Computational Tools for Business Analytics

Assignment No1 - Flora Fyka - mpked2349

I am executing the 'analyze' function for each one of the three failure modes, utilizing the corresponding dataset and varying window sizes across different experiments.

The first two plots Sensor data and Log likelihood of changepoint in raw sensor data are the result of Test case num1 Bayesian Online Changepoint Detection on raw sensor data. (This group of plots correspond to Test Case 1)

The second two plots Kurtosis and Log likelihood of changepoint in Kurtosis are the result of Test case num1 Bayesian Online Changepoint Detection on the Kurtosis feature. (This group of plots correspond to Test Case 2)

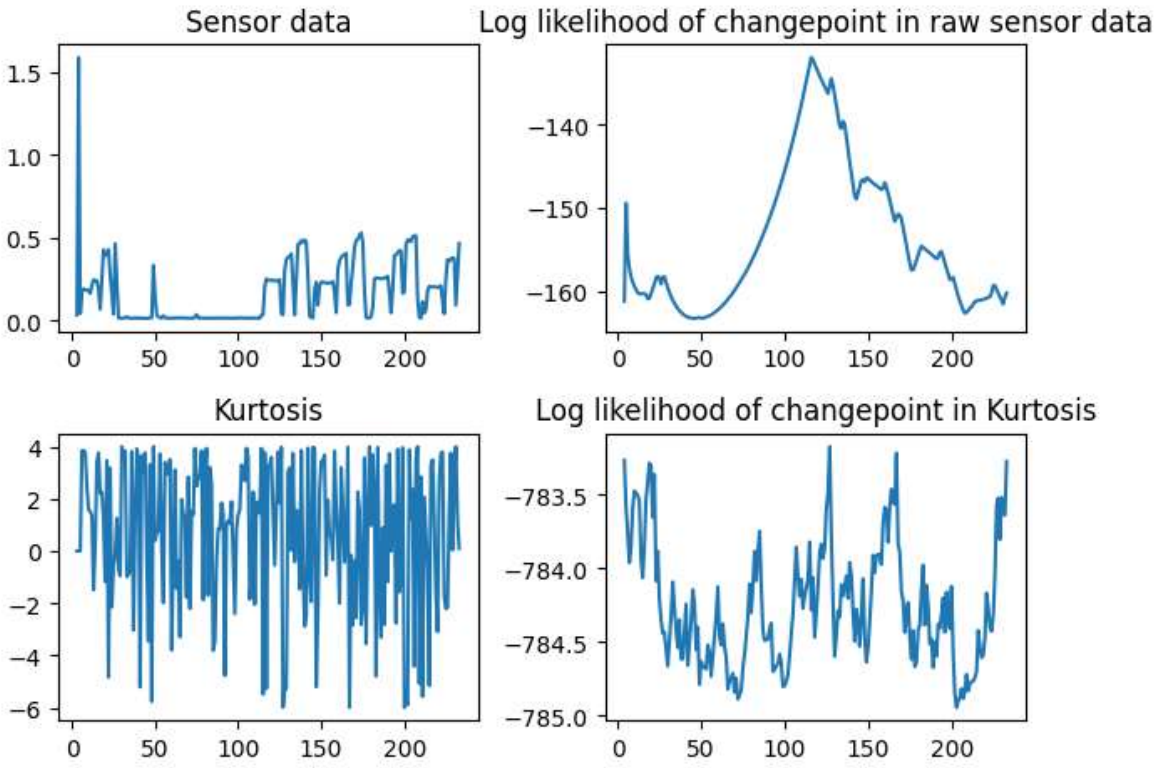
Each experiment features a different rolling window size. Summing up each dataset, I include comments, comparisons, and draw a conclusion.

In [1]:
```
import BayesianOnlineChangepointDetection as detect
```

Crash

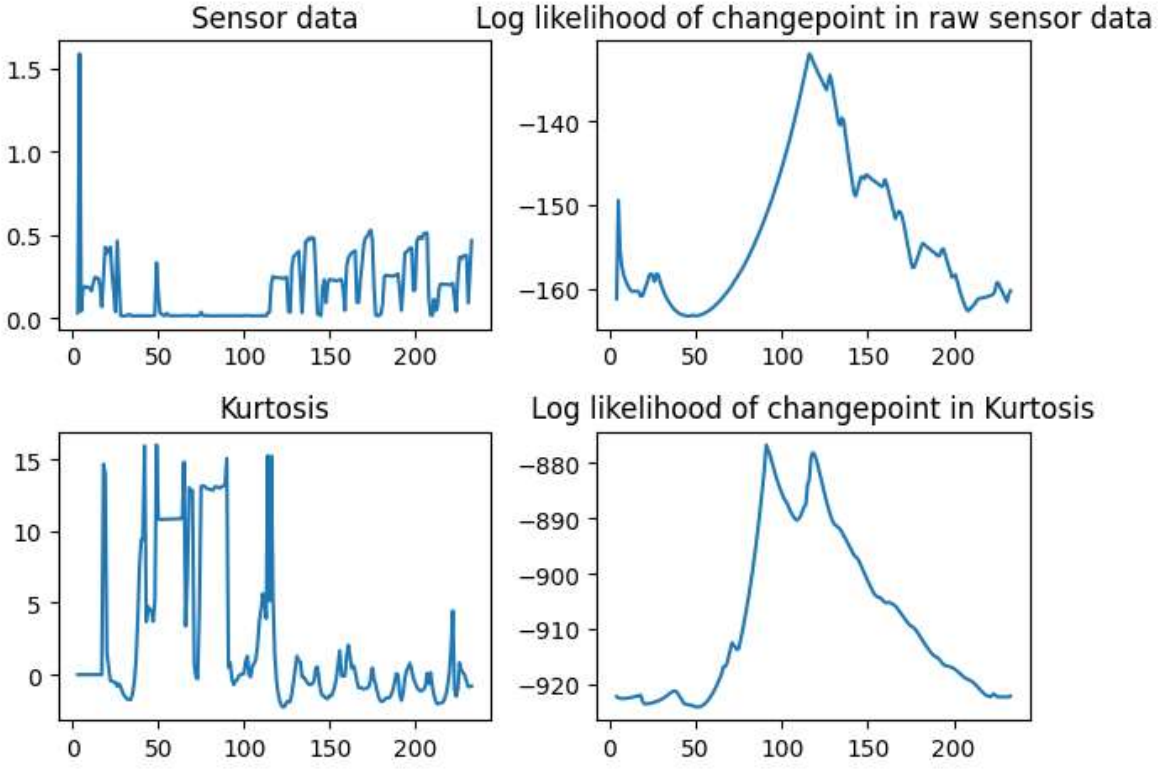
Dataset Num1

In [2]:
```
detect.analyze("Datasets/Frakarisma1.csv", 4)
```




```
Changepoint time: 112
Alert at time: 123
```

In [3]:
```
detect.analyze("Datasets/Frakarisma1.csv", 16)
```




```
Changepoint time: 112
Alert at time: 87
```

The initial size window was 4, the alert time is later than the changepoint. Also on test case 2 we can't observe the likelihood of the changepoint based on the Kurtosis feature.

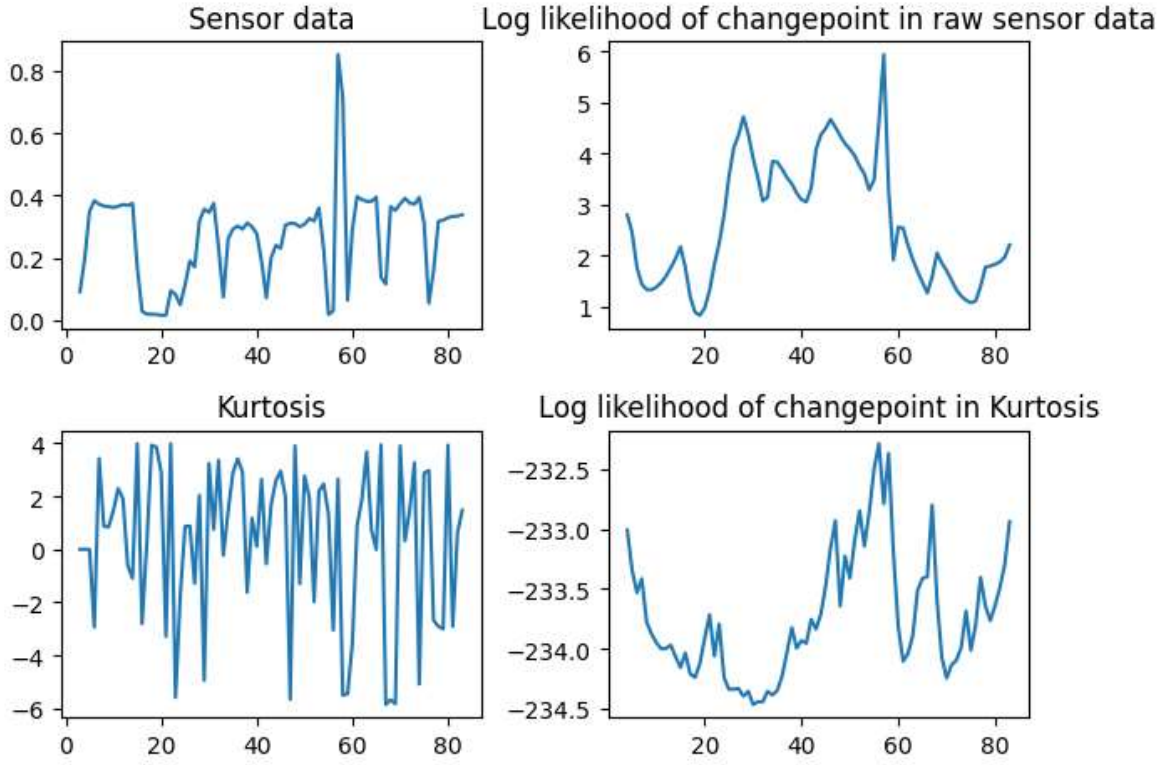
After testing some other window sizes like 14, 16, 18 The size window 16, seems to be a sufficient one. The alert time is earlier than the changepoint. Also on test case 2 we can

observe the changepoint.

Burst

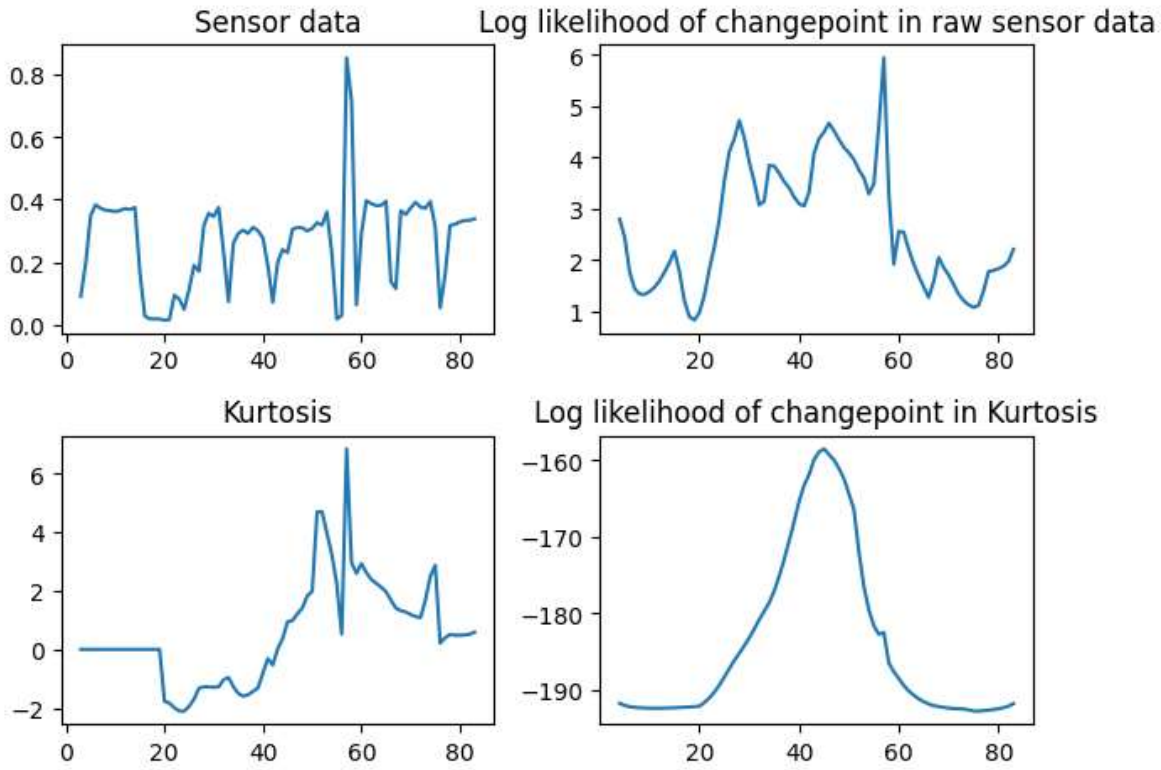
Dataset Num1:

In [4]:
```
detect.analyze("Datasets/Skasimo1.csv", 4)
```



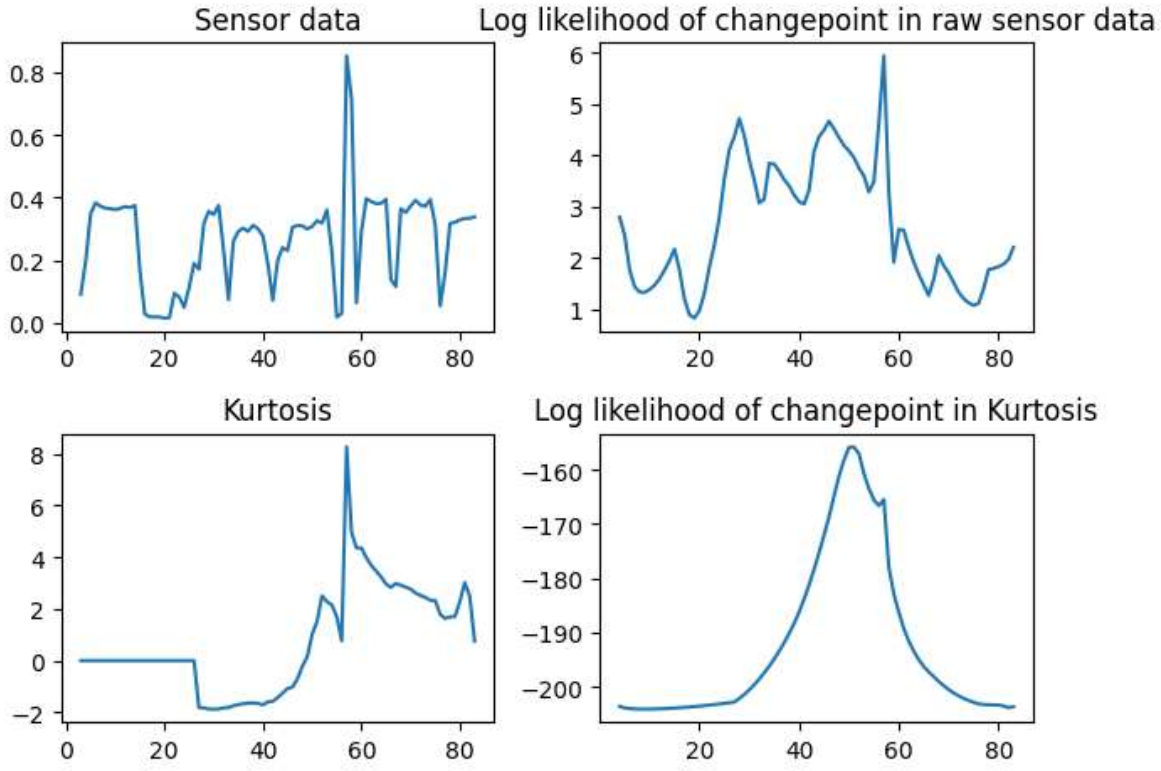

```
Changepoint time: 53
Alert at time: 52
```

In [5]:
```
detect.analyze("Datasets/Skasimo1.csv", 18)
```




```
Changepoint time: 53
Alert at time: 41
```

In [6]:
```
detect.analyze("Datasets/Skasimo1.csv", 25)
```



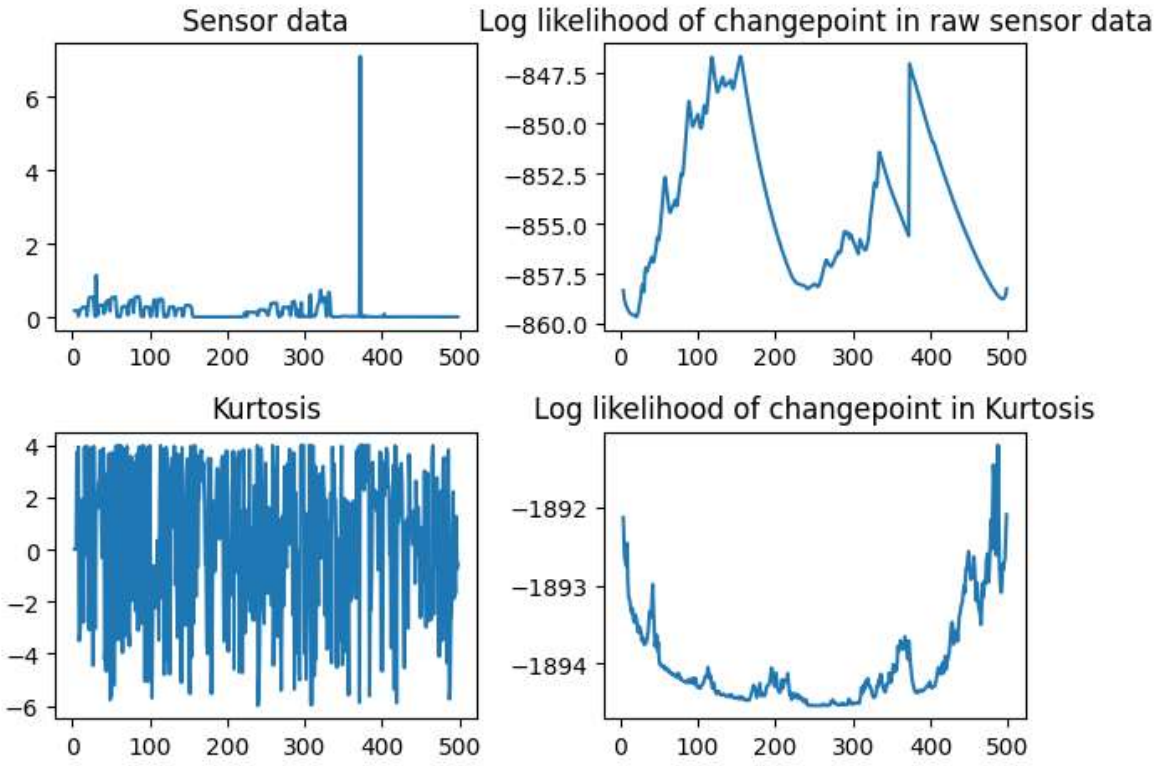

```
Changepoint time: 53
Alert at time: 47
```

In this experiment, size 4 gives us a very accurate alert time but in case of slight variation it might be too late. Also on test case 2 we can't observe the likelihood of the changepoint based on the Kurtosis feature. Window size we can observe the changepoint on test case 2, and it is still early enough. By sizing up the window size to

25, we get a better result, but the computational const rises. Depending on the needs, we can either choose 18 (decent efficiency, cheaper) or 25 (efficient, more costly)

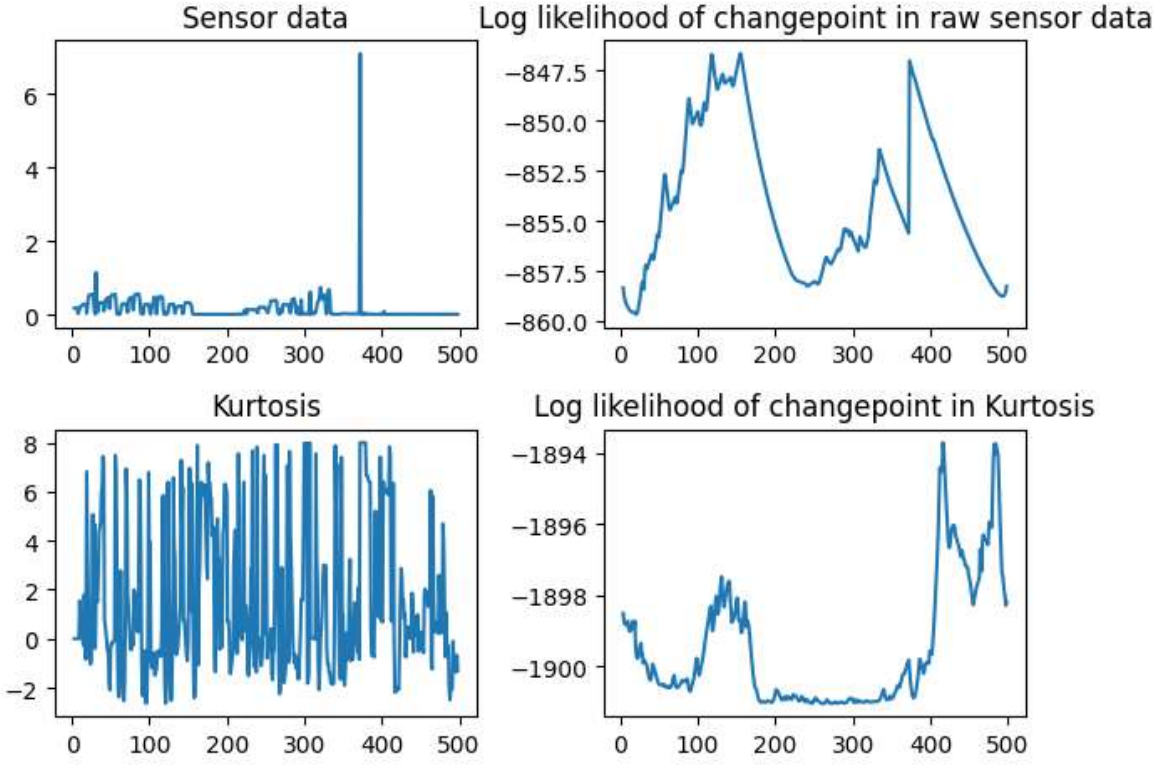
Dataset Num2

In [7]:
```
detect.analyze("Datasets/Skasimo2.csv", 4)
```



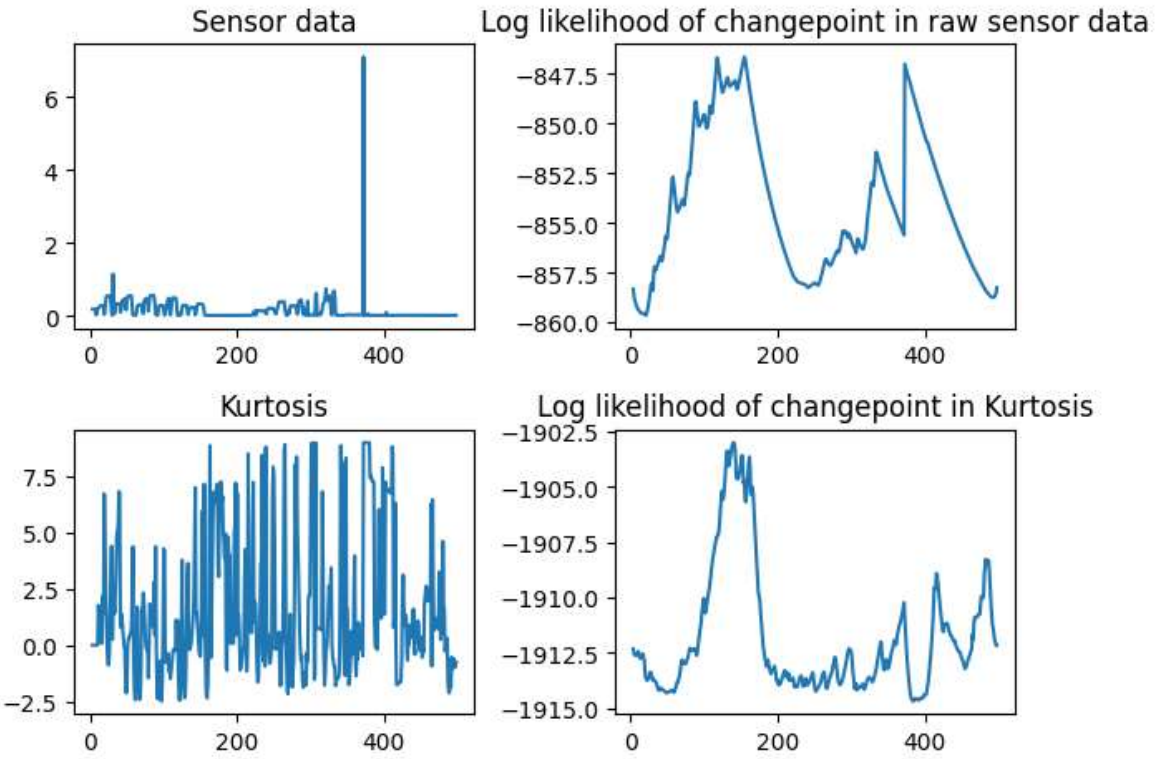

```
Changepoint time: 151
Alert at time: 483
```

In [8]:
```
detect.analyze("Datasets/Skasimo2.csv", 8)
```



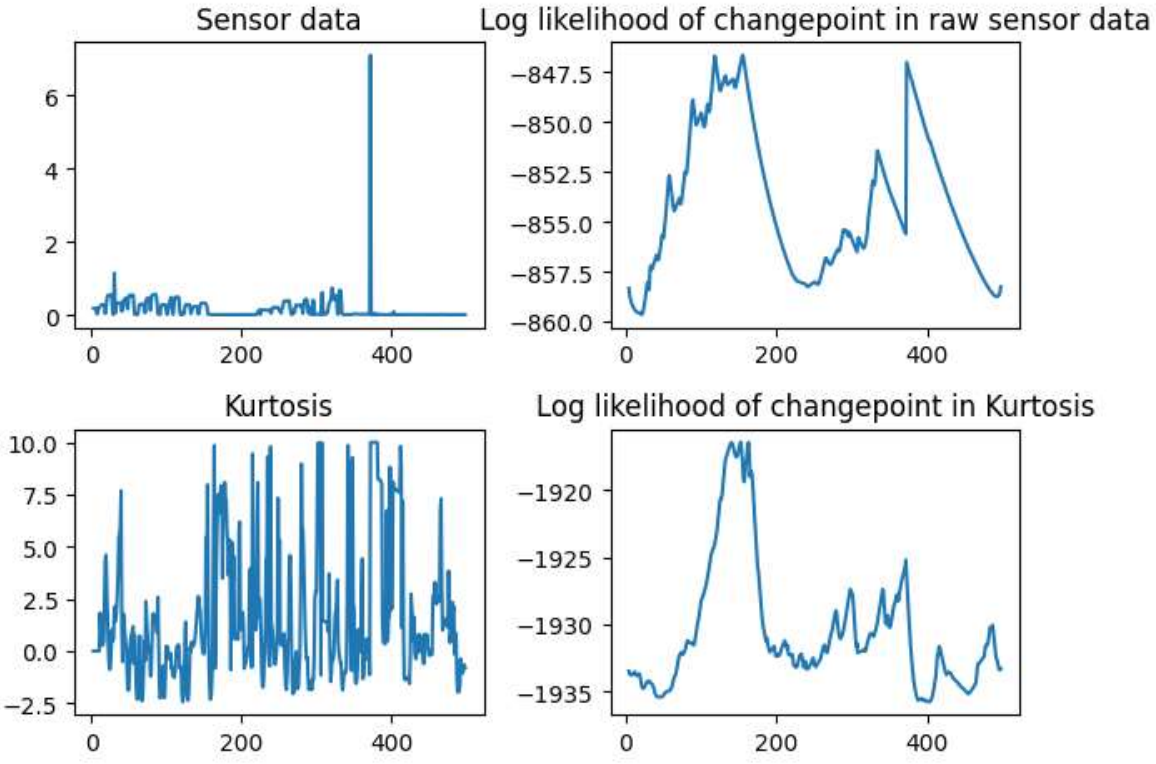
```
Changepoint time: 151
Alert at time: 412
```

In [9]:
```
detect.analyze("Datasets/Skasimo2.csv", 9)
```




```
Changepoint time: 151
Alert at time: 136
```

In [10]:
```
detect.analyze("Datasets/Skasimo2.csv", 10)
```

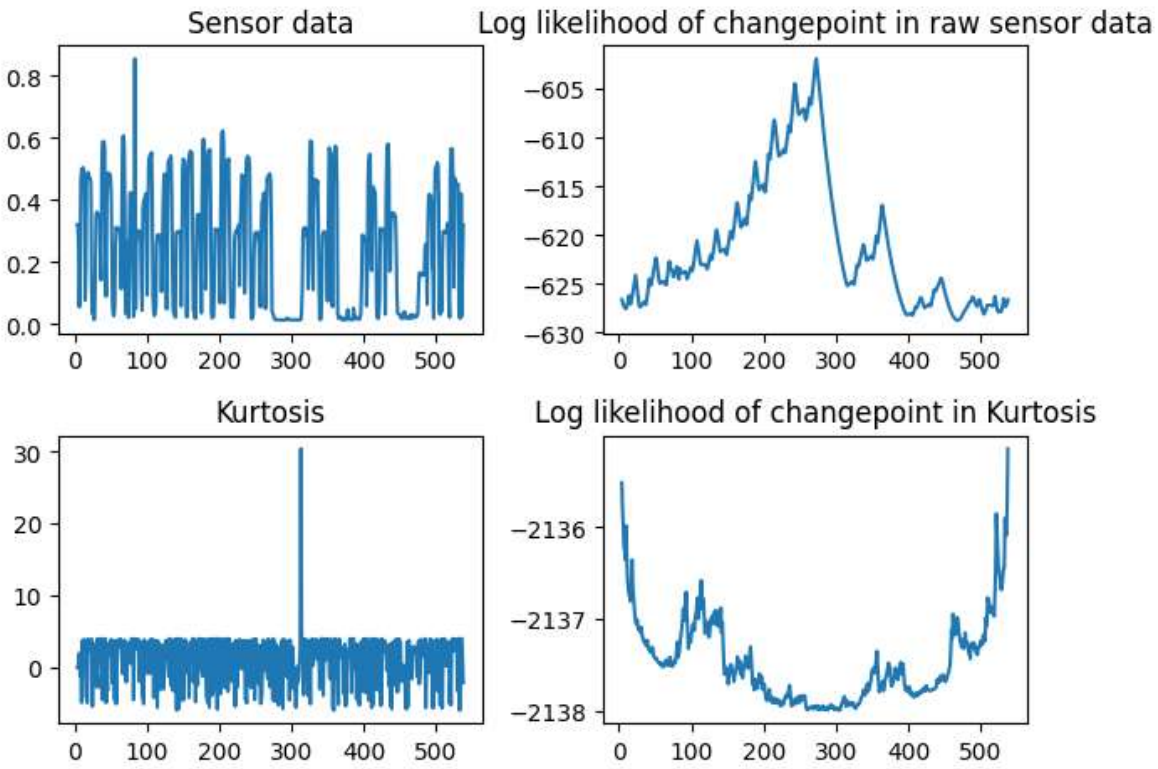



```
Changepoint time: 151
Alert at time: 149
```

What is interesting about this experiment is that window sizes 4,8 is too late while 9 is efficient giving alert early and 10 is giving an alert just in time. Neither of them though gives us on test case 2 a clear observation of the changepoint. Note: Trying higher sizes doesn't improve it.

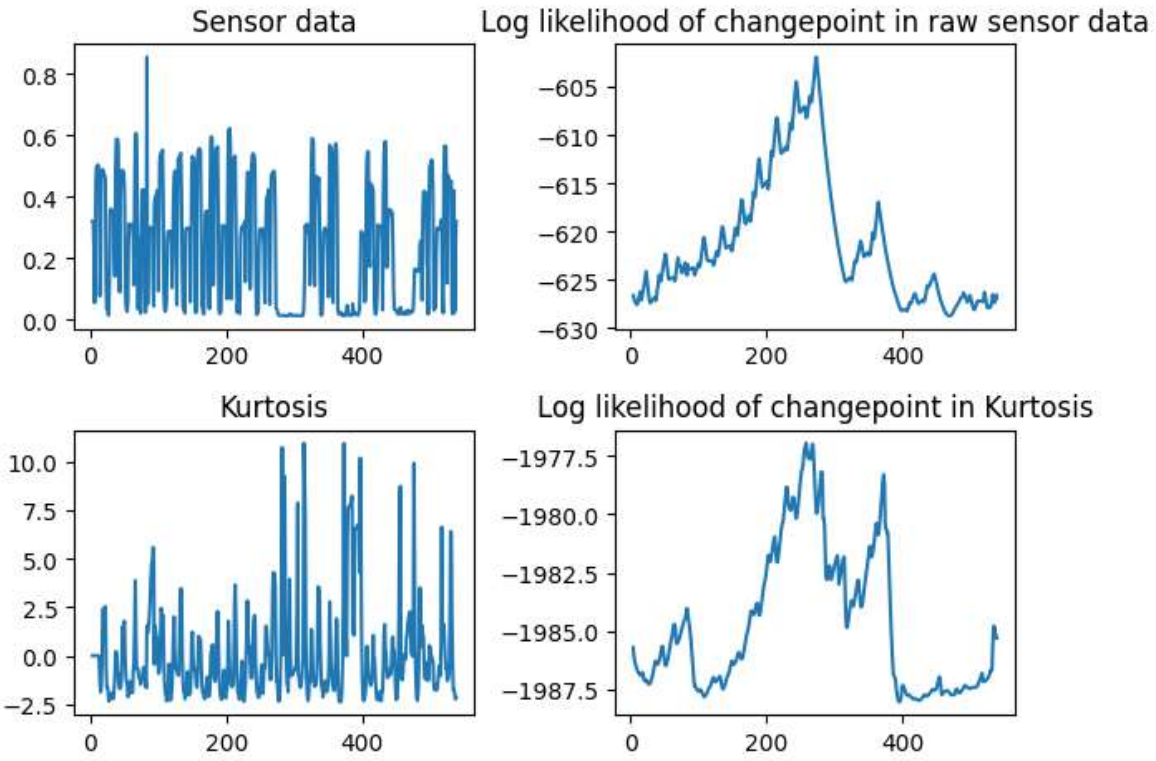
Dataset Num3:

In [11]:
```
detect.analyze("Datasets/Skasimo3.csv", 4)
```



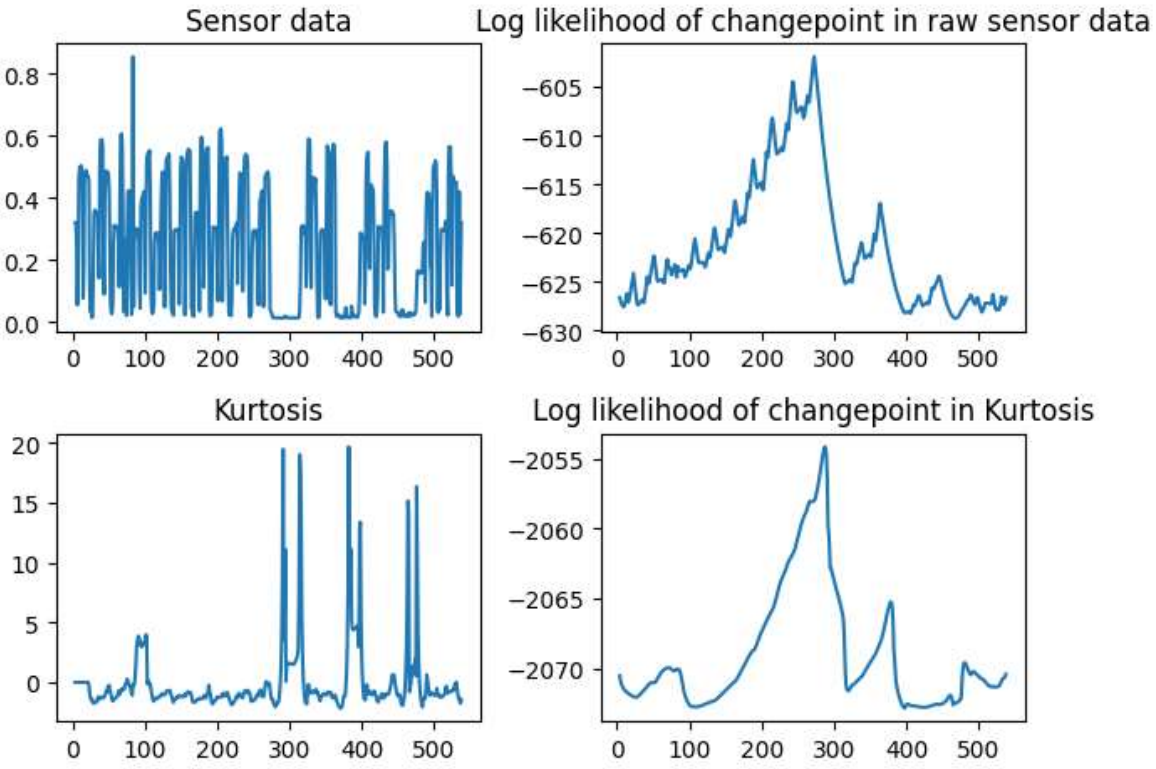

```
Changepoint time: 269
Alert at time: 534
```

In [12]:
```
detect.analyze("Datasets/Skasimo3.csv", 11)
```




```
Changepoint time: 269
Alert at time: 254
```

In [13]:
```
detect.analyze("Datasets/Skasimo3.csv", 20)
```



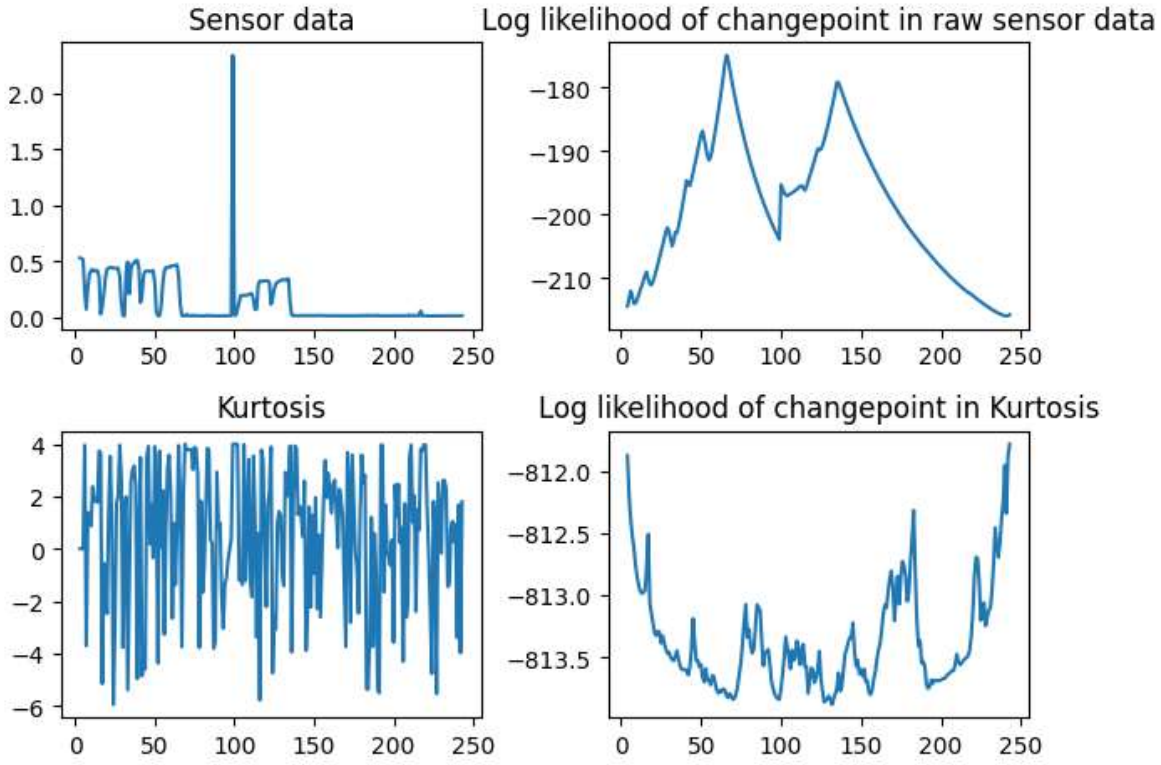

```
Changepoint time: 269
Alert at time: 284
```

Window sizes 4 and 20 are too late while size 11 is efficient and we can also observe the changepoint on test case 2.

Signs

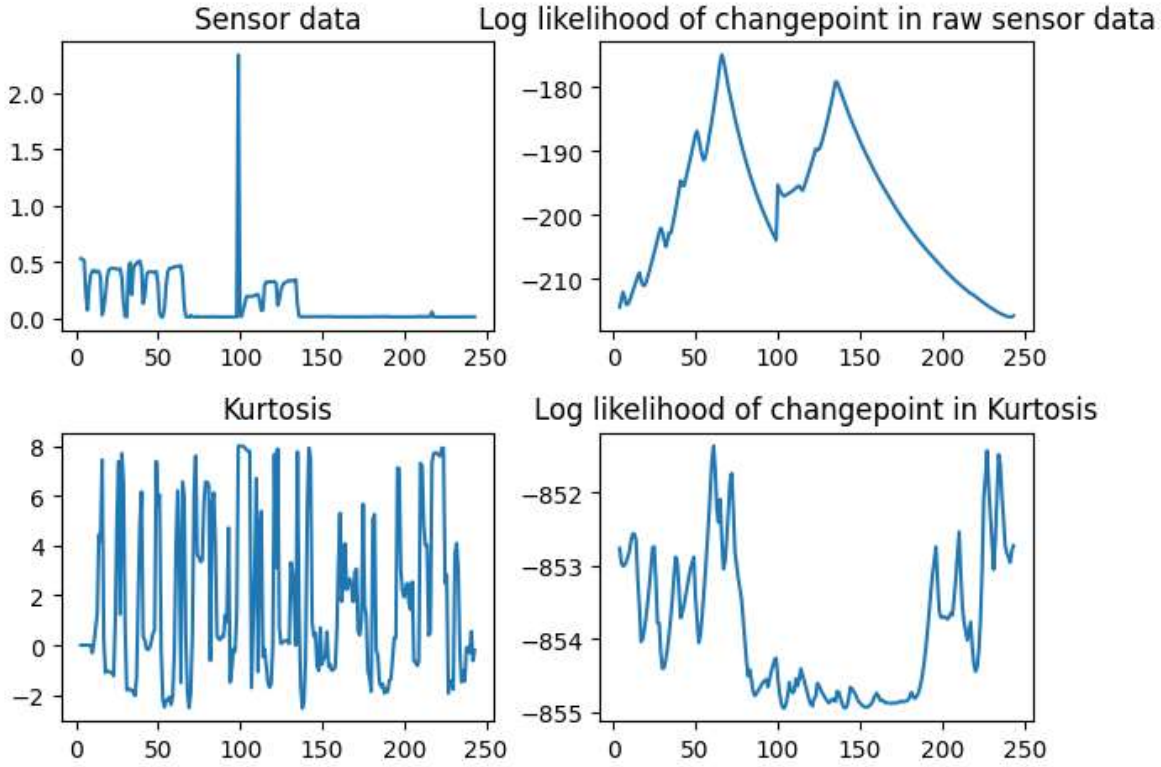
Dataset Num1:

In [14]:
```
detect.analyze("Datasets/Simadia1.csv", 4)
```



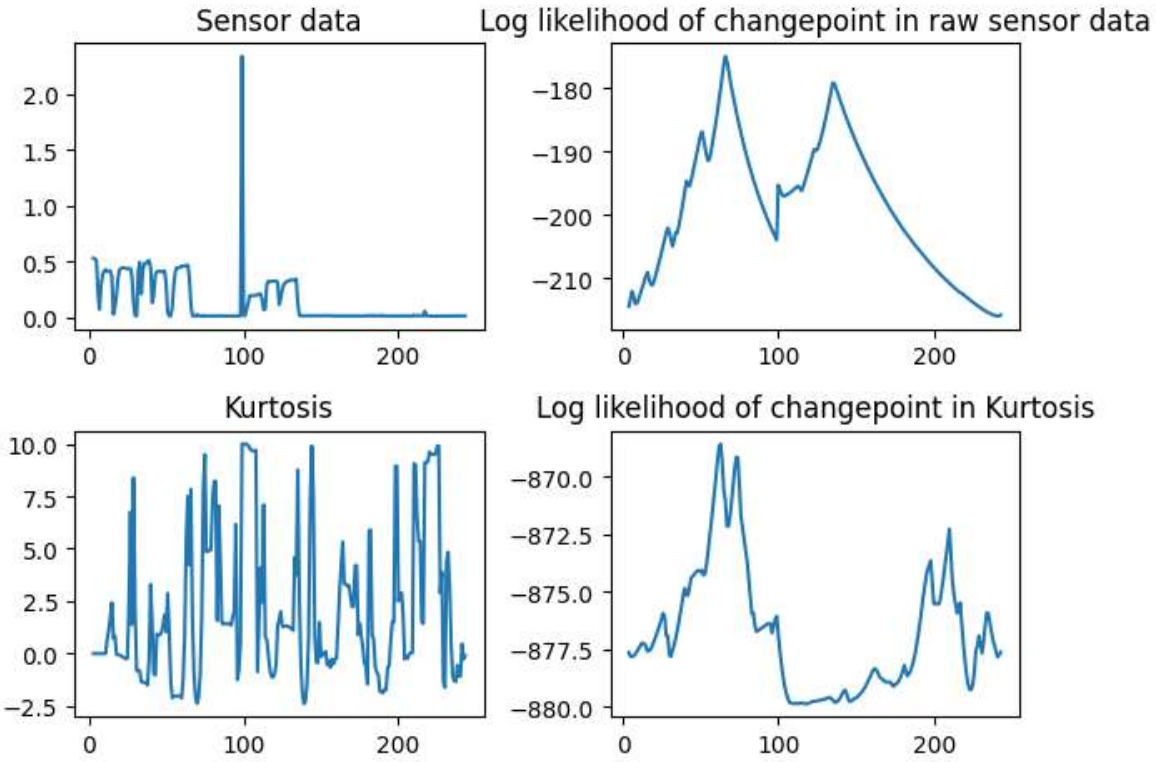

```
Changepoint time: 62
Alert at time: 239
```

In [15]:
```
detect.analyze("Datasets/Simadia1.csv", 8)
```




```
Changepoint time: 62
Alert at time: 57
```

In [16]:
```
detect.analyze("Datasets/Simadia1.csv", 10)
```

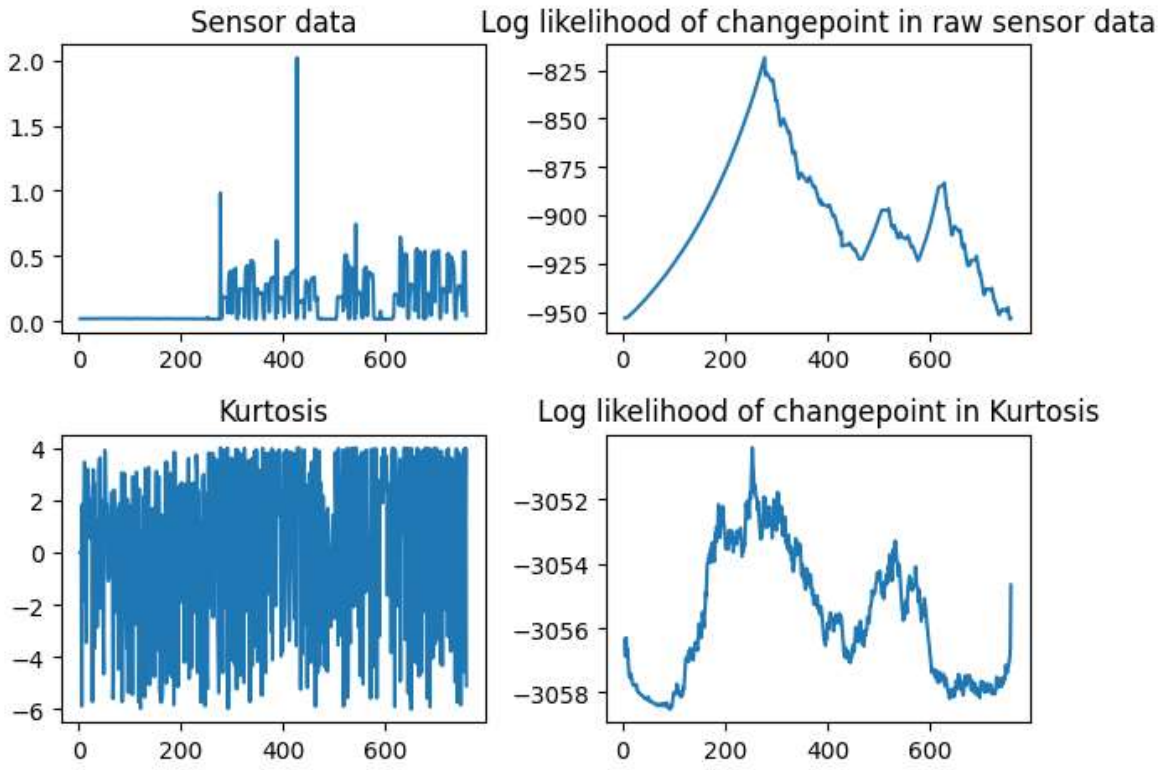



```
Changepoint time: 62
Alert at time: 59
```

Size 4, the alert time exceeds the change point is too late. SIze 8, is efficient, but we *can't* observe *only one* potential changepoint on test case 2 plot. Size 10, is still efficient and we *can* observe *one* potential changepoint, making it a good choice.

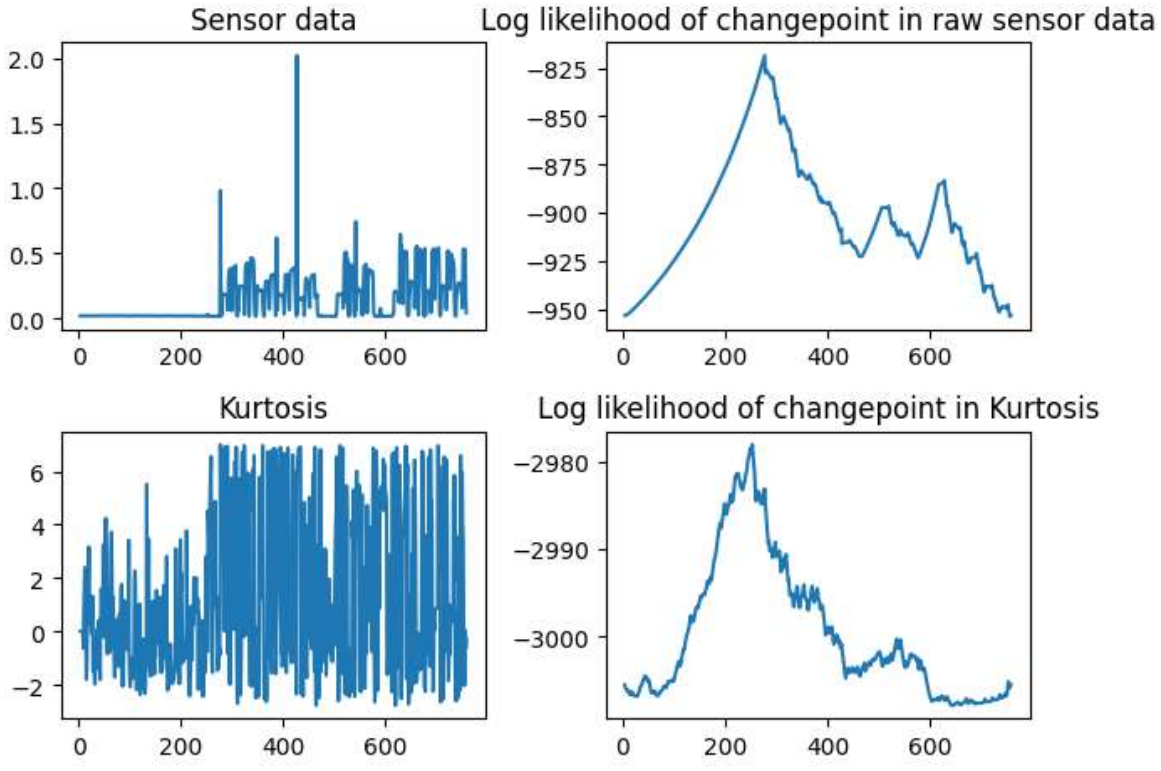
Dataset Num2:

In [17]:
```
detect.analyze("Datasets/Simadia2.csv", 4)
```



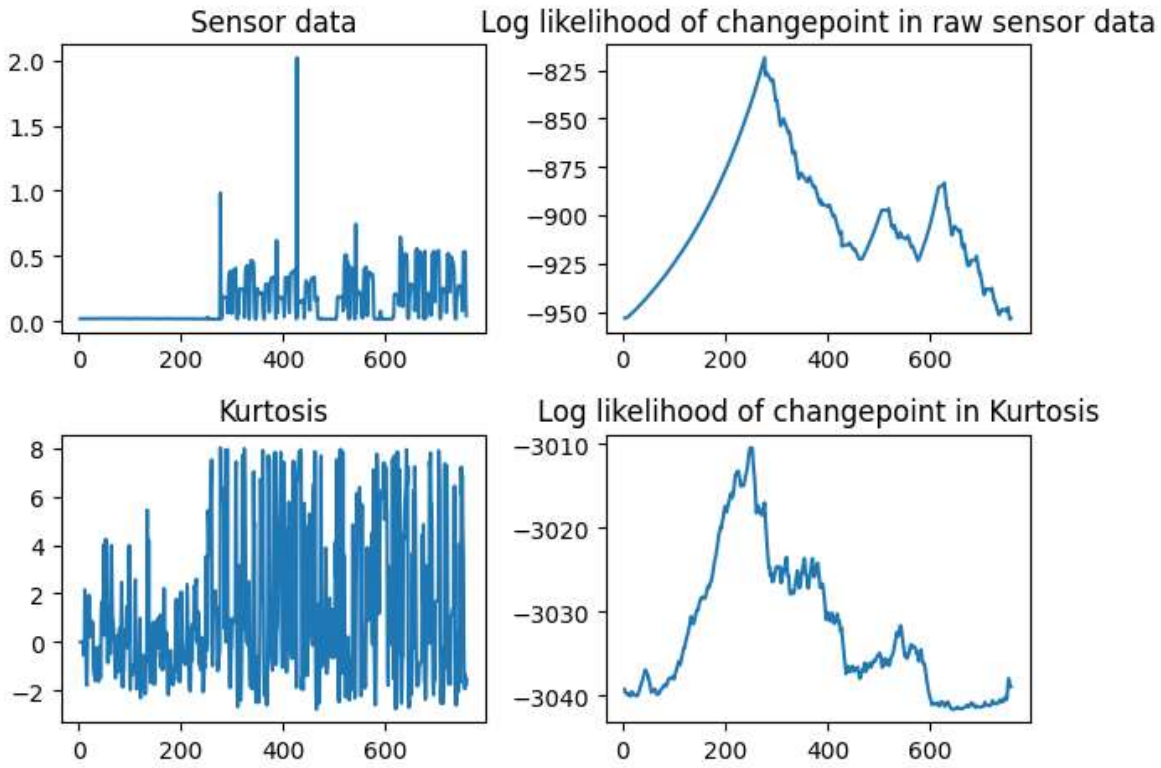

```
Changepoint time: 273
Alert at time: 249
```

In [18]:
```
detect.analyze("Datasets/Simadia2.csv", 7)
```



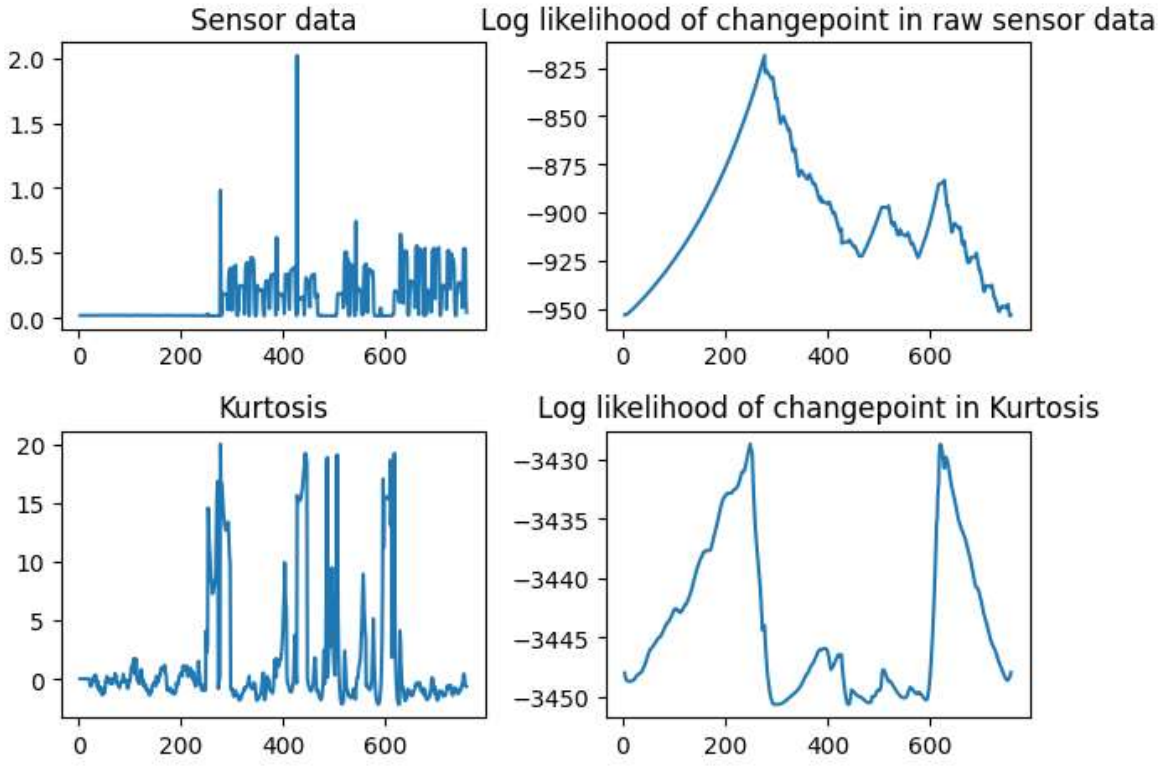

```
Changepoint time: 273
Alert at time: 249
```

In [19]:
```
detect.analyze("Datasets/Simadia2.csv", 8)
```




```
Changepoint time: 273
Alert at time: 249
```

In [20]:
```
detect.analyze("Datasets/Simadia2.csv", 20)
```



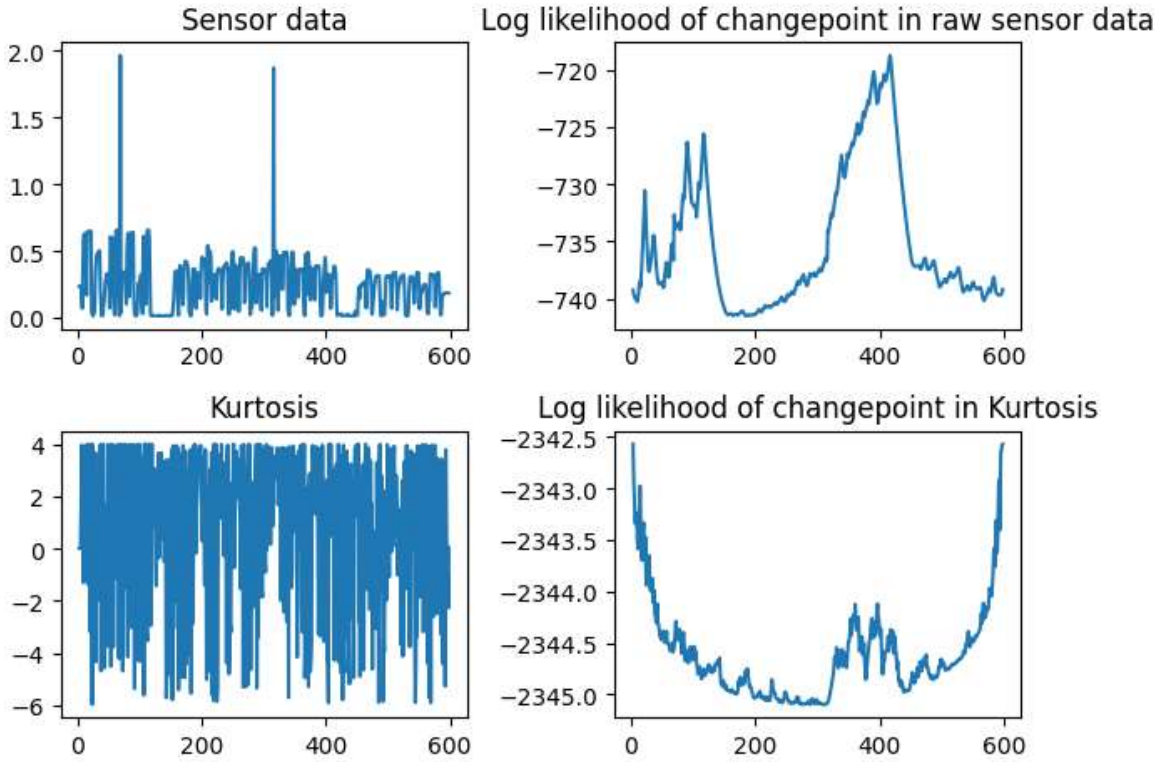

```
Changepoint time: 273
Alert at time: 245
```

Size 4 to 8 gives the same result. Size 20 makes no significant difference. Size 4 is efficient and the changepoint is notable both on test case 2 and test case one.

In [ ]:
```
# Replacement due to scheduled maintenance
```

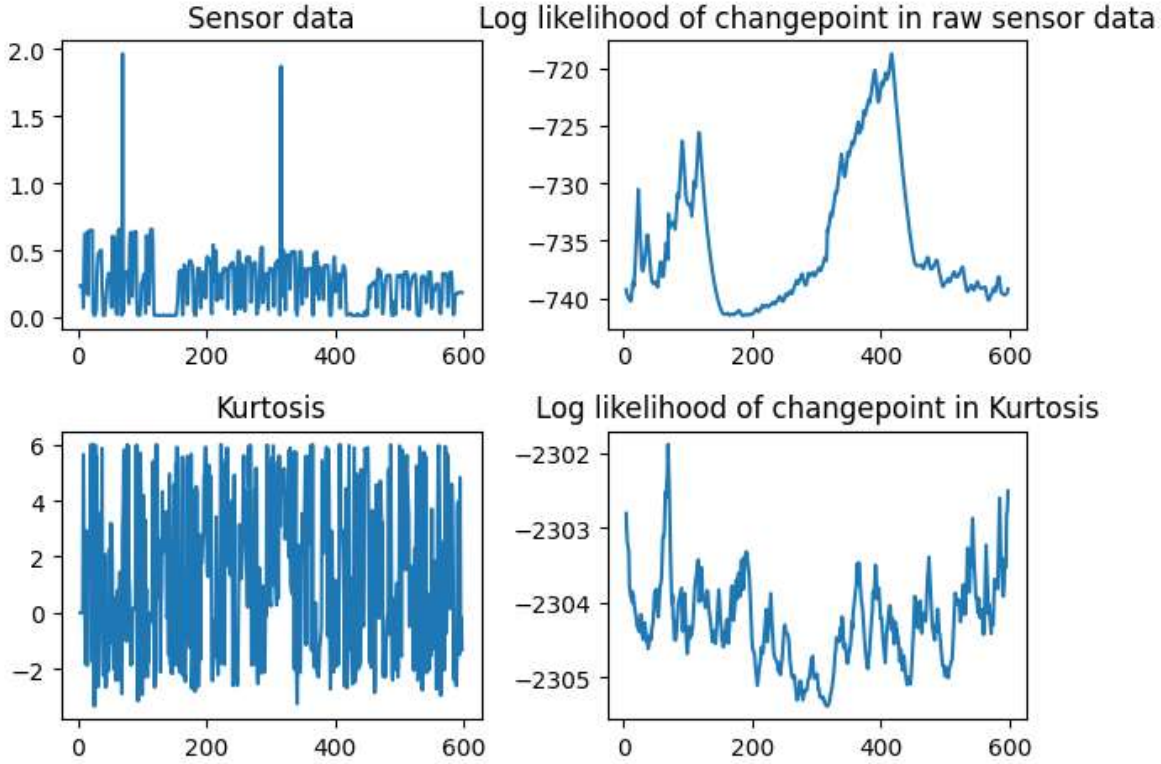
Dataset Num1:

In [27]:
