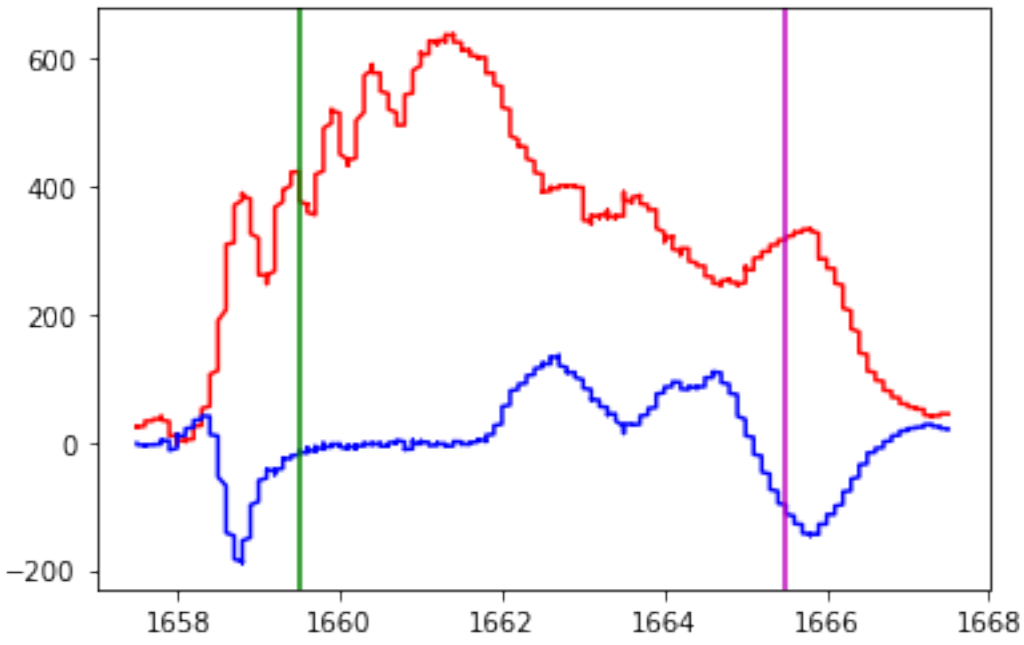

10
8
6
4
2
0
1564
1566
1568
1570
1572

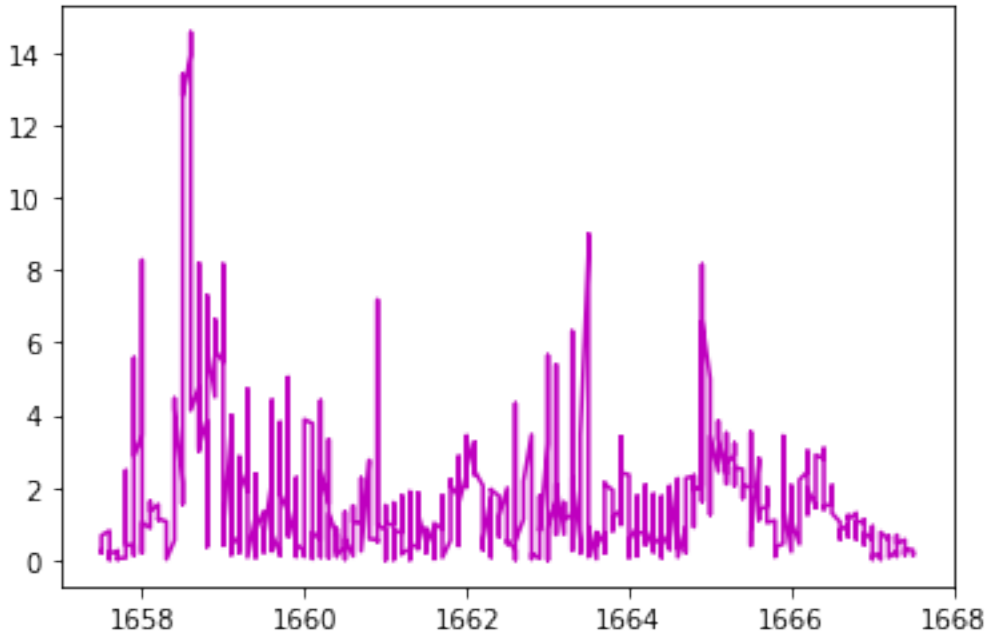

500
400
300
200
100
0
−100
1592
1594
1596
1598
1600

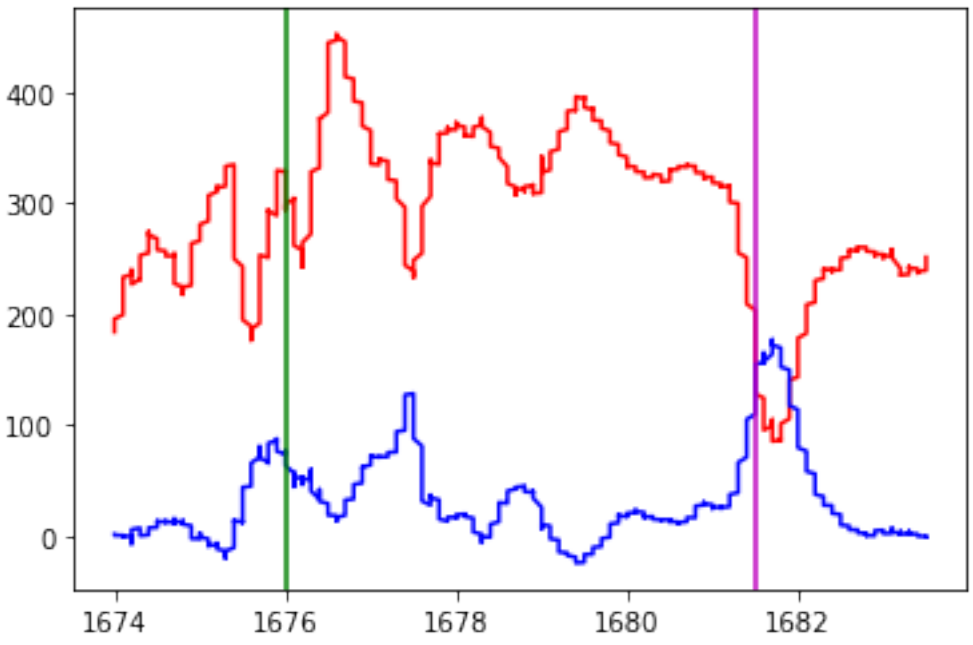

10
8
6
4
2
0
1592
1594
1596
1598
1600

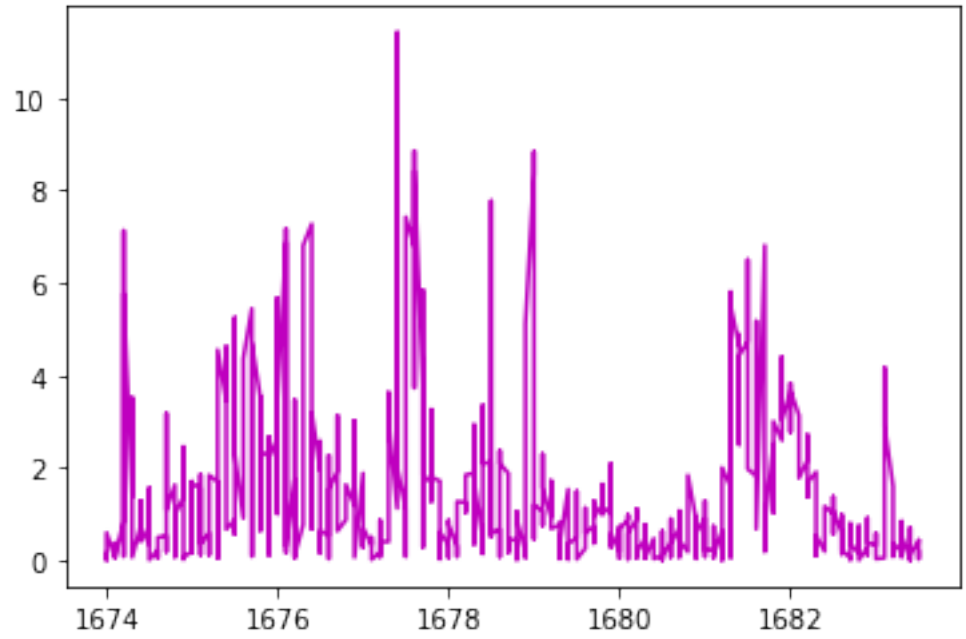

600
500
400
300
200
100
0
−100
1626
1628
1630
1632
1634

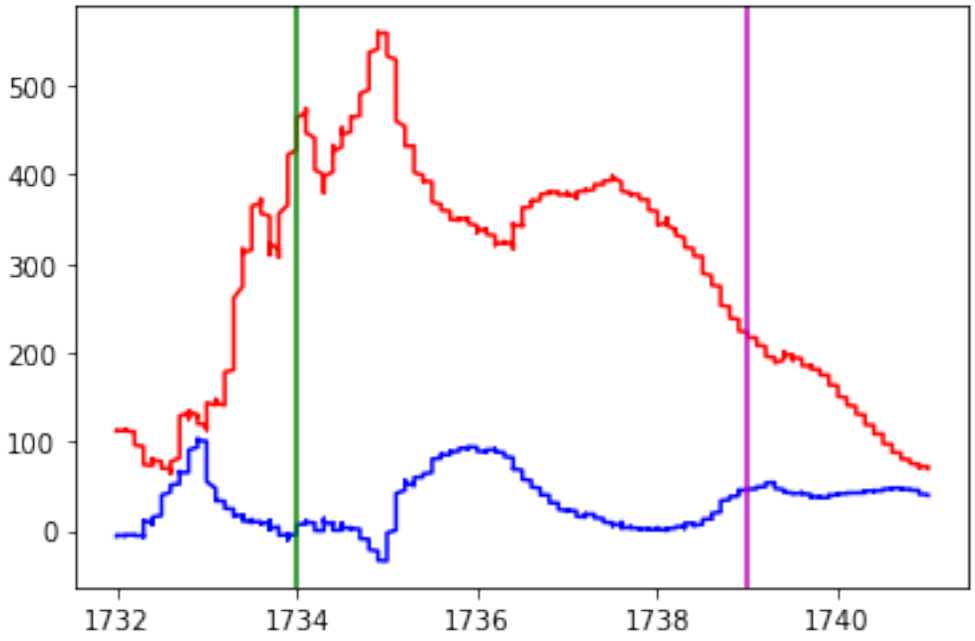

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1626
1628
1630
1632
1634

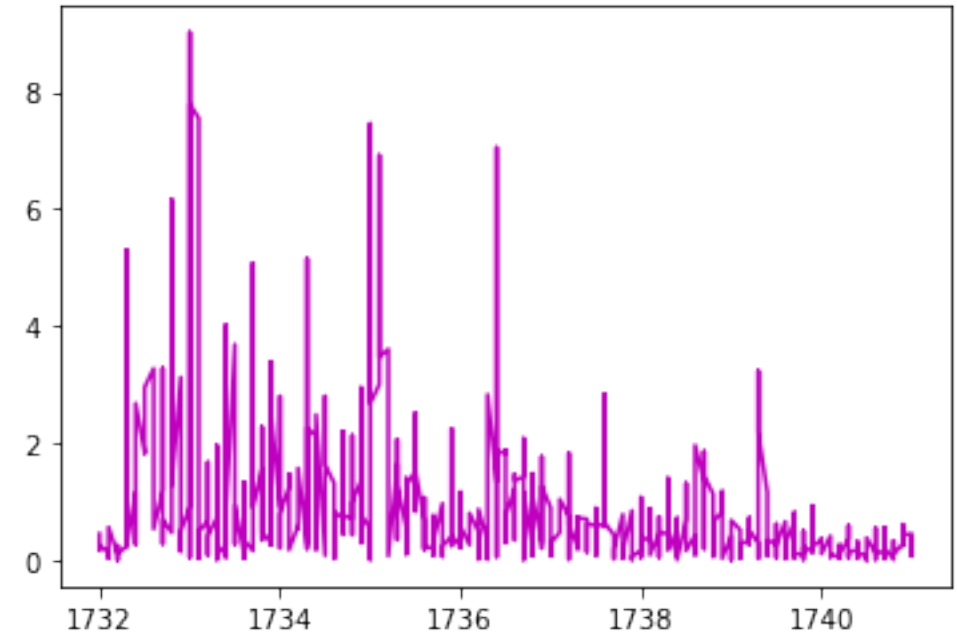

600
400
200
0
−200
1658
1660
1662
1664
1666
1668

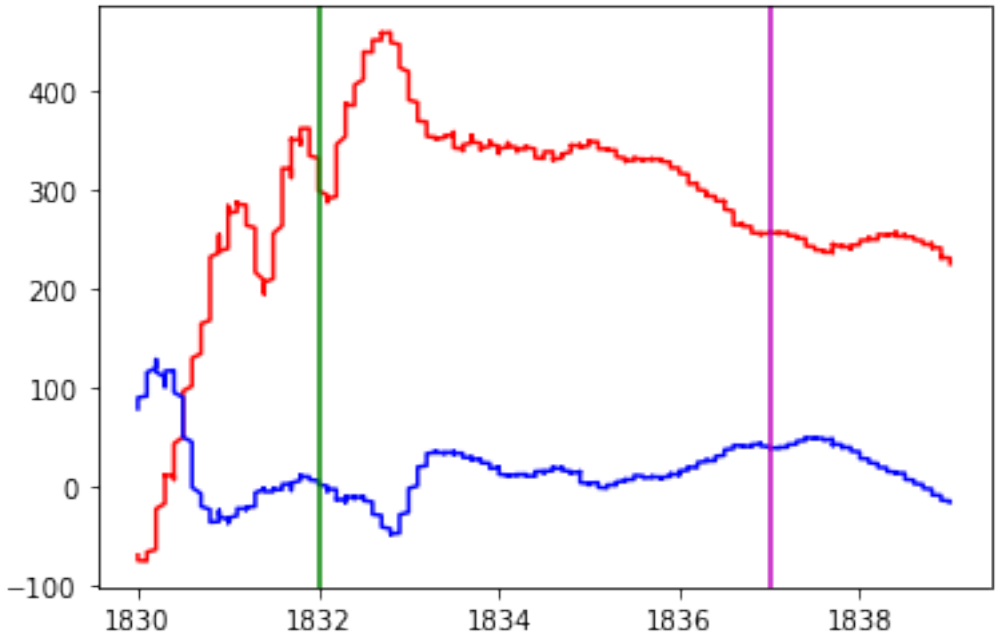

14
12
10
8
6
4
2
0
1658
1660
1662
1664
1666
1668

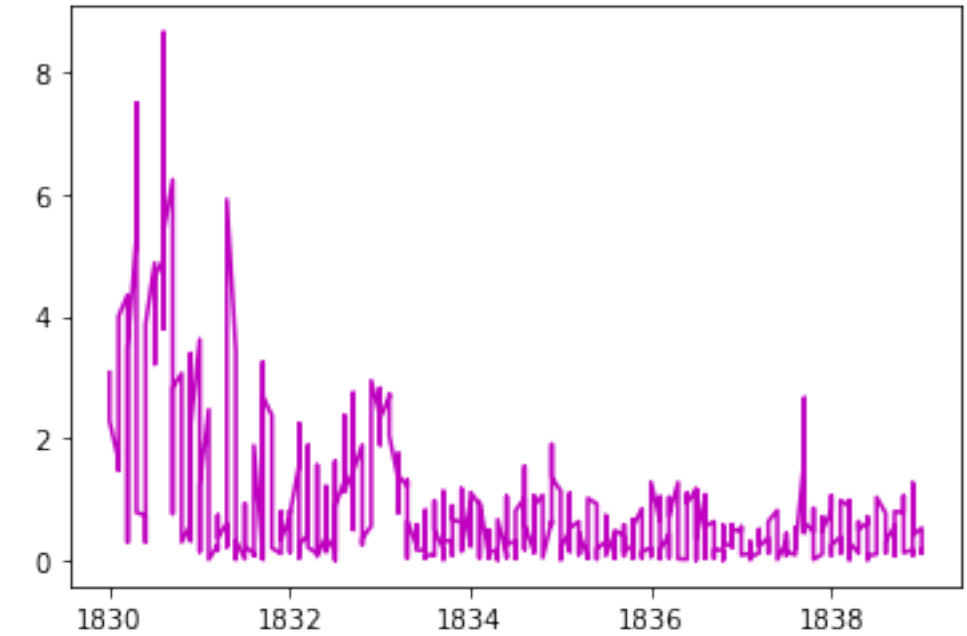

400
300
200
100
0
1674
1676
1678
1680
1682

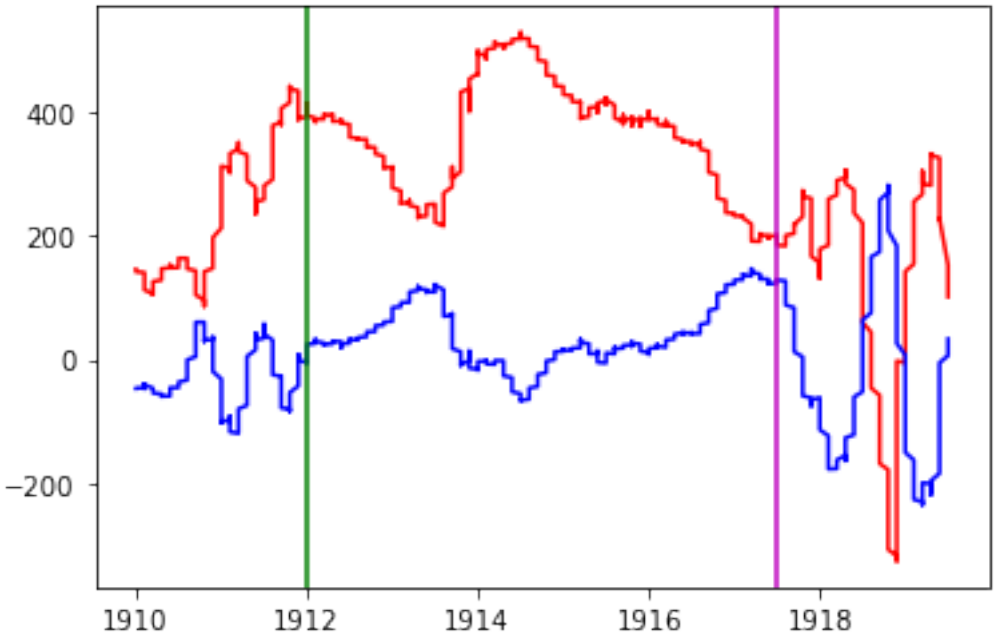

10
8
6
4
2
0
1674
1676
1678
1680
1682

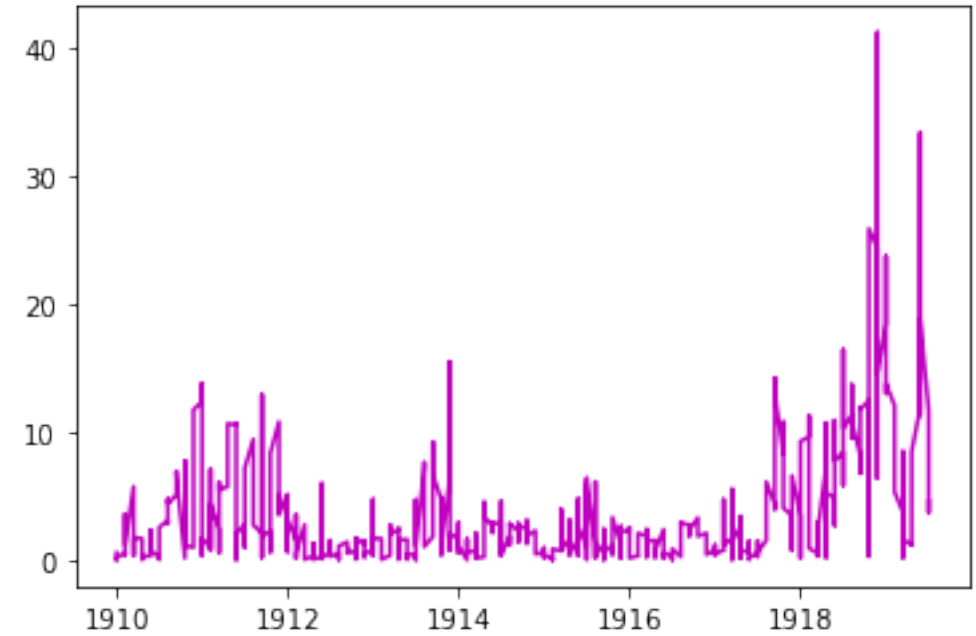

500
400
300
200
100
0
1732
1734
1736
1738
1740

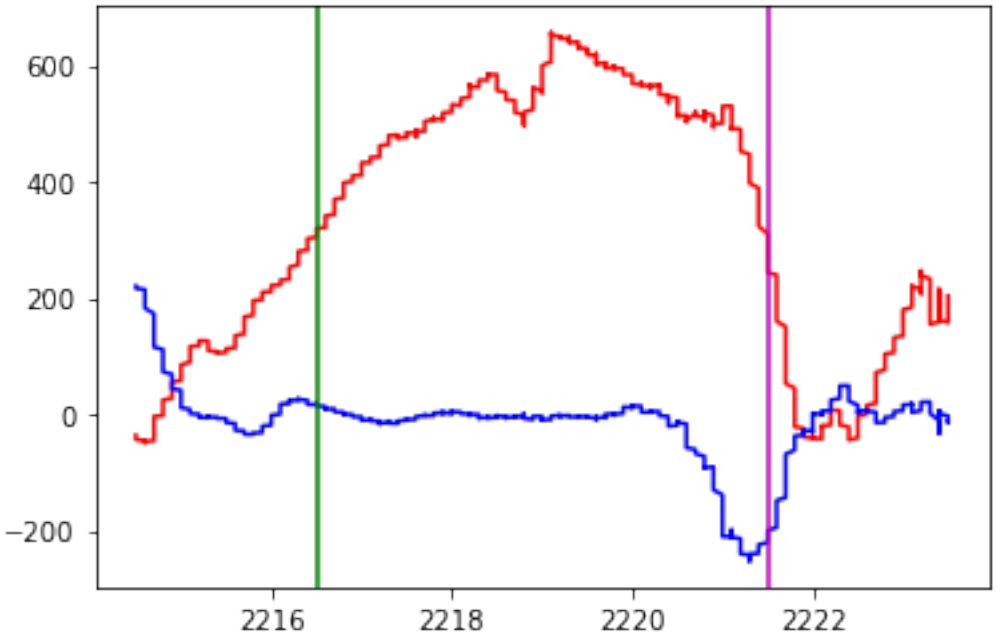

8
6
4
2
0
1732
1734
1736
1738
1740

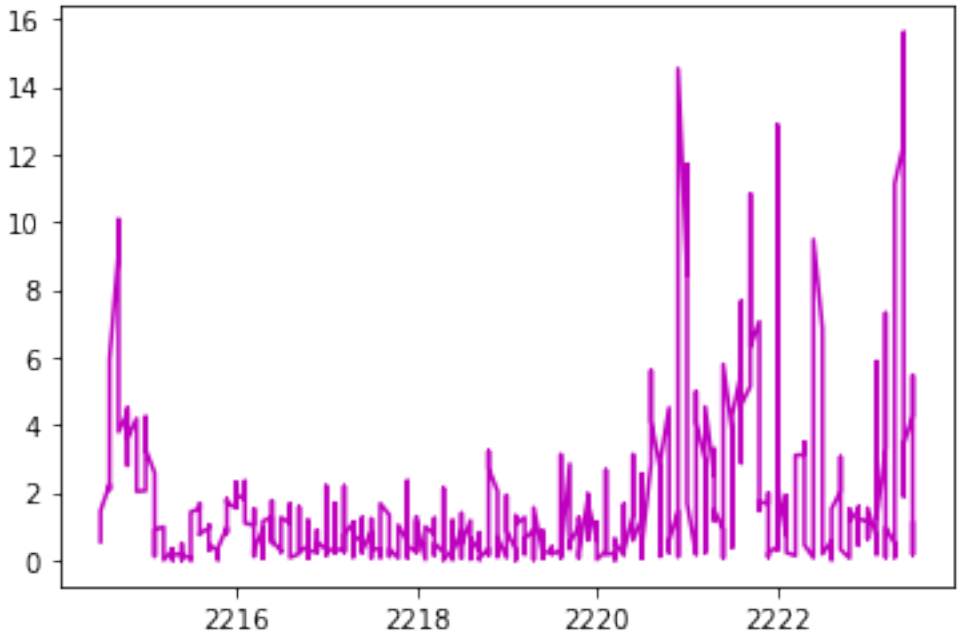

400
300
200
100
0
−100
1830
1832
1834
1836
1838

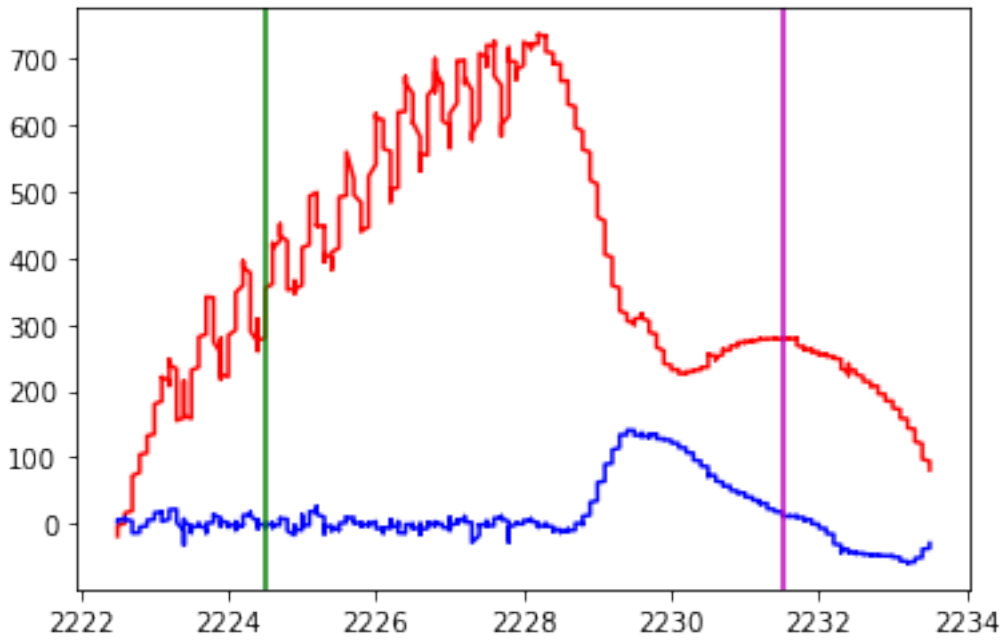

8
6
4
2
0
1830
1832
1834
1836
1838

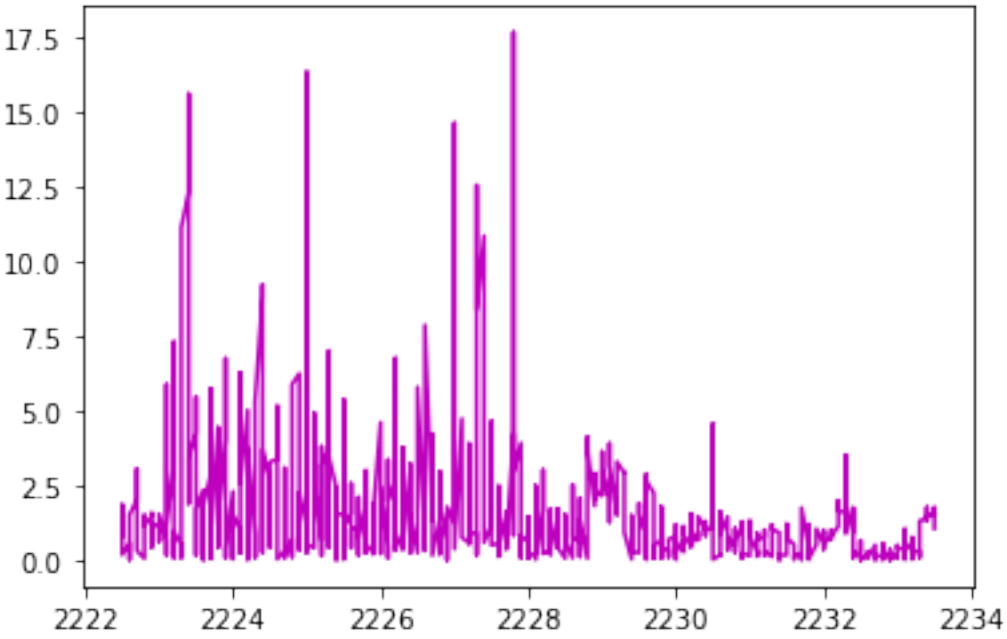

400
200
0
−200
1910
1912
1914
1916
1918

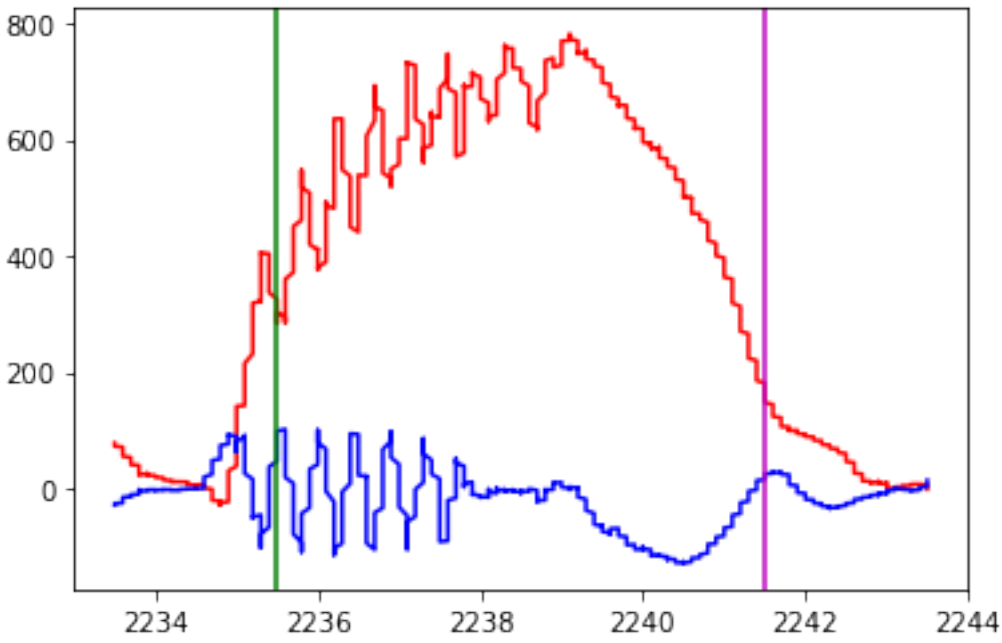

40
30
20
10
0
1910
1912
1914
1916
1918

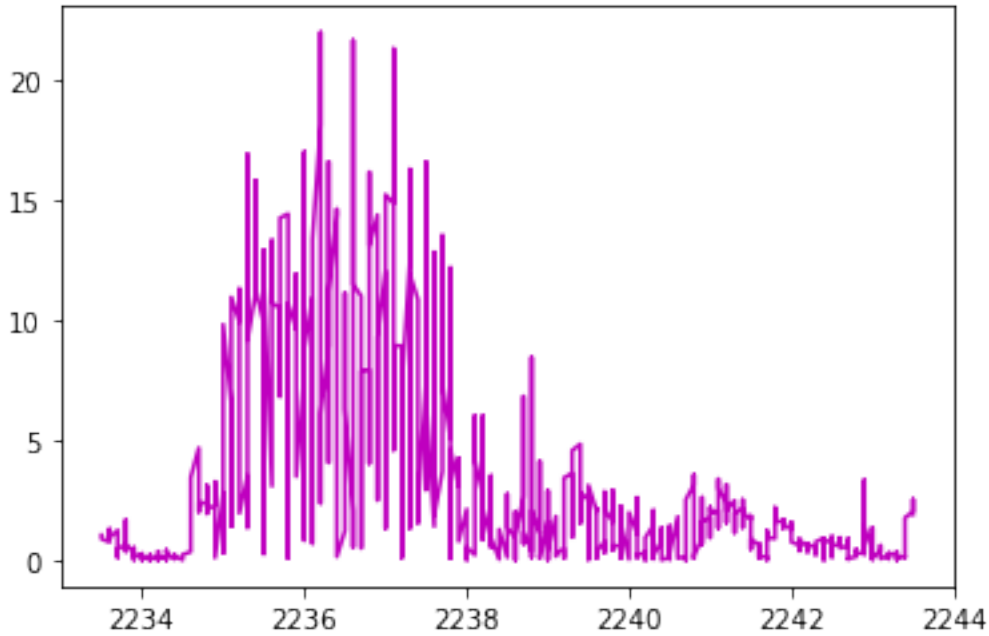

600
400
200
0
−200
2216
2218
2220
2222

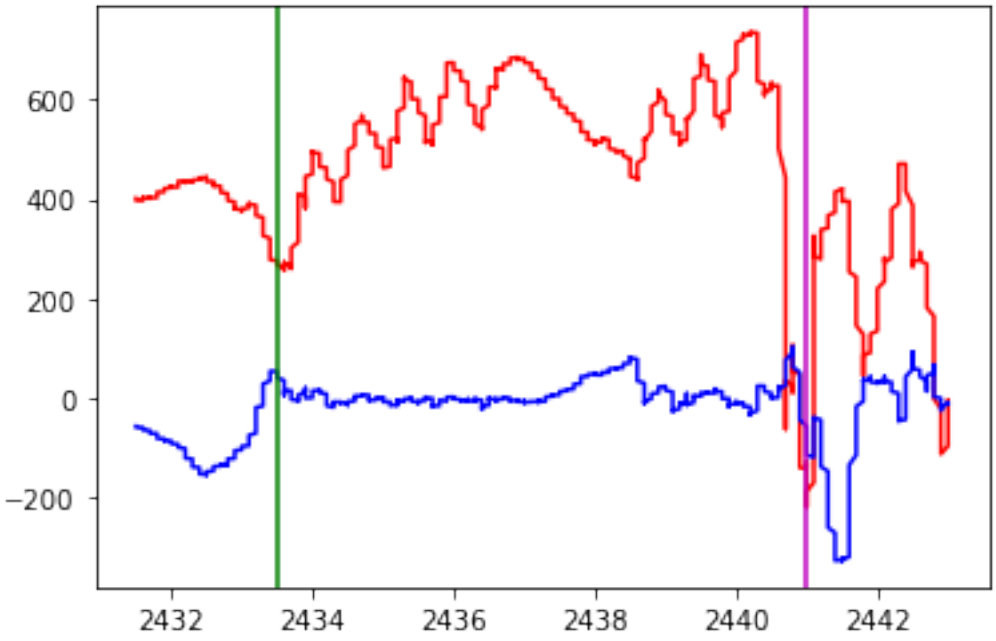

16
14
12
10
8
6
4
2
0
2216
2218
2220
2222

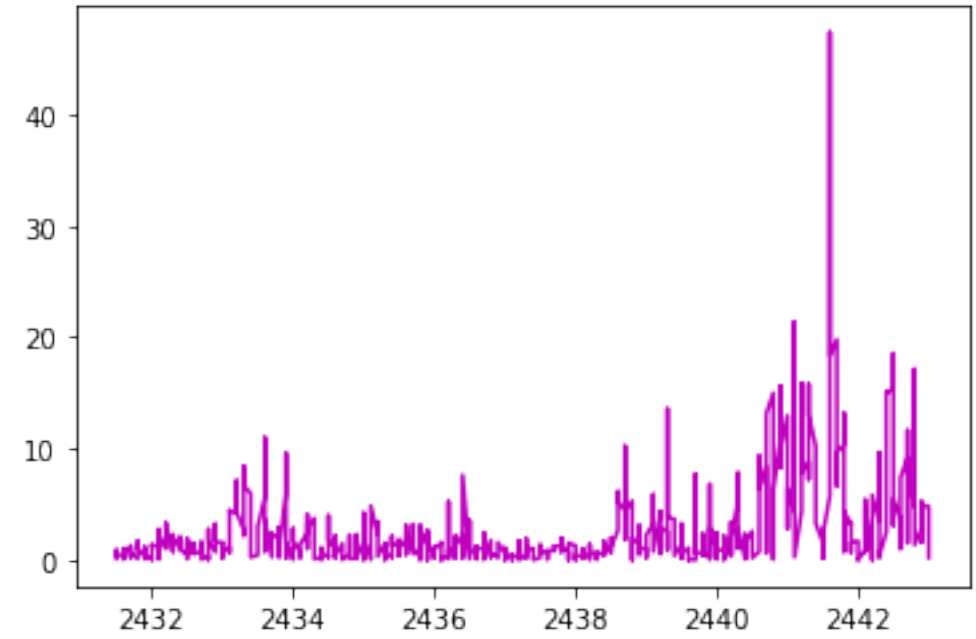

700
600
500
400
300
200
100
0
2222
2224
2226
2228
2230
2232
2234

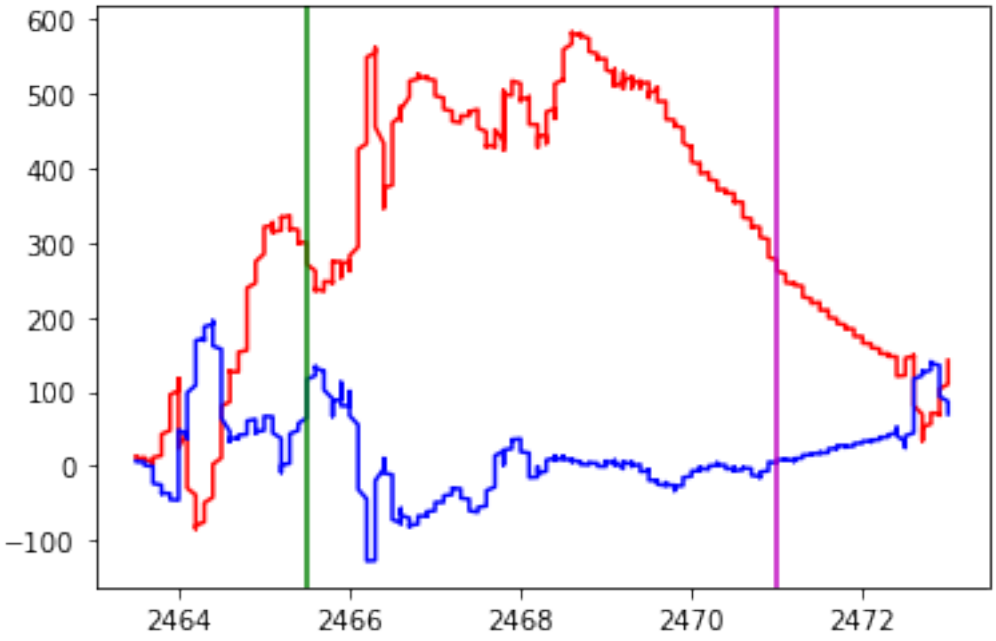

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2222
2224
2226
2228
2230
2232
2234

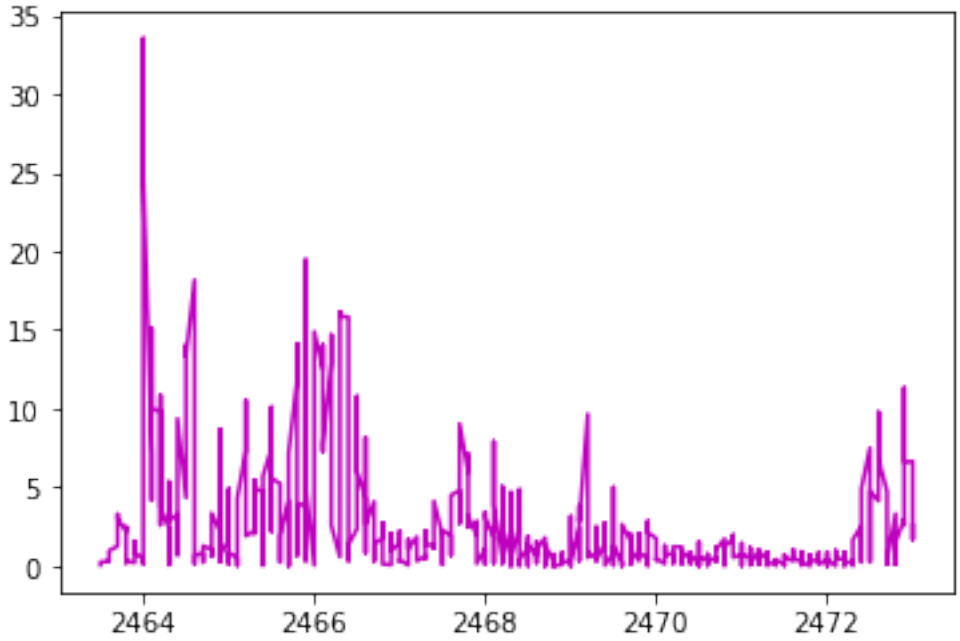

800
600
400
200
0
2234
2236
2238
2240
2242
2244

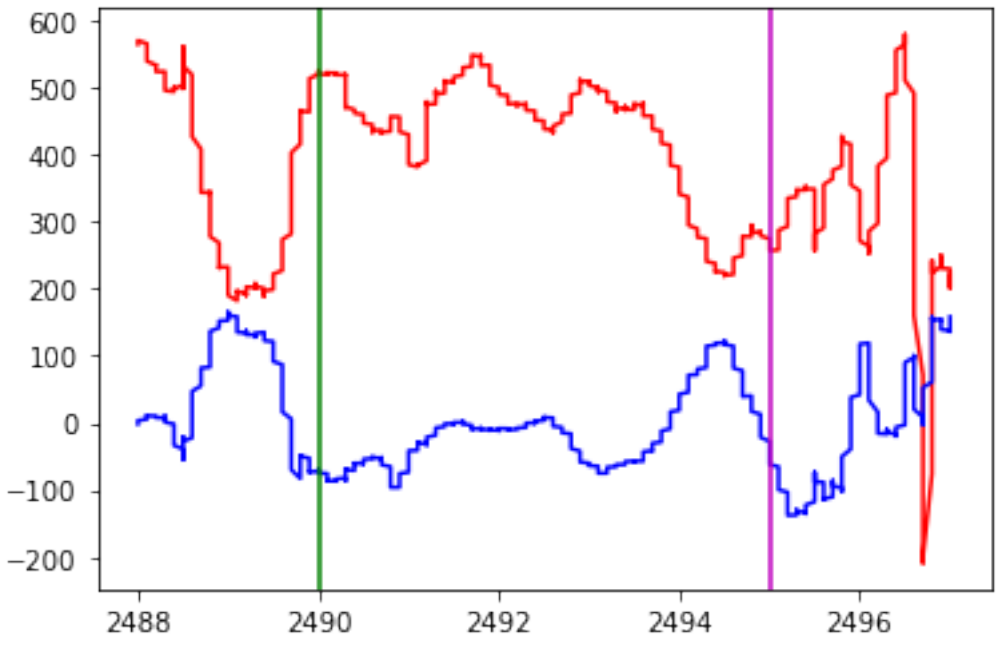

20
15
10
5
0
2234
2236
2238
2240
2242
2244

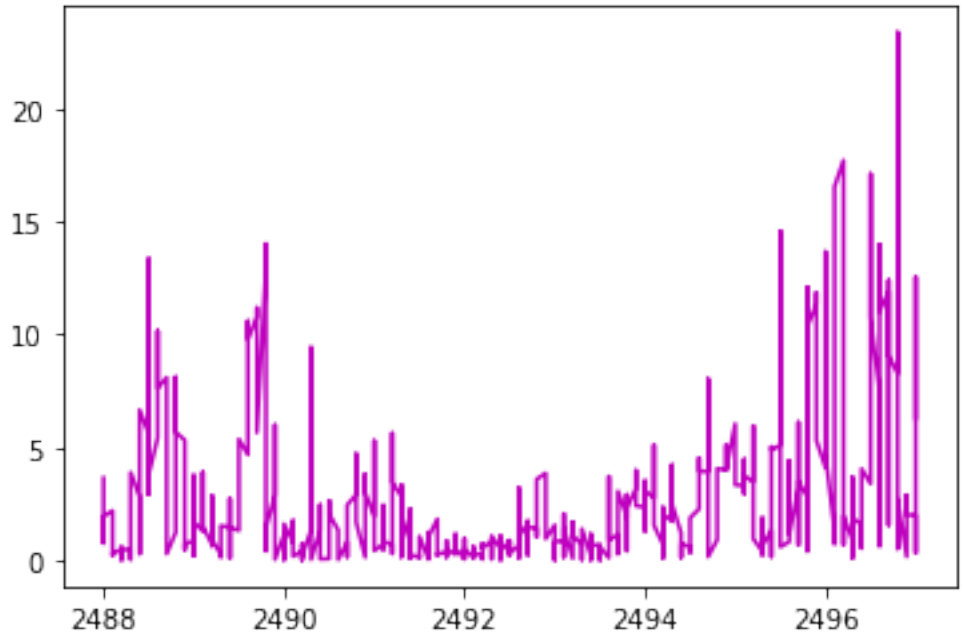

600
400
200
0
−200
2432
2434
2436
2438
2440
2442

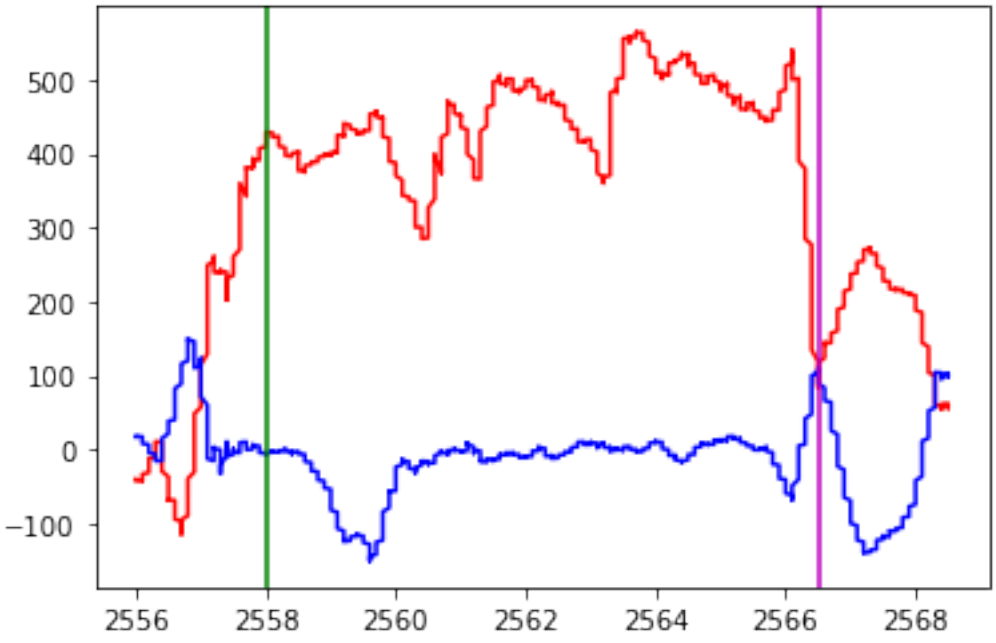

40
30
20
10
0
2432
2434
2436
2438
2440
2442

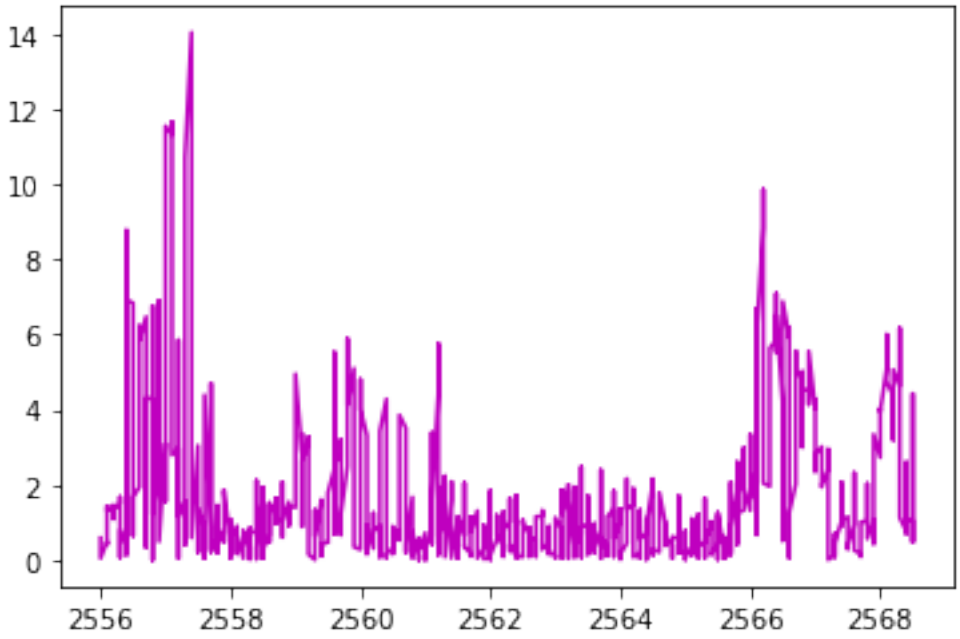

600
500
400
300
200
100
0
−100
2464
2466
2468
2470
2472

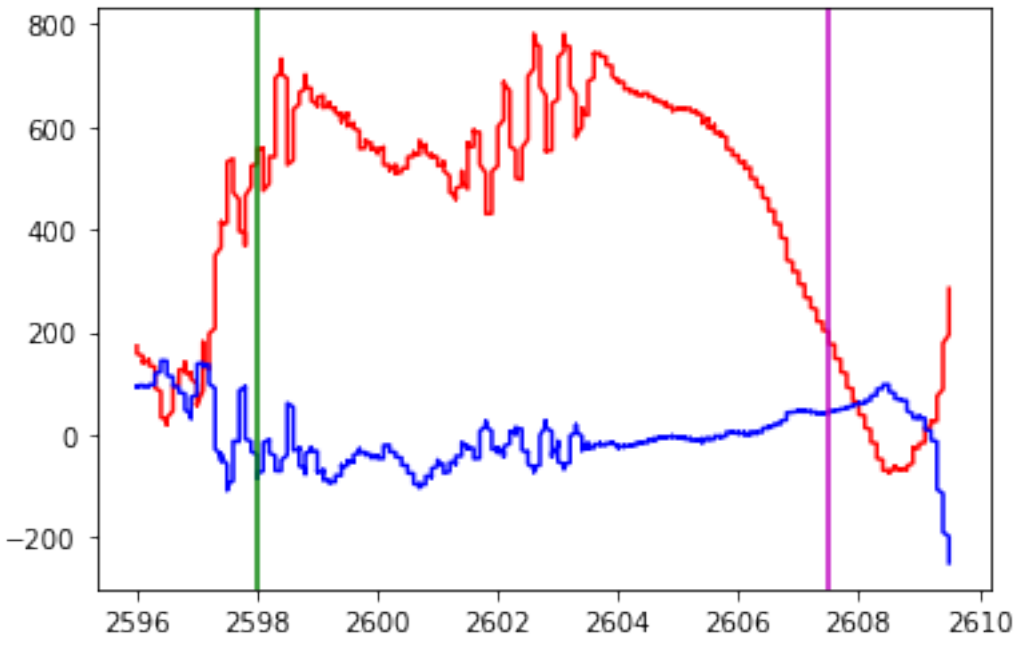

35
30
25
20
15
10
5
0
2464
2466
2468
2470
2472

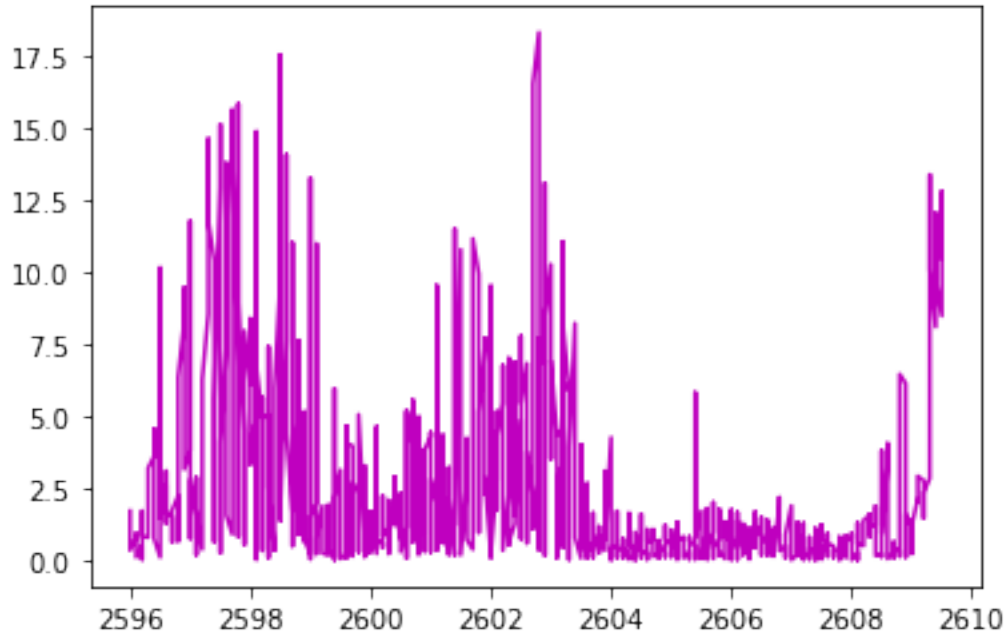

600
500
400
300
200
100
0
−100
−200
2488
2490
2492
2494
2496

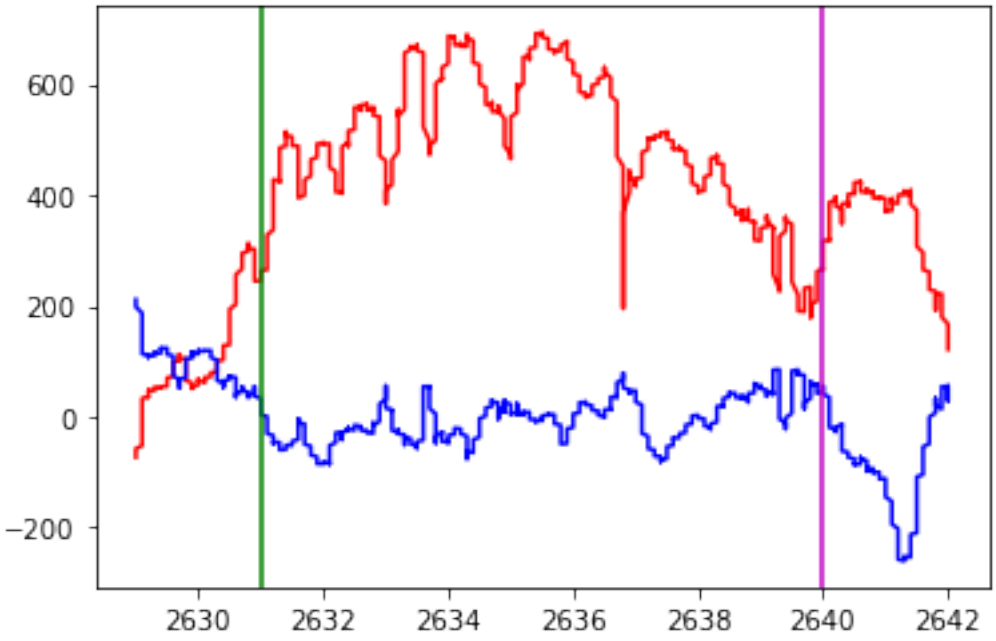

20
15
10
5
0
2488
2490
2492
2494
2496

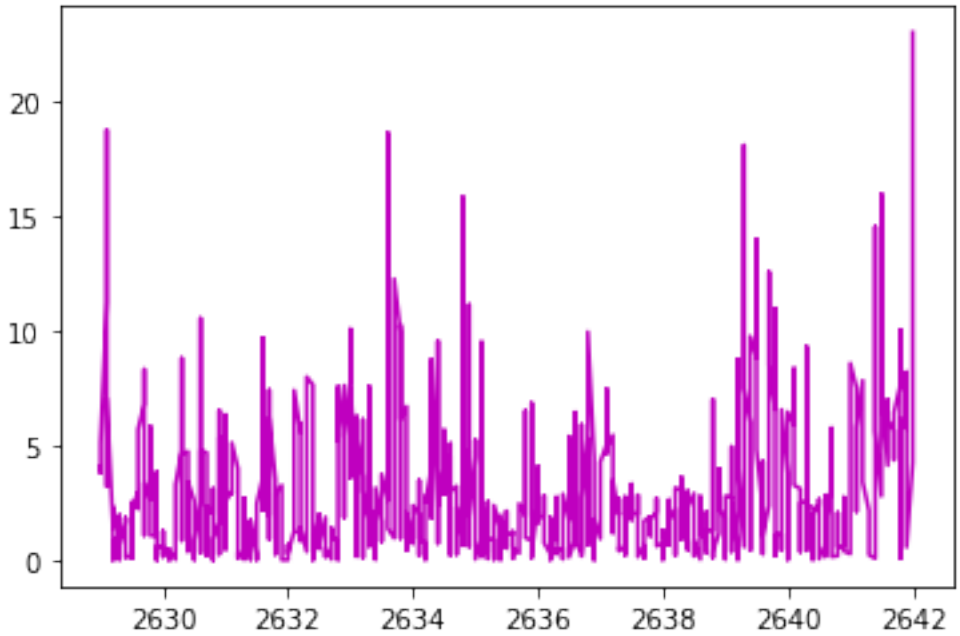

500
400
300
200
100
0
−100
2556
2558
2560
2562
2564
2566
2568

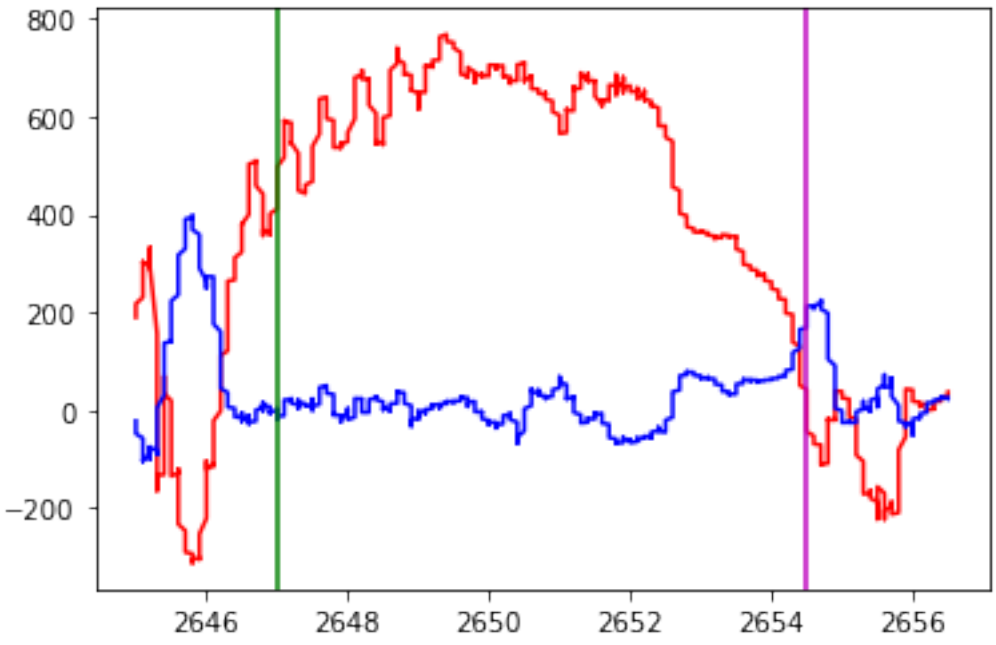

14
12
10
8
6
4
2
0
2556
2558
2560
2562
2564
2566
2568

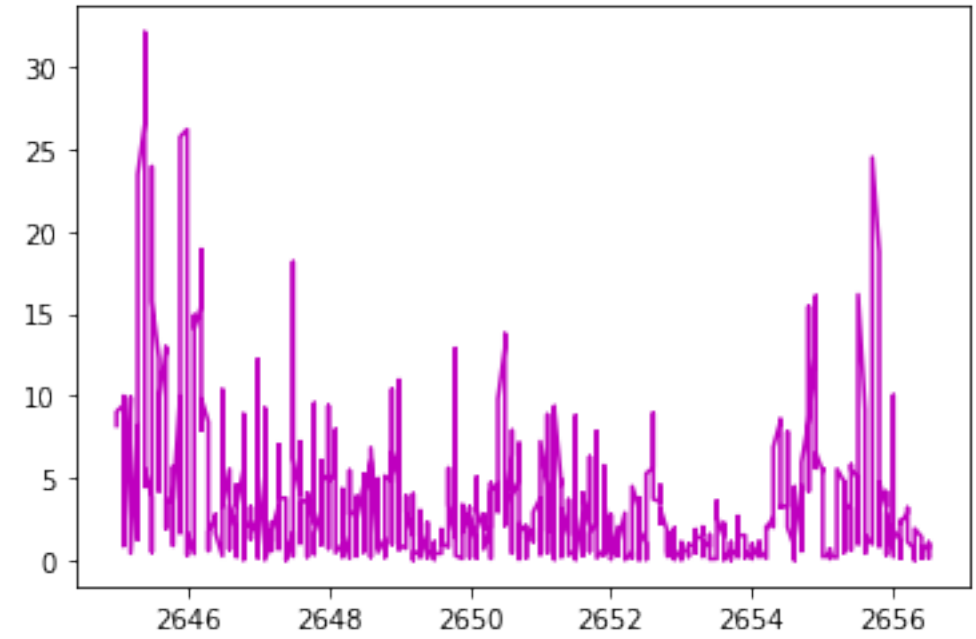

800
600
400
200
0
−200
2596
2598
2600
2602
2604
2606
2608
2610

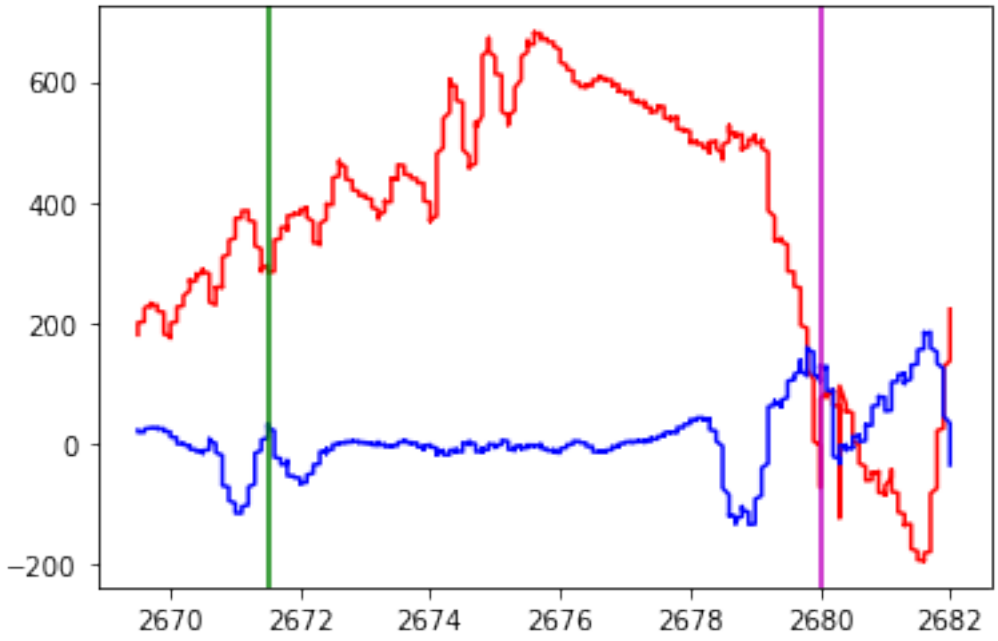

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2596
2598
2600
2602
2604
2606
2608
2610

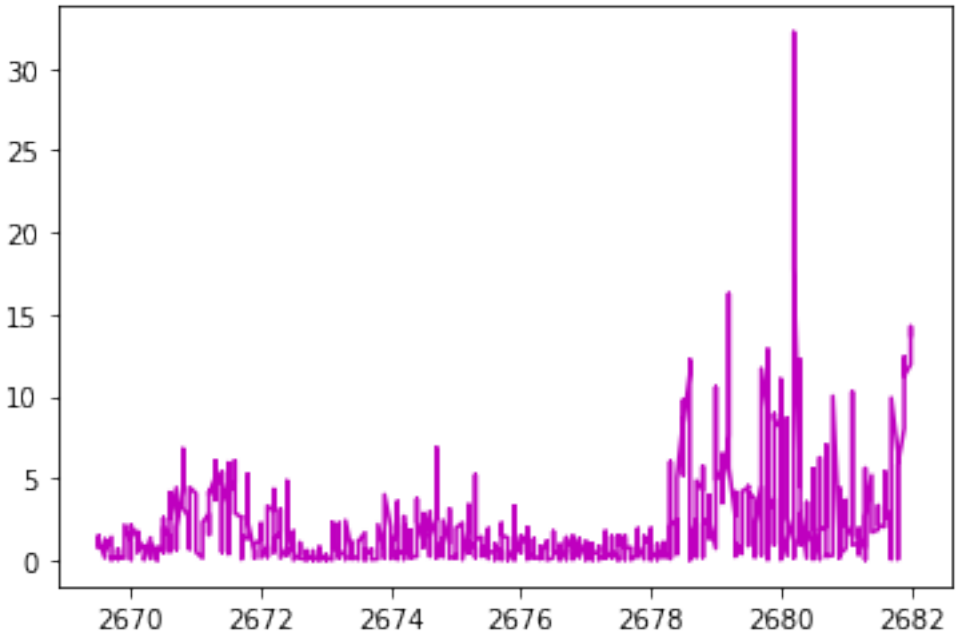

600
400
200
0
−200
2630
2632
2634
2636
2638
2640
2642

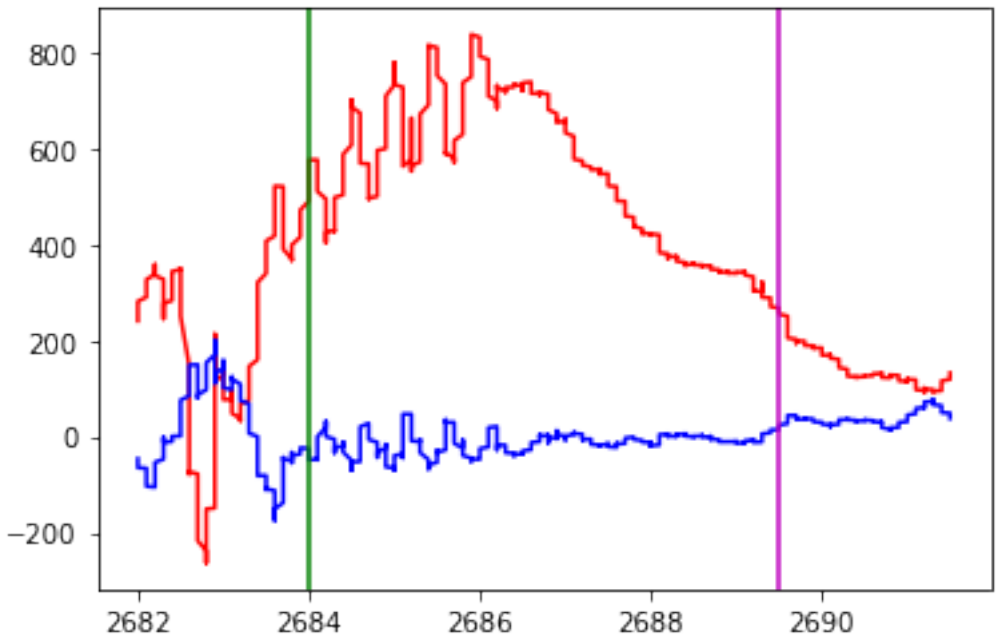

20
15
10
5
0
2630
2632
2634
2636
2638
2640
2642

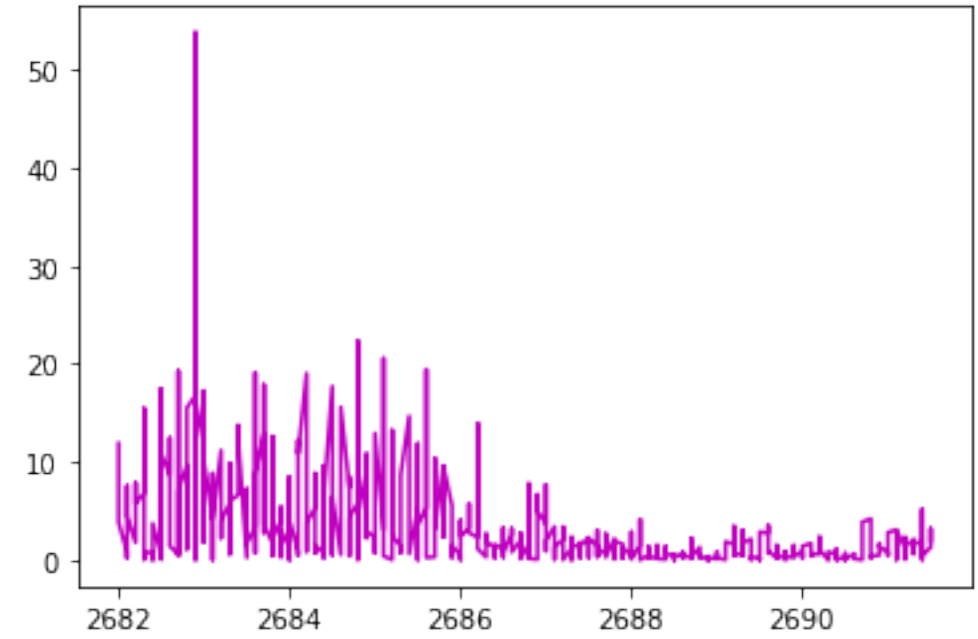

800
600
400
200
0
−200
2646
2648
2650
2652
2654
2656

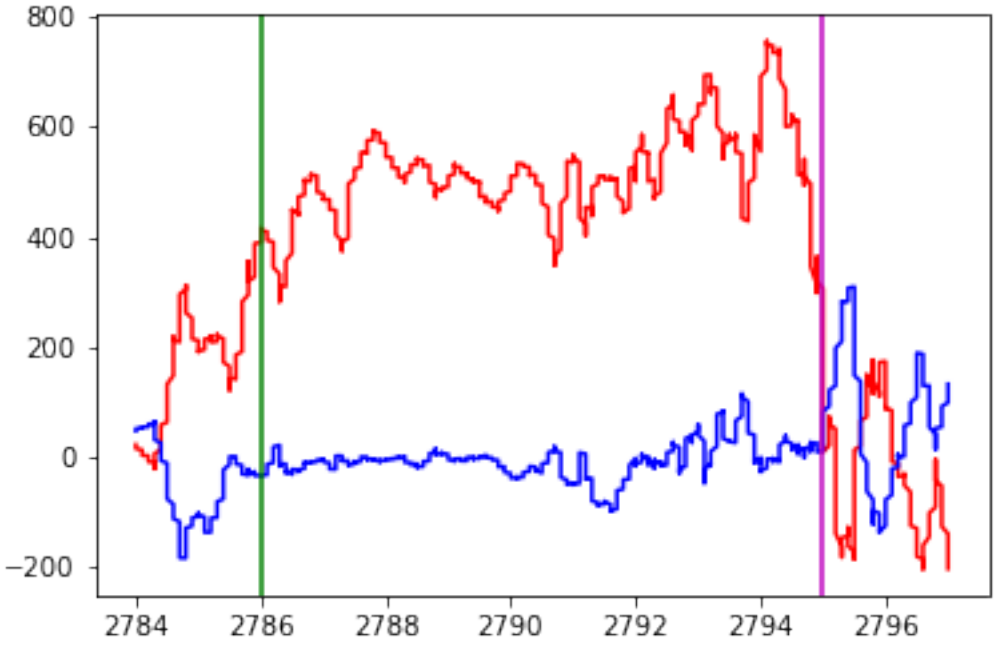

30
25
20
15
10
5
0
2646
2648
2650
2652
2654
2656

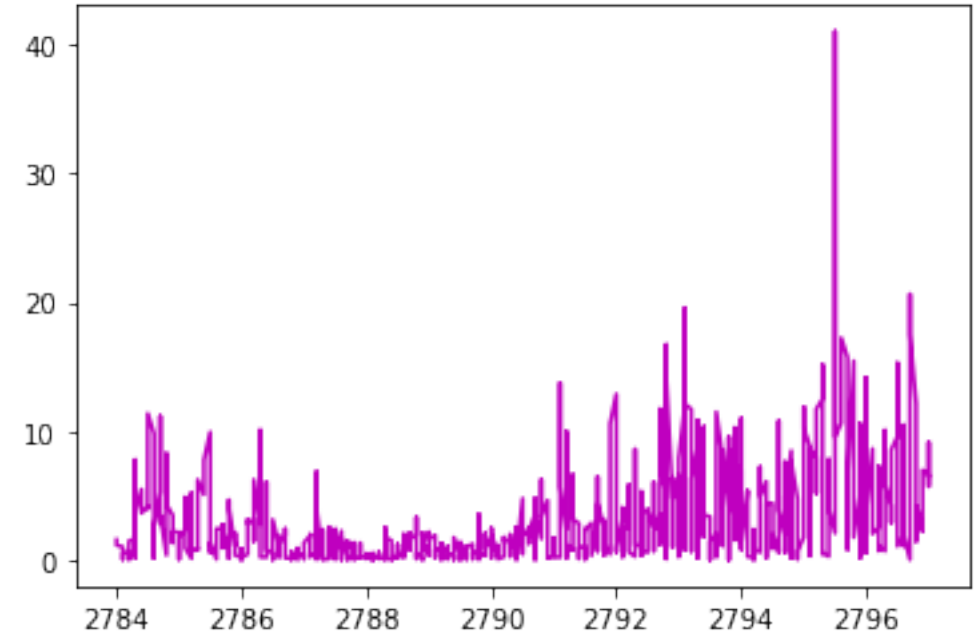

600
400
200
0
−200
2670
2672
2674
2676
2678
2680
2682

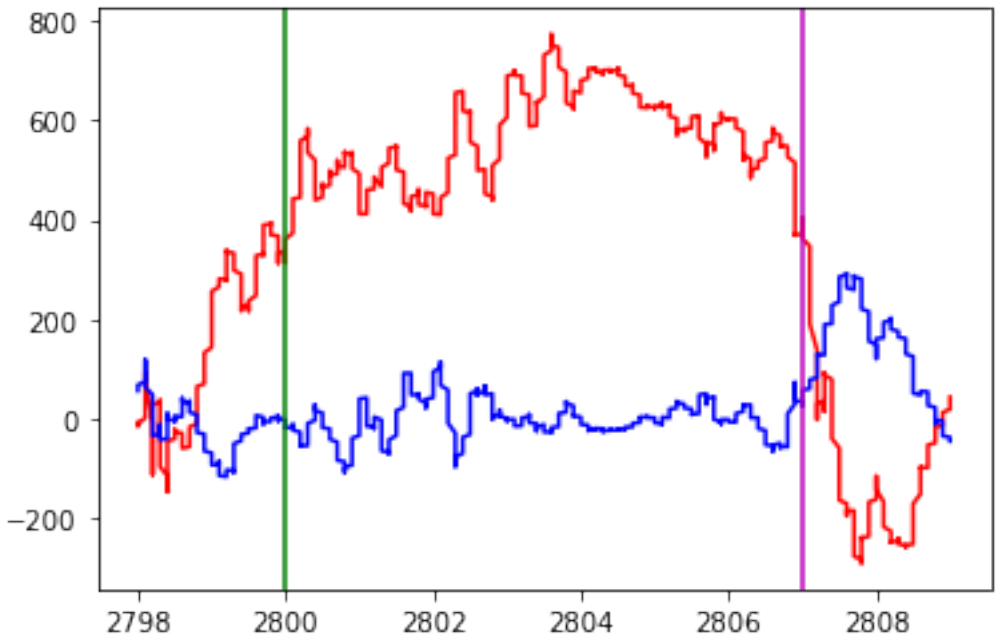

30
25
20
15
10
5
0
2670
2672
2674
2676
2678
2680
2682

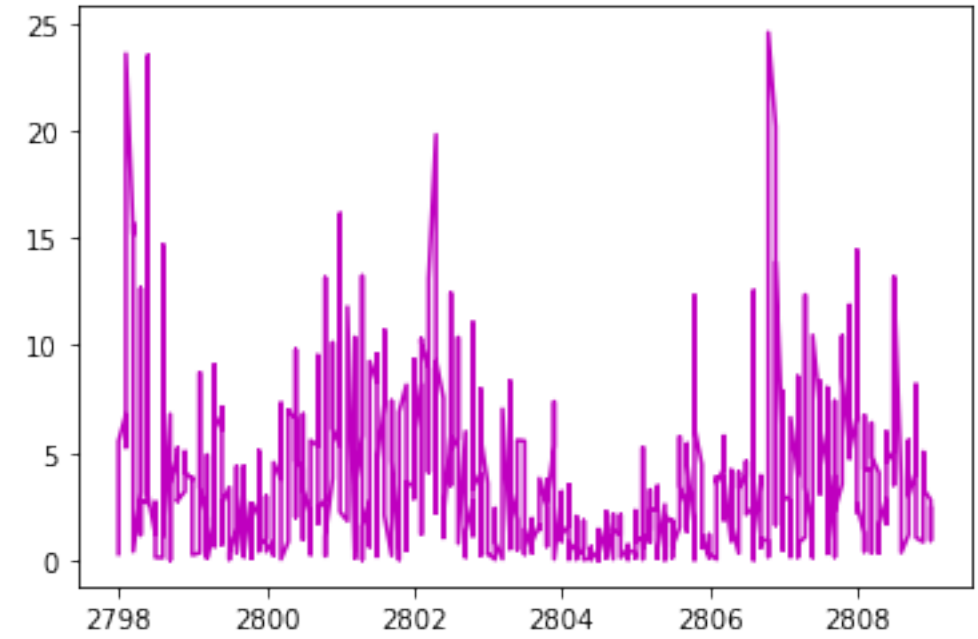

800
600
400
200
0
−200
2682
2684
2686
2688
2690

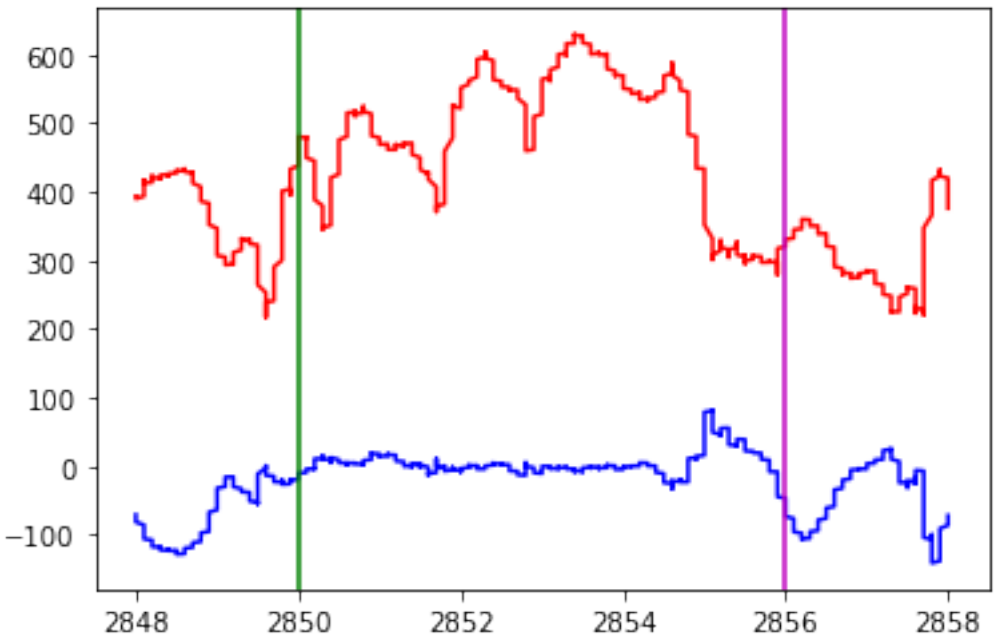

50
40
30
20
10
0
2682
2684
2686
2688
2690

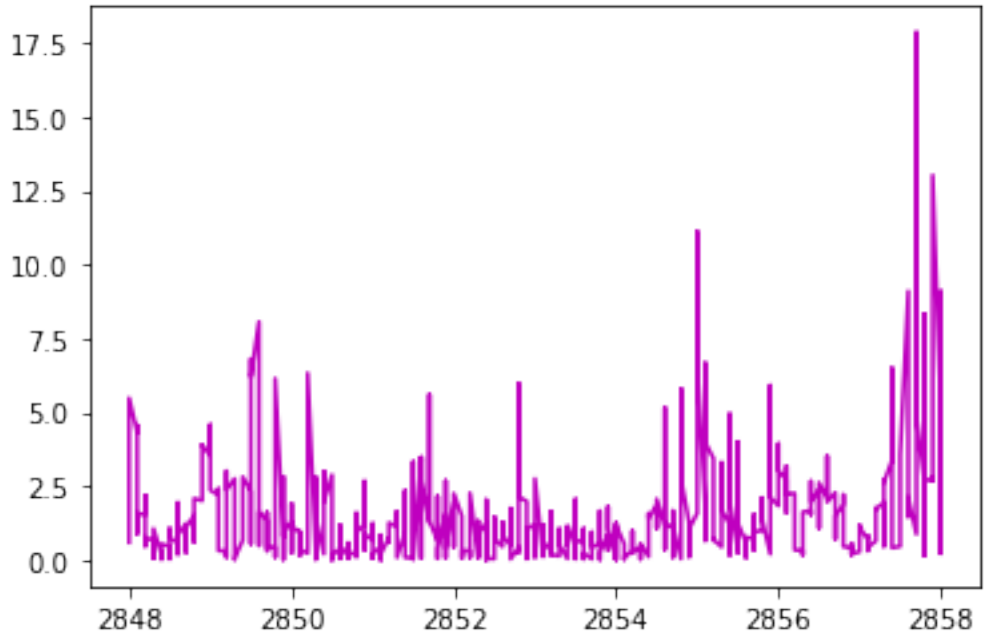

800
600
400
200
0
−200
2784
2786
2788
2790
2792
2794
2796

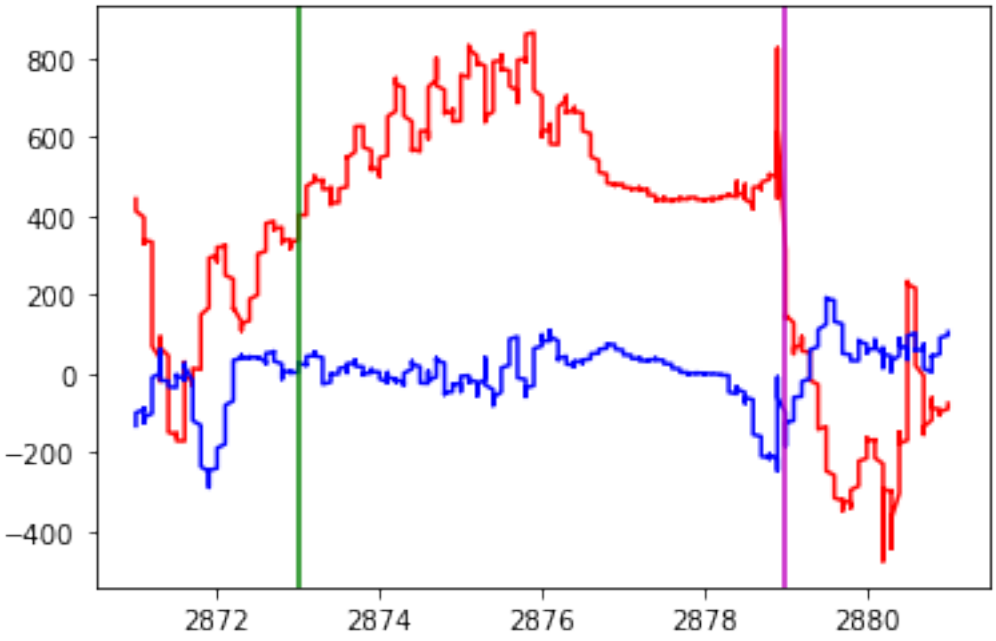

40
30
20
10
0
2784
2786
2788
2790
2792
2794
2796

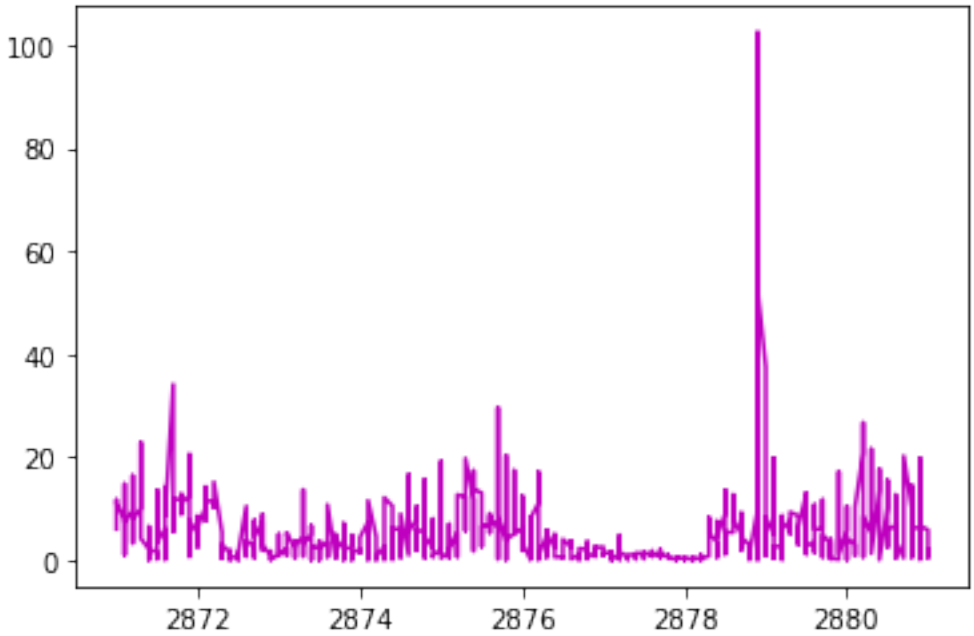

800
600
400
200
0
−200
2798
2800
2802
2804
2806
2808

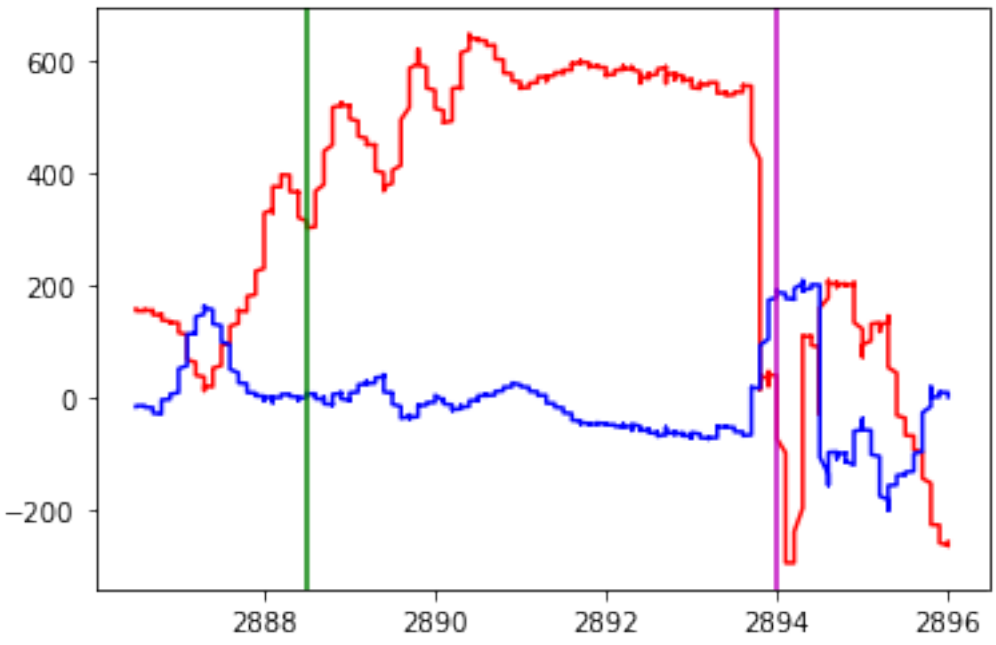

25
20
15
10
5
0
2798
2800
2802
2804
2806
2808

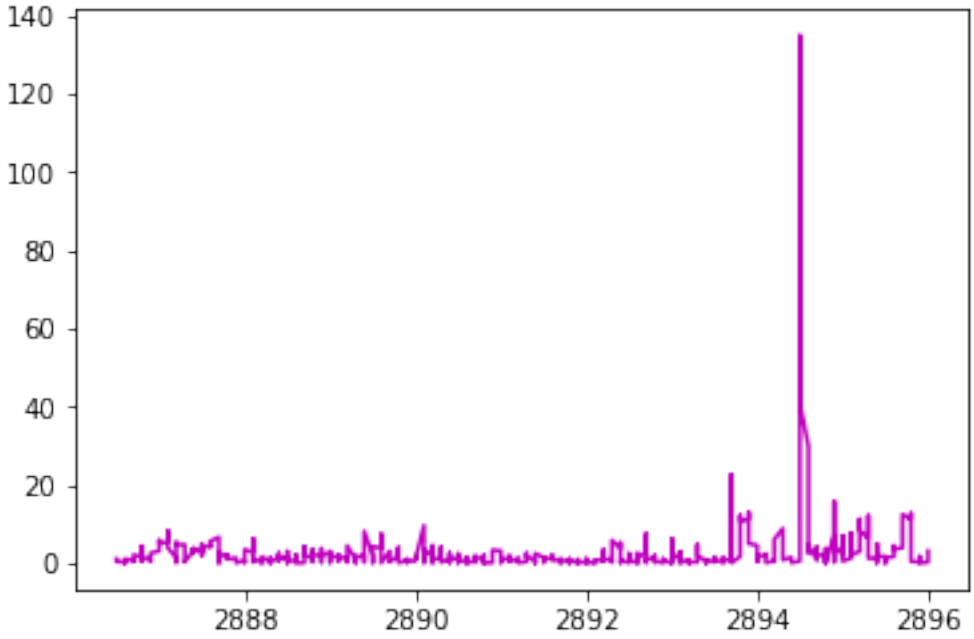

600
500
400
300
200
100
0
−100
2848
2850
2852
2854
2856
2858

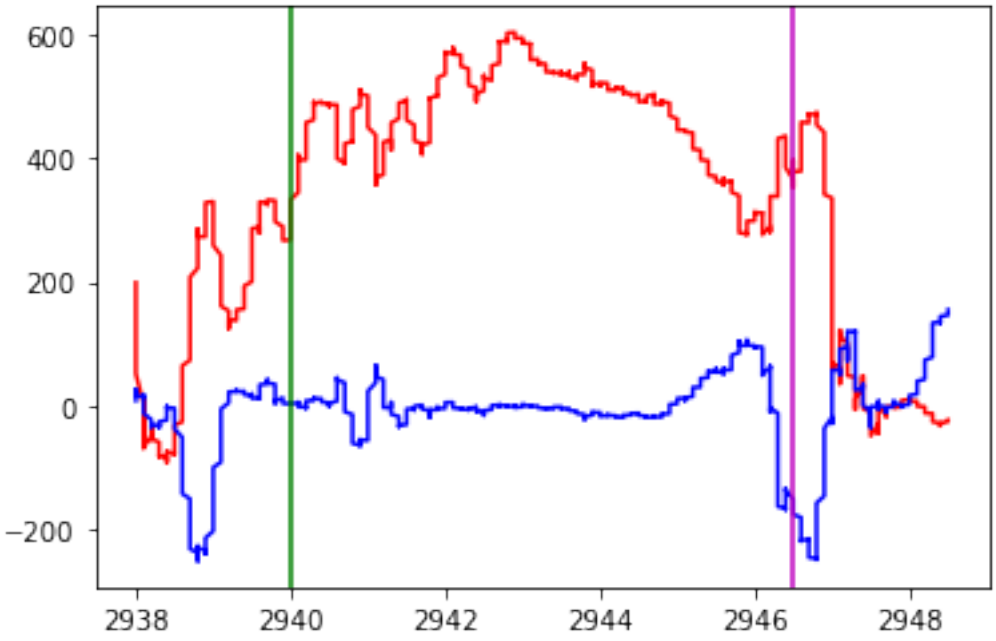

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2848
2850
2852
2854
2856
2858

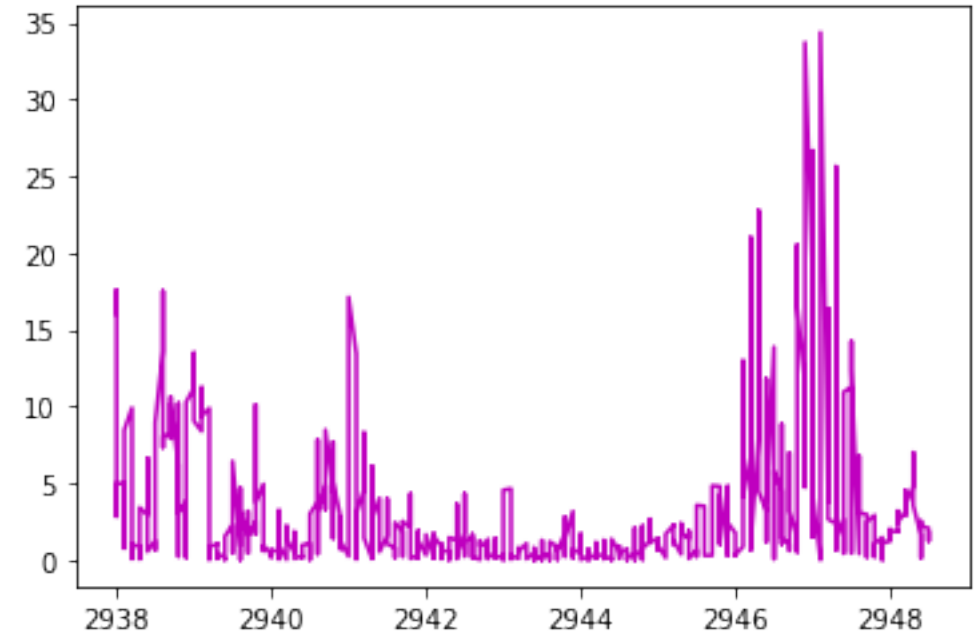

800
600
400
200
0
−200
−400
2872
2874
2876
2878
2880

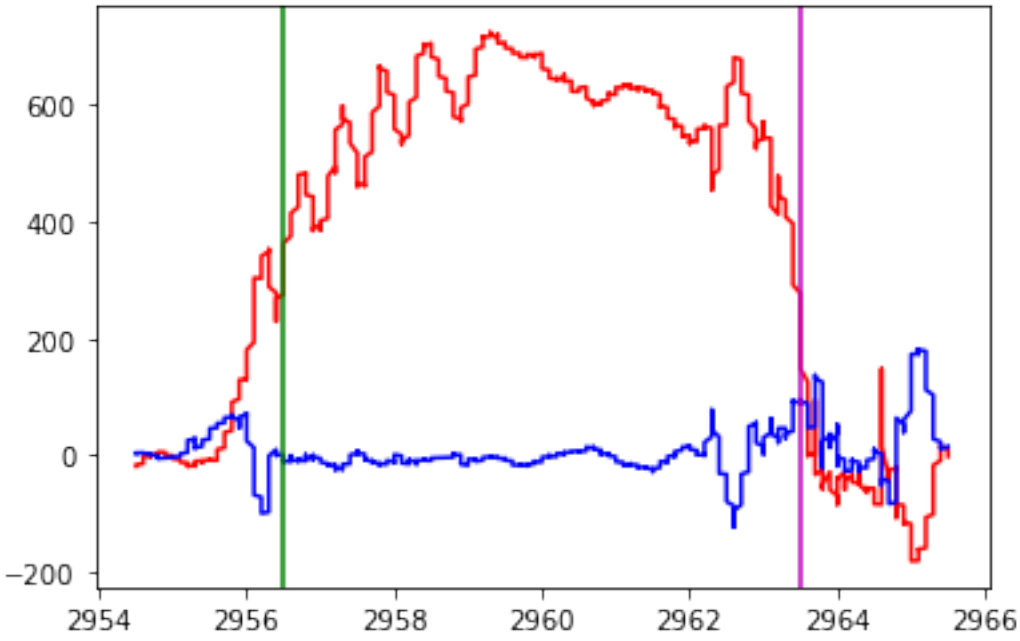

100
80
60
40
20
0
2872
2874
2876
2878
2880

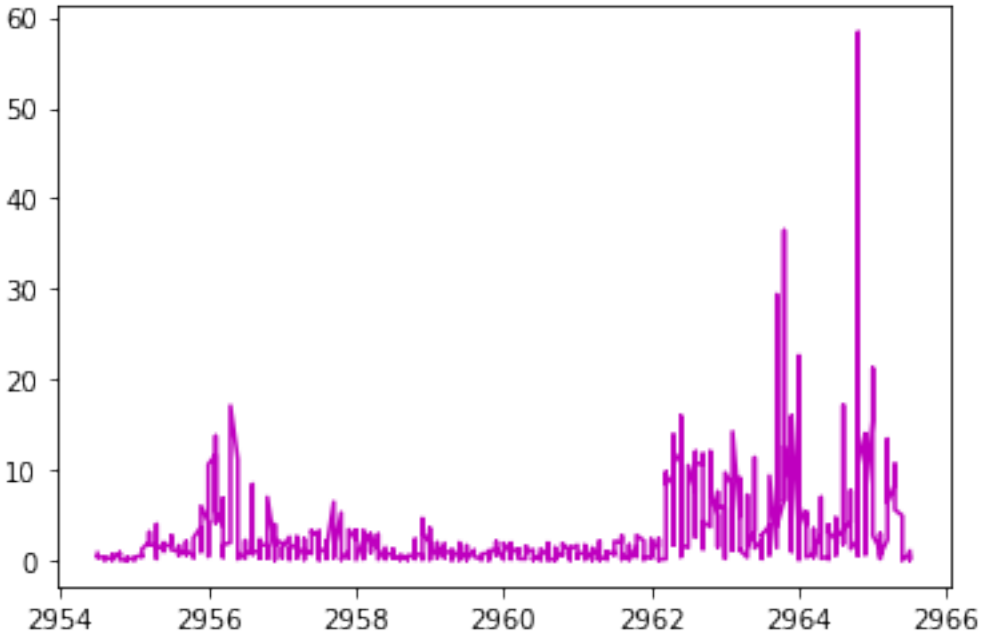

600
400
200
0
−200
2888
2890
2892
2894
2896

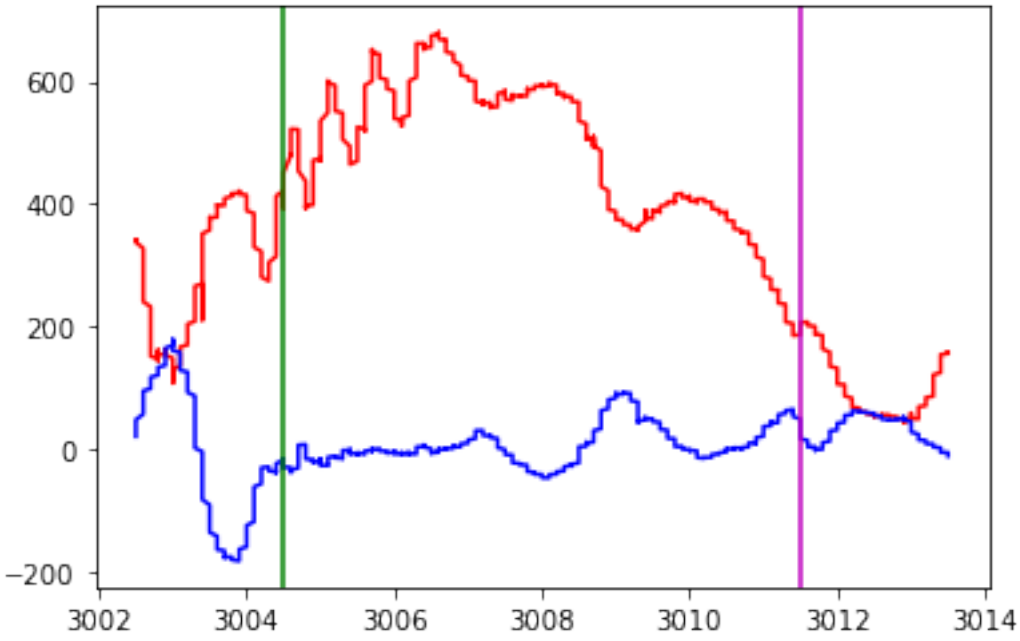

140
120
100
80
60
40
20
0
2888
2890
2892
2894
2896

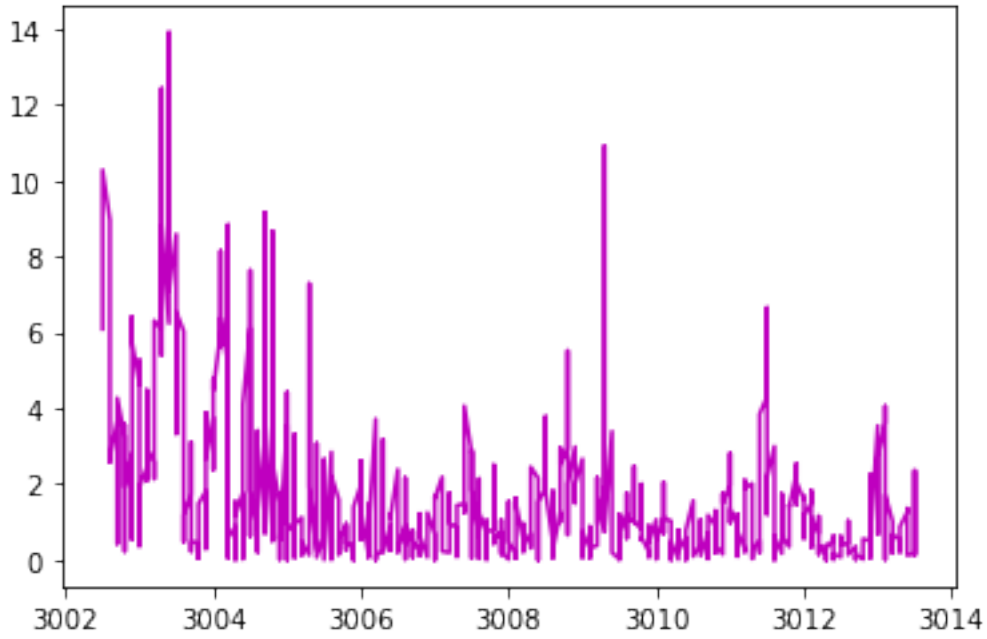

600
400
200
0
−200
2938
2940
2942
2944
2946
2948

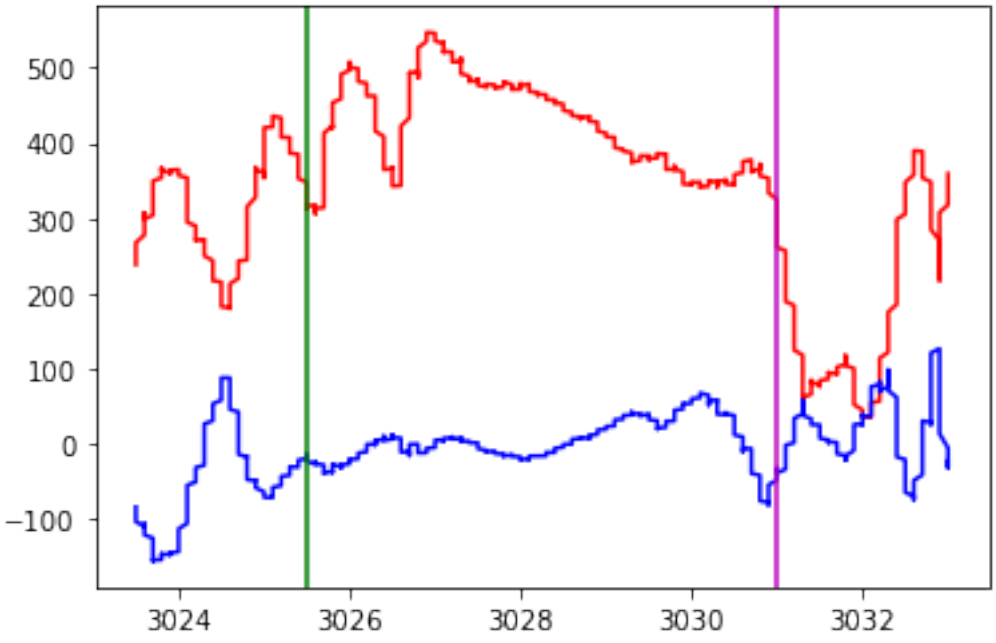

35
30
25
20
15
10
5
0
2938
2940
2942
2944
2946
2948

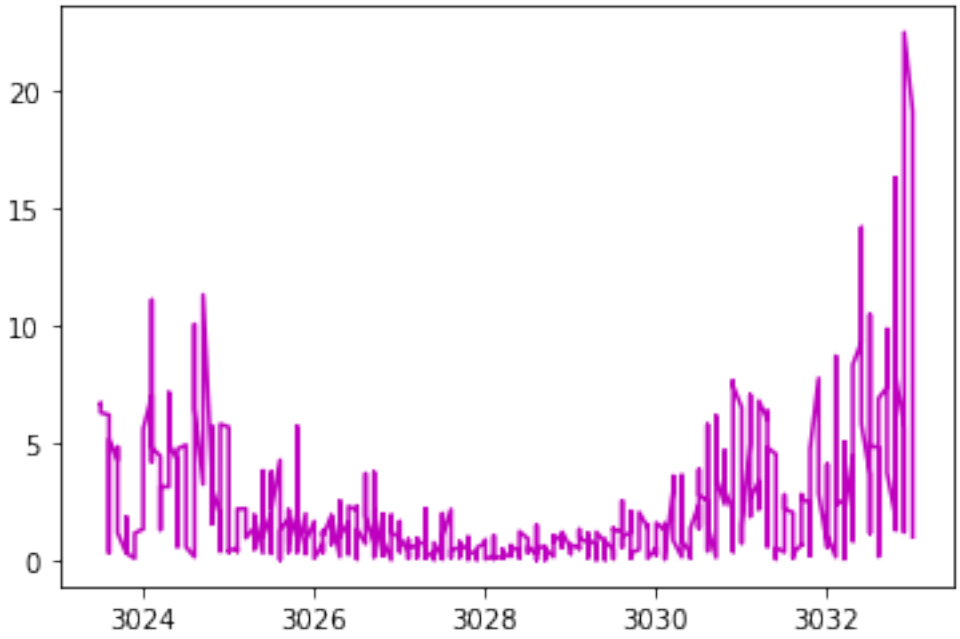

600
400
200
0
−200
2954
2956
2958
2960
2962
2964
2966

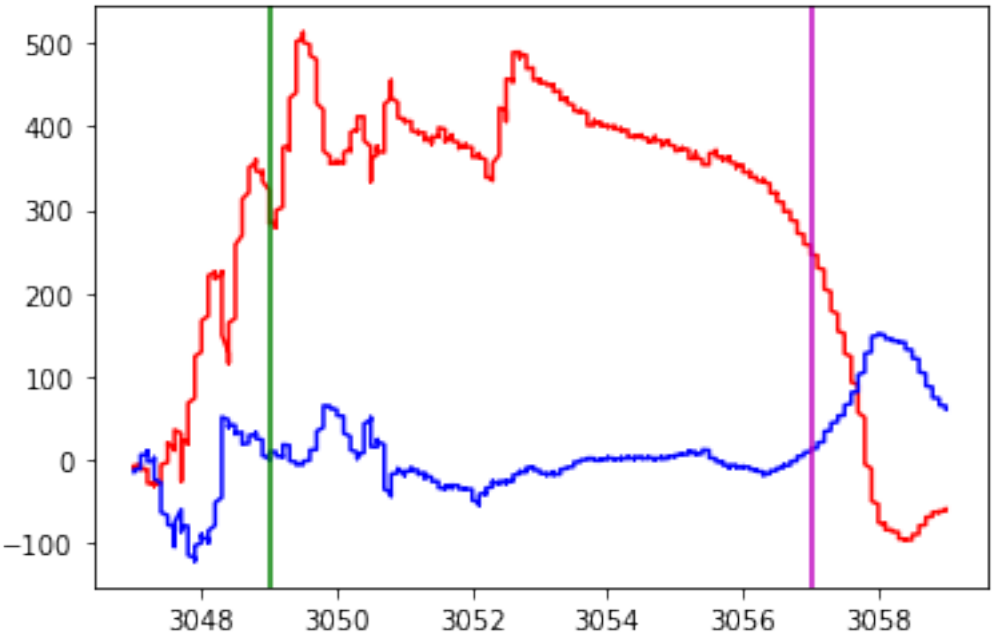

60
50
40
30
20
10
0
2954
2956
2958
2960
2962
2964
2966

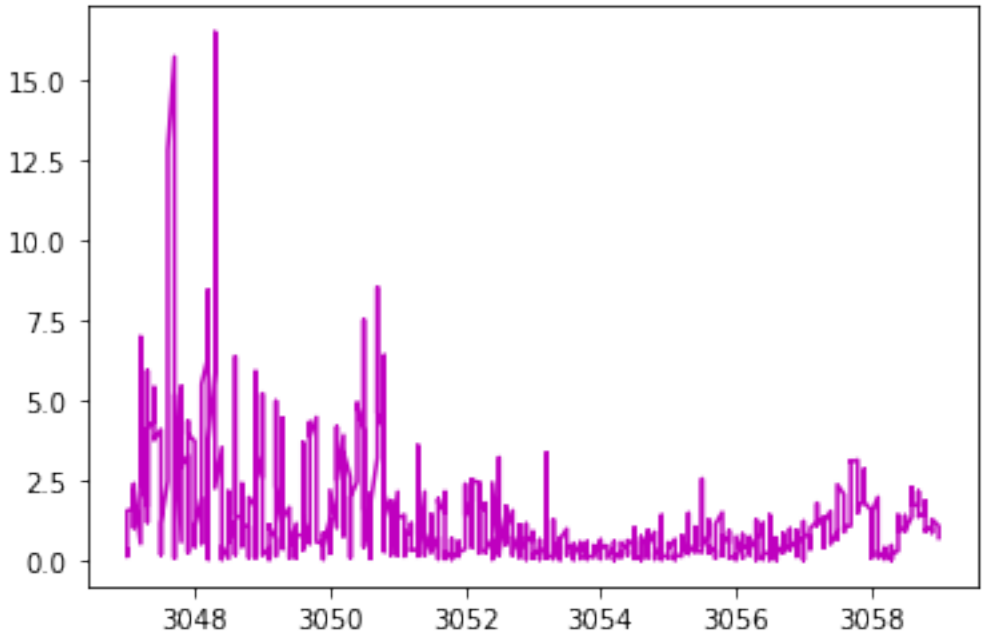

600
400
200
0
−200
3002
3004
3006
3008
3010
3012
3014

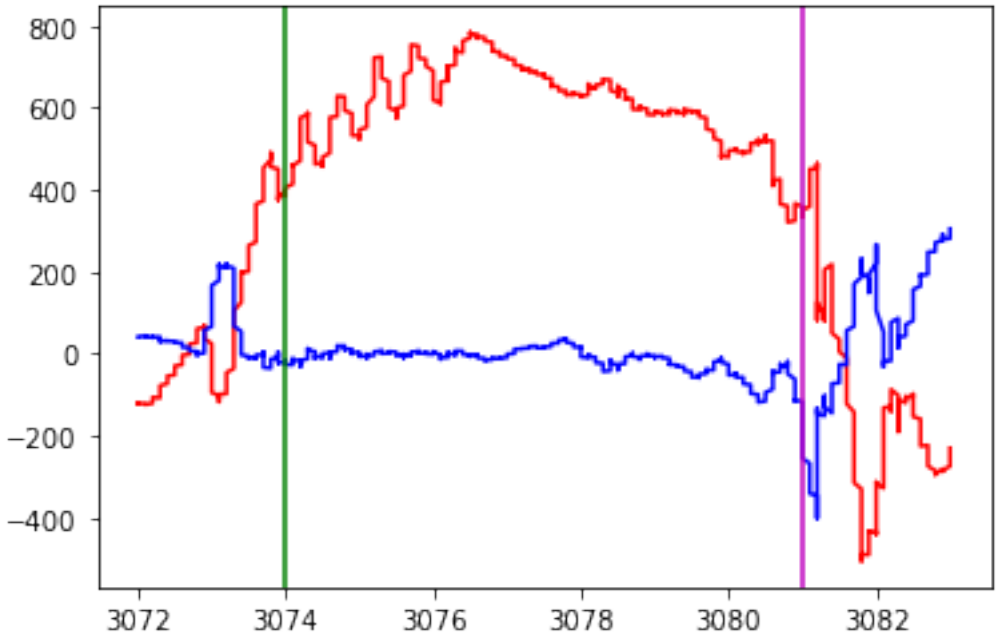

14
12
10
8
6
4
2
0
3002
3004
3006
3008
3010
3012
3014

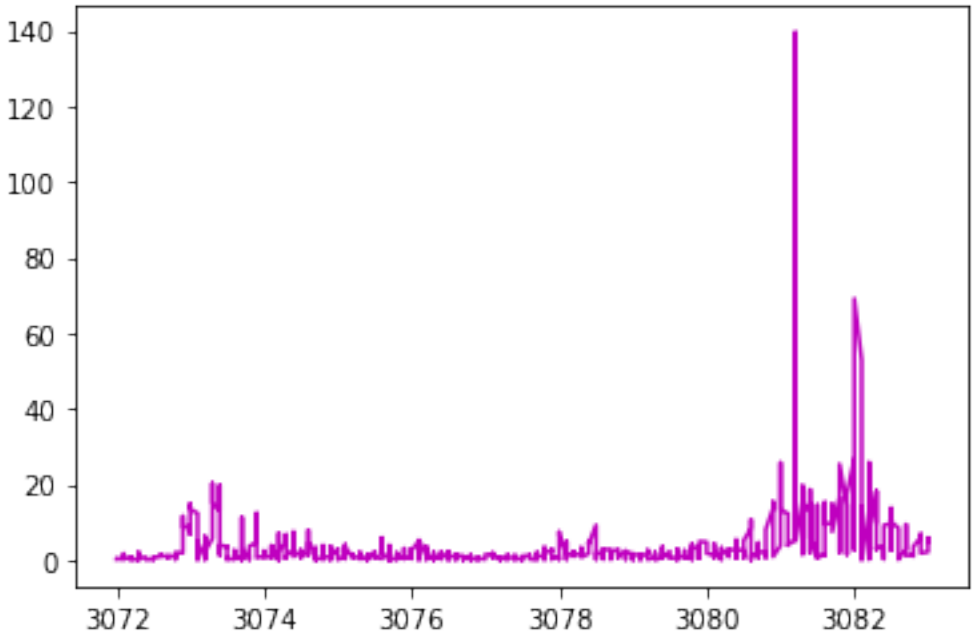

500
400
300
200
100
0
−100
3024
3026
3028
3030
3032

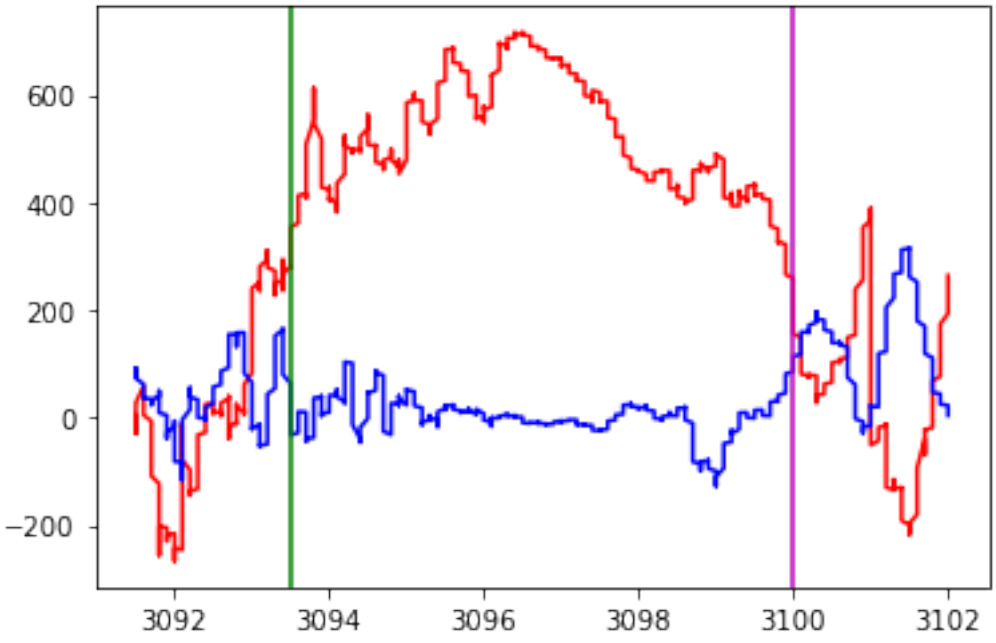

20
15
10
5
0
3024
3026
3028
3030
3032

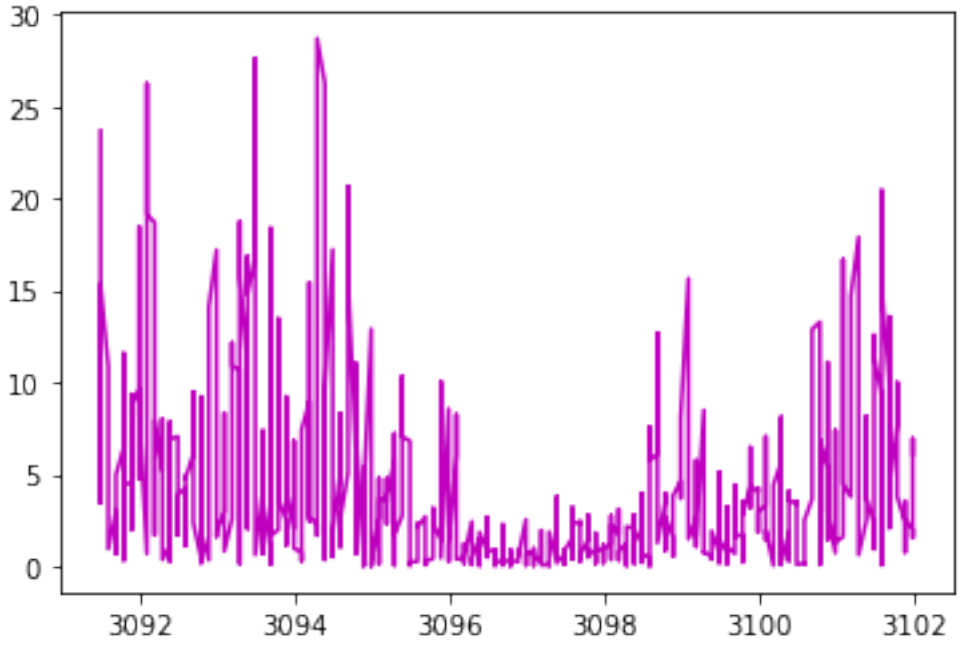

500
400
300
200
100
0
−100
3048
3050
3052
3054
3056
3058

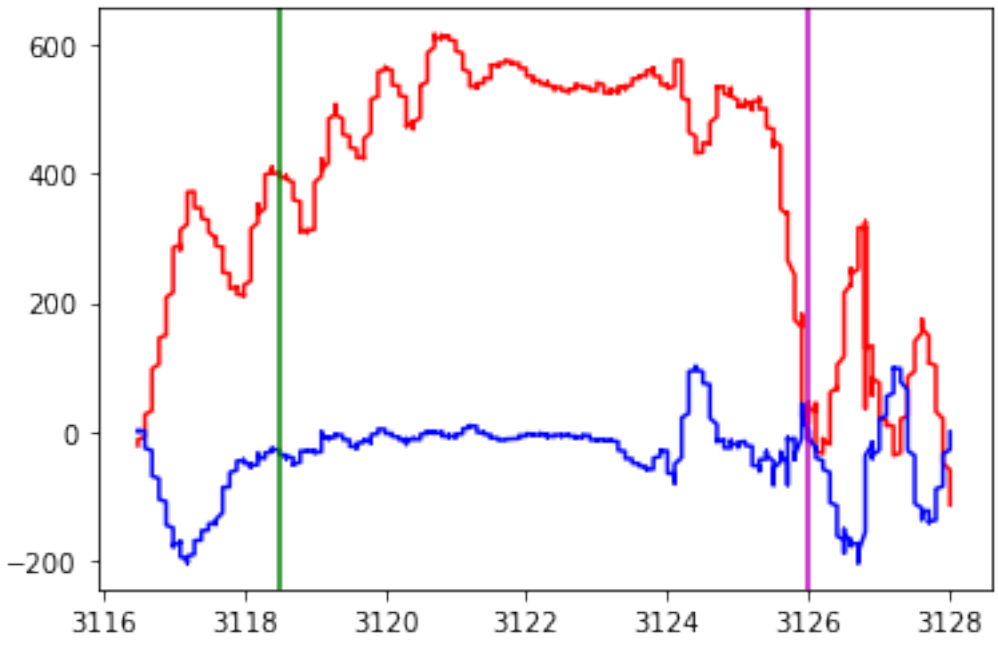

15.0
12.5
10.0
7.5
5.0
2.5
0.0
3048
3050
3052
3054
3056
3058

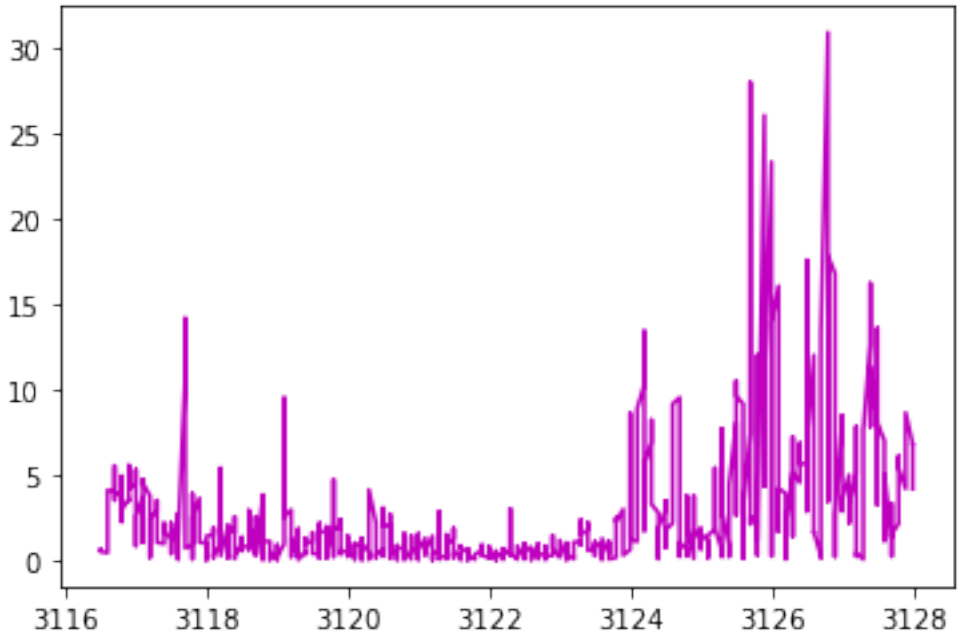

800
600
400
200
0
−200
−400
3072
3074
3076
3078
3080
3082

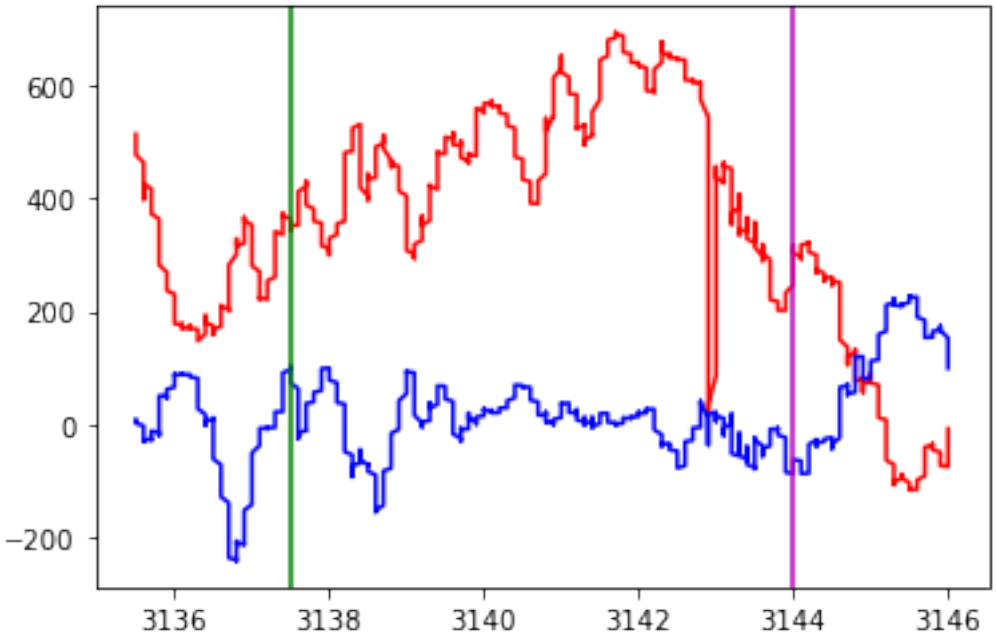

140
120
100
80
60
40
20
0
3072
3074
3076
3078
3080
3082

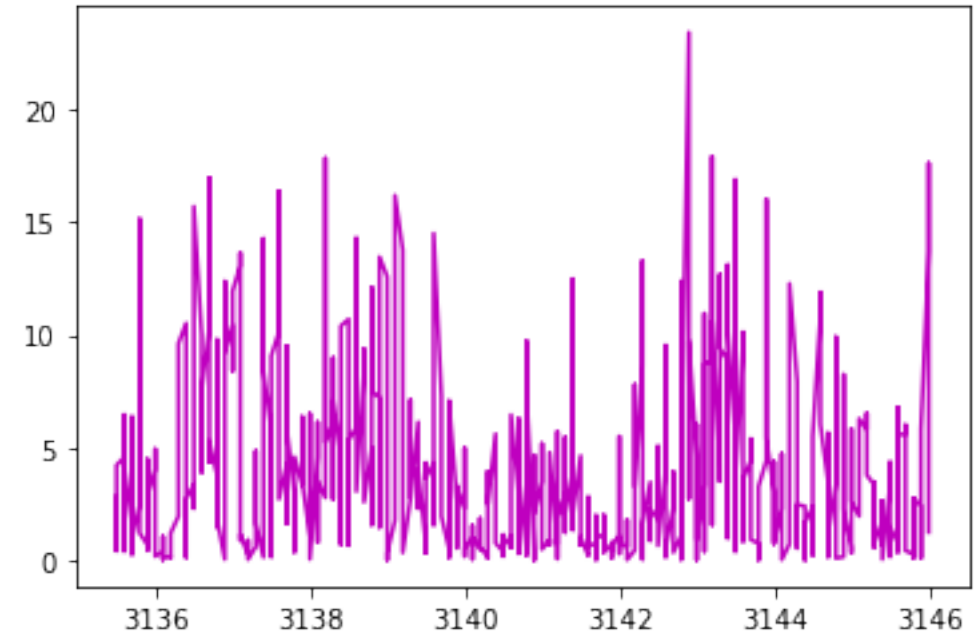

600
400
200
0
−200
3092
3094
3096
3098
3100
3102

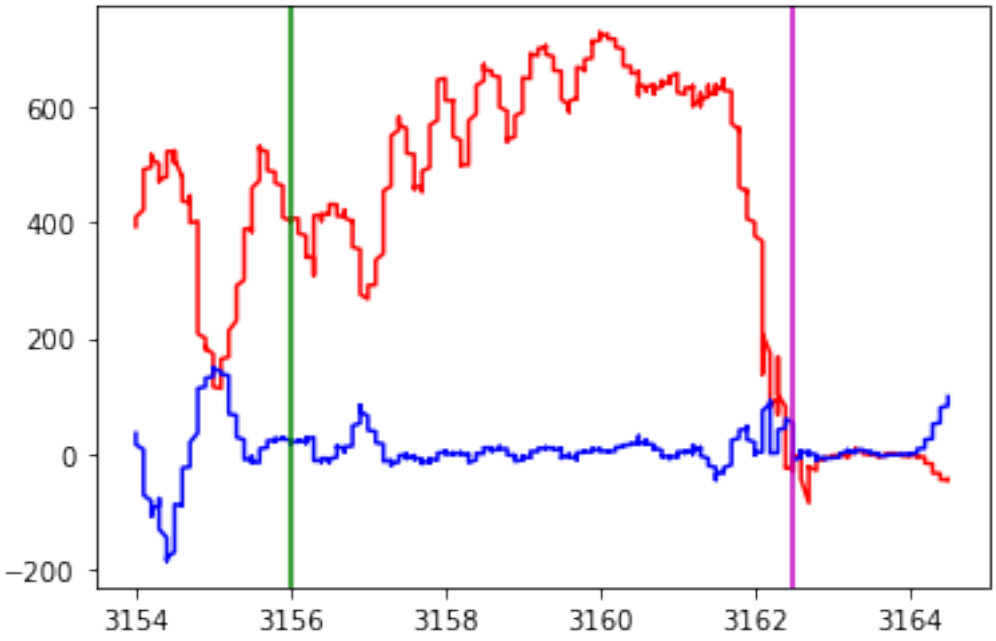

30
25
20
15
10
5
0
3092
3094
3096
3098
3100
3102

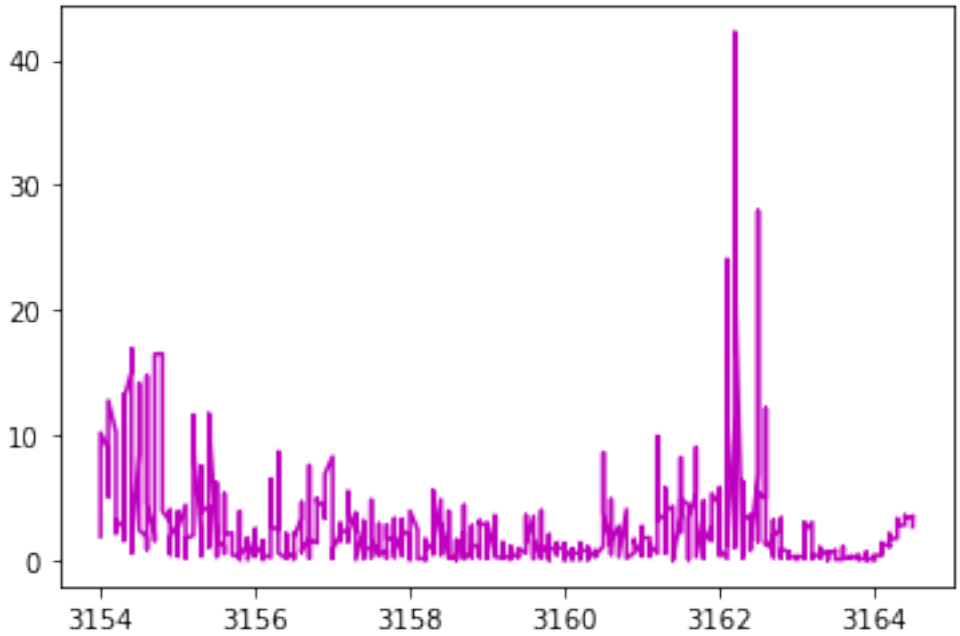

600
400
200
0
−200
3116
3118
3120
3122
3124
3126
3128

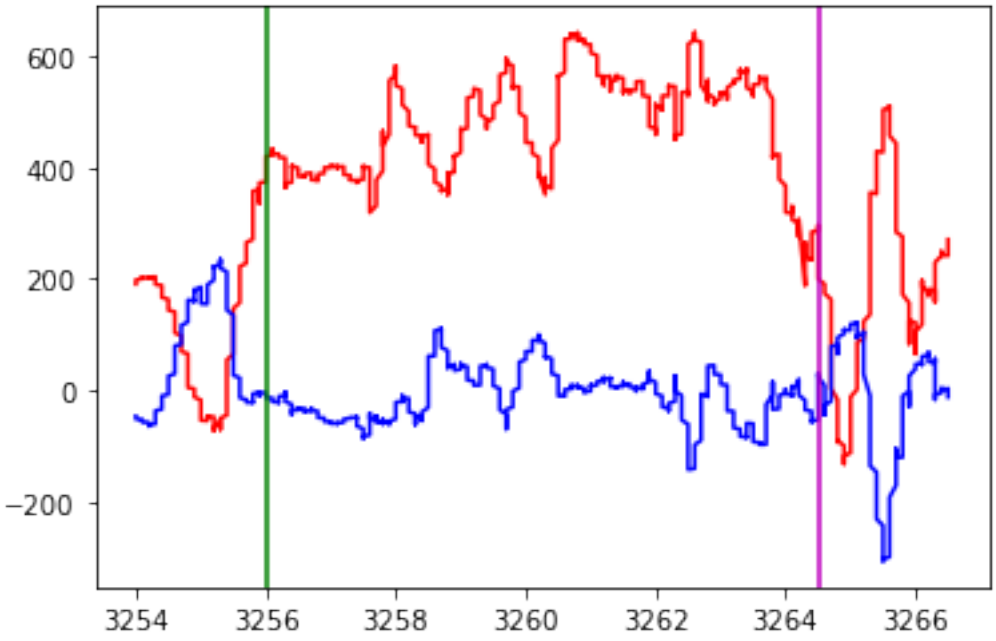

30
25
20
15
10
5
0
3116
3118
3120
3122
3124
3126
3128

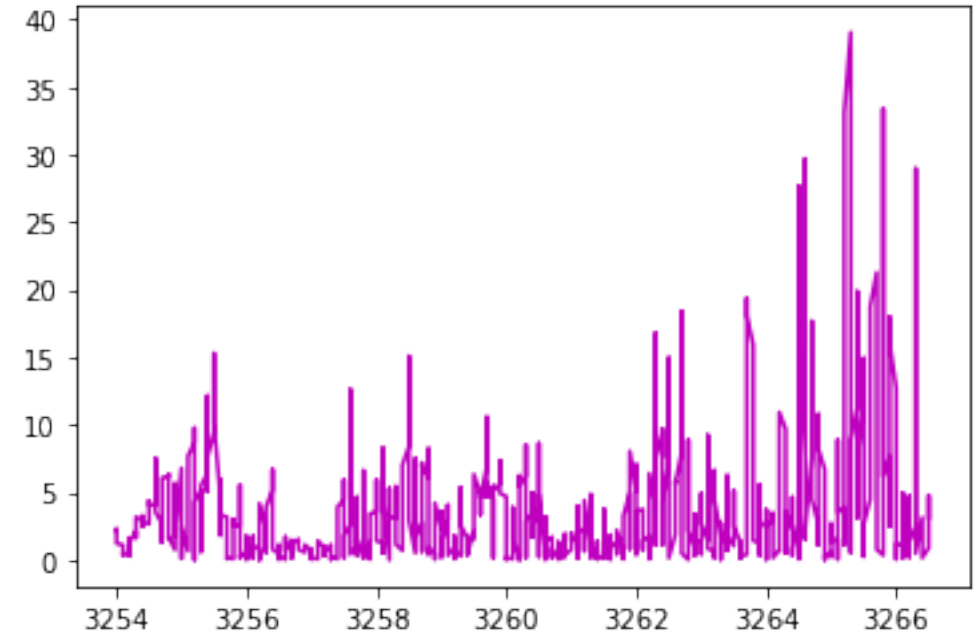

600
400
200
0
−200
3136
3138
3140
3142
3144
3146

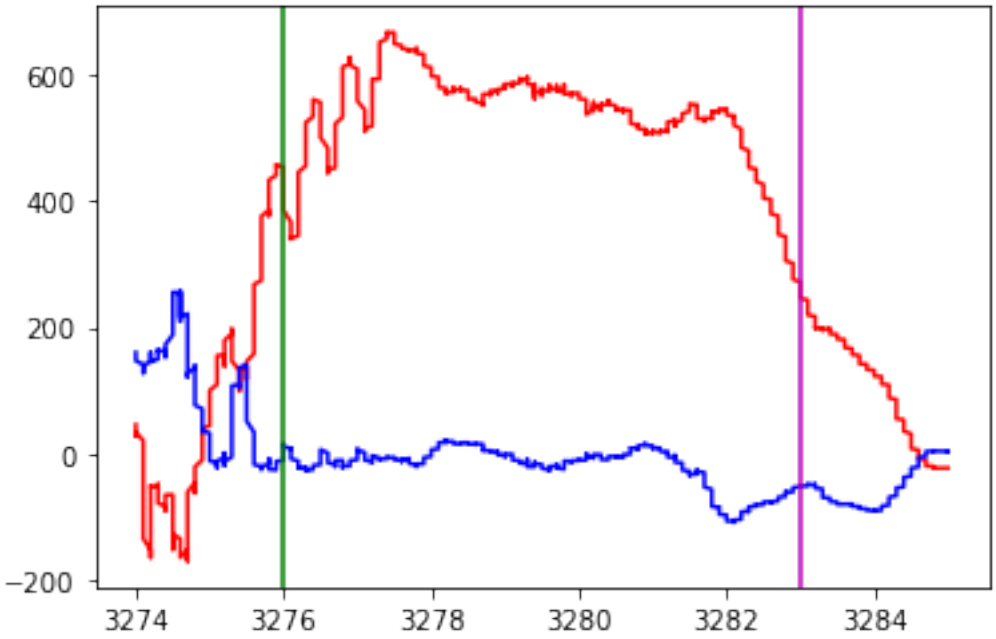

20
15
10
5
0
3136
3138
3140
3142
3144
3146

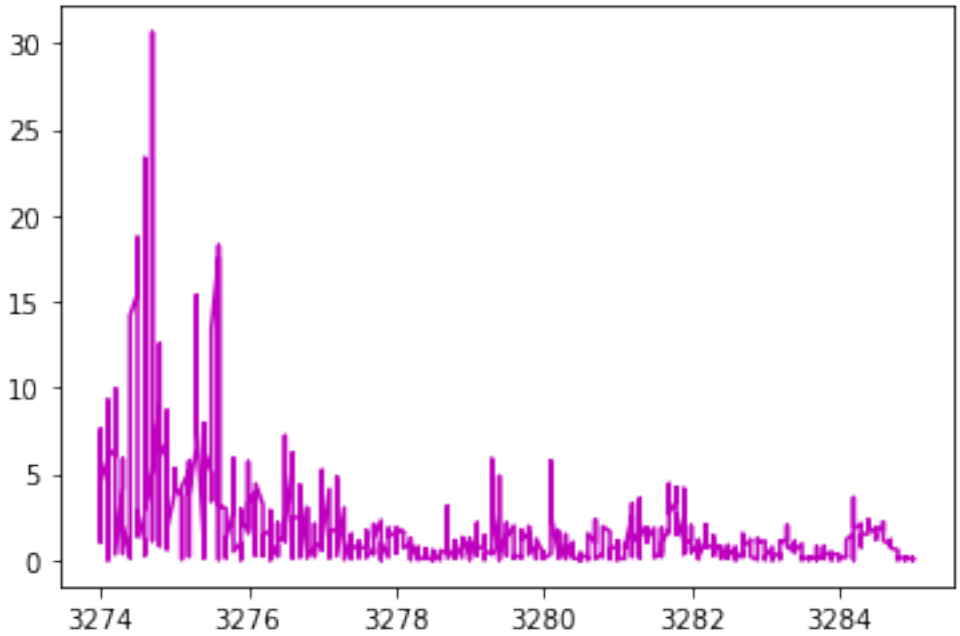

600
400
200
0
−200
3154
3156
3158
3160
3162
3164

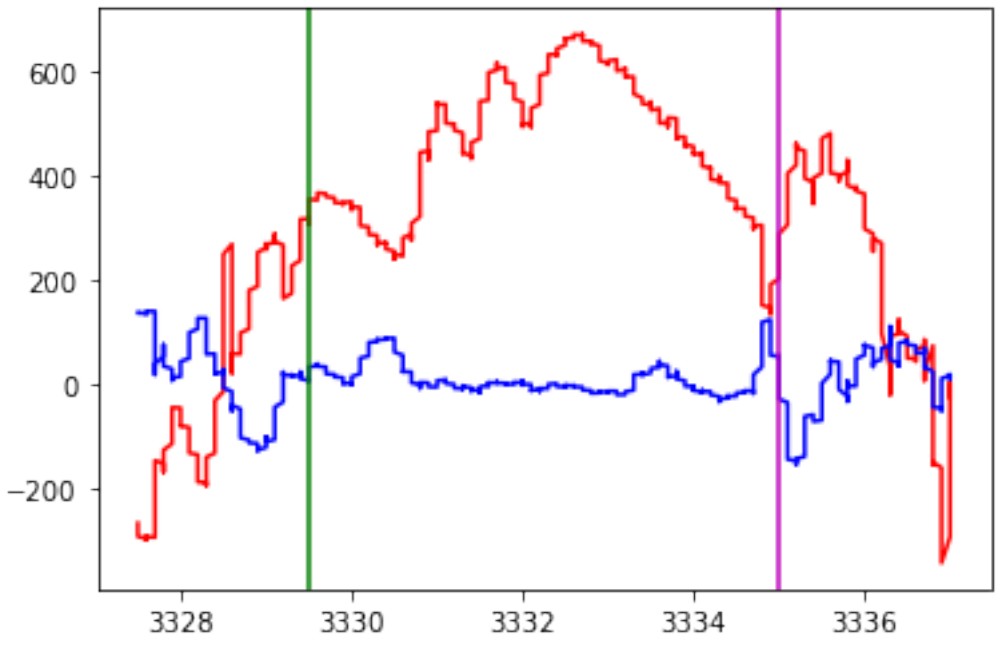

40
30
20
10
0
3154
3156
3158
3160
3162
3164

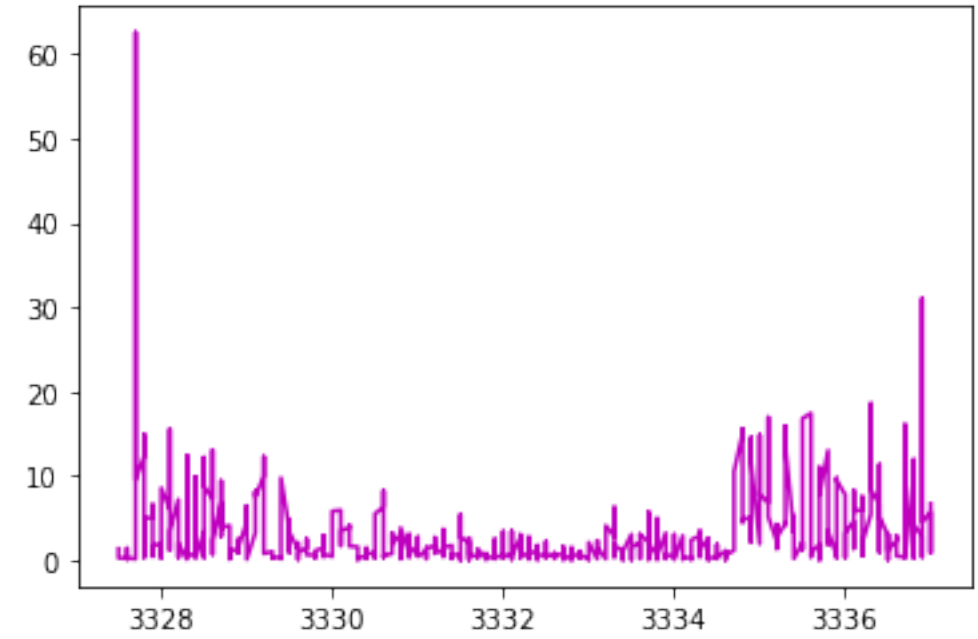

600
400
200
0
−200
3254
3256
3258
3260
3262
3264
3266

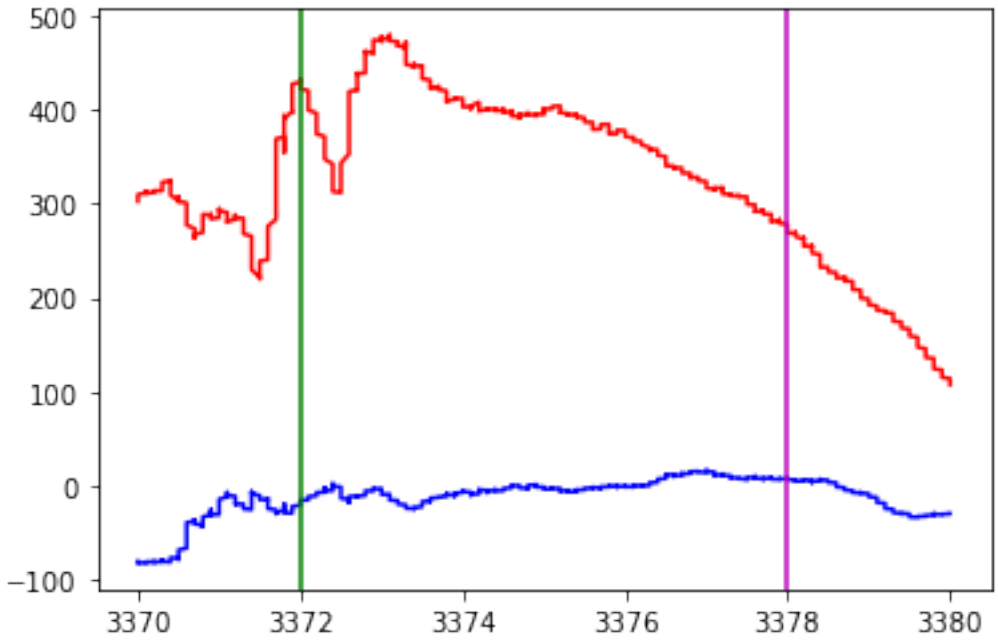

40
35
30
25
20
15
10
5
0
3254
3256
3258
3260
3262
3264
3266

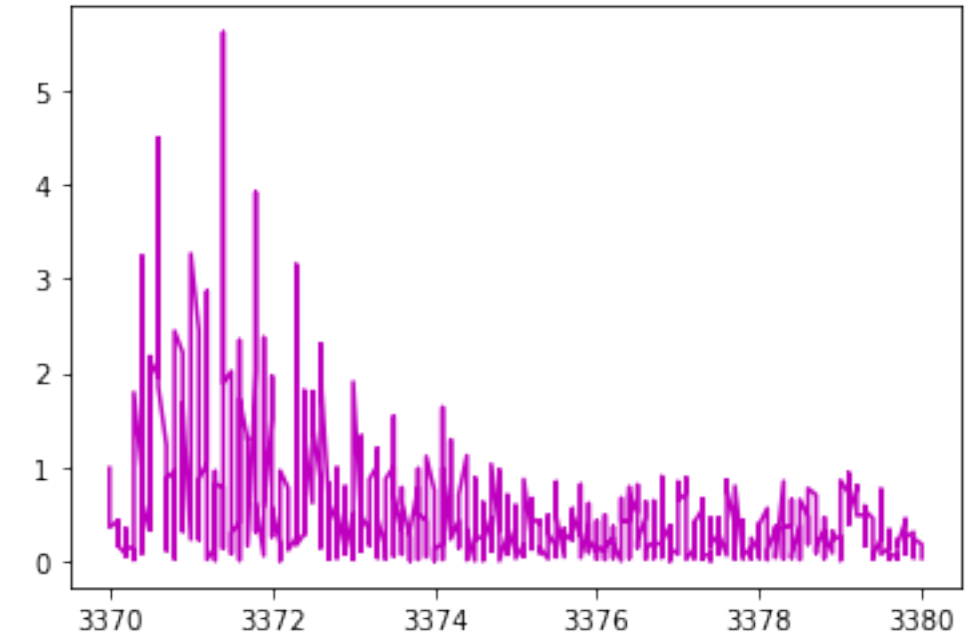

600
400
200
0
−200
3274
3276
3278
3280
3282
3284

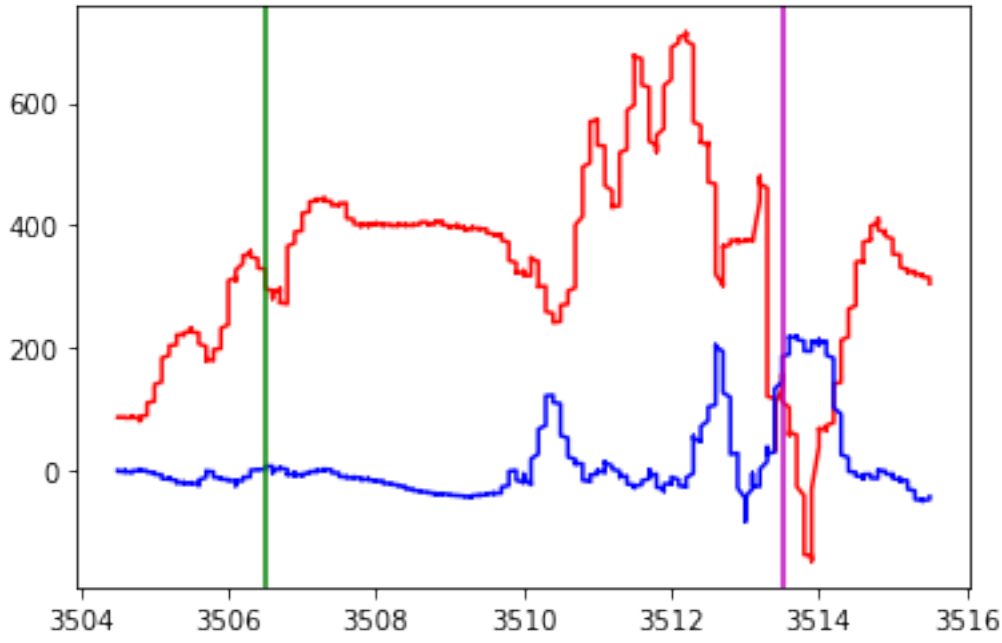

30
25
20
15
10
5
0
3274
3276
3278
3280
3282
3284

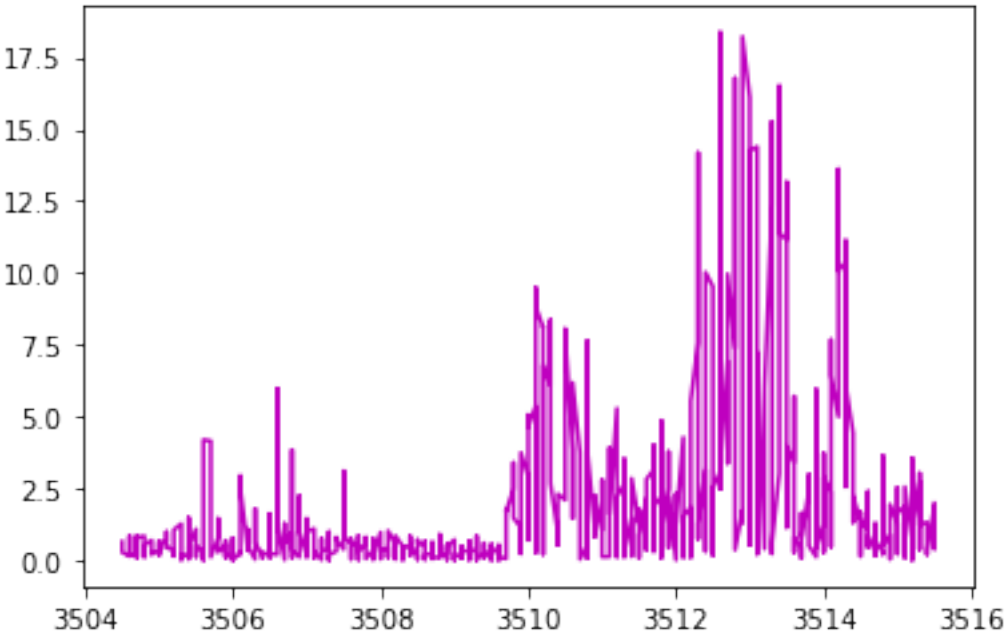

600
400
200
0
−200
3328
3330
3332
3334
3336

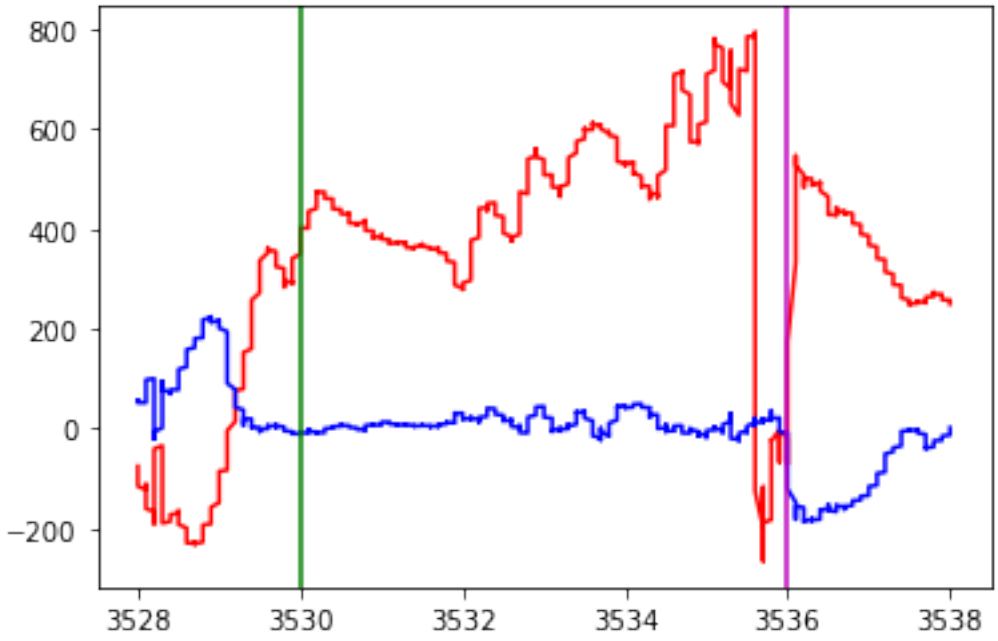

60
50
40
30
20
10
0
3328
3330
3332
3334
3336

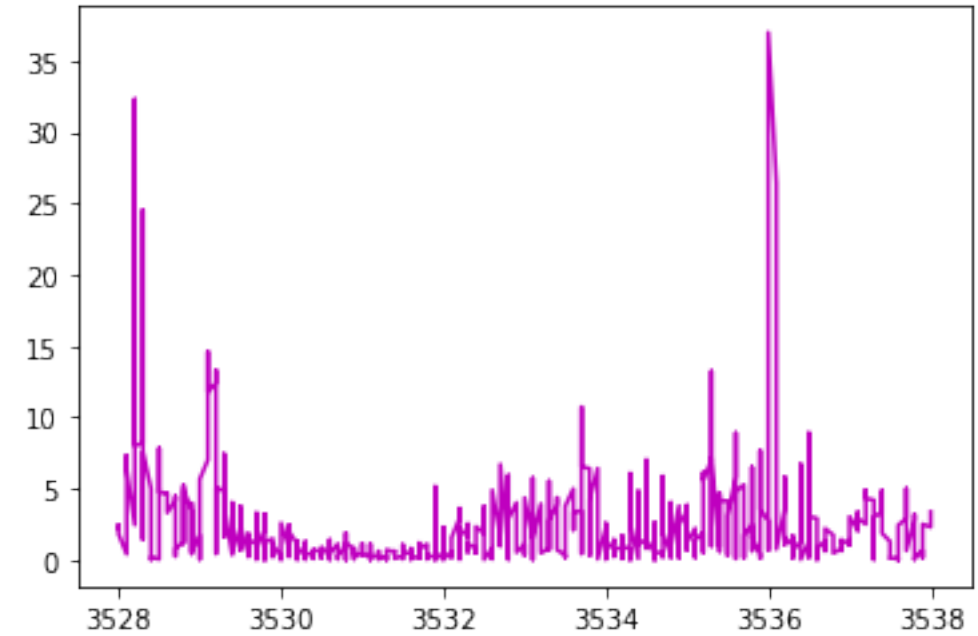

500
400
300
200
100
0
−100
3370
3372
3374
3376
3378
3380

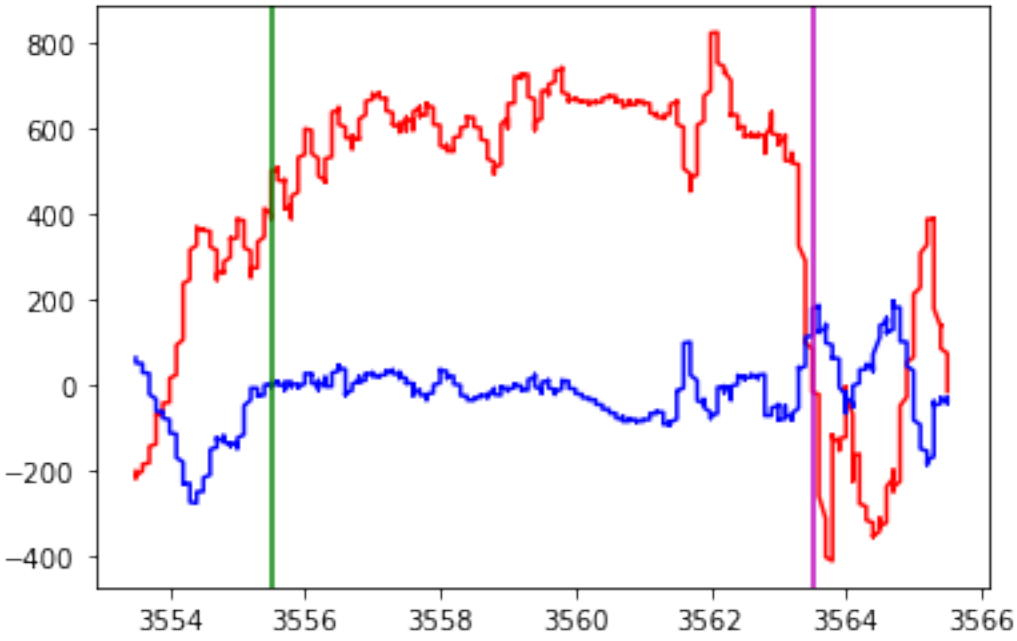

5
4
3
2
1
0
3370
3372
3374
3376
3378
3380

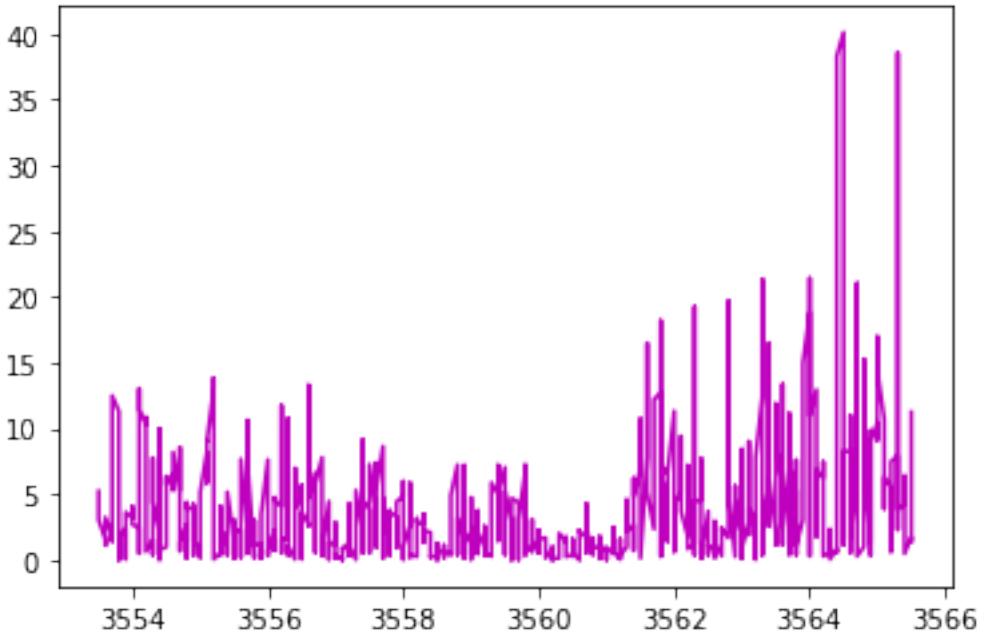

600
400
200
0
3504
3506
3508
3510
3512
3514
3516

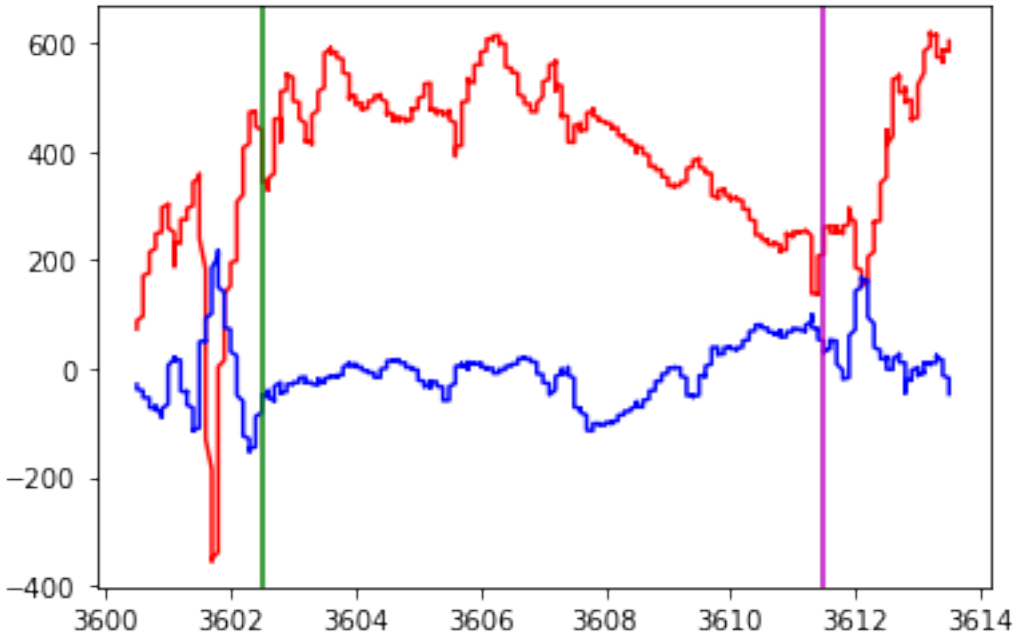

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
3504
3506
3508
3510
3512
3514
3516

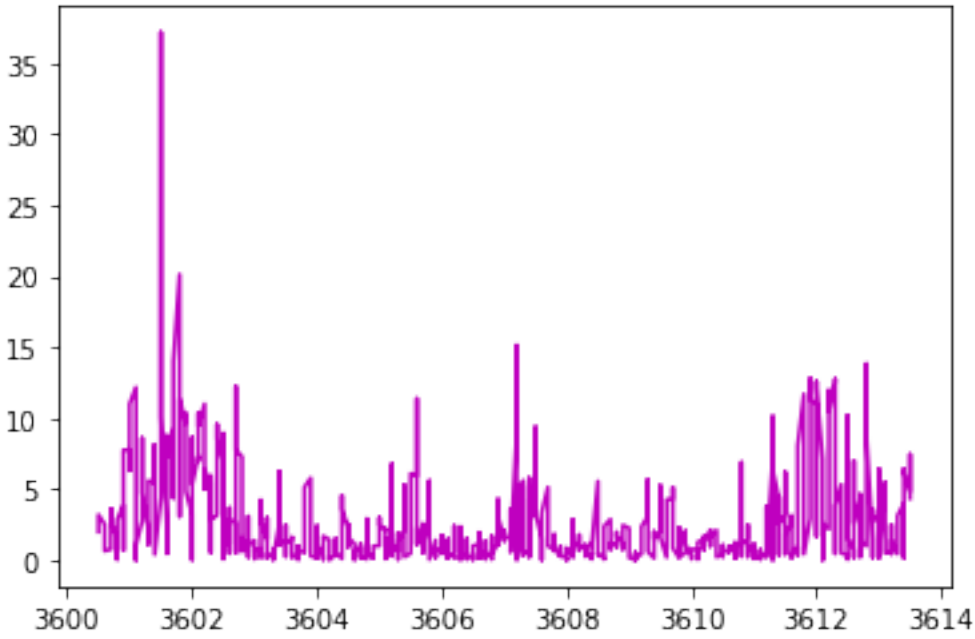

800
600
400
200
0
−200
3528
3530
3532
3534
3536
3538

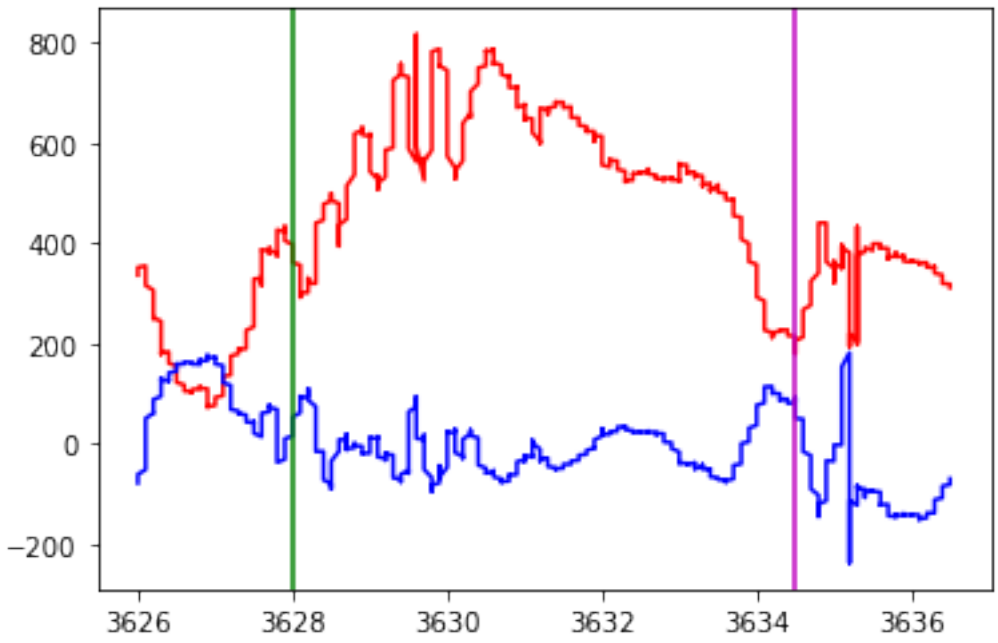

35
30
25
20
15
10
5
0
3528
3530
3532
3534
3536
3538

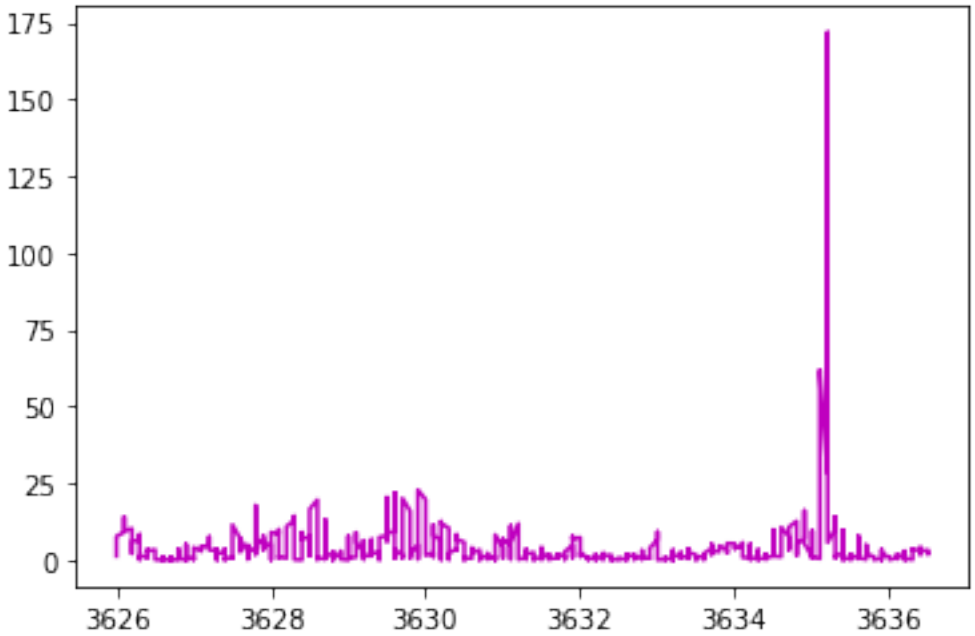

800
600
400
200
0
−200
−400
3554
3556
3558
3560
3562
3564
3566

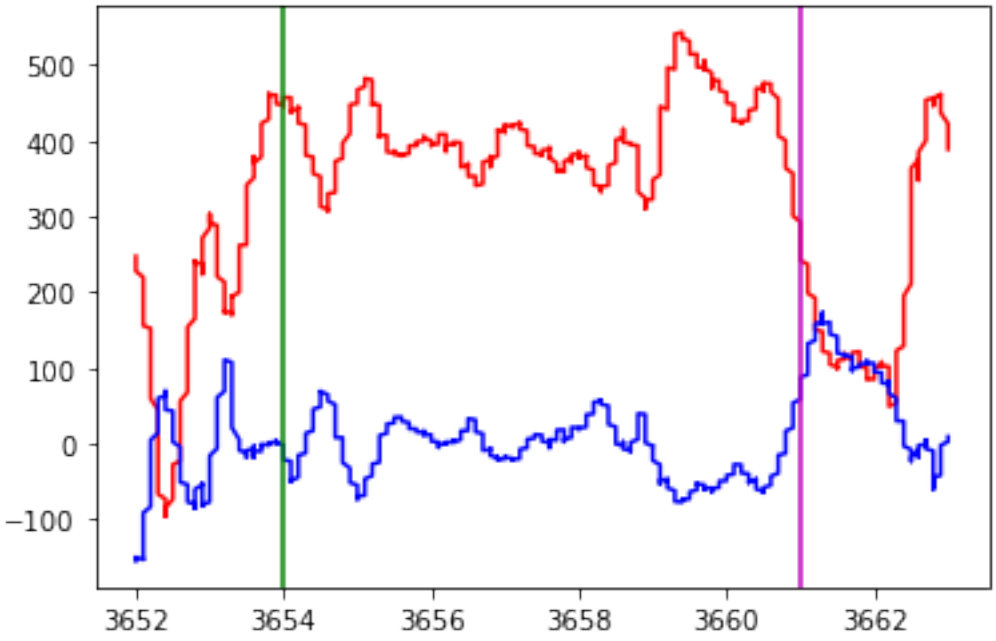

40
35
30
25
20
15
10
5
0
3554
3556
3558
3560
3562
3564
3566

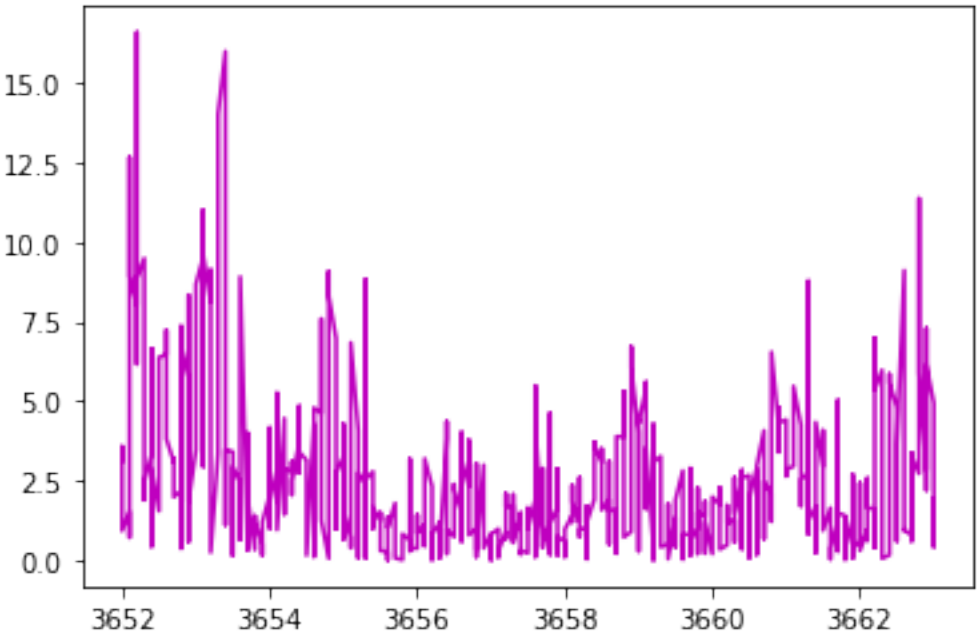

600
400
200
0
−200
−400
3600
3602
3604
3606
3608
3610
3612
3614

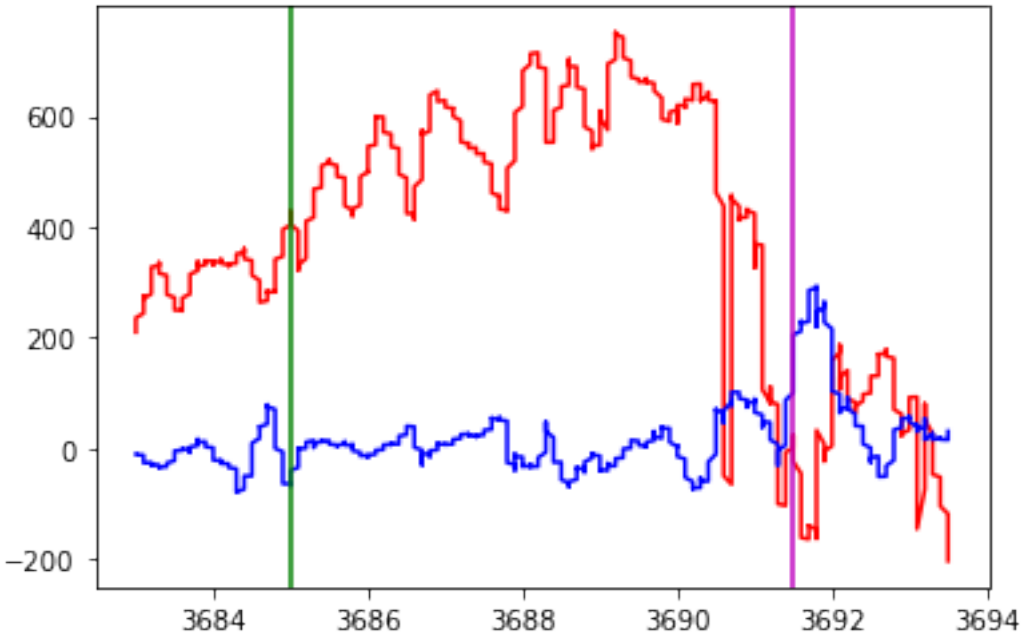

35
30
25
20
15
10
5
0
3600
3602
3604
3606
3608
3610
3612
3614

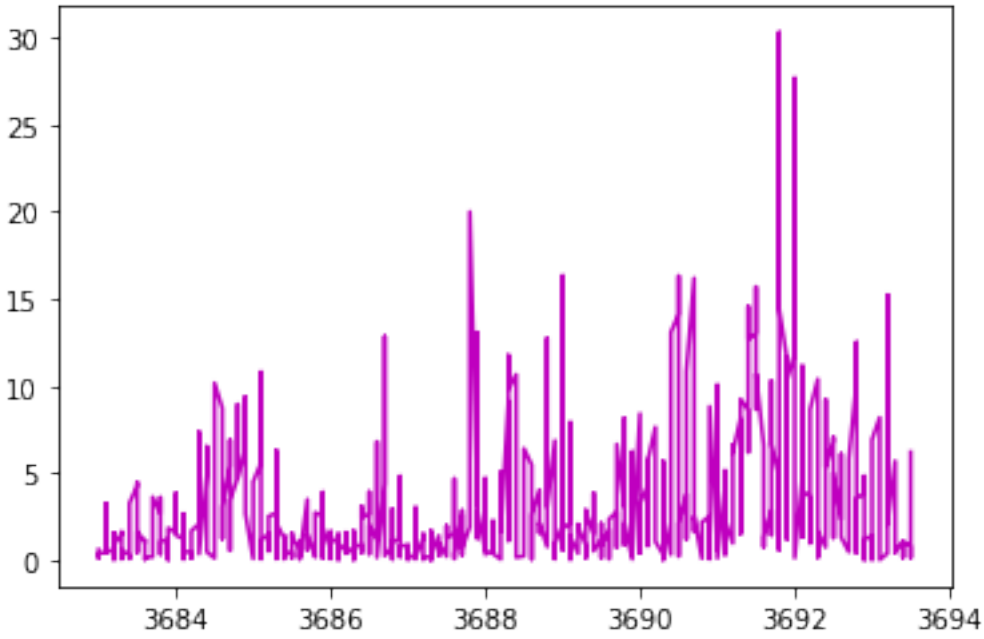

800
600
400
200
0
−200
3626
3628
3630
3632
3634
3636

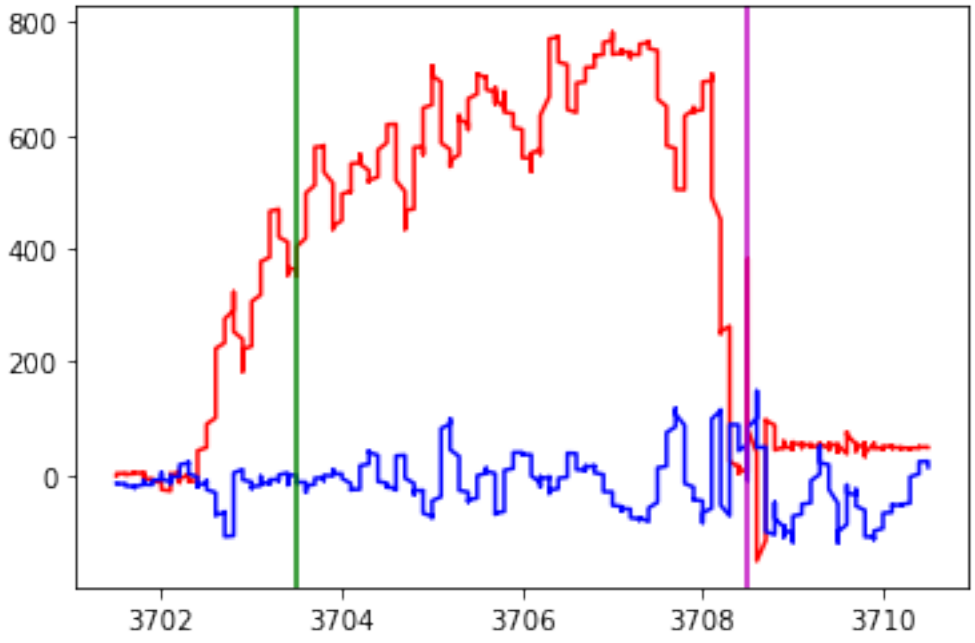

175
150
125
100
75
50
25
0
3626
3628
3630
3632
3634
3636

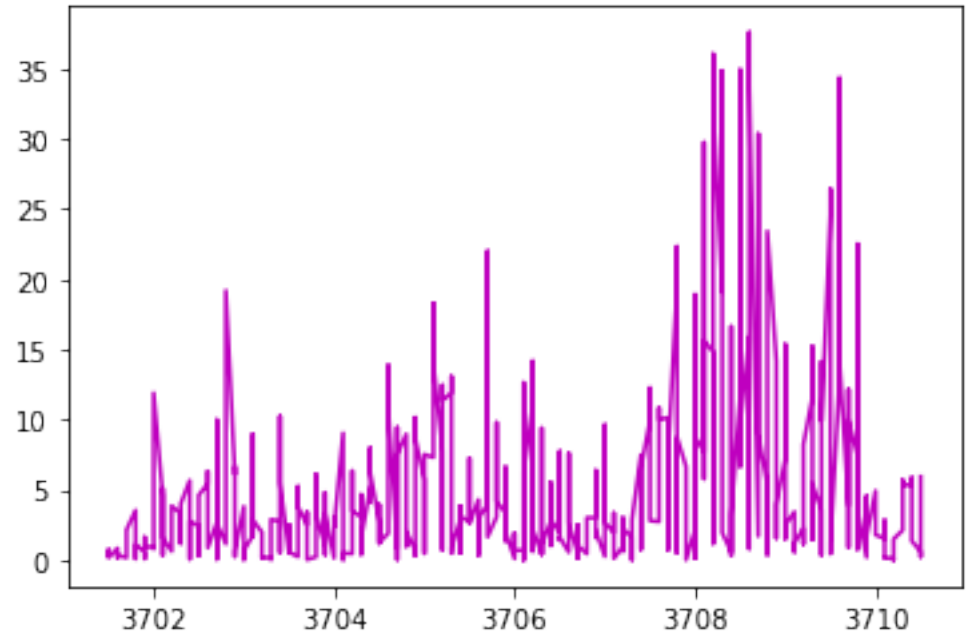

500
400
300
200
100
0
−100
3652
3654
3656
3658
3660
3662

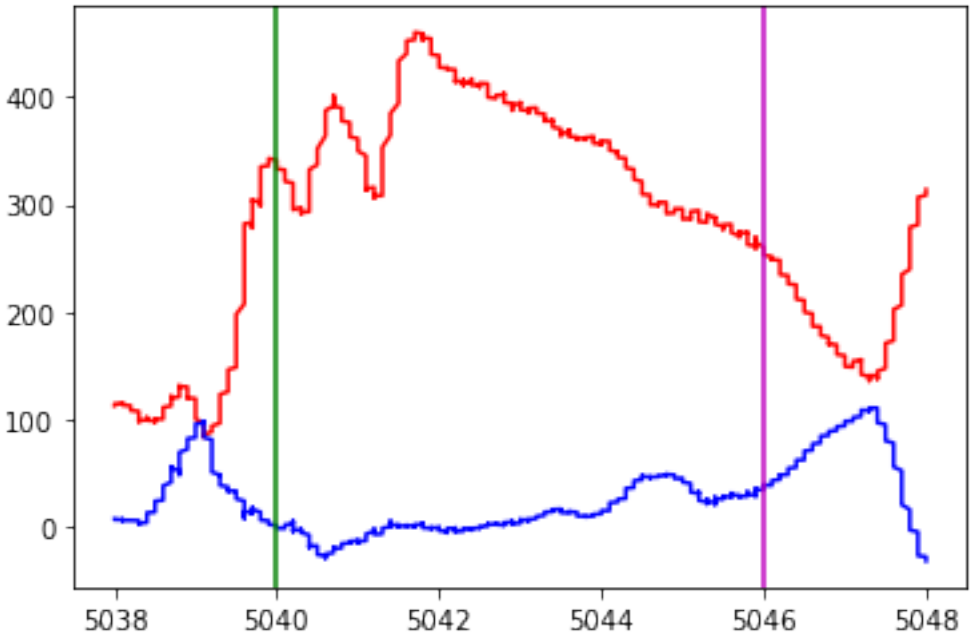

15.0
12.5
10.0
7.5
5.0
2.5
0.0
3652
3654
3656
3658
3660
3662

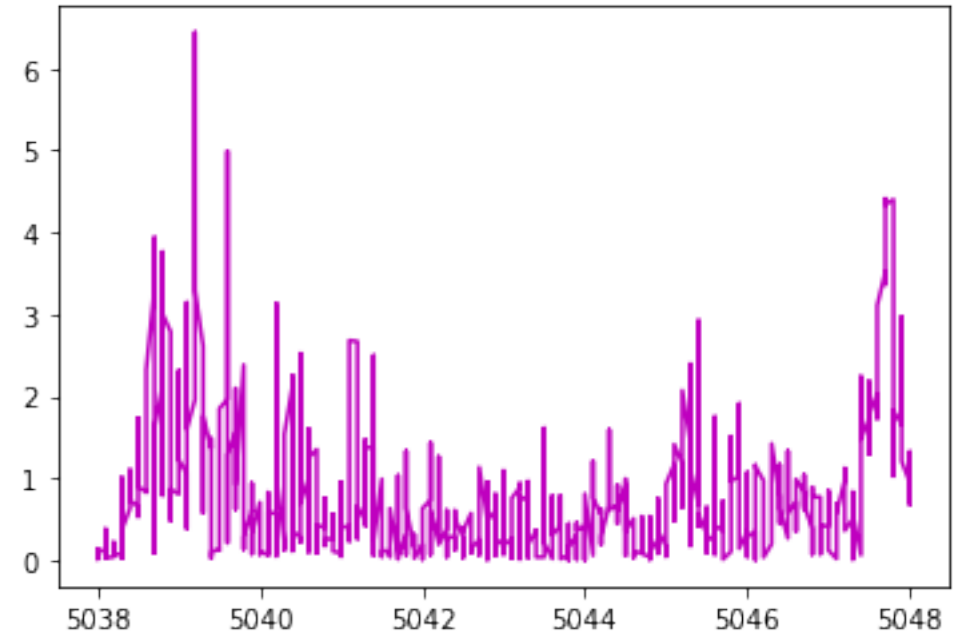

600
400
200
0
−200
3684
3686
3688
3690
3692
3694

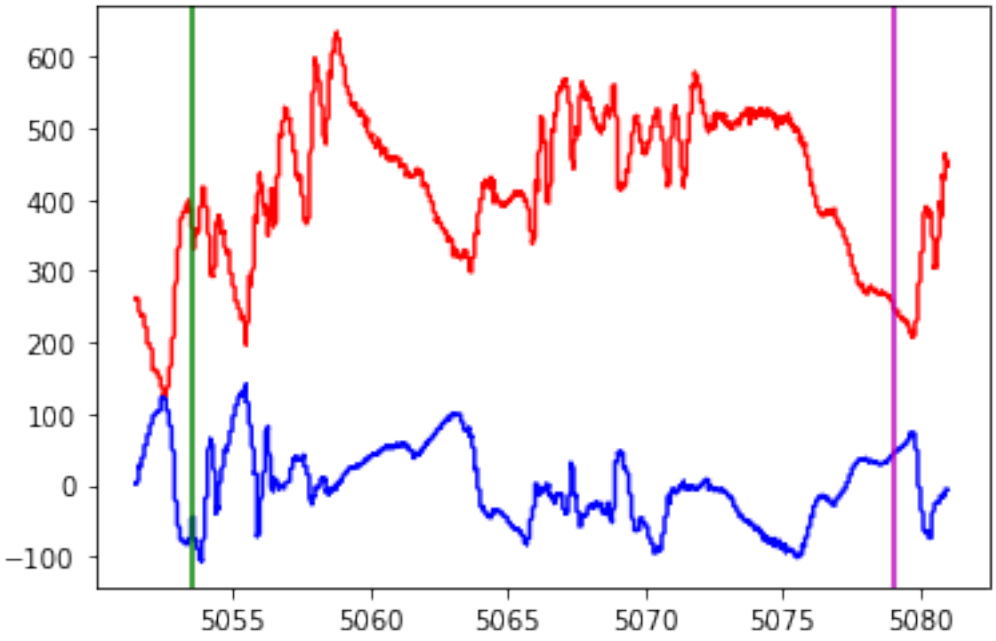

30
25
20
15
10
5
0
3684
3686
3688
3690
3692
3694

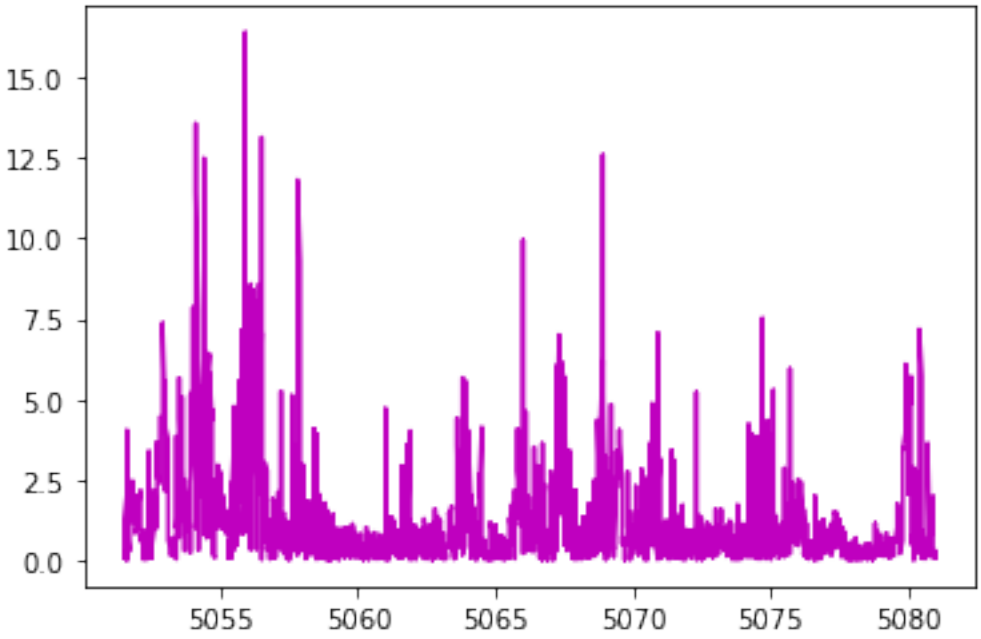

800
600
400
200
0
3702
3704
3706
3708
3710

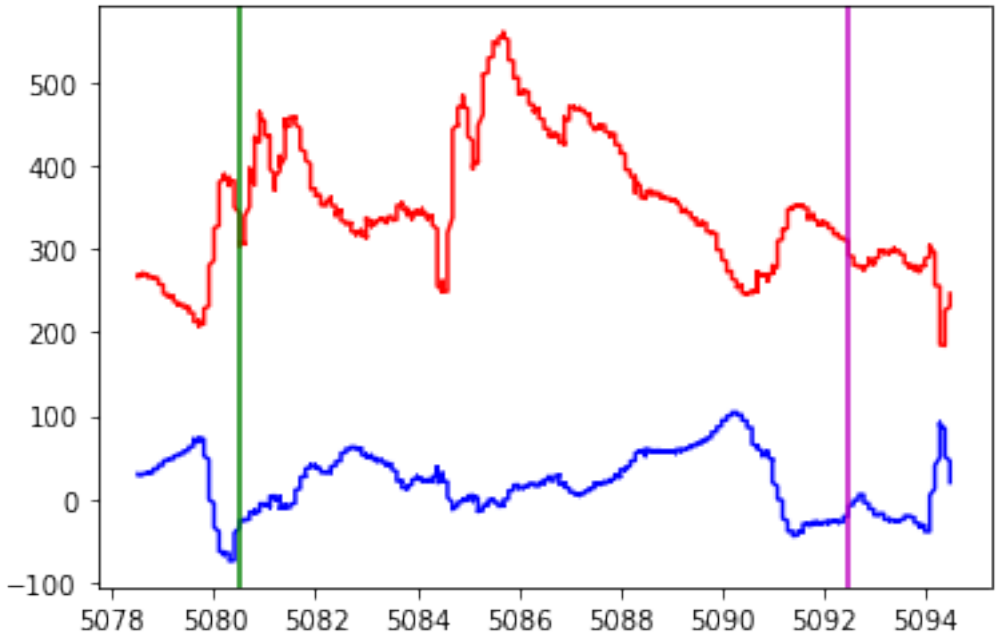

35
30
25
20
15
10
5
0
3702
3704
3706
3708
3710

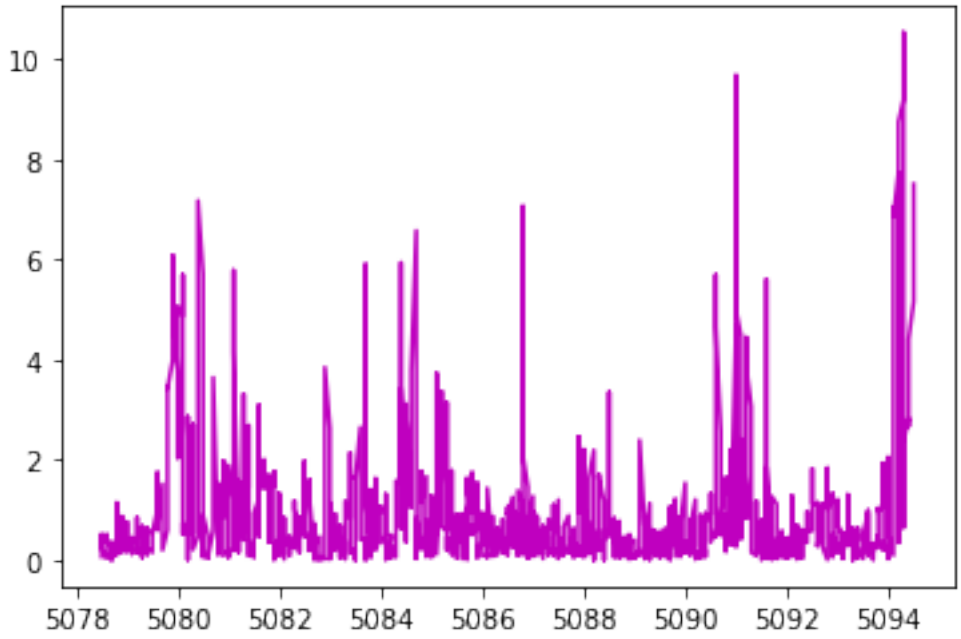

400
300
200
100
0
5038
5040
5042
5044
5046
5048

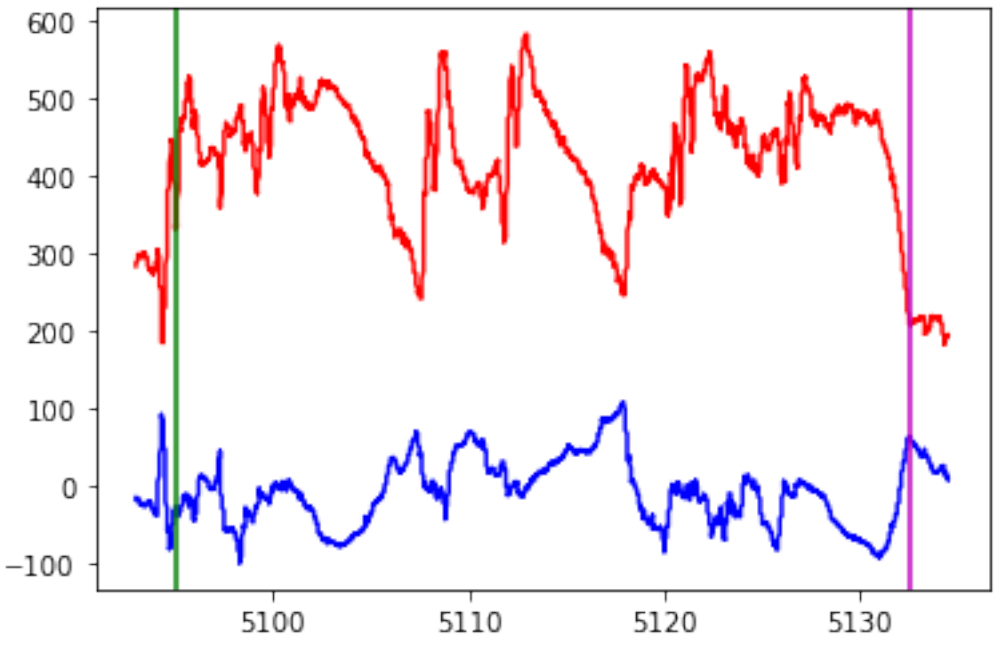

6
5
4
3
2
1
0
5038
5040
5042
5044
5046
5048

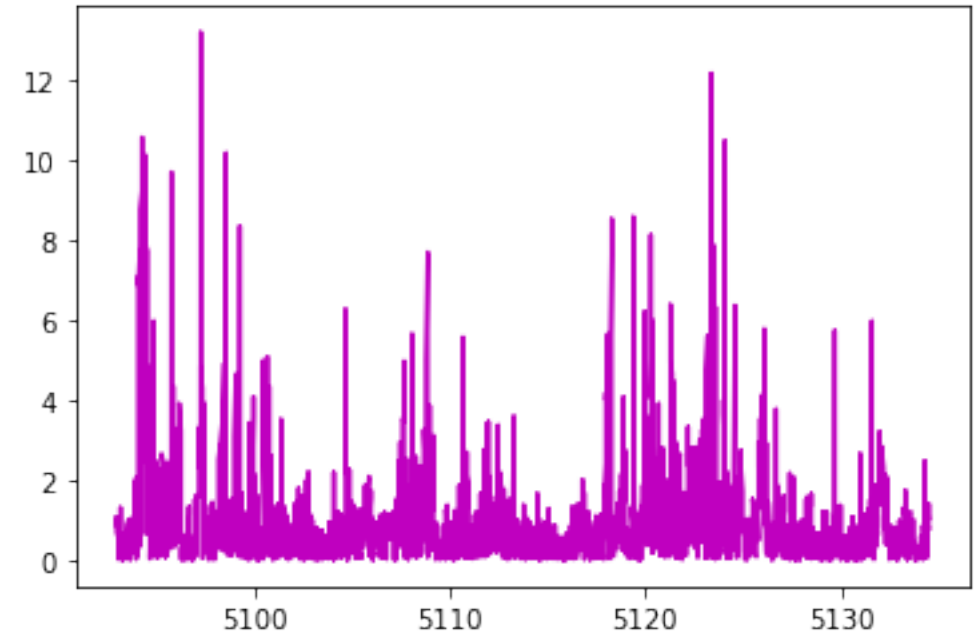

600
500
400
300
200
100
0
−100
5055
5060
5065
5070
5075
5080

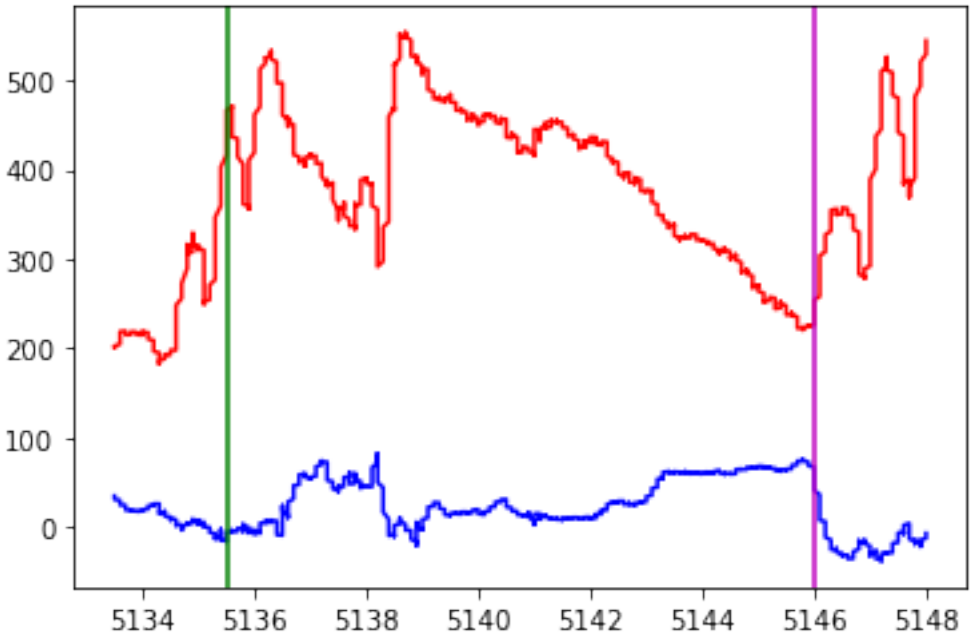

15.0
12.5
10.0
7.5
5.0
2.5
0.0
5055
5060
5065
5070
5075
5080

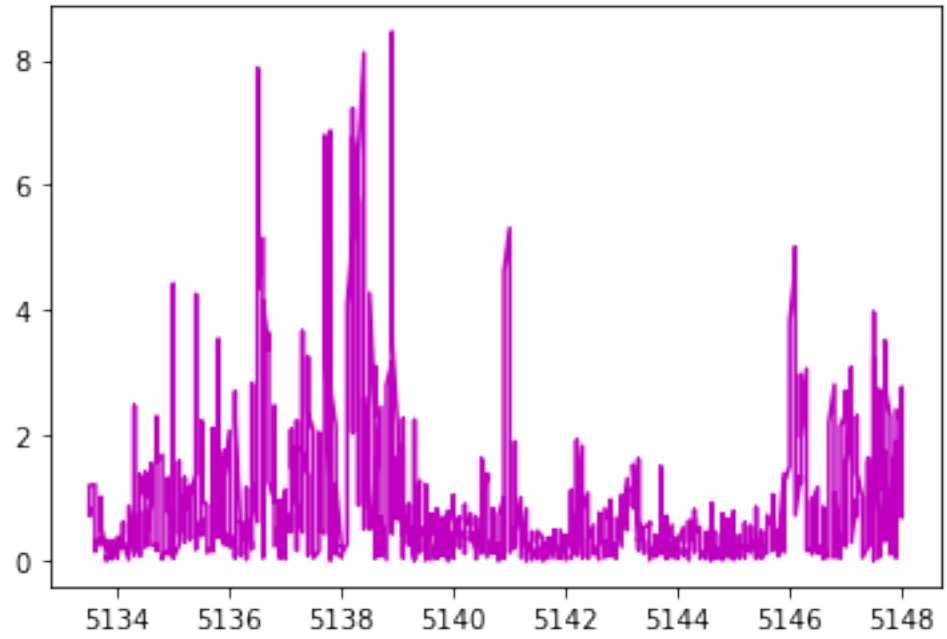

500
400
300
200
100
0
−100
5078
5080
5082
5084
5086
5088
5090
5092
5094

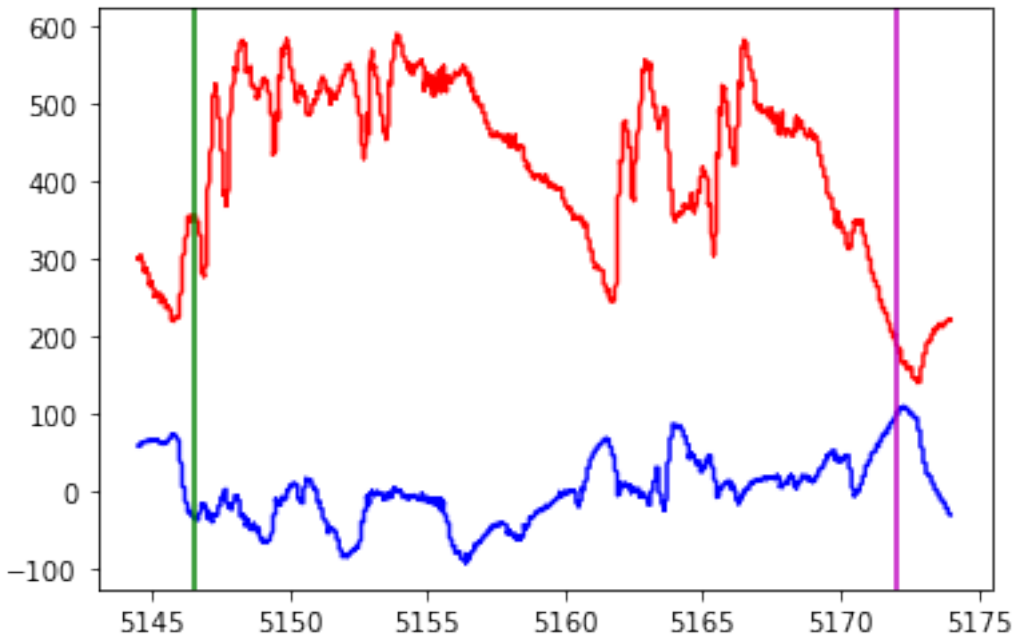

10
8
6
4
2
0
5078
5080
5082
5084
5086
5088
5090
5092
5094

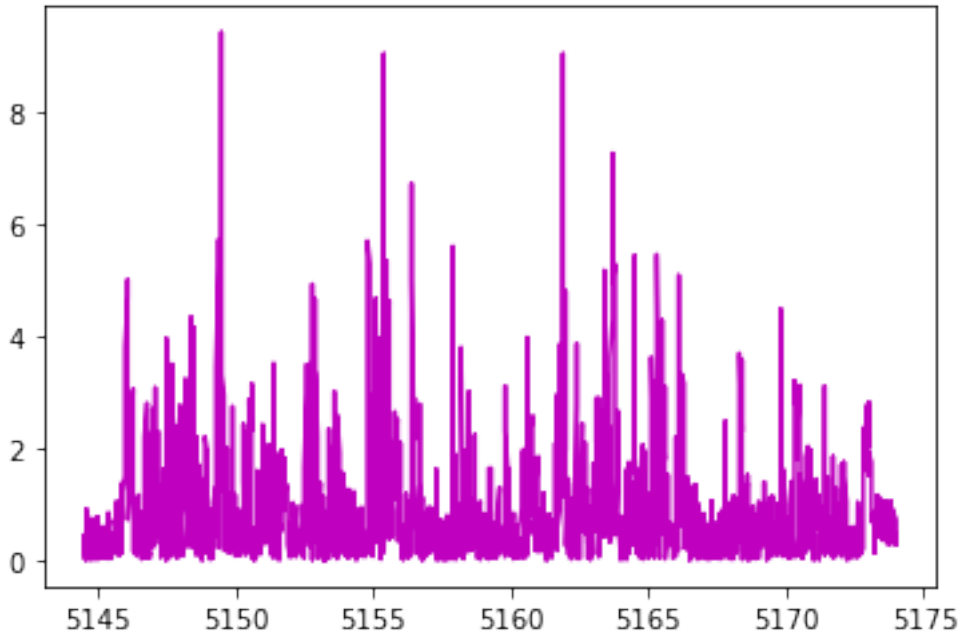

600
500
400
300
200
100
0
−100
5100
5110
5120
5130

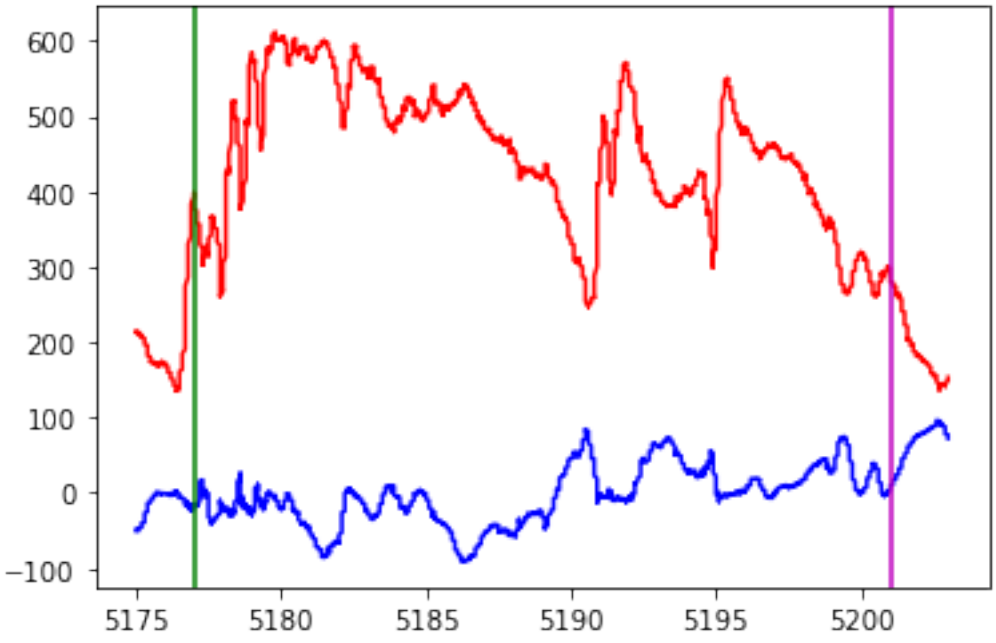

12
10
8
6
4
2
0
5100
5110
5120
5130

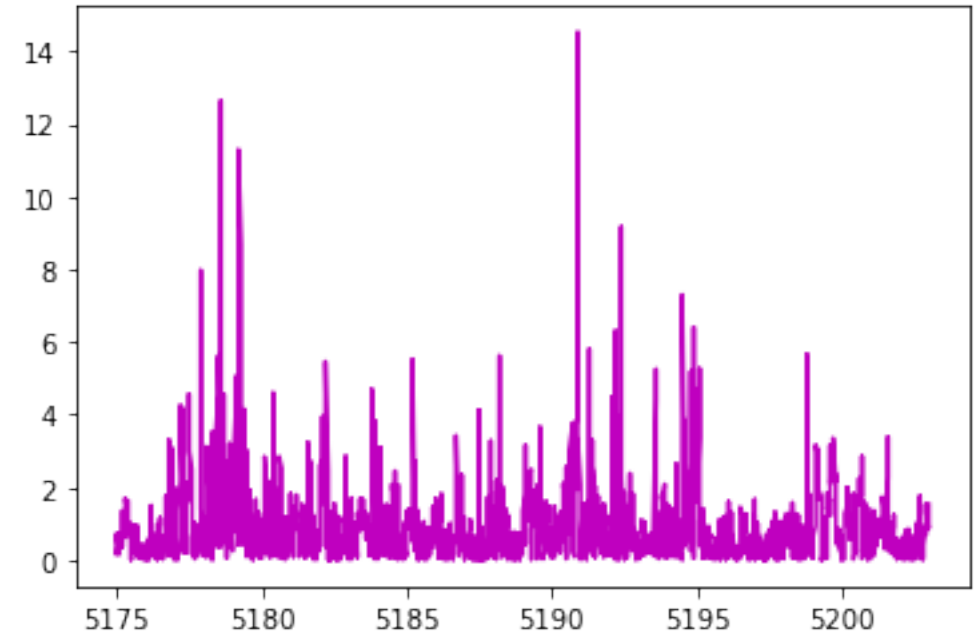

500
400
300
200
100
0
5134
5136
5138
5140
5142
5144
5146
5148

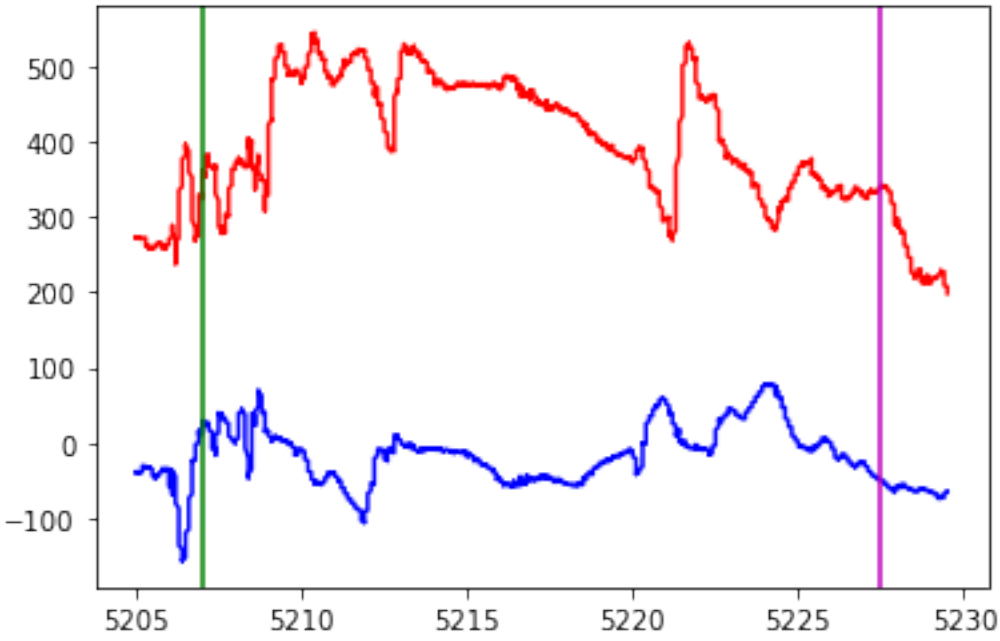

8
6
4
2
0
5134
5136
5138
5140
5142
5144
5146
5148

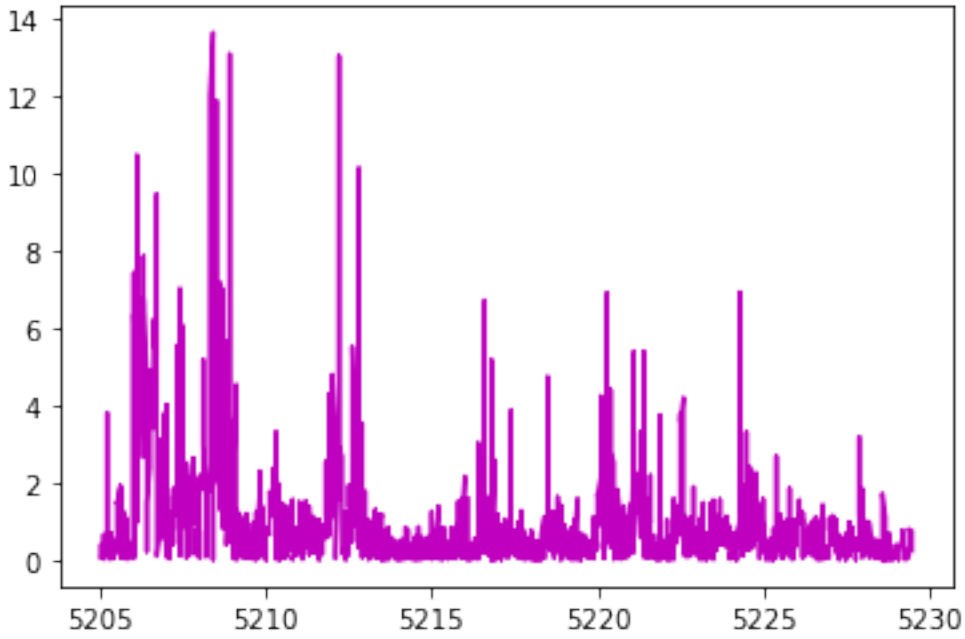

600
500
400
300
200
100
0
−100
5145
5150
5155
5160
5165
5170
5175

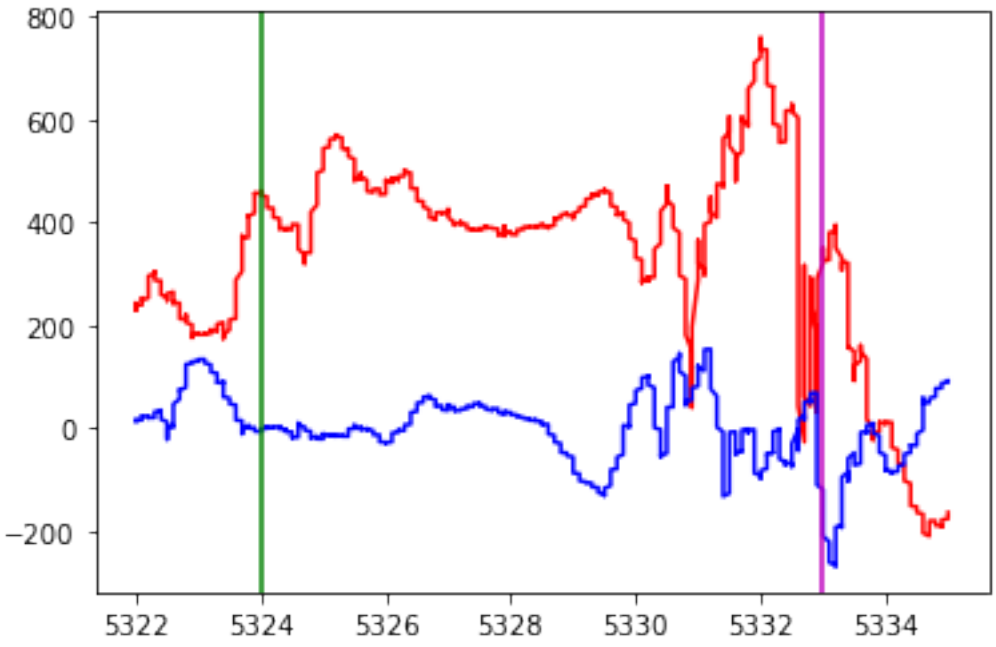

8
6
4
2
0
5145
5150
5155
5160
5165
5170
5175

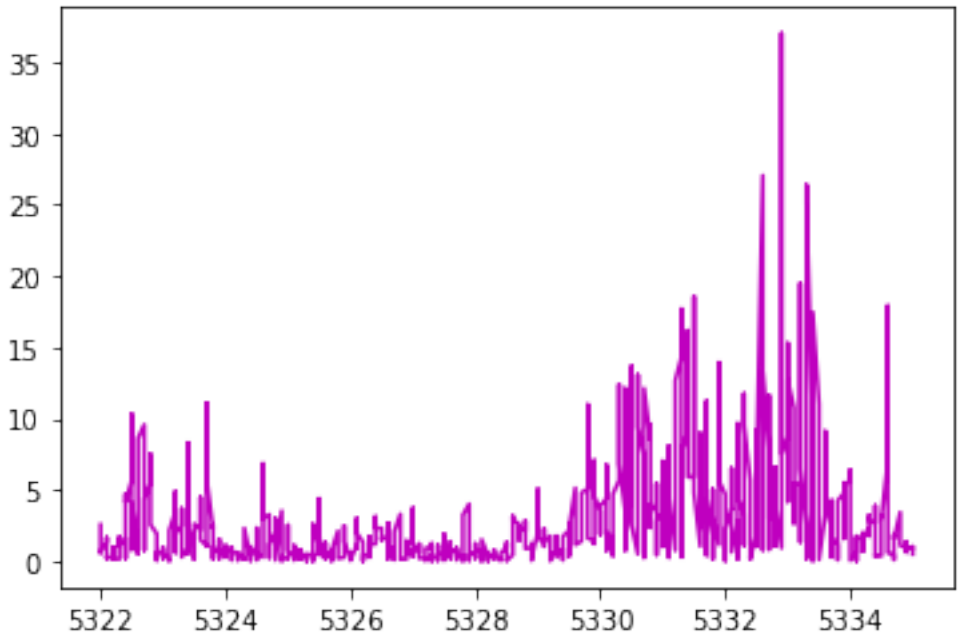

600
500
400
300
200
100
0
−100
5175
5180
5185
5190
5195
5200

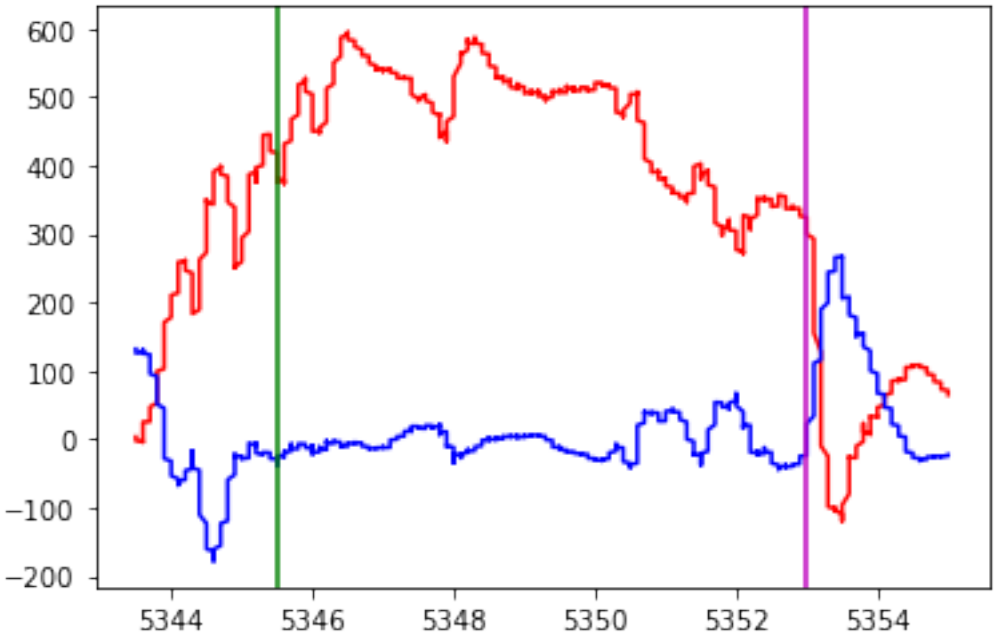

14
12
10
8
6
4
2
0
5175
5180
5185
5190
5195
5200

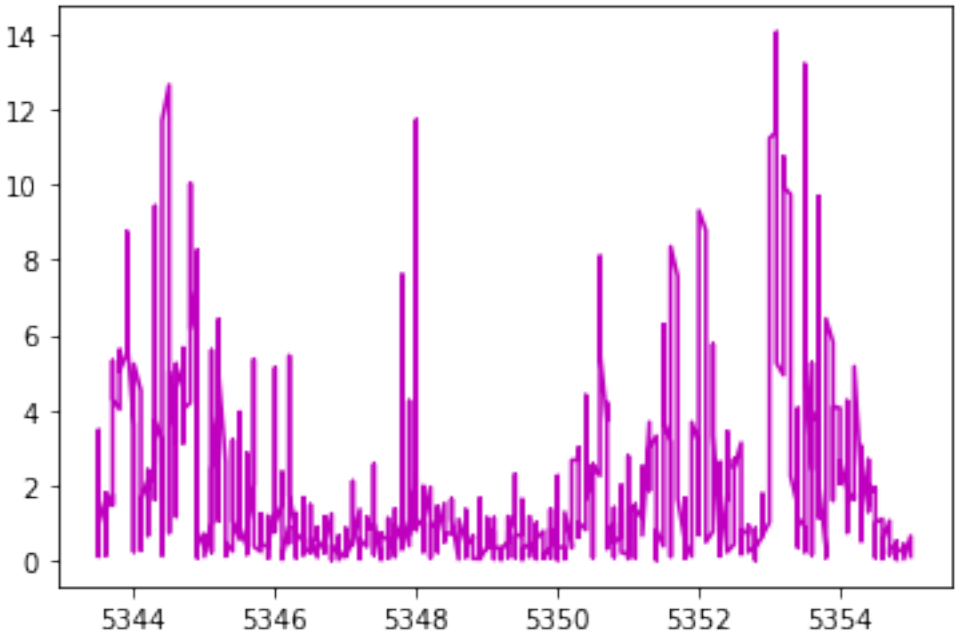

500
400
300
200
100
0
−100
5205
5210
5215
5220
5225
5230

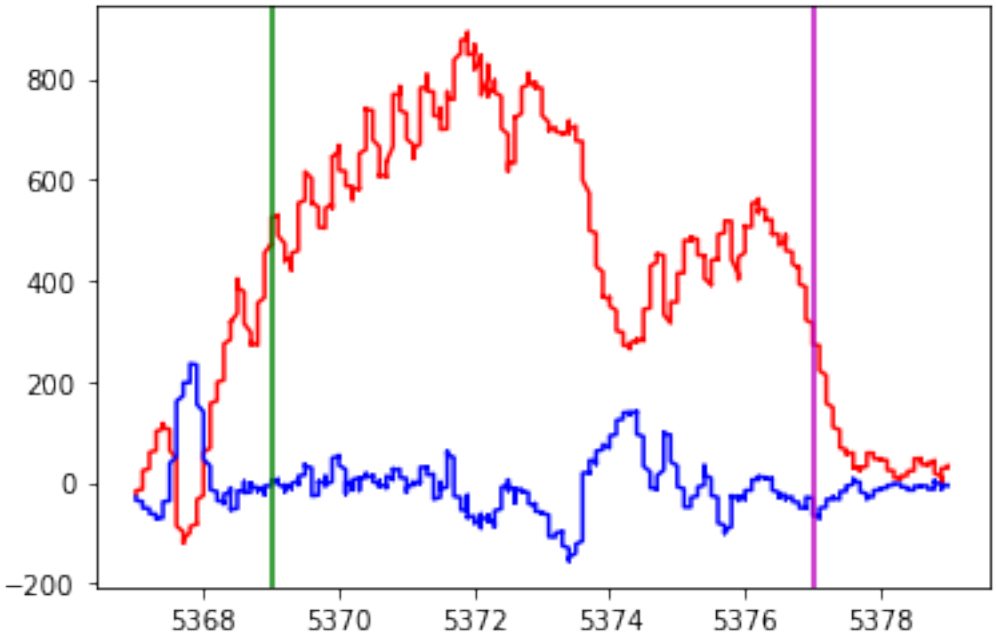

14
12
10
8
6
4
2
0
5205
5210
5215
5220
5225
5230

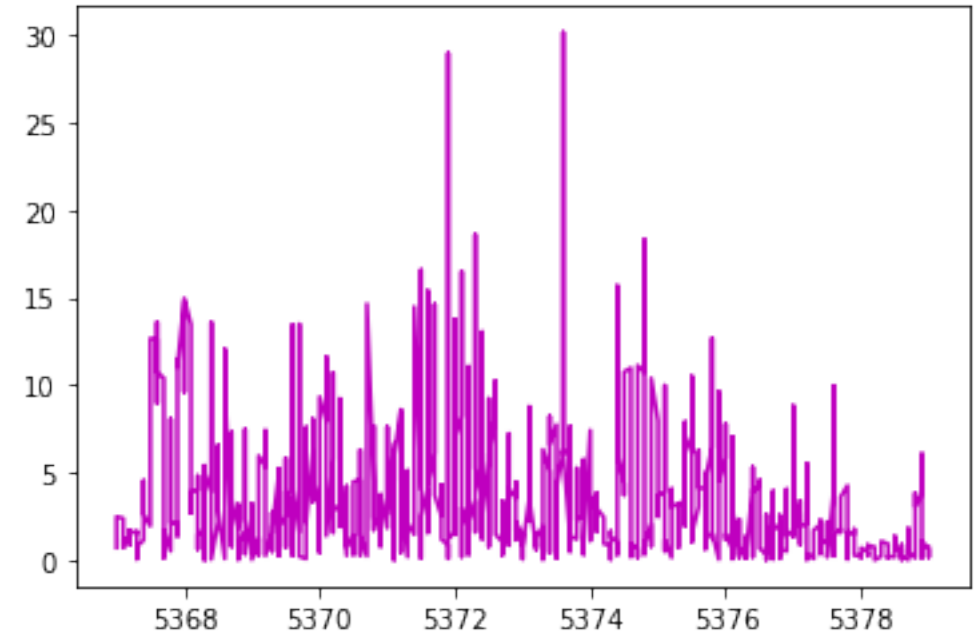

800
600
400
200
0
−200
5322
5324
5326
5328
5330
5332
5334

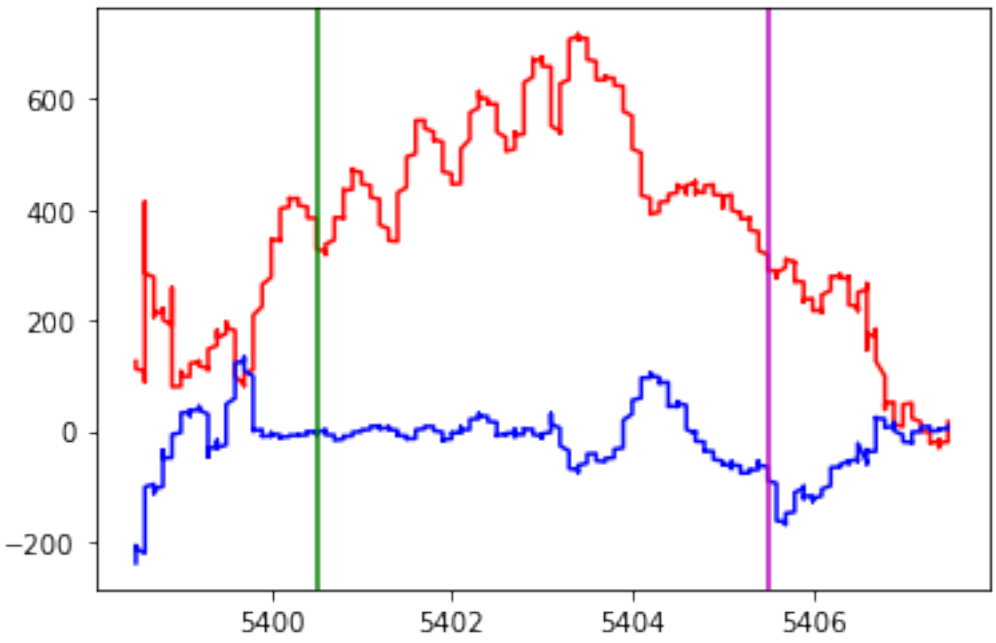

35
30
25
20
15
10
5
0
5322
5324
5326
5328
5330
5332
5334

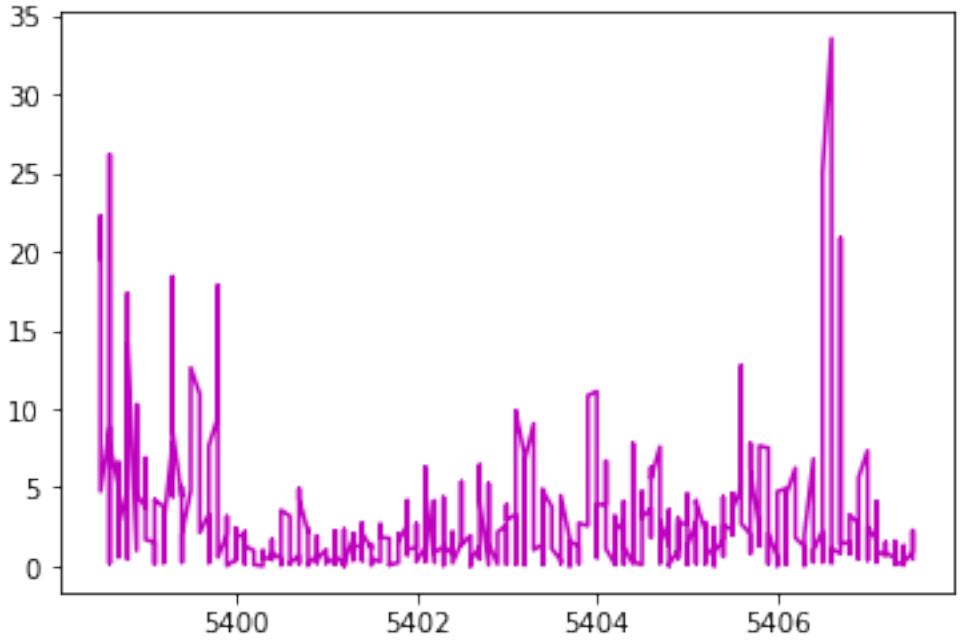

600
500
400
300
200
100
0
−100
−200
5344
5346
5348
5350
5352
5354

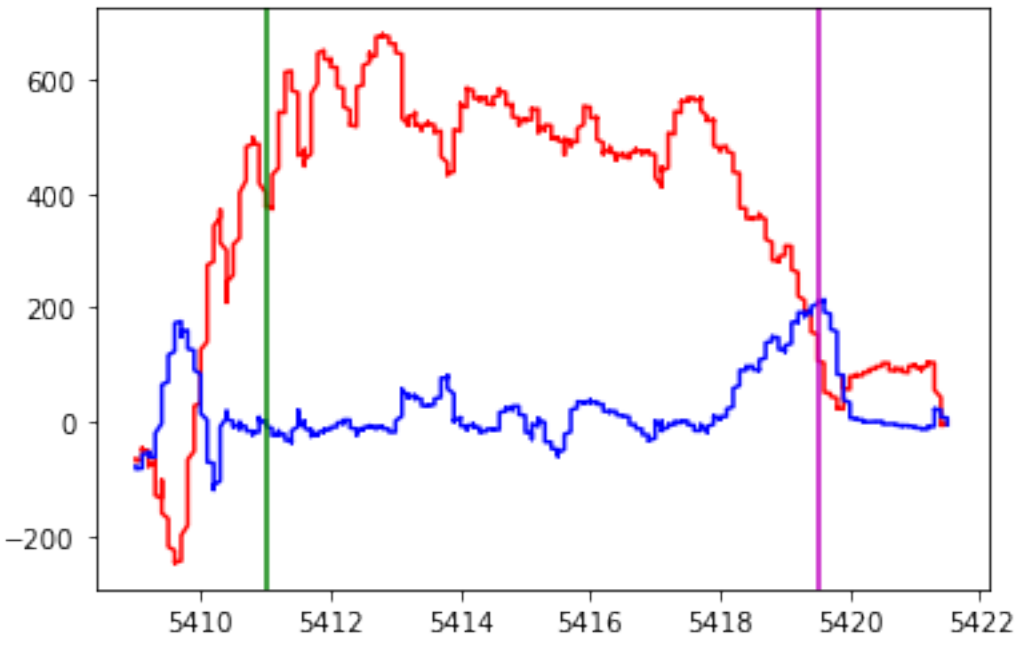

14
12
10
8
6
4
2
0
5344
5346
5348
5350
5352
5354

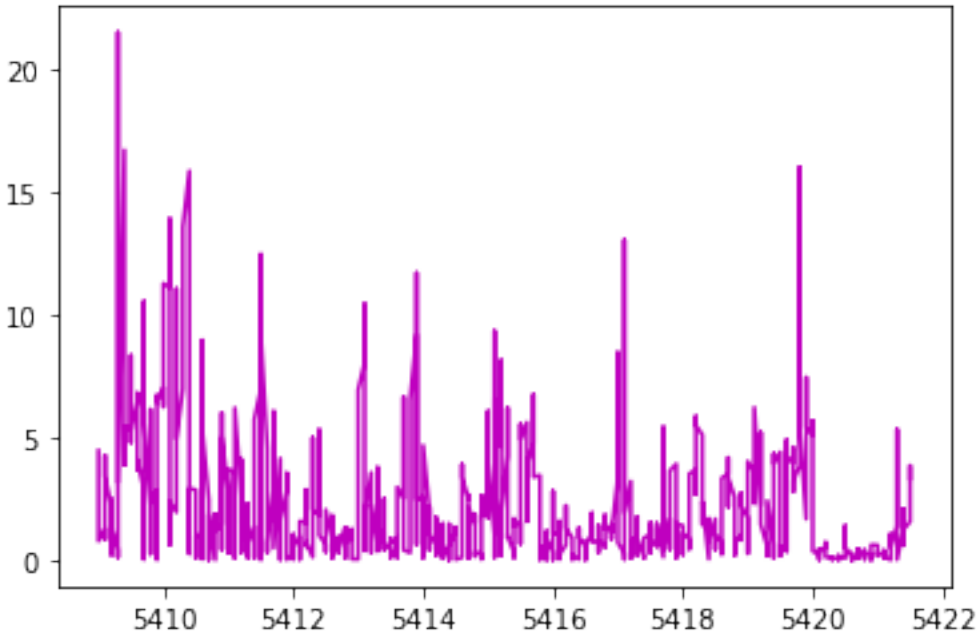

800
600
400
200
0
−200
5368
5370
5372
5374
5376
5378

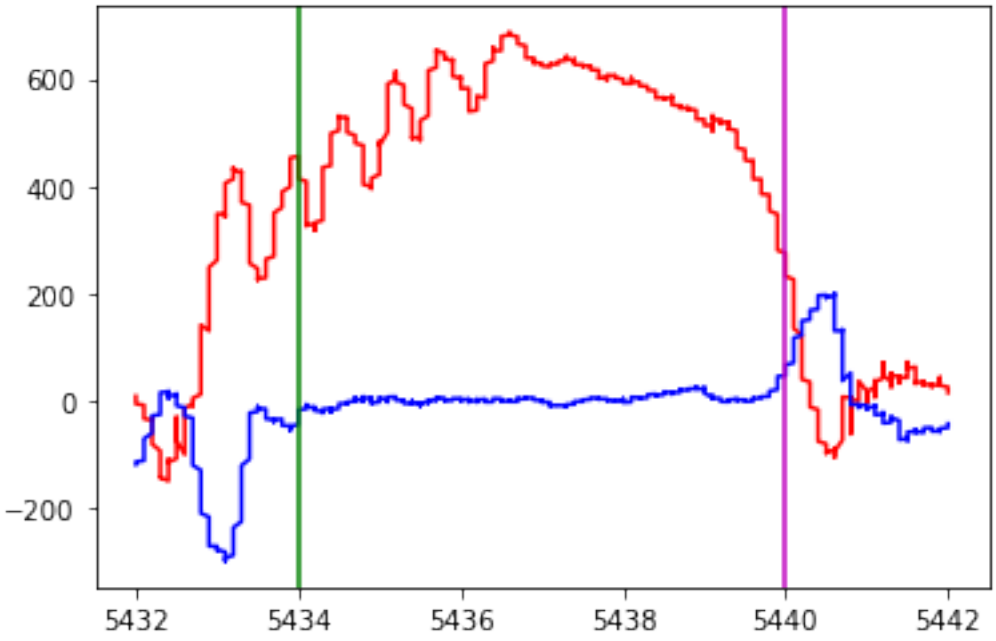

30
25
20
15
10
5
0
5368
5370
5372
5374
5376
5378

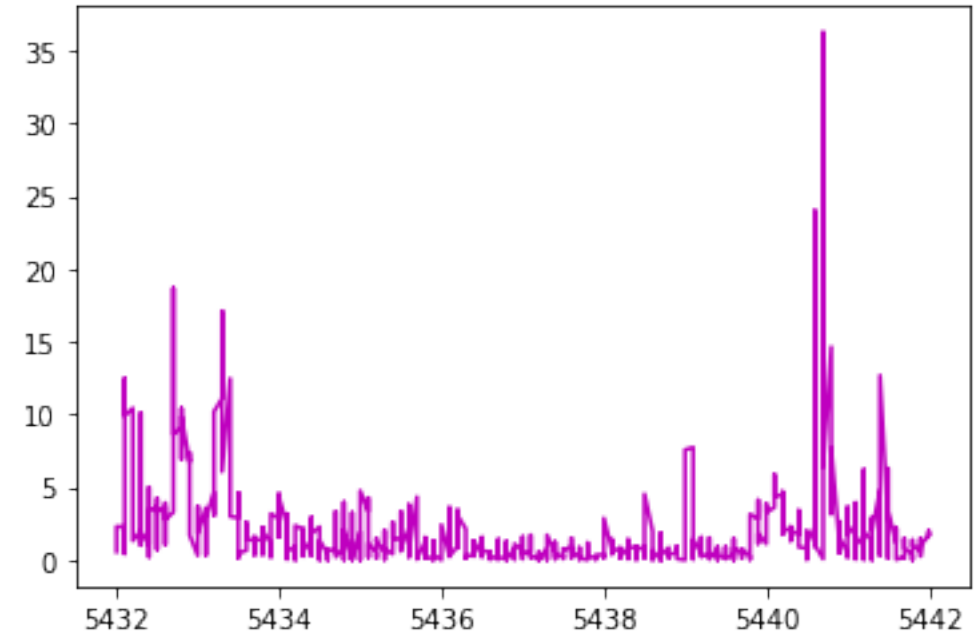

600
400
200
0
−200
5400
5402
5404
5406

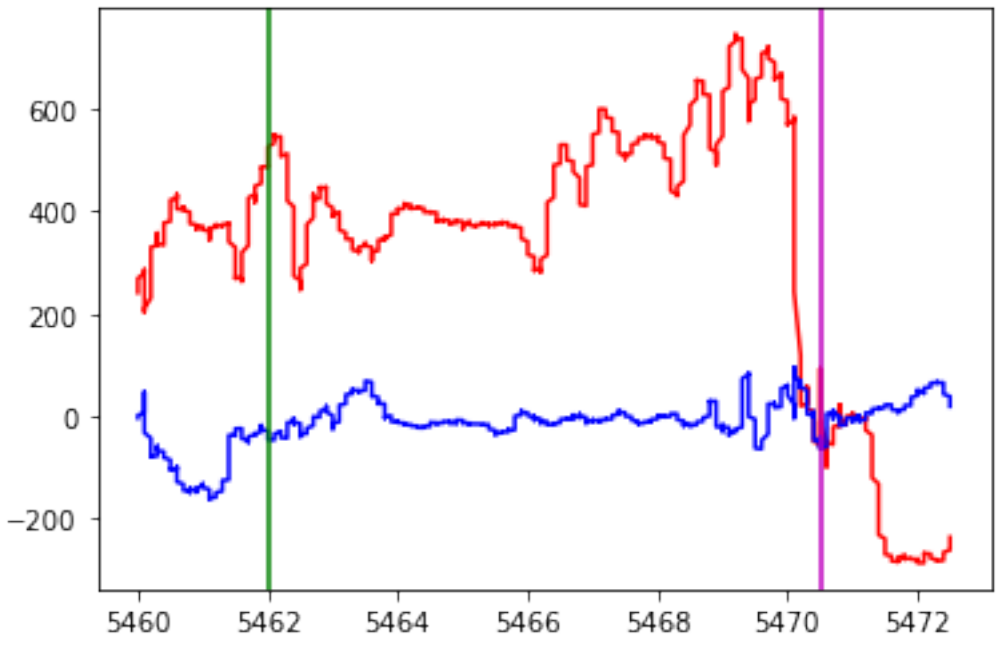

35
30
25
20
15
10
5
0
5400
5402
5404
5406

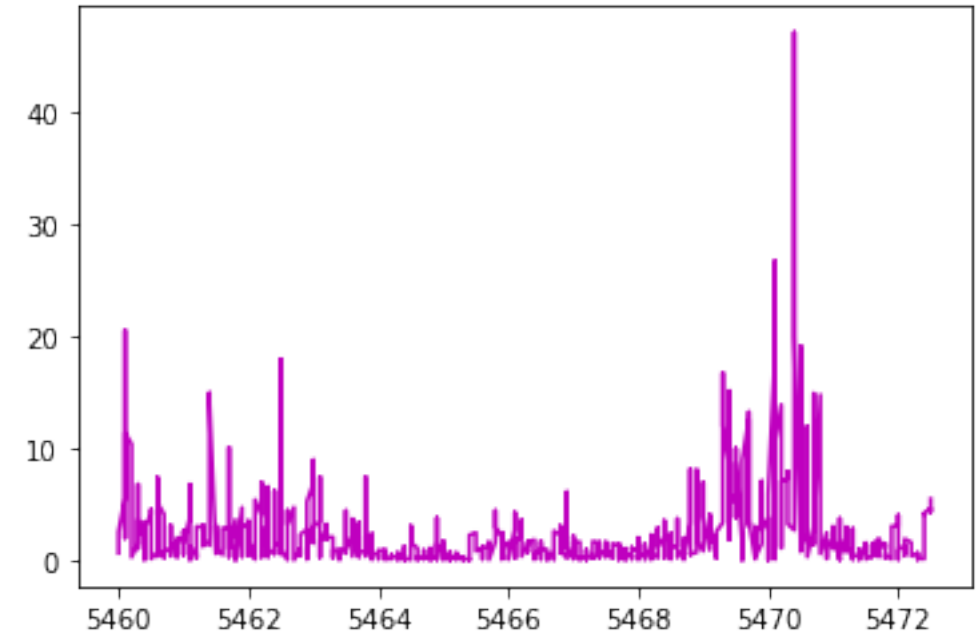

600
400
200
0
−200
5410
5412
5414
5416
5418
5420
5422

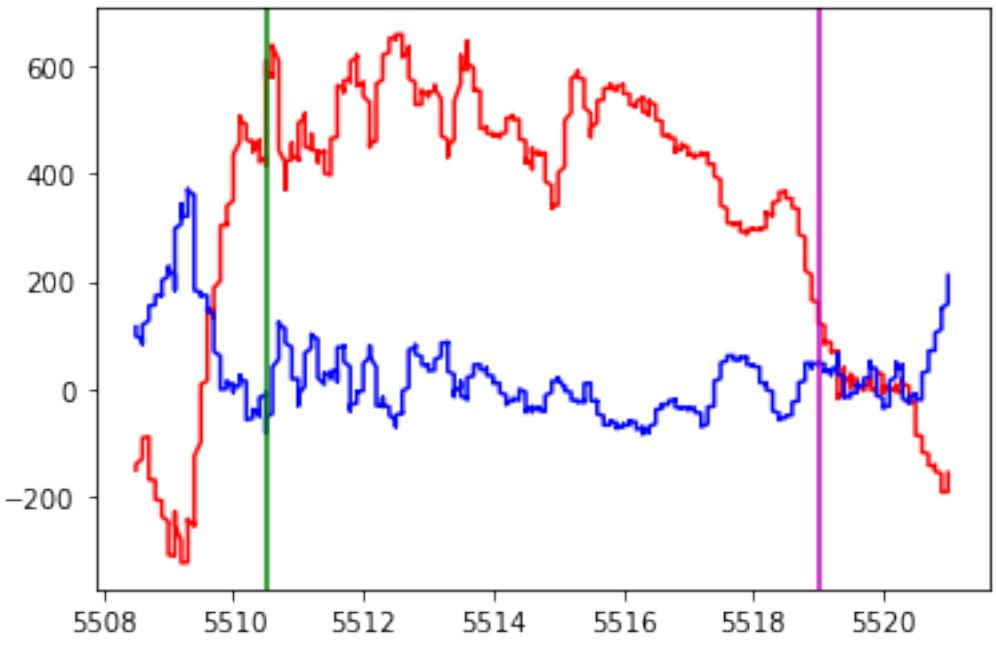

20
15
10
5
0
5410
5412
5414
5416
5418
5420
5422

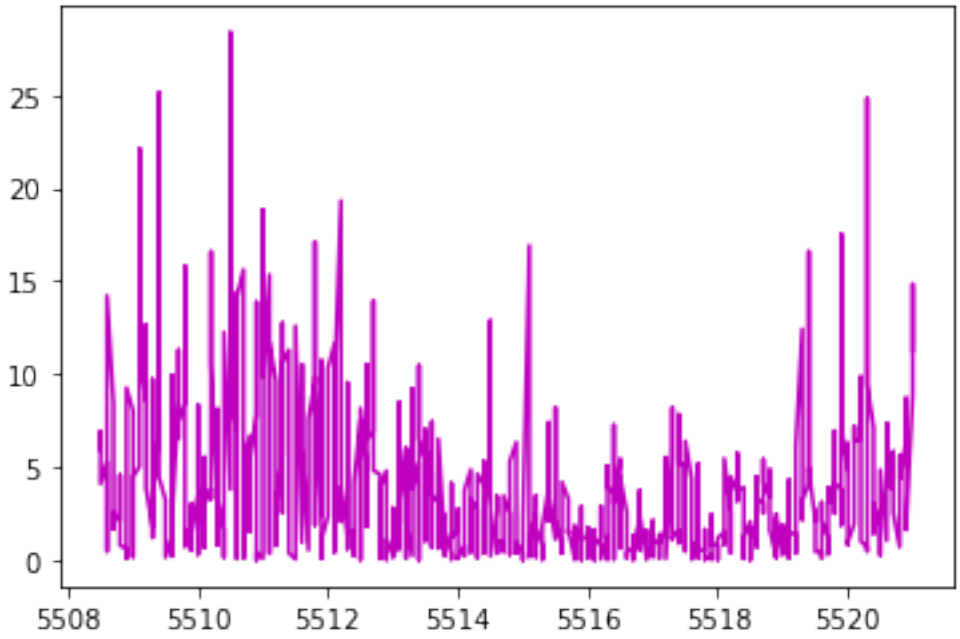

600
400
200
0
−200
5432
5434
5436
5438
5440
5442

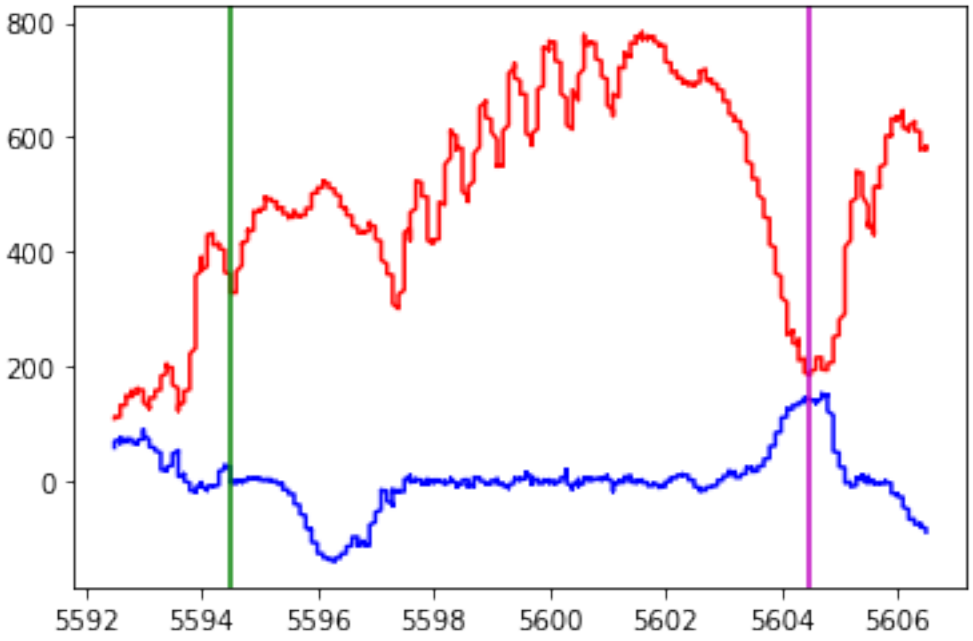

35
30
25
20
15
10
5
0
5432
5434
5436
5438
5440
5442

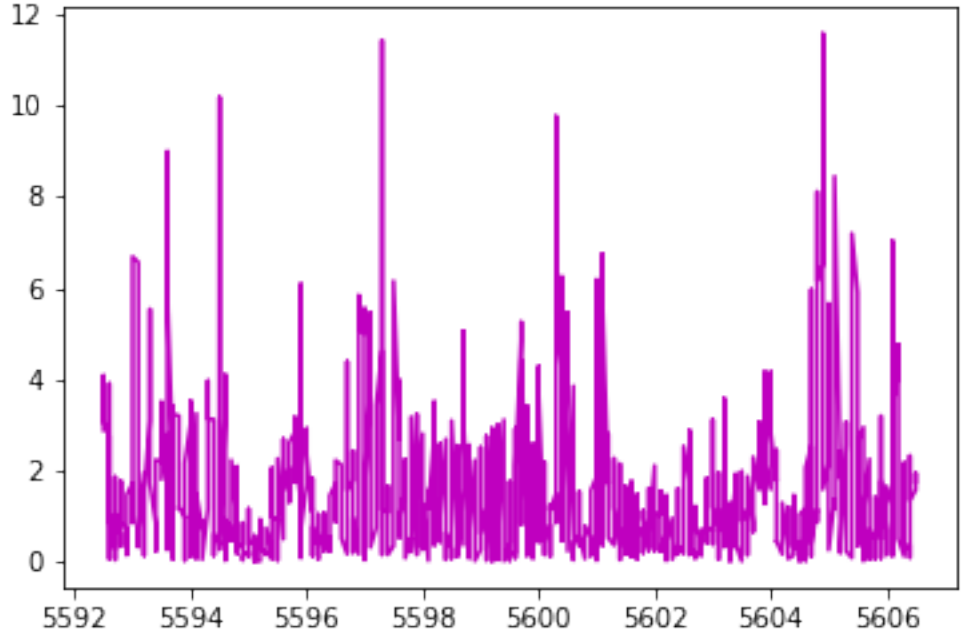

600
400
200
0
−200
5460
5462
5464
5466
5468
5470
5472

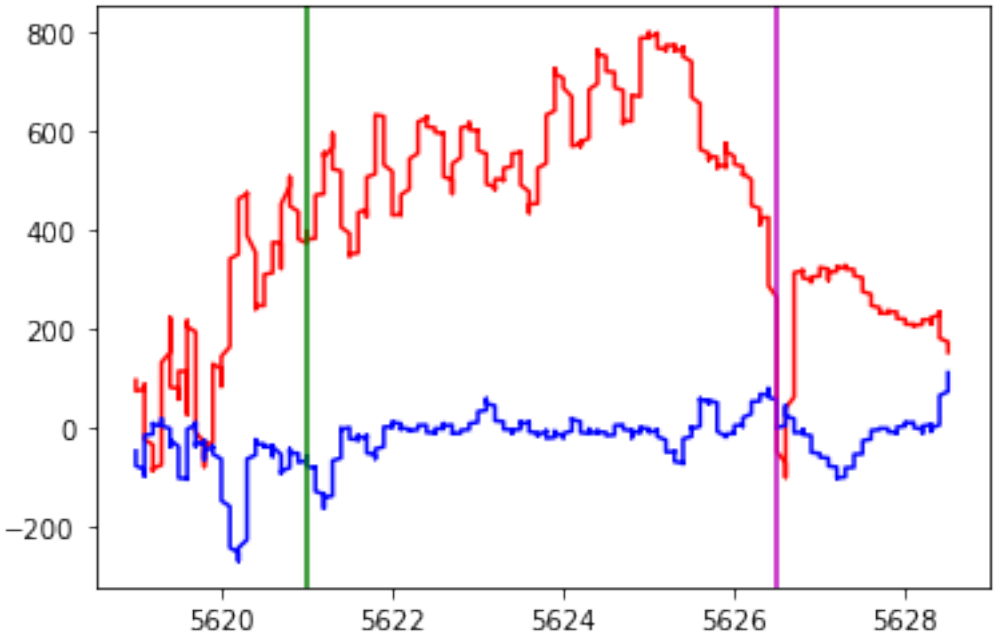

40
30
20
10
0
5460
5462
5464
5466
5468
5470
5472

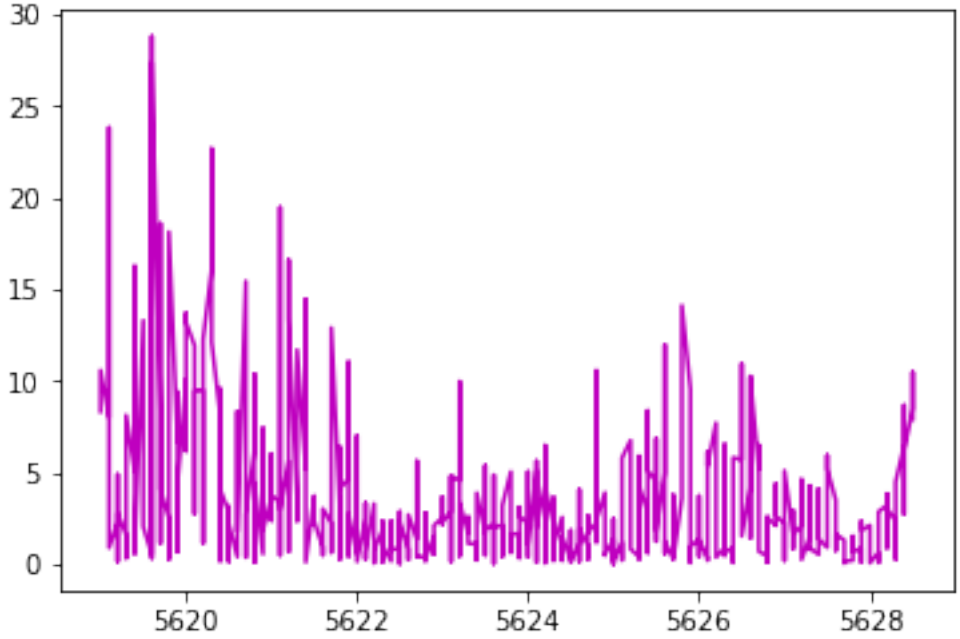

600
400
200
0
−200
5508
5510
5512
5514
5516
5518
5520

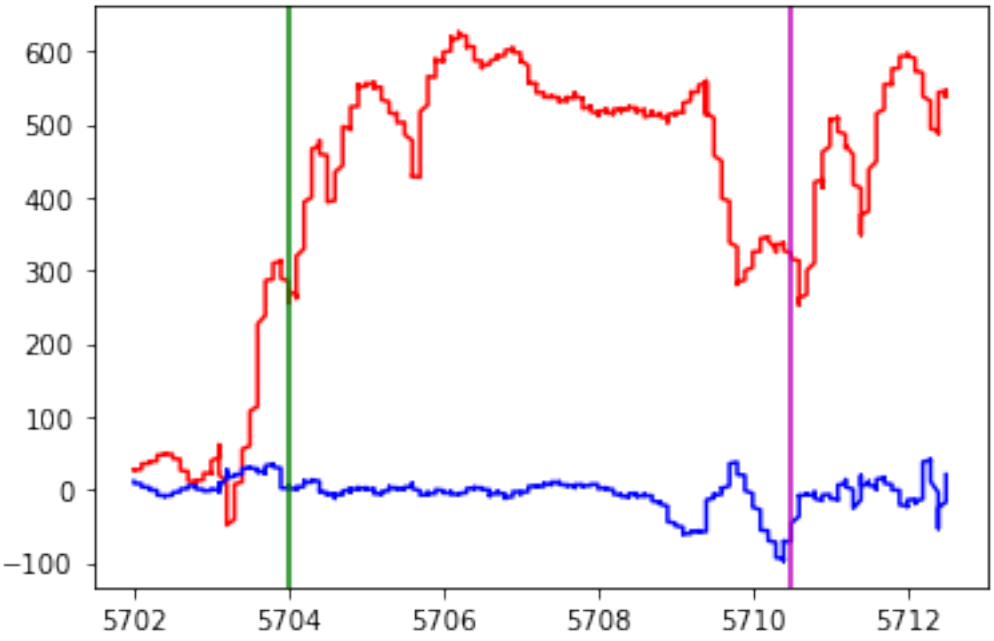

25
20
15
10
5
0
5508
5510
5512
5514
5516
5518
5520

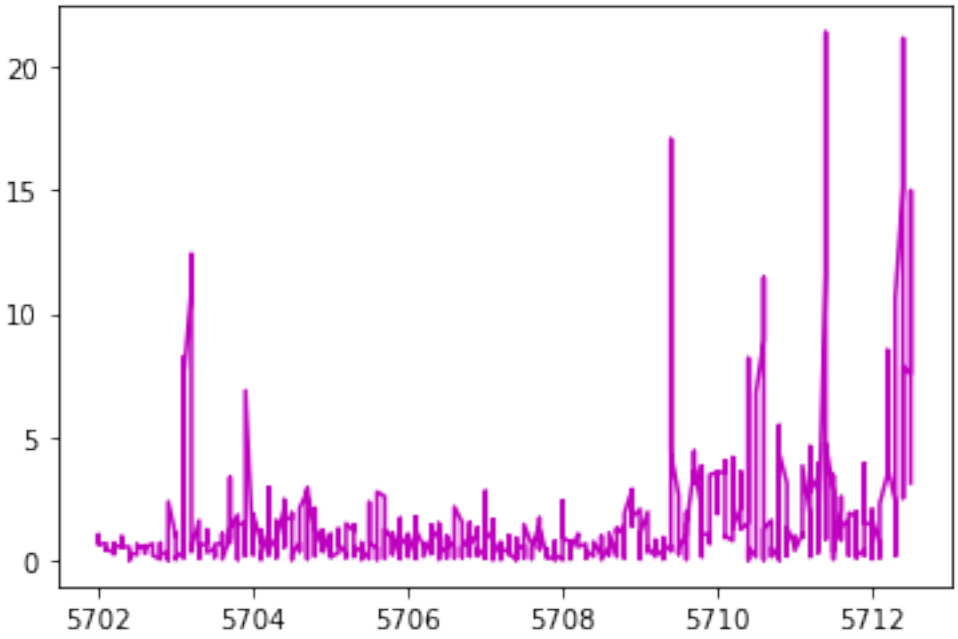

800
600
400
200
0
5592
5594
5596
5598
5600
5602
5604
5606

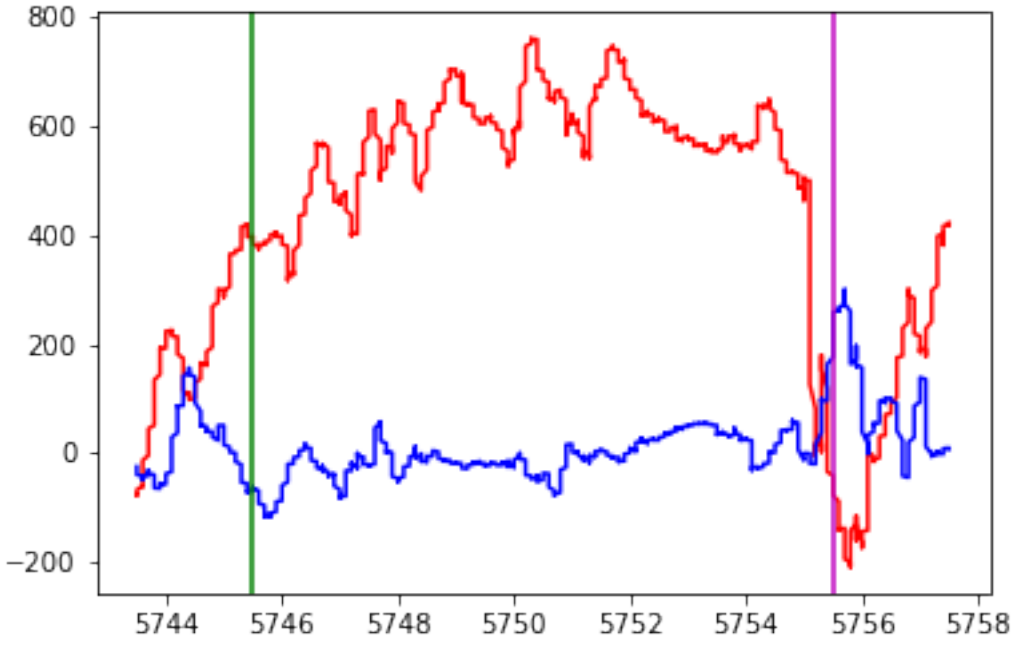

12
10
8
6
4
2
0
5592
5594
5596
5598
5600
5602
5604
5606

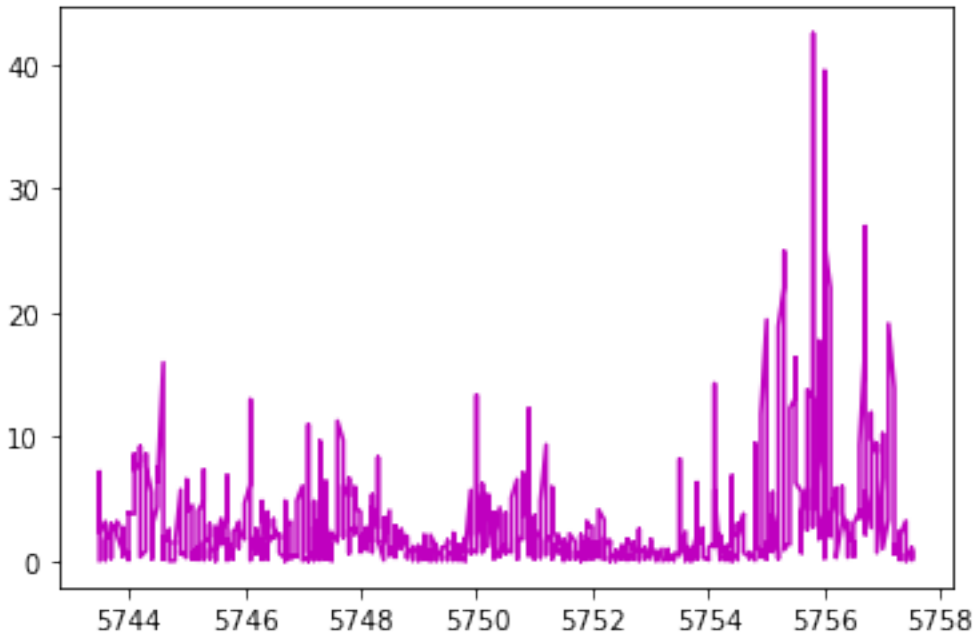

800
600
400
200
0
−200
5620
5622
5624
5626
5628

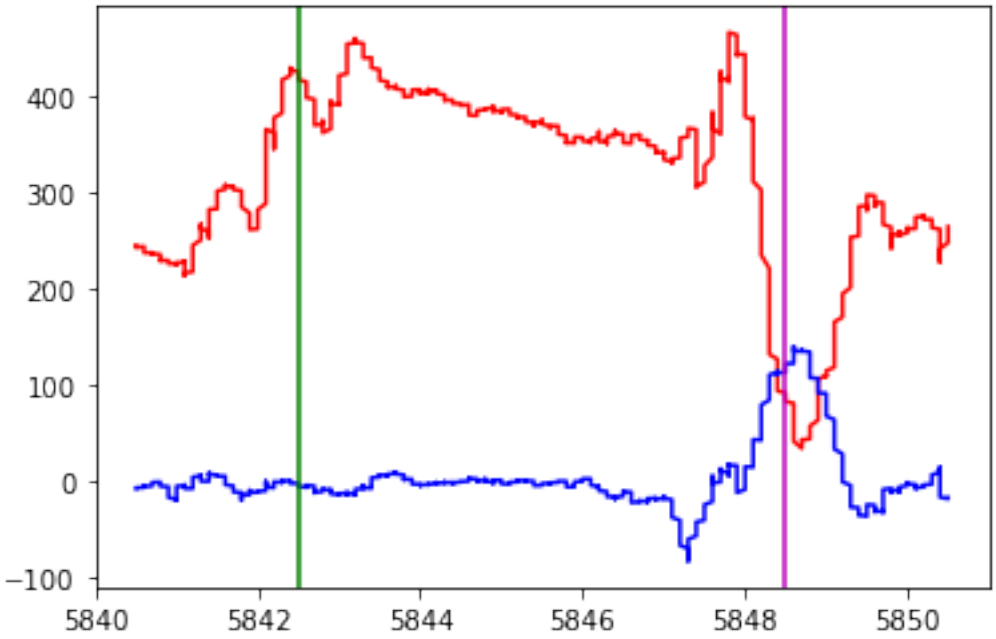

30
25
20
15
10
5
0
5620
5622
5624
5626
5628

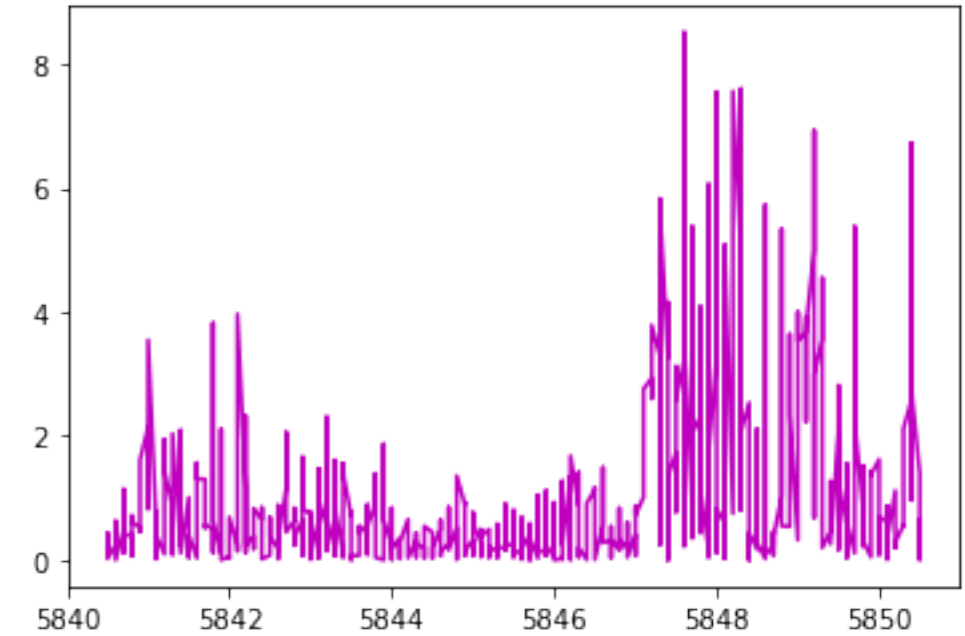

600
500
400
300
200
100
0
−100
5702
5704
5706
5708
5710
5712

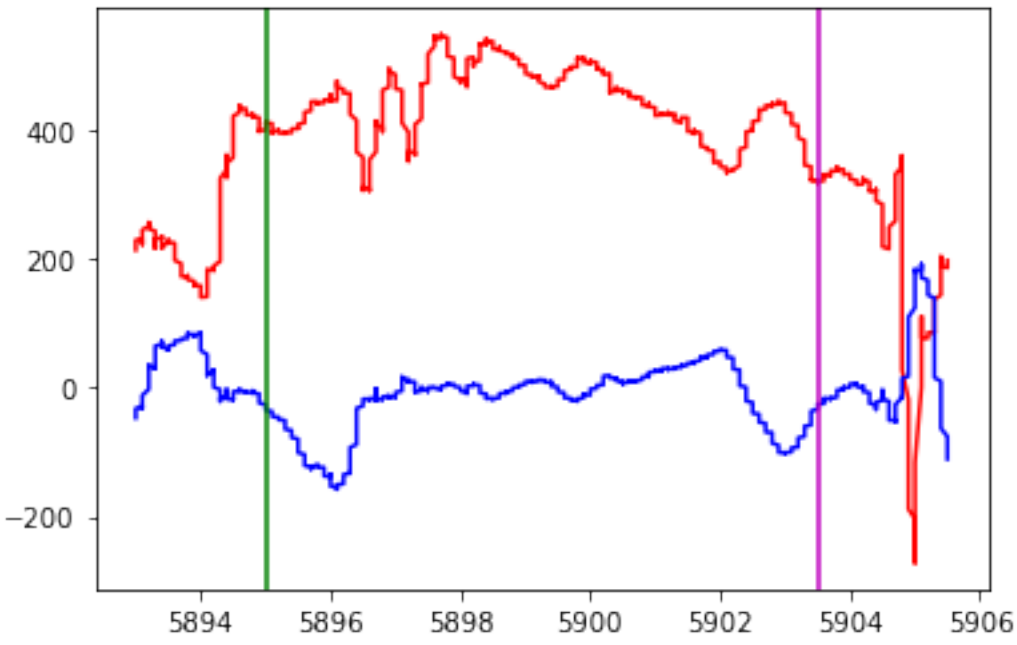

20
15
10
5
0
5702
5704
5706
5708
5710
5712

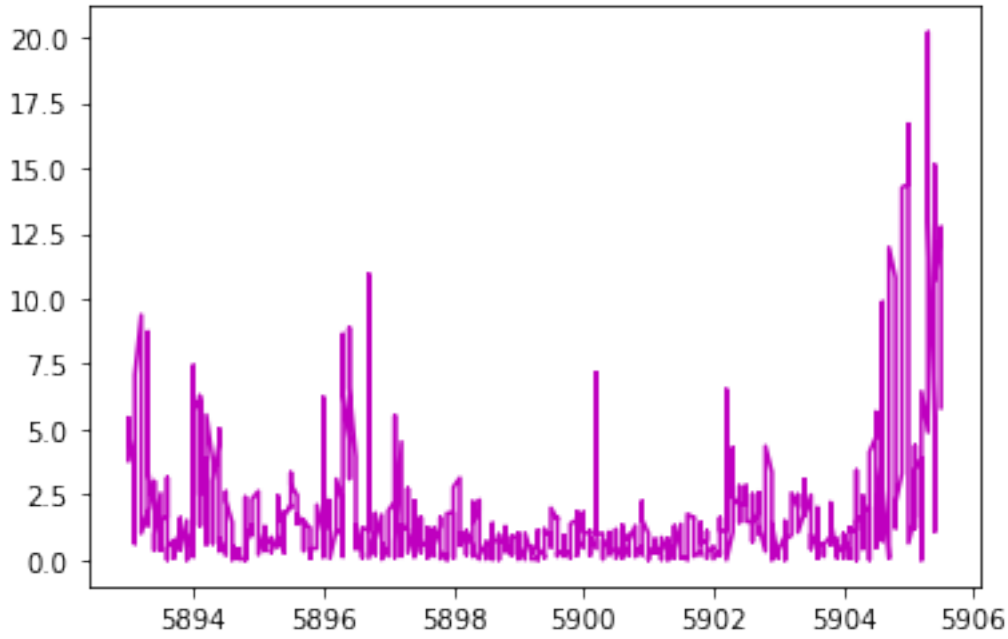

800
600
400
200
0
−200
5744
5746
5748
5750
5752
5754
5756
5758

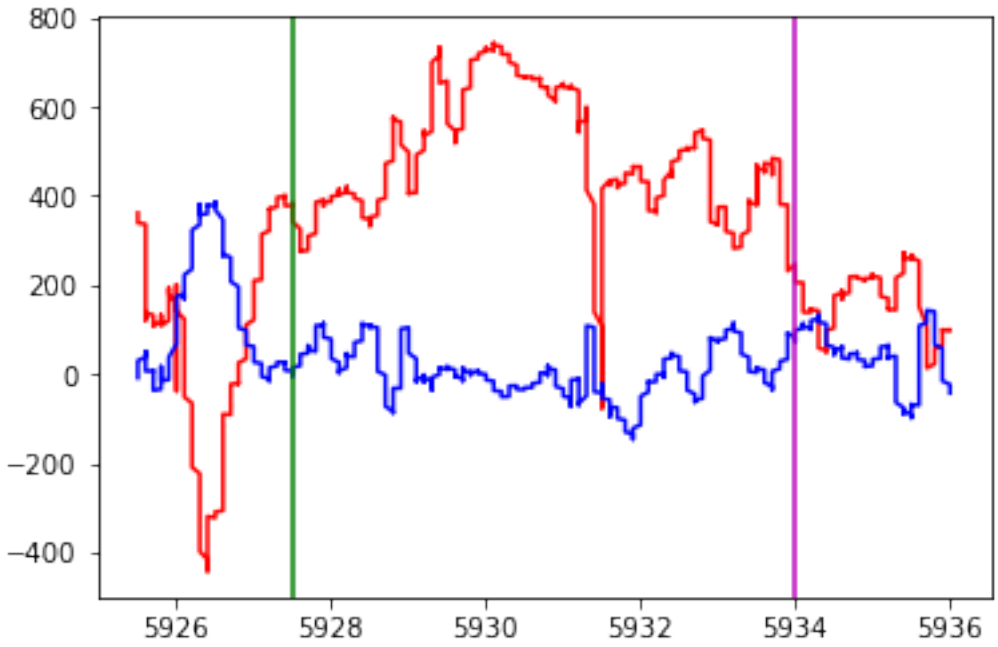

40
30
20
10
0
5744
5746
5748
5750
5752
5754
5756
5758

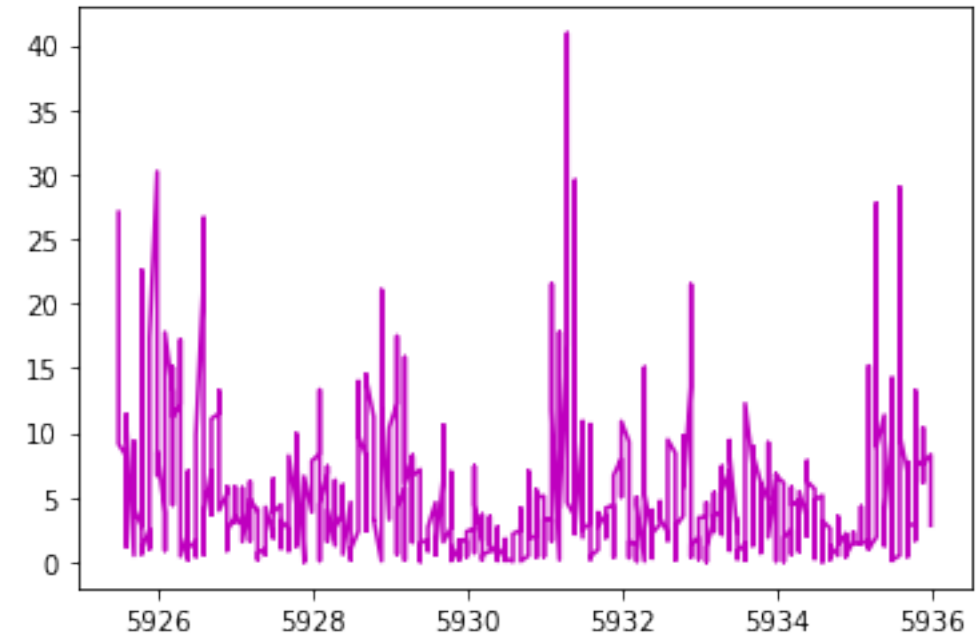

400
300
200
100
0
−100
5840
5842
5844
5846
5848
5850

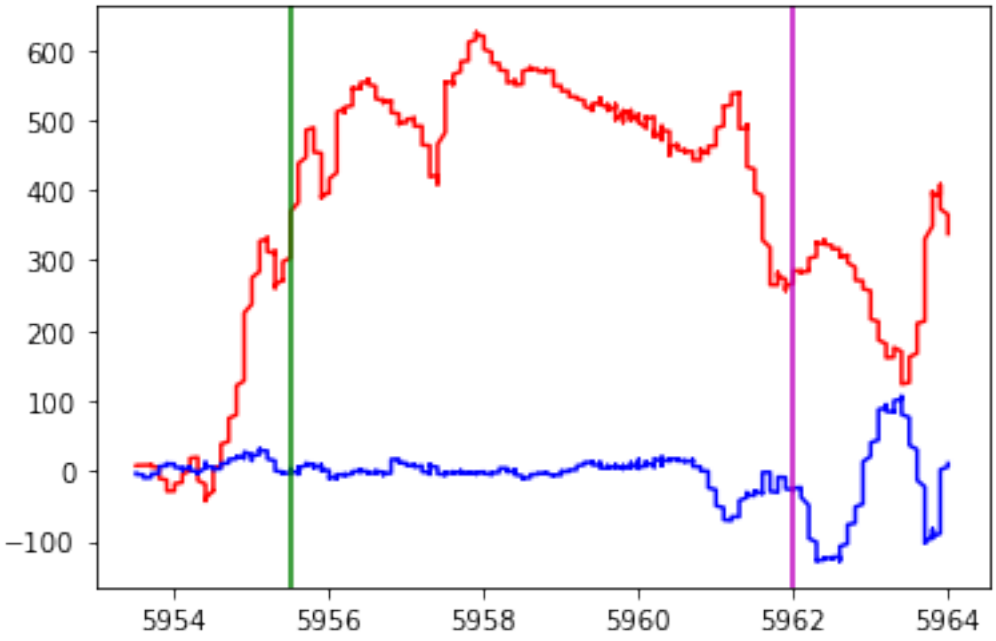

8
6
4
2
0
5840
5842
5844
5846
5848
5850

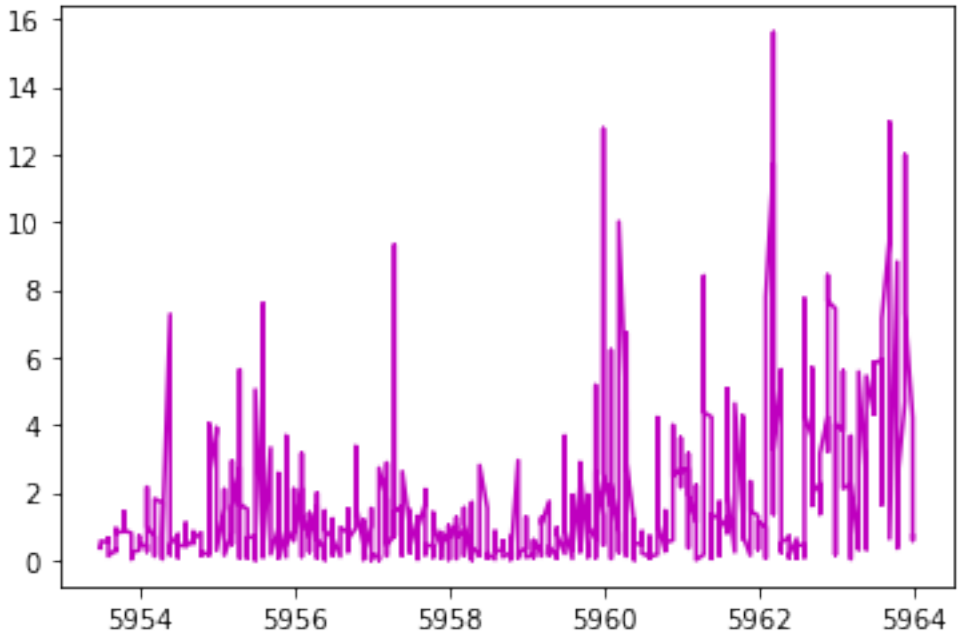

400
200
0
−200
5894
5896
5898
5900
5902
5904
5906

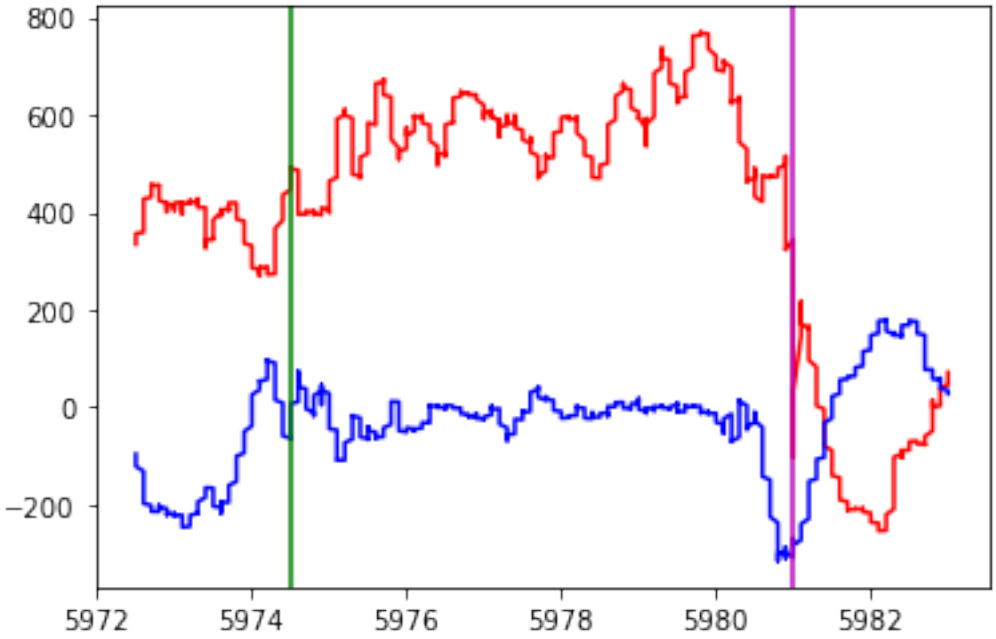

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
5894
5896
5898
5900
5902
5904
5906

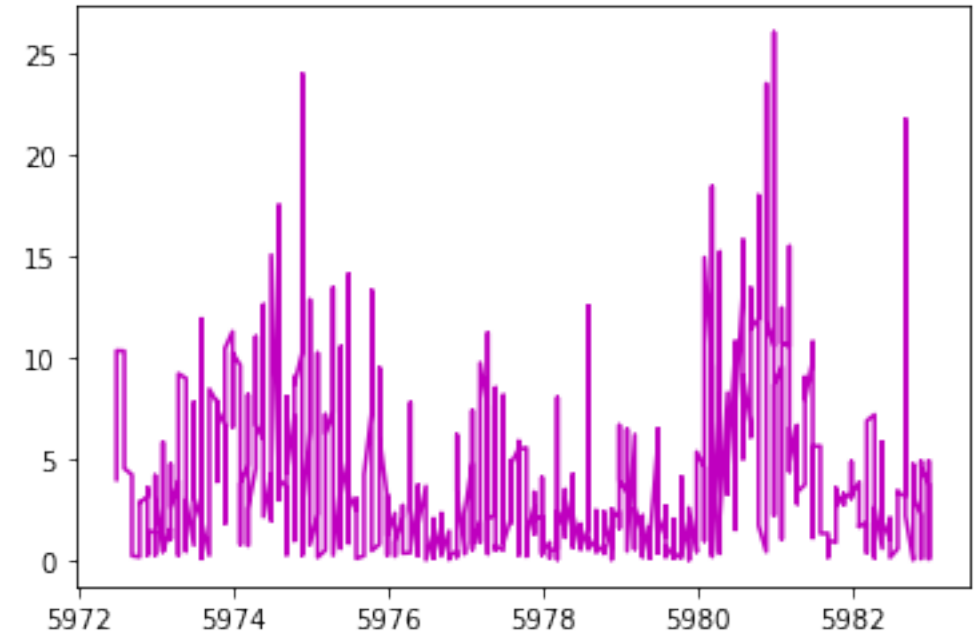

800
600
400
200
0
−200
−400
5926
5928
5930
5932
5934
5936

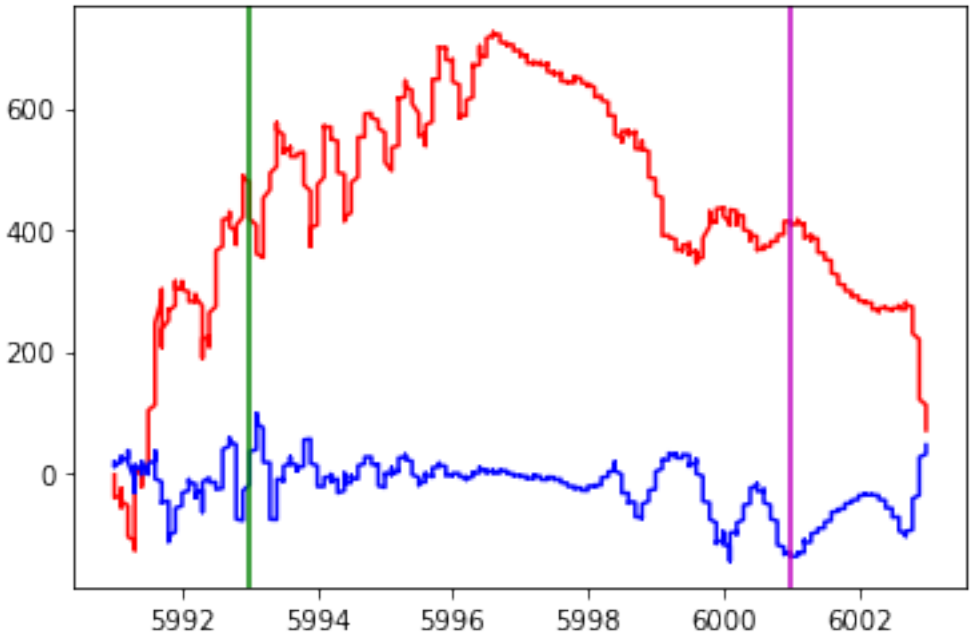

40
35
30
25
20
15
10
5
0
5926
5928
5930
5932
5934
5936

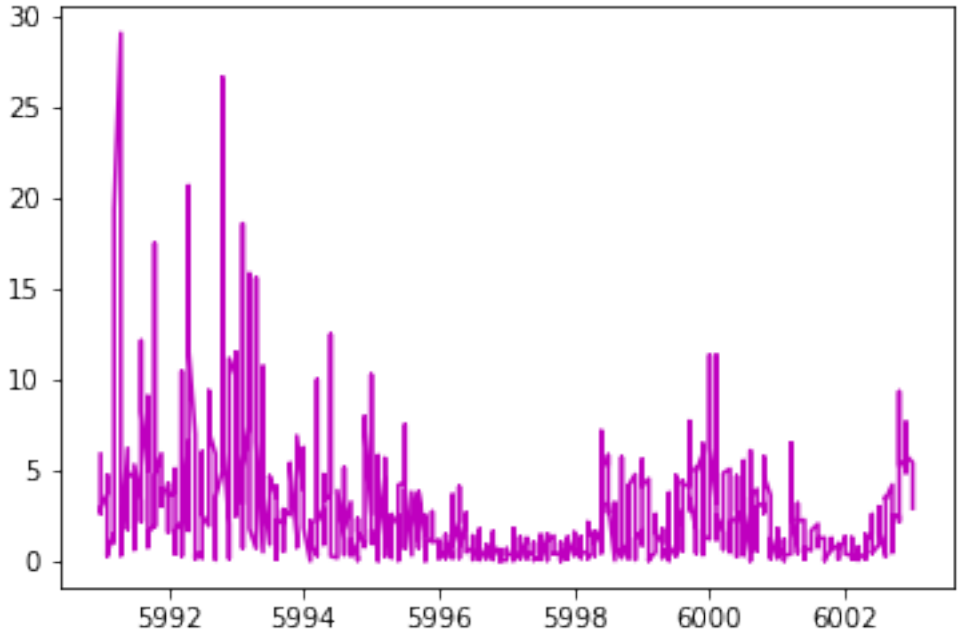

600
500
400
300
200
100
0
−100
5954
5956
5958
5960
5962
5964

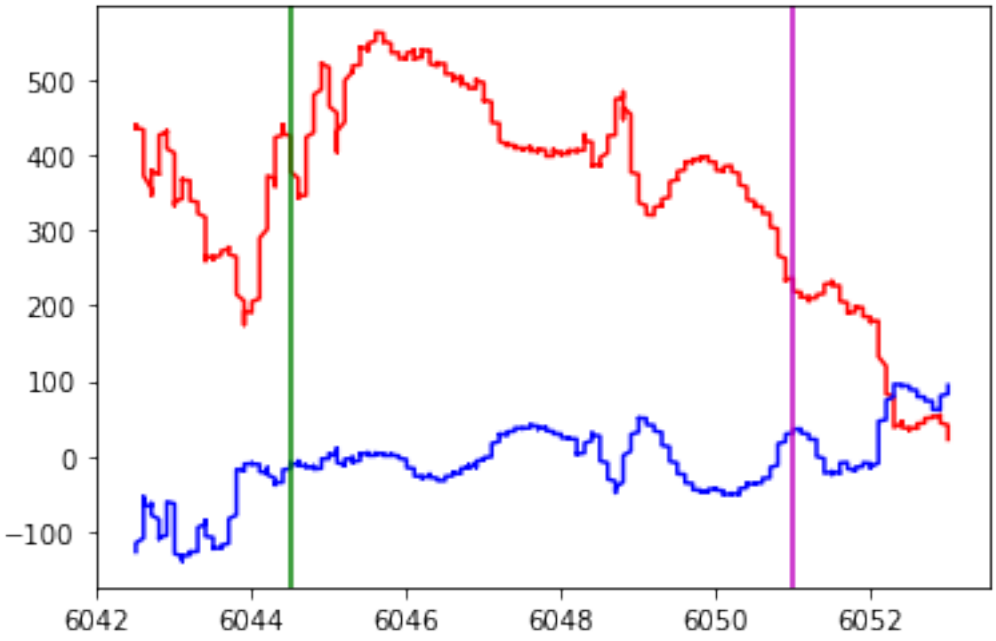

16
14
12
10
8
6
4
2
0
5954
5956
5958
5960
5962
5964

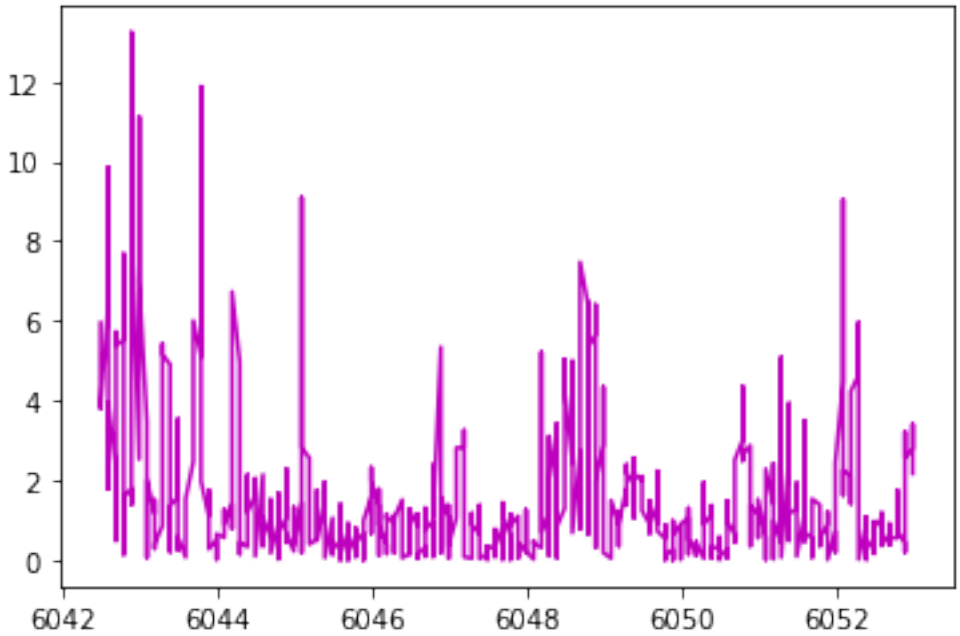

800
600
400
200
0
−200
5972
5974
5976
5978
5980
5982

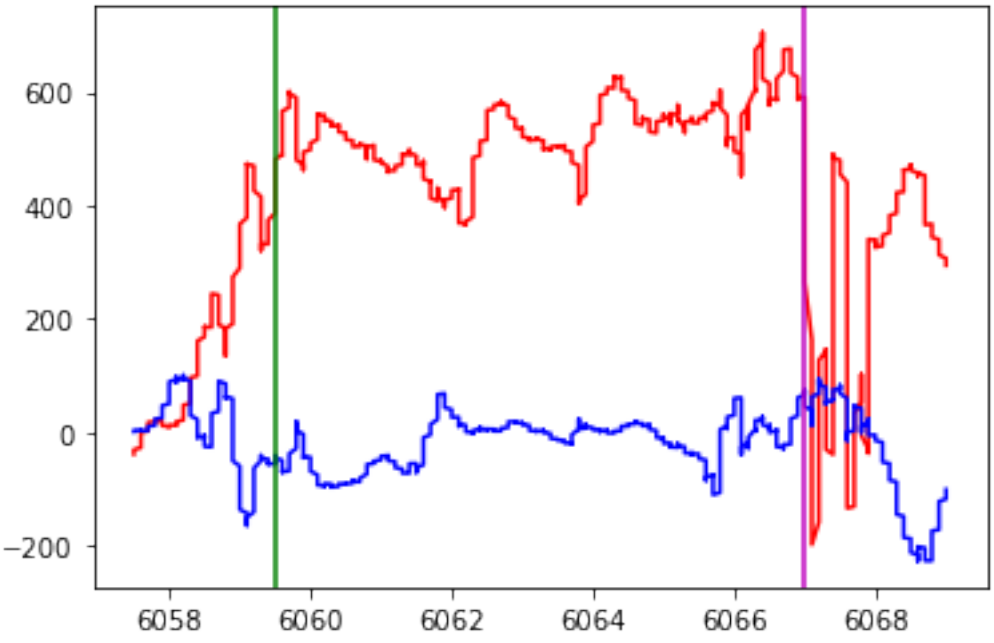

25
20
15
10
5
0
5972
5974
5976
5978
5980
5982

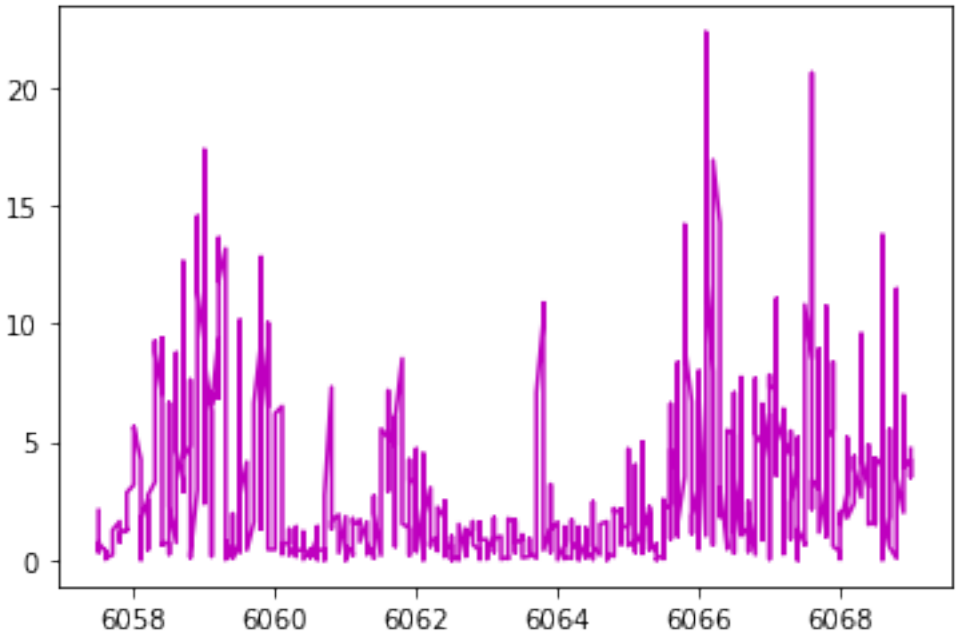

600
400
200
0
5992
5994
5996
5998
6000
6002

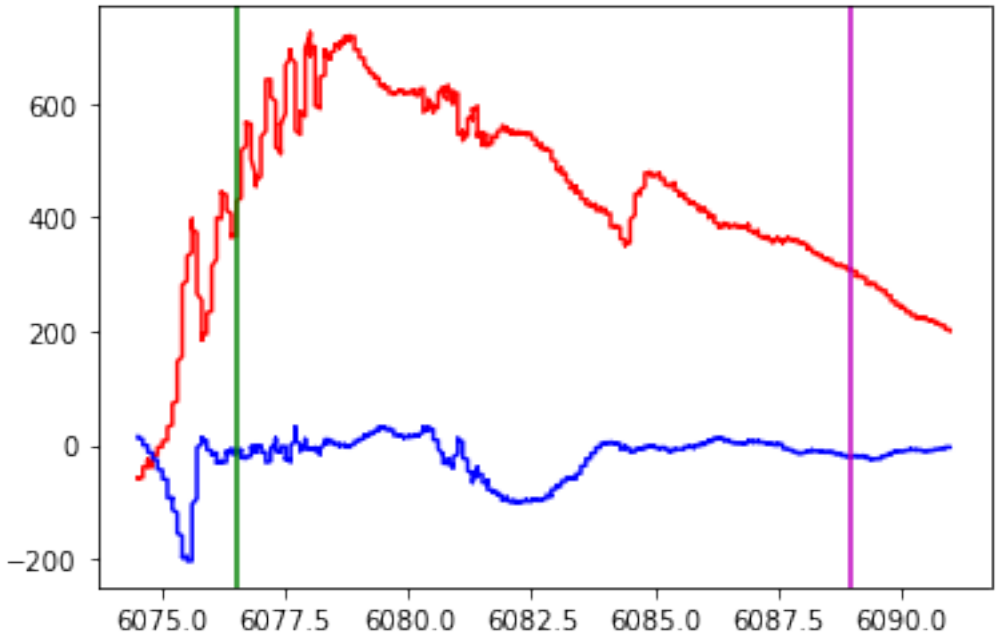

30
25
20
15
10
5
0
5992
5994
5996
5998
6000
6002

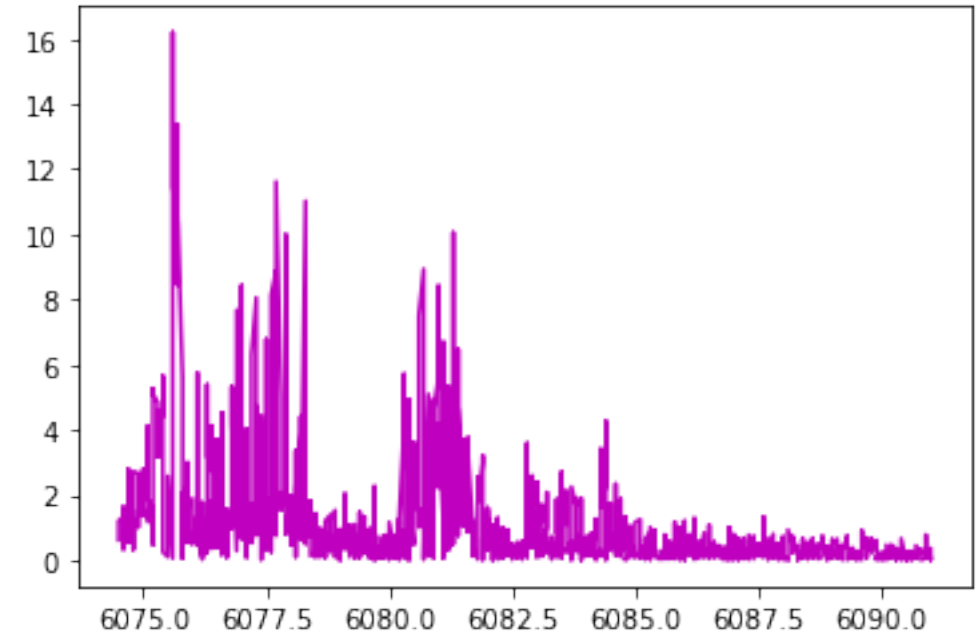

500
400
300
200
100
0
−100
6042
6044
6046
6048
6050
6052

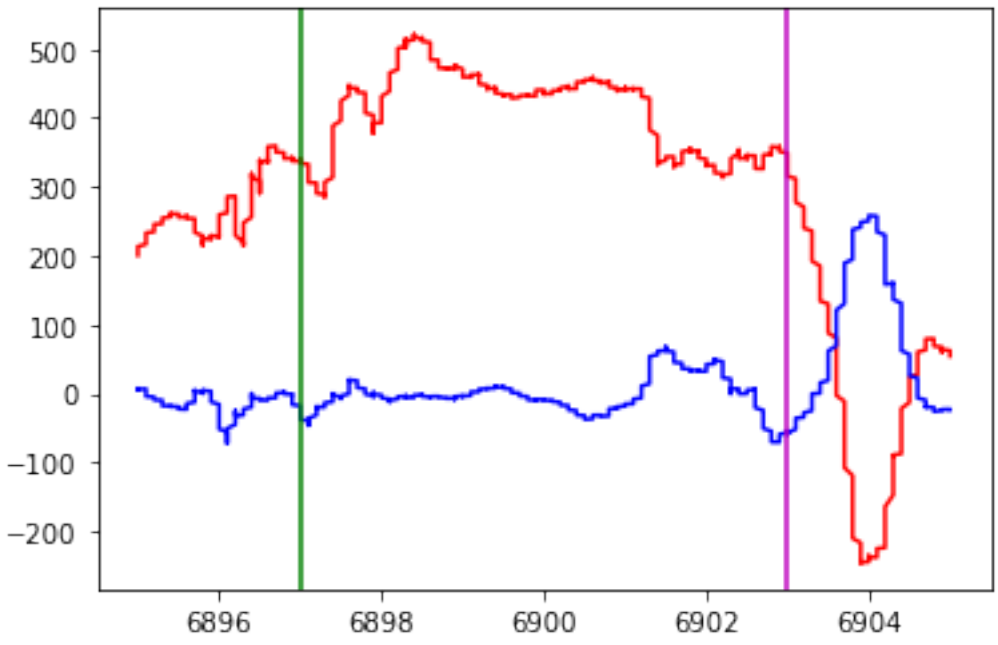

12
10
8
6
4
2
0
6042
6044
6046
6048
6050
6052

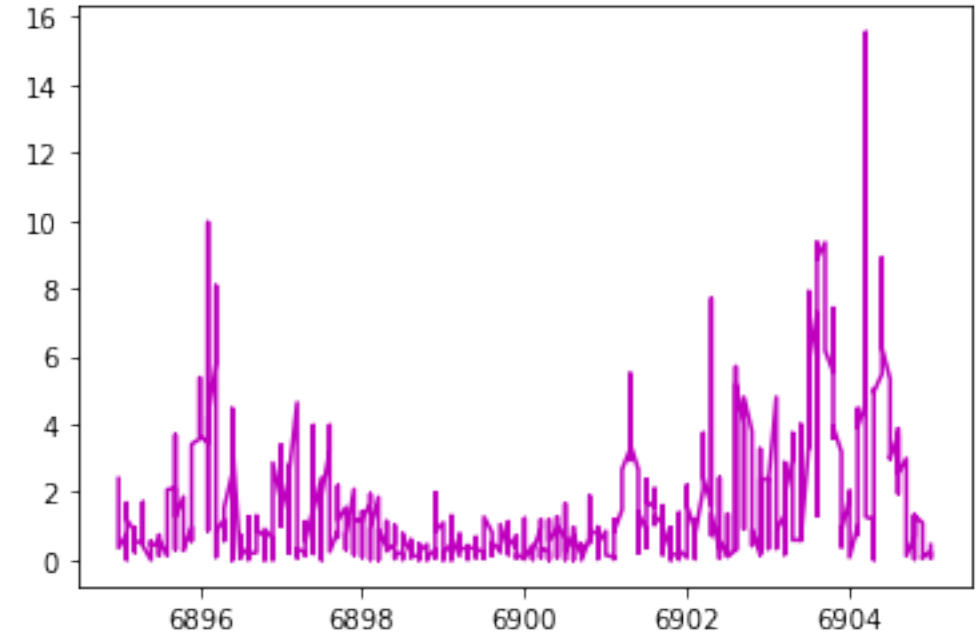

600
400
200
0
−200
6058
6060
6062
6064
6066
6068

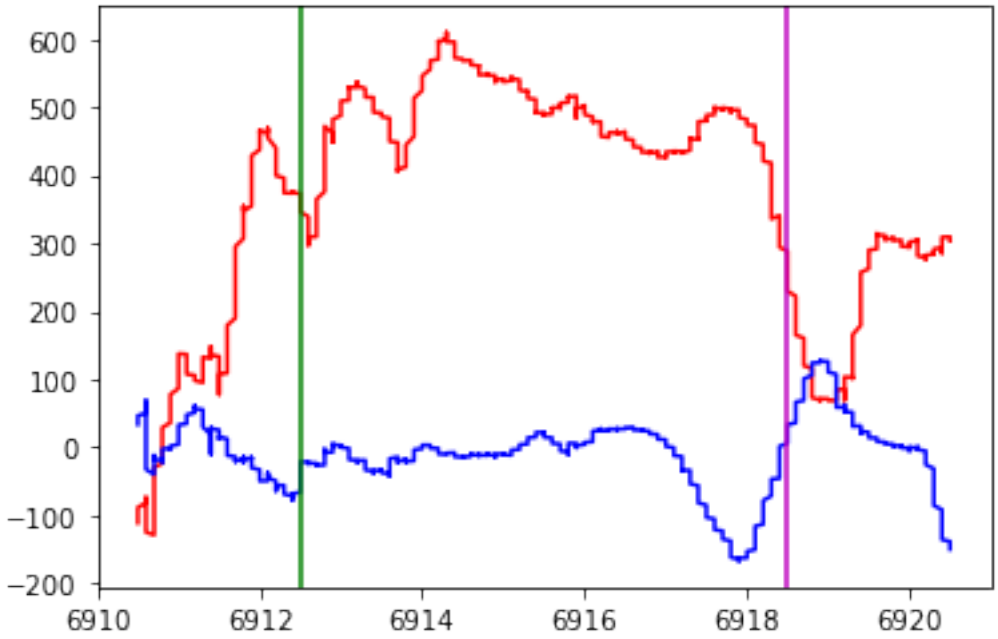

20
15
10
5
0
6058
6060
6062
6064
6066
6068

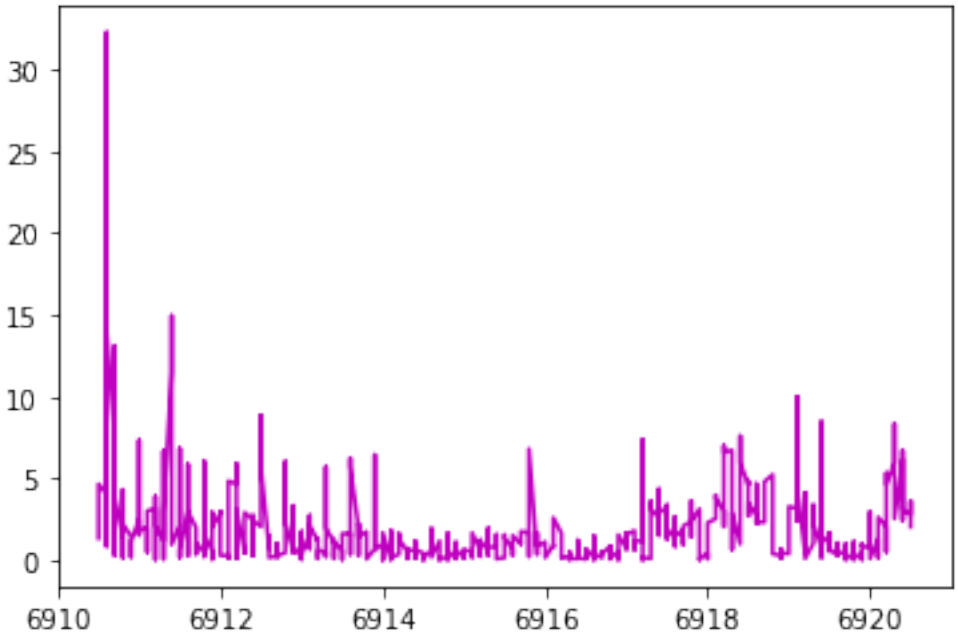

600
400
200
0
−200
6075.0
6077.5
6080.0
6082.5
6085.0
6087.5
6090.0

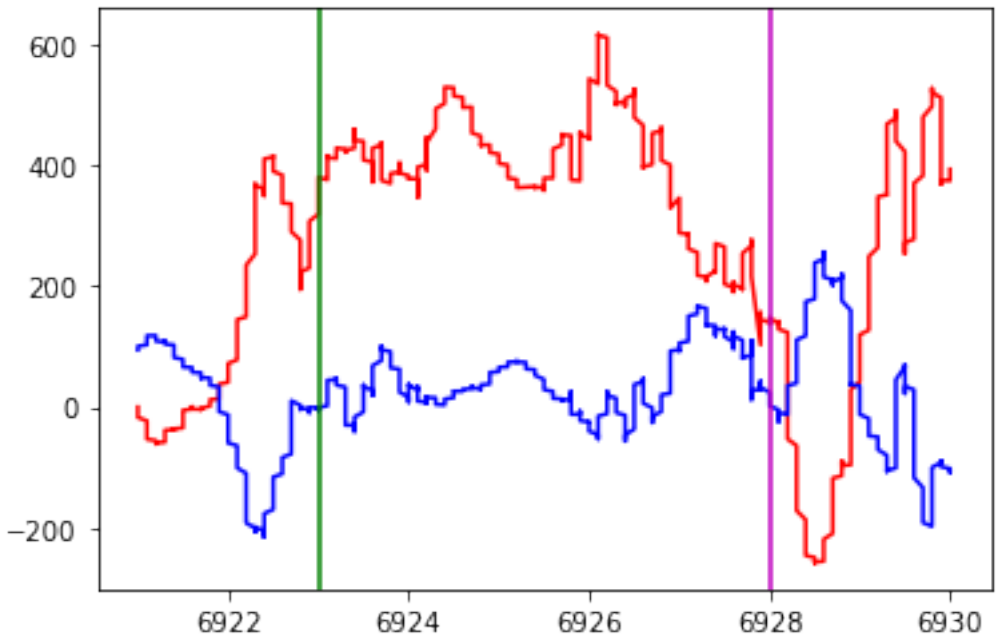

16
14
12
10
8
6
4
2
0
6075.0
6077.5
6080.0
6082.5
6085.0
6087.5
6090.0

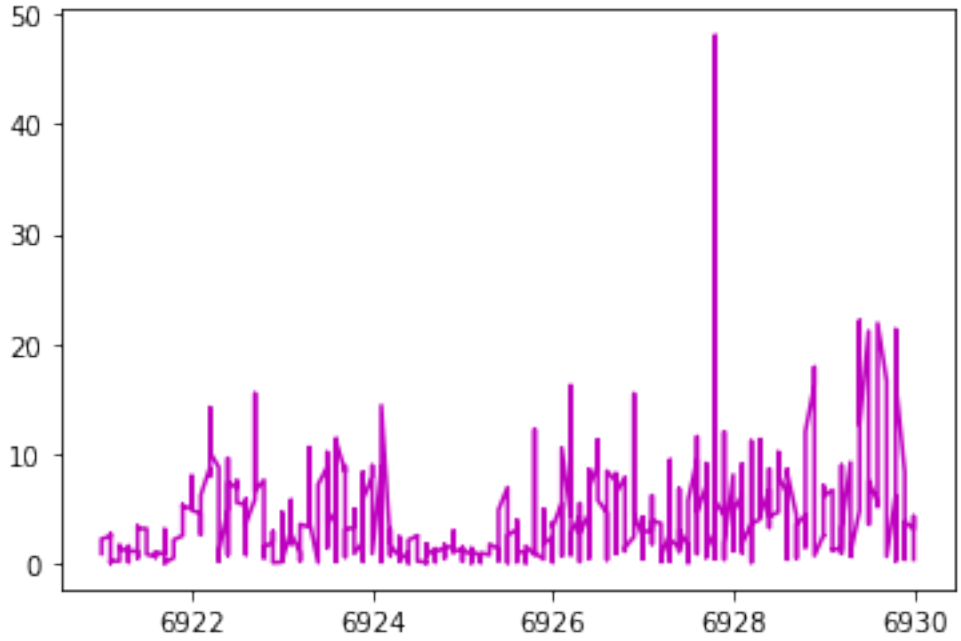

500
400
300
200
100
0
−100
−200
6896
6898
6900
6902
6904

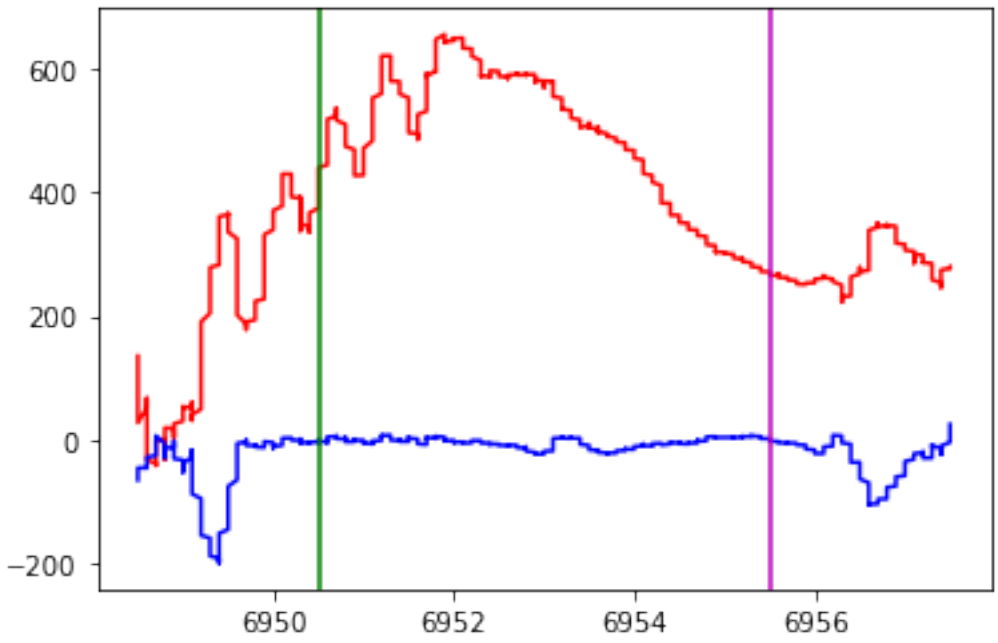

16
14
12
10
8
6
4
2
0
6896
6898
6900
6902
6904

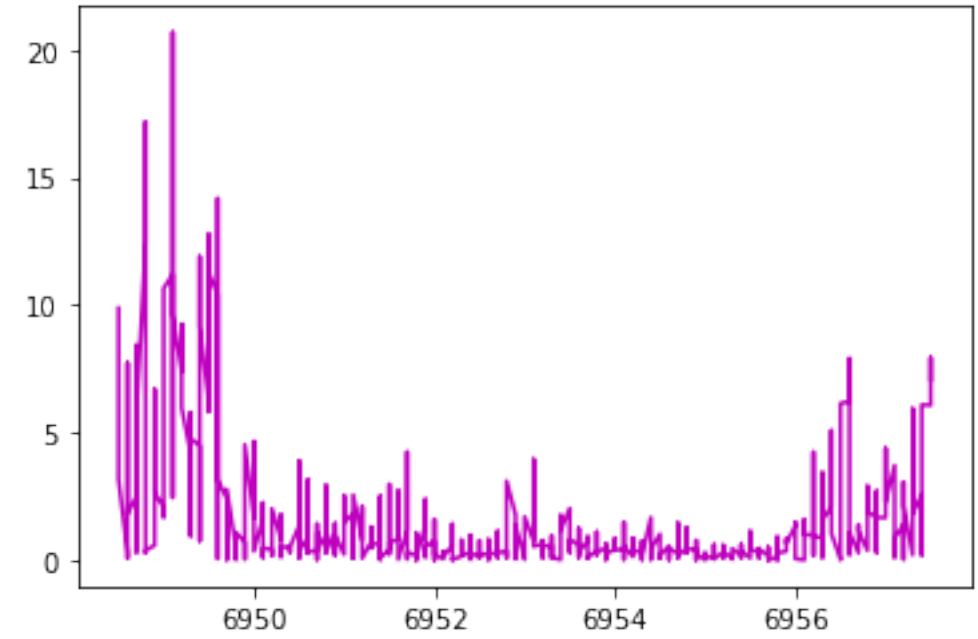

600
500
400
300
200
100
0
−100
−200
6910
6912
6914
6916
6918
6920

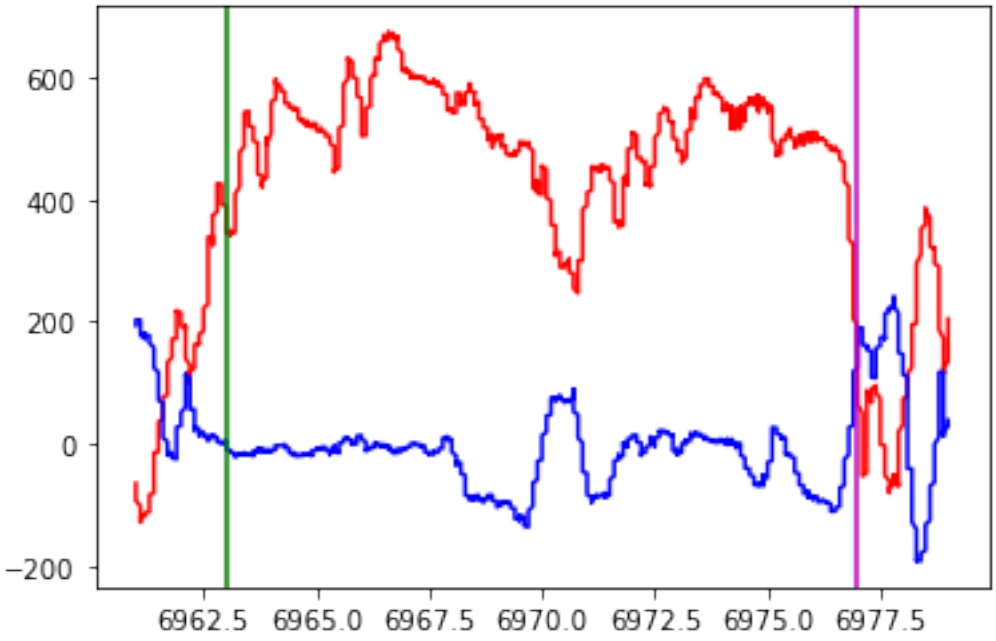

30
25
20
15
10
5
0
6910
6912
6914
6916
6918
6920

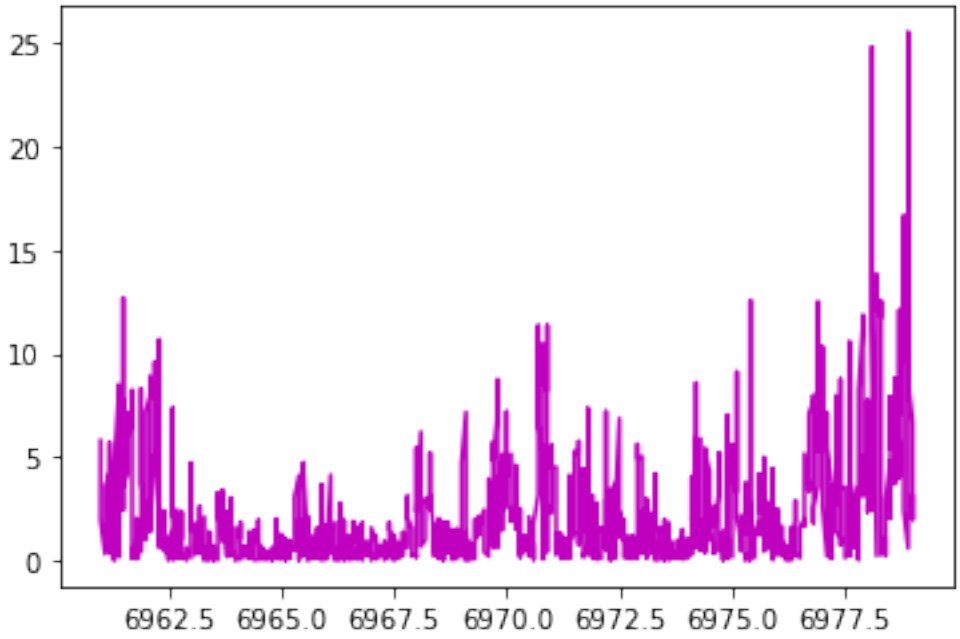

600
400
200
0
−200
6922
6924
6926
6928
6930

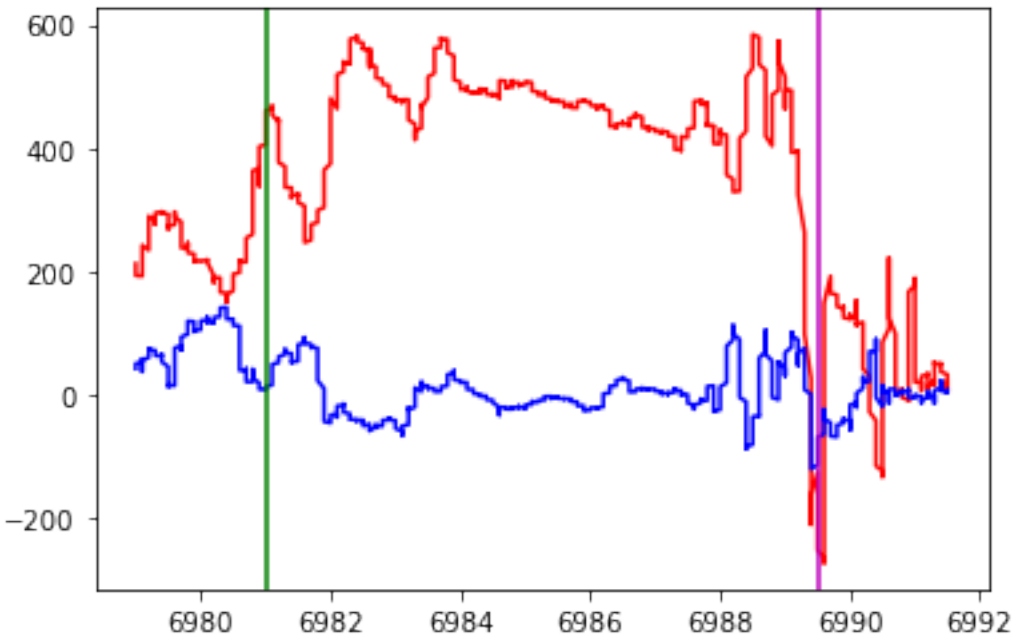

50
40
30
20
10
0
6922
6924
6926
6928
6930

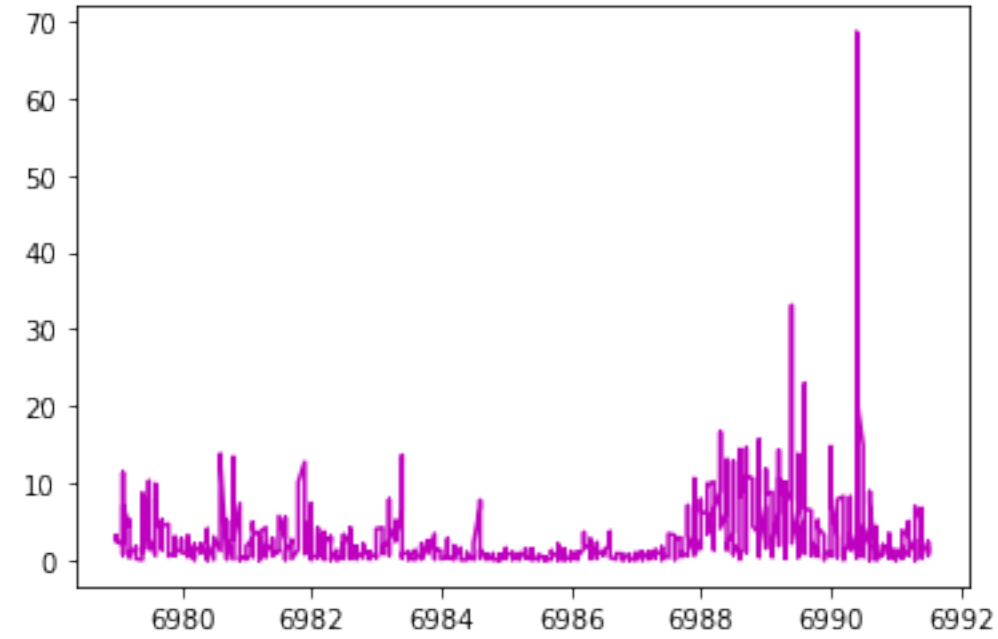

600
400
200
0
−200
6950
6952
6954
6956

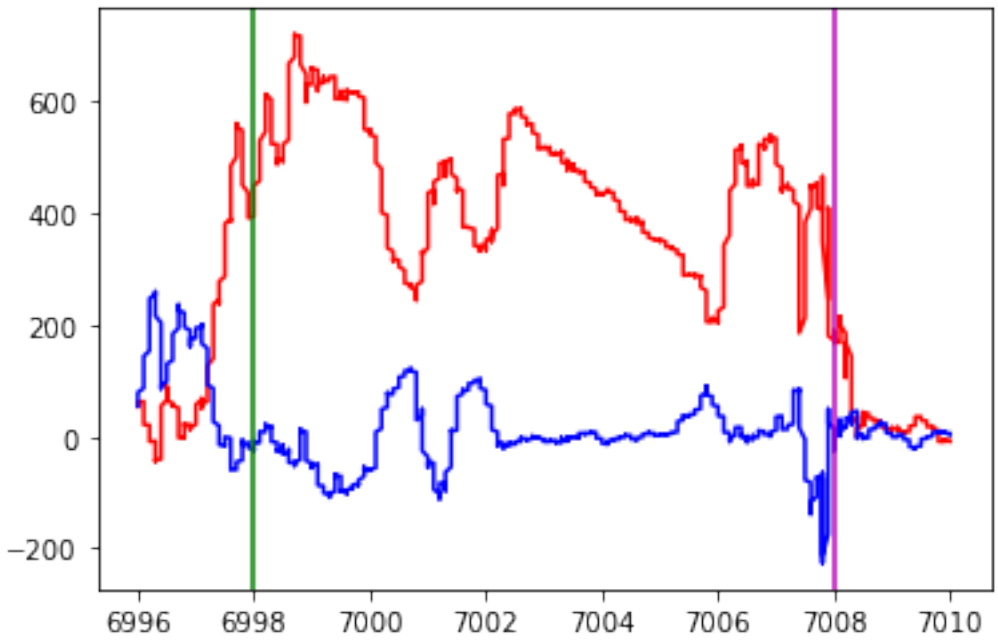

20
15
10
5
0
6950
6952
6954
6956

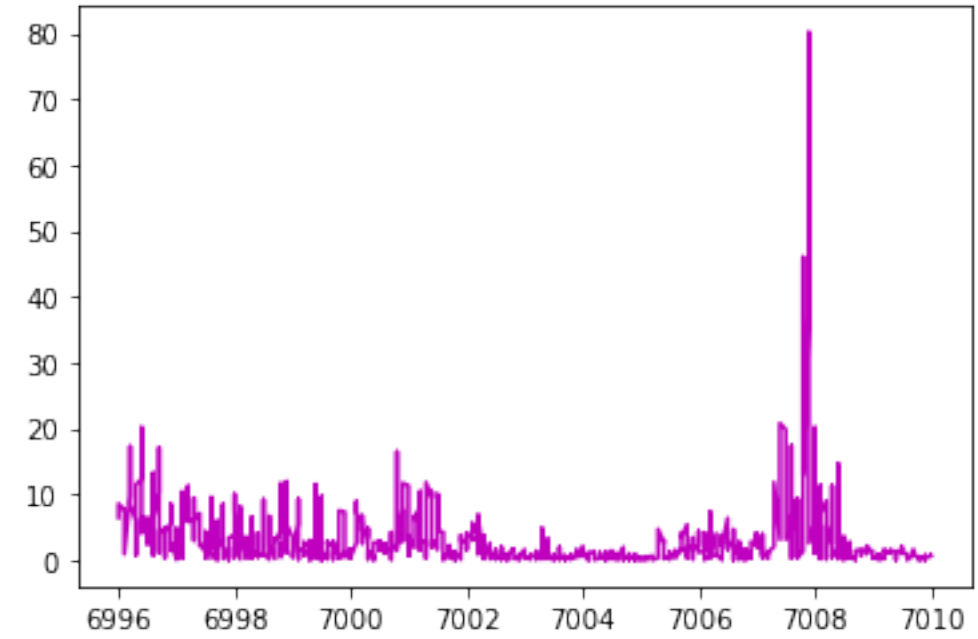

600
400
200
0
−200
6962.5
6965.0
6967.5
6970.0
6972.5
6975.0
6977.5

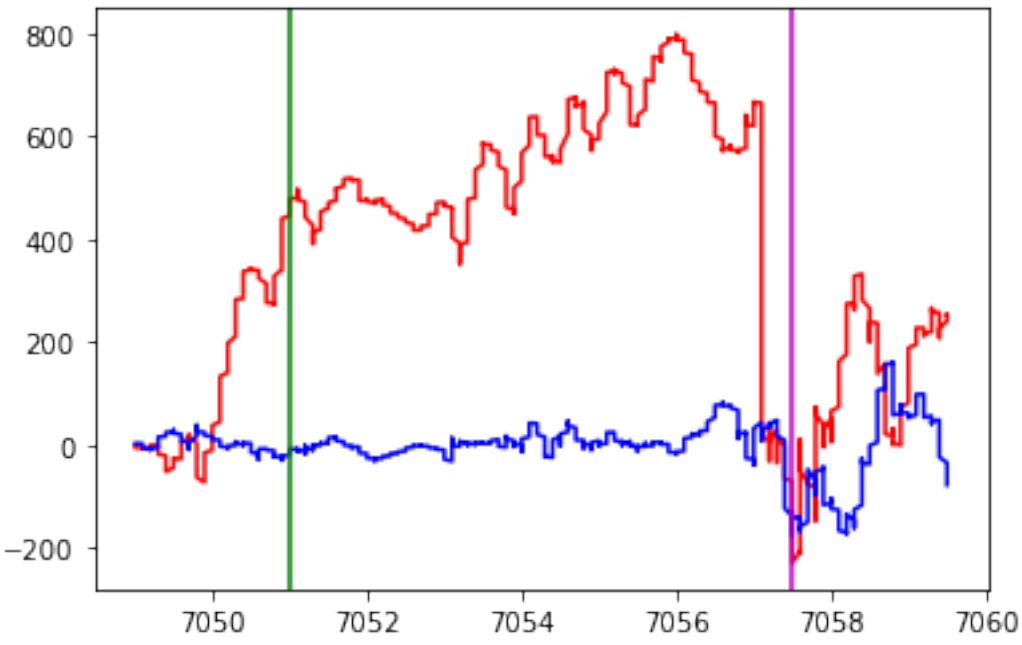

25
20
15
10
5
0
6962.5
6965.0
6967.5
6970.0
6972.5
6975.0
6977.5

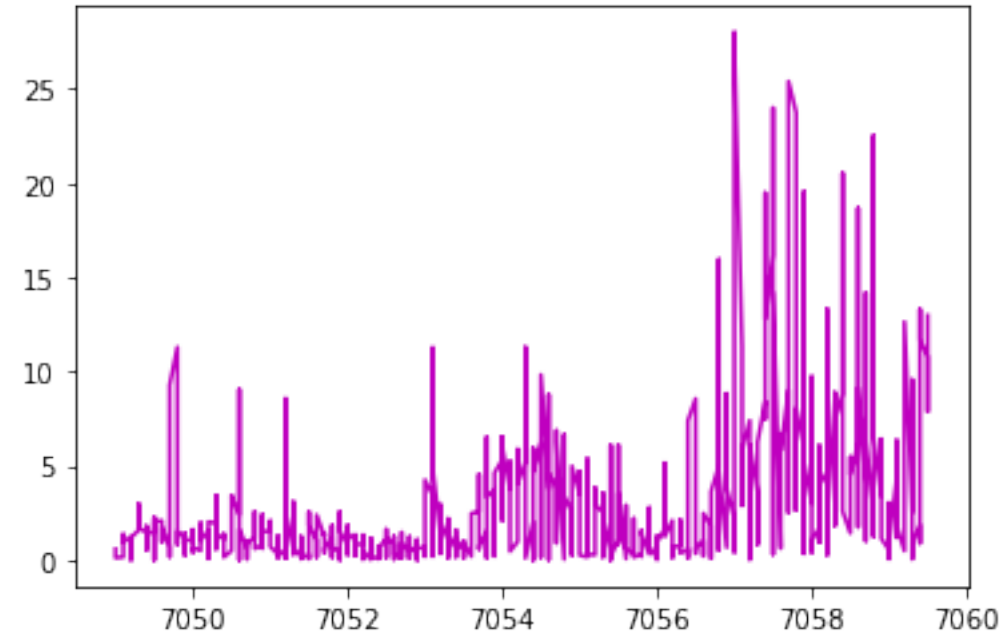

600
400
200
0
−200
6980
6982
6984
6986
6988
6990
6992

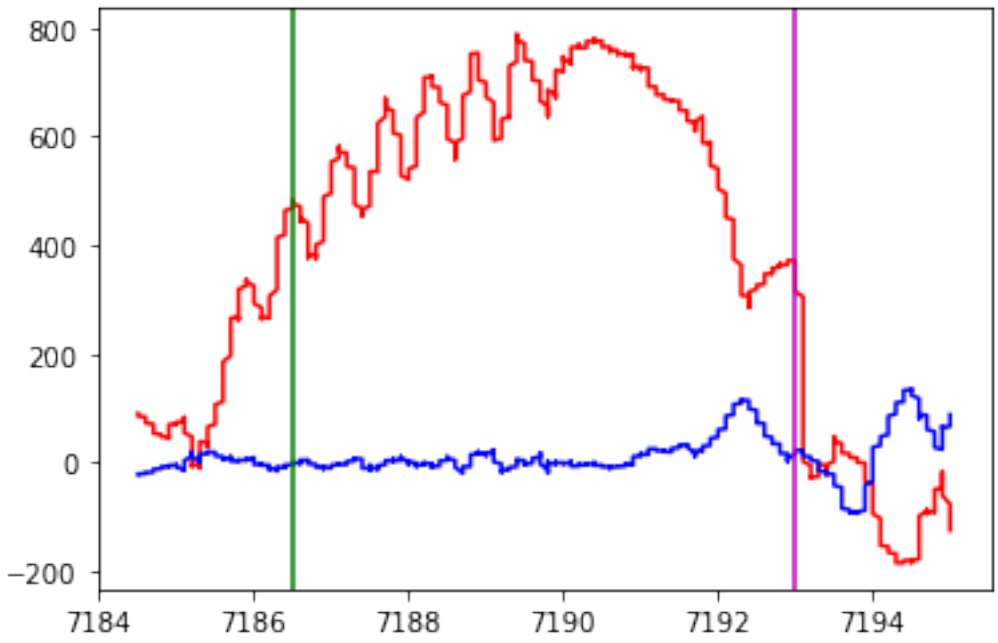

70
60
50
40
30
20
10
0
6980
6982
6984
6986
6988
6990
6992

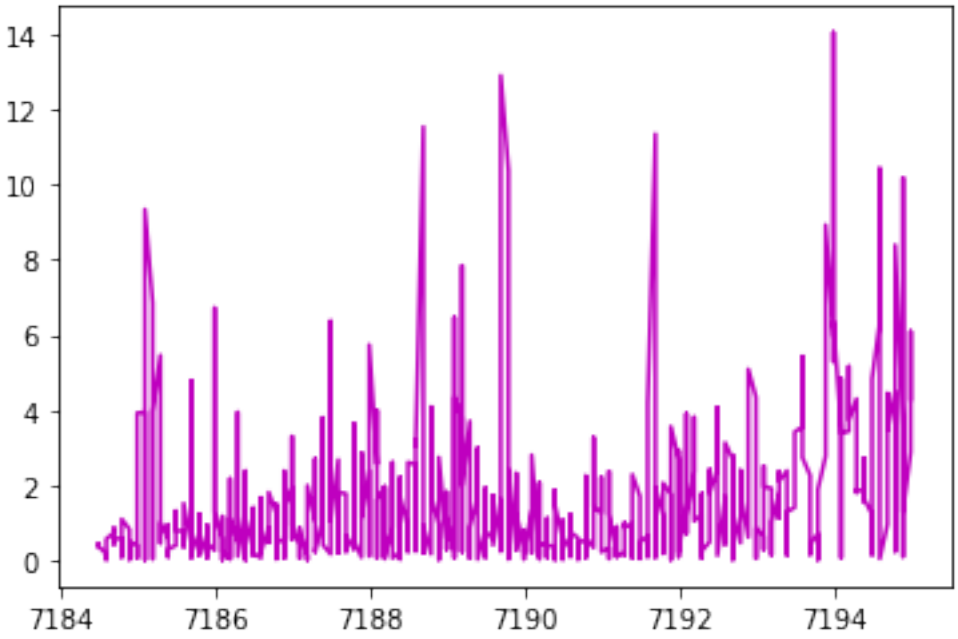

600
400
200
0
−200
6996
6998
7000
7002
7004
7006
7008
7010

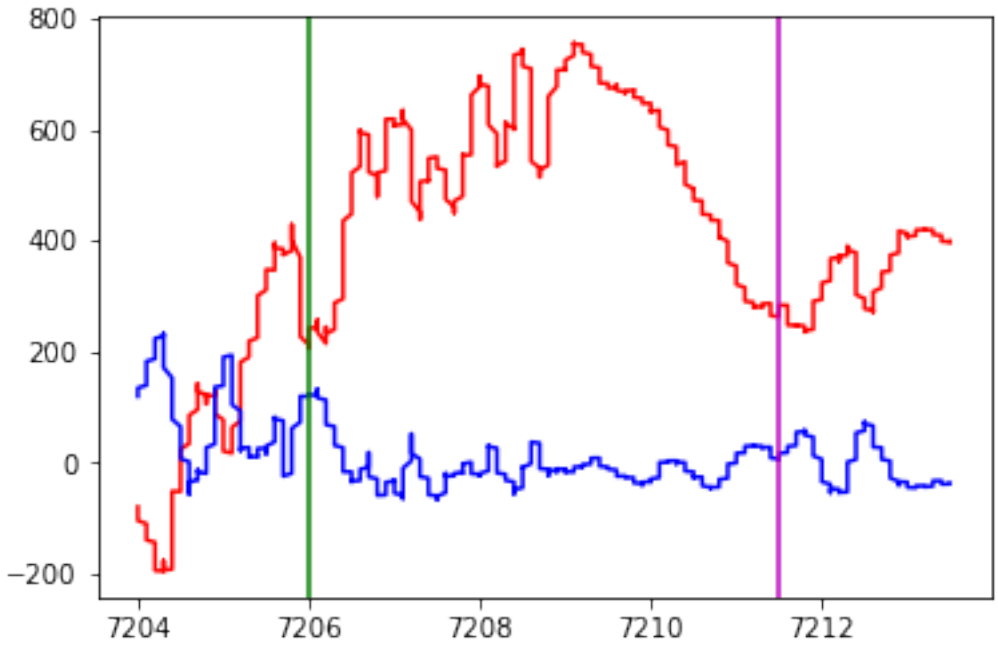

80
70
60
50
40
30
20
10
0
6996
6998
7000
7002
7004
7006
7008
7010

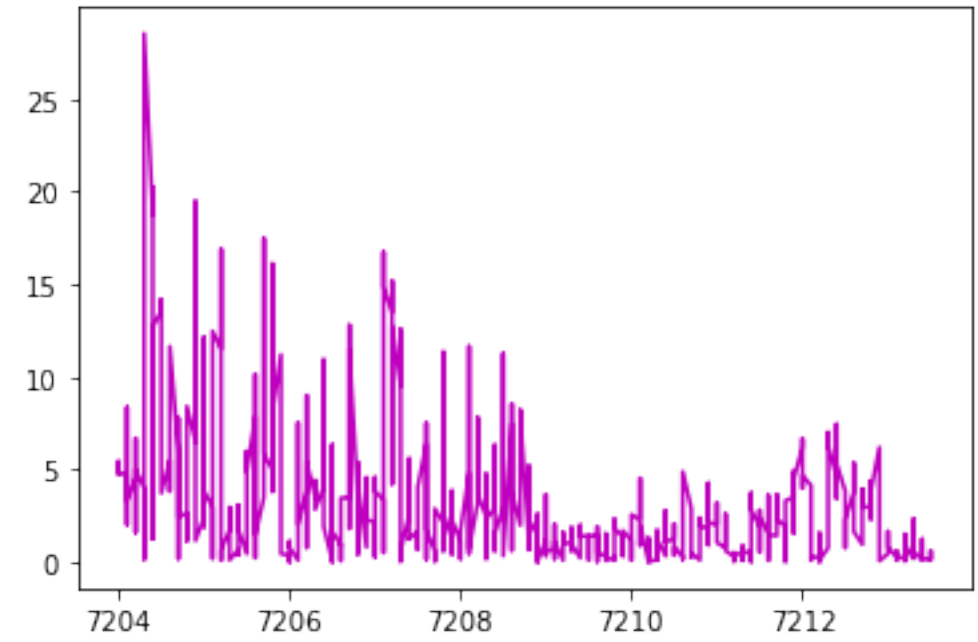

800
600
400
200
0
−200
7050
7052
7054
7056
7058
7060

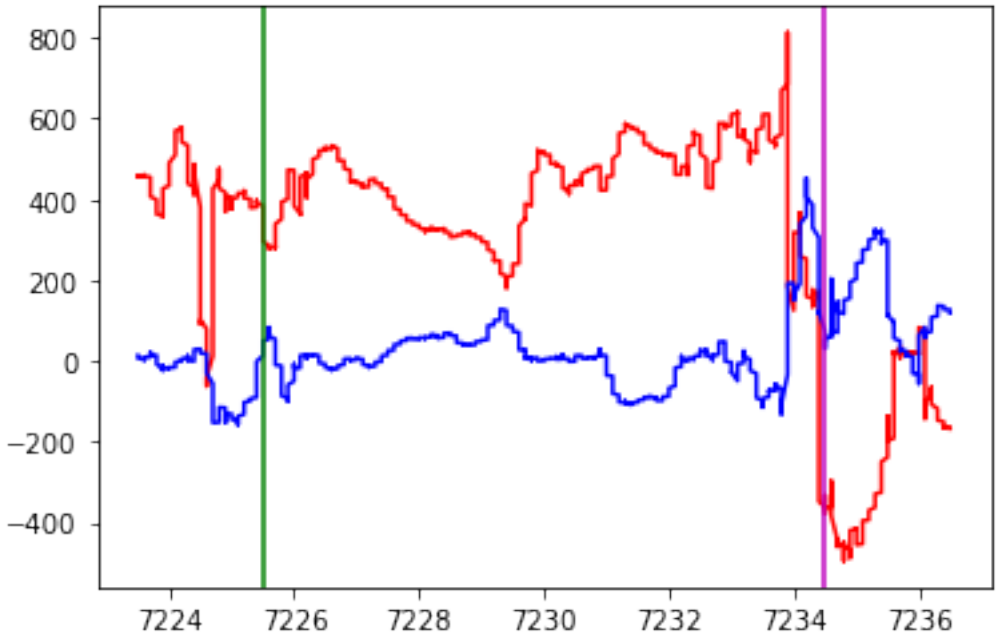

25
20
15
10
5
0
7050
7052
7054
7056
7058
7060

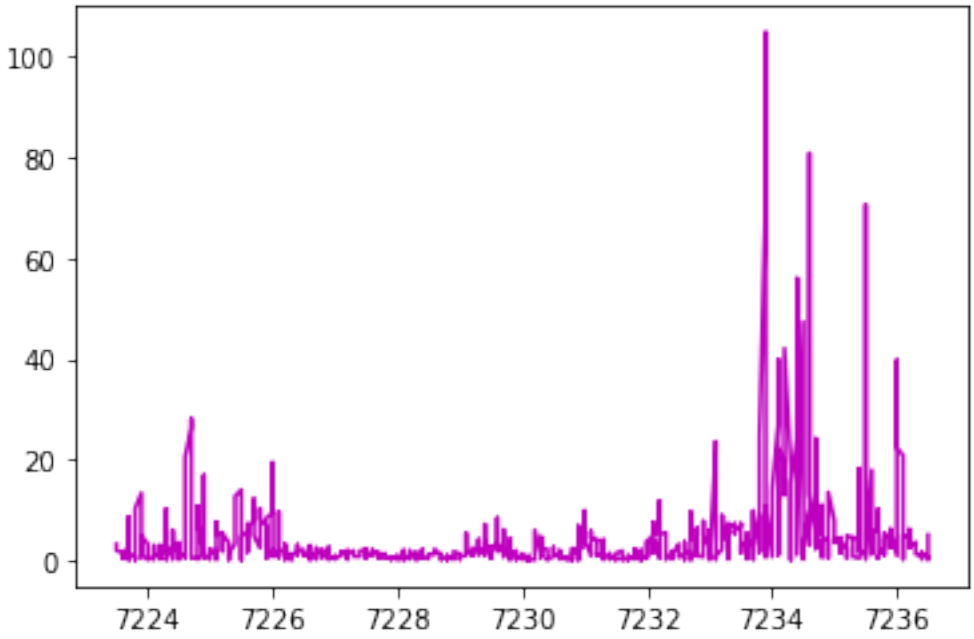

800
600
400
200
0
−200
7184
7186
7188
7190
7192
7194

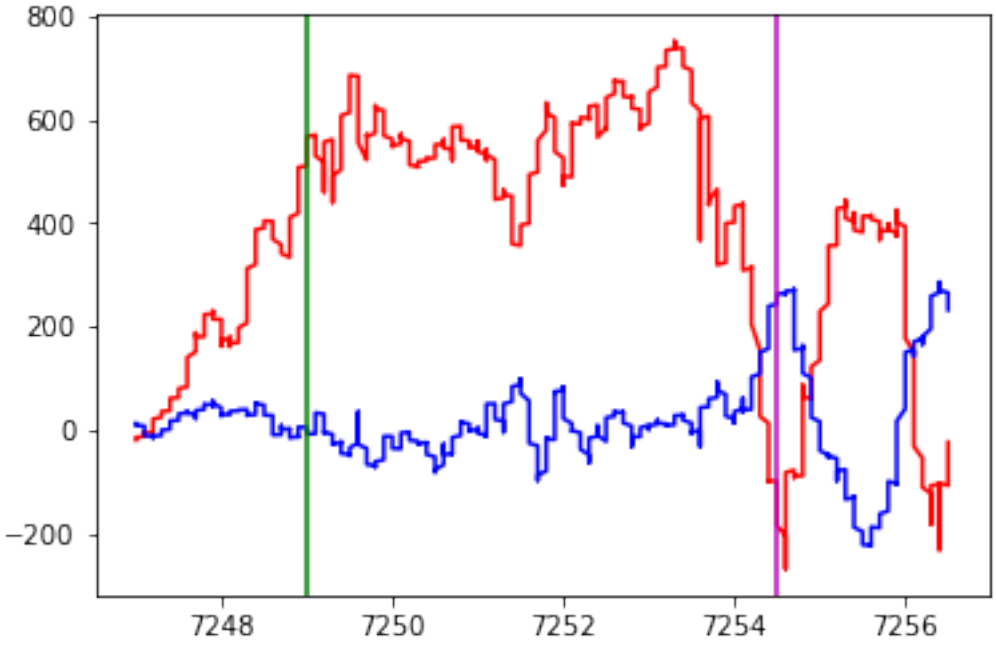

14
12
10
8
6
4
2
0
7184
7186
7188
7190
7192
7194

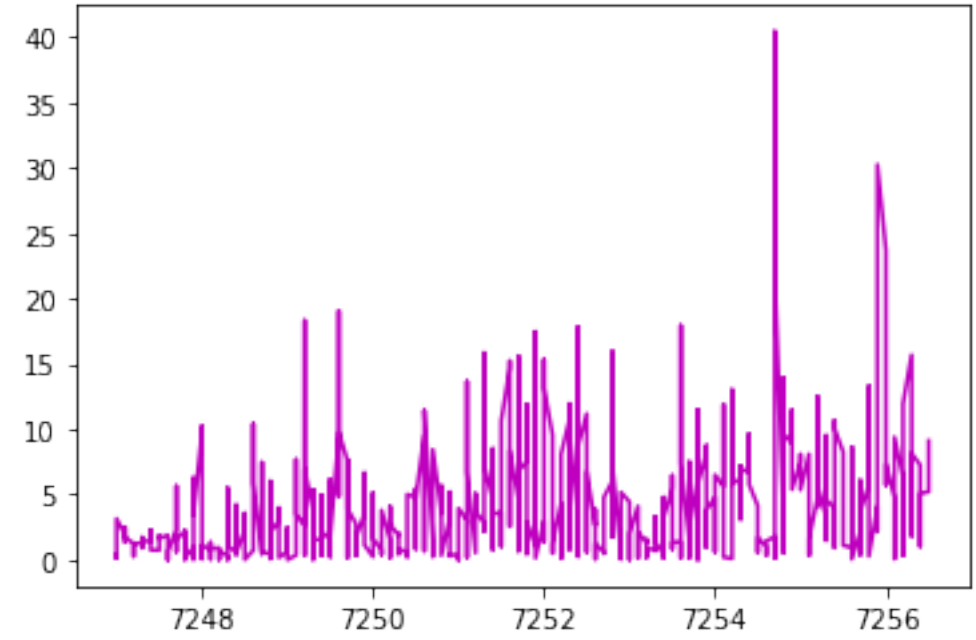

800
600
400
200
0
−200
7204
7206
7208
7210
7212

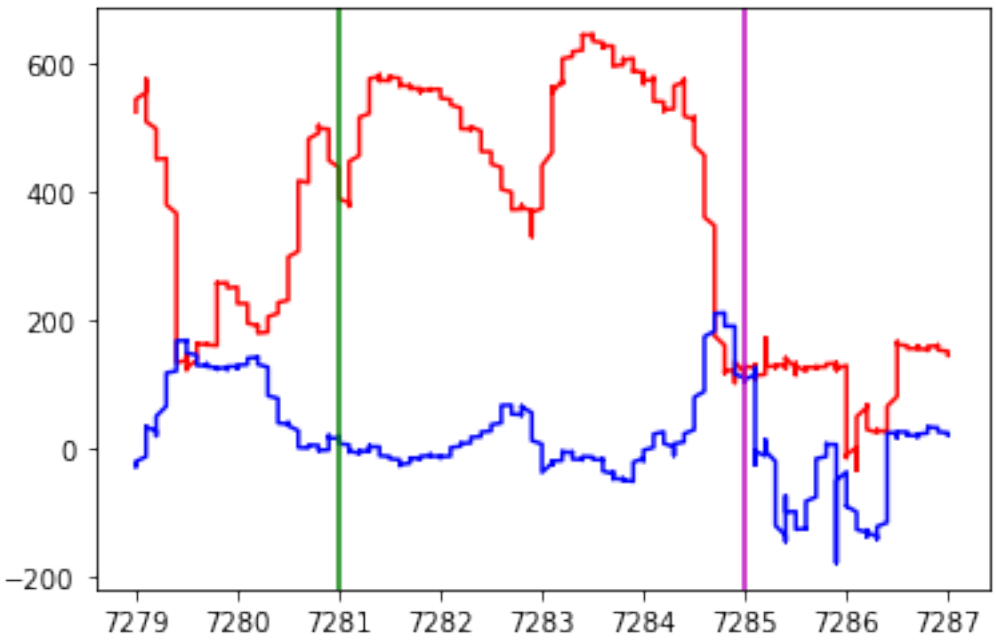

25
20
15
10
5
0
7204
7206
7208
7210
7212

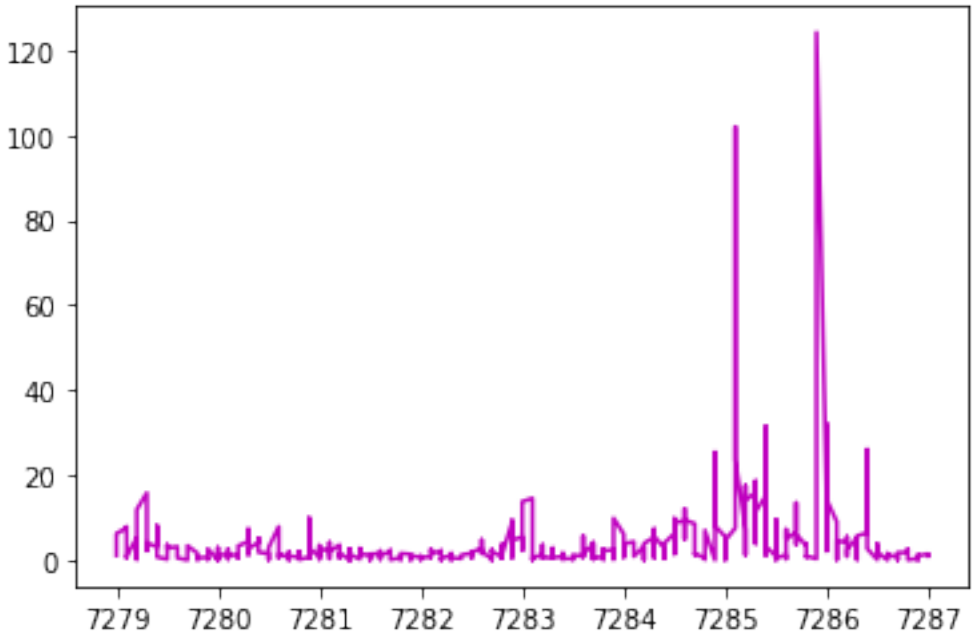

800
600
400
200
0
−200
−400
7224
7226
7228
7230
7232
7234
7236


100
80
60
40
20
0
7224
7226
7228
7230
7232
7234
7236


800
600
400
200
0
−200
7248
7250
7252
7254
7256


40
35
30
25
20
15
10
5
0
7248
7250
7252
7254
7256





[25]:
```
Starting =[]
Stopping = []

df_Filter = df_Player
```

```
df_Filter = df_Filter.fillna(0)

Order = 5
cutoff_freq = 1.5
sampling_freq = 100
sampling_duration = len(df_Filter.wheelRotationalSpeedX)/100


normalized_cutoff_freq = 2 * cutoff_freq / sampling_freq
numerator_coeffs, denominator_coeffs = signal.butter(Order,␣
 ↪normalized_cutoff_freq)

filtered_WheelRotationspeedX = signal.lfilter(numerator_coeffs,␣
 ↪denominator_coeffs, df_Filter.wheelRotationalSpeedX)
filtered_FrameRotationspeedZ = signal.lfilter(numerator_coeffs,␣
 ↪denominator_coeffs, df_Filter.frameRotationalSpeedZ)


df_Player['Sum_WheelX_FrameZ']  = df_Player.wheelRotationalSpeedX + df_Player.
 ↪frameRotationalSpeedZ
df_Player['Div_FrameZ_WheelX']  = df_Player.frameRotationalSpeedZ / df_Player.
 ↪wheelRotationalSpeedX
df_Player['Filt_WheelX'] = filtered_WheelRotationspeedX
df_Player['Filt_FrameZ'] = filtered_FrameRotationspeedZ

print(Startsprint, Stopsprint)

df_Player['Action'] = ""

for i  in range(0,len(Startsprint)):
    df_Player['Action'].iloc[int(Startsprint[i]*100):int(Stopsprint[i]*100)] = 1


df_Player.to_csv('matrix_Player_' + str(Player) + '_game_' + str(Game) +␣
 ↪'_All_Action.csv')
```

```
[997.01, 1019.0, 1041.5, 1062.5, 1078.0, 1086.0, 1093.5, 1105.5, 1138.0, 1161.0,
1181.5, 1201.0, 1232.5, 1248.5, 1257.0, 1427.5, 1485.5, 1511.5, 1538.0, 1566.0,
1593.0, 1626.5, 1659.5, 1676.0, 1734.0, 1832.0, 1912.0, 2216.5, 2224.5, 2235.5,
2433.5, 2465.5, 2490.0, 2558.0, 2598.0, 2631.0, 2647.0, 2671.5, 2684.0, 2786.0,
2800.0, 2850.0, 2873.0, 2888.5, 2940.0, 2956.5, 3004.5, 3025.5, 3049.0, 3074.0,
3093.5, 3118.5, 3137.5, 3156.0, 3256.0, 3276.0, 3329.5, 3372.0, 3506.5, 3530.0,
3555.5, 3602.5, 3628.0, 3654.0, 3685.0, 3703.5, 5040.0, 5053.5, 5080.5, 5095.0,
5135.5, 5146.5, 5177.0, 5207.0, 5324.0, 5345.5, 5369.0, 5400.5, 5411.0, 5434.0,
5462.0, 5510.5, 5594.5, 5621.0, 5704.0, 5745.5, 5842.5, 5895.0, 5927.5, 5955.5,
5974.5, 5993.0, 6044.5, 6059.5, 6076.5, 6897.0, 6912.5, 6923.0, 6950.5, 6963.0,
6981.0, 6998.0, 7051.0, 7186.5, 7206.0, 7225.5, 7249.0, 7281.0] [1008.0, 1041.0,
```

```
1047.0, 1074.0, 1083.5, 1093.0, 1103.0, 1121.0, 1148.5, 1181.0, 1186.5, 1219.5,
1246.0, 1254.0, 1262.0, 1433.5, 1490.5, 1516.5, 1543.5, 1571.0, 1598.0, 1632.0,
1665.5, 1681.5, 1739.0, 1837.0, 1917.5, 2221.5, 2231.5, 2241.5, 2441.0, 2471.0,
2495.0, 2566.5, 2607.5, 2640.0, 2654.5, 2680.0, 2689.5, 2795.0, 2807.0, 2856.0,
2879.0, 2894.0, 2946.5, 2963.5, 3011.5, 3031.0, 3057.0, 3081.0, 3100.0, 3126.0,
3144.0, 3162.5, 3264.5, 3283.0, 3335.0, 3378.0, 3513.5, 3536.0, 3563.5, 3611.5,
3634.5, 3661.0, 3691.5, 3708.5, 5046.0, 5079.0, 5092.5, 5132.5, 5146.0, 5172.0,
5201.0, 5227.5, 5333.0, 5353.0, 5377.0, 5405.5, 5419.5, 5440.0, 5470.5, 5519.0,
5604.5, 5626.5, 5710.5, 5755.5, 5848.5, 5903.5, 5934.0, 5962.0, 5981.0, 6001.0,
6051.0, 6067.0, 6089.0, 6903.0, 6918.5, 6928.0, 6955.5, 6977.0, 6989.5, 7008.0,
7057.5, 7193.0, 7211.5, 7234.5, 7254.5, 7285.0]

/opt/jupyterhub/anaconda/lib/python3.8/site-
packages/pandas/core/indexing.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

[26]:
```
print(len(Stoprotate))

print(len(Startrotate))
```

```
688
688
```

[ ]:
```

```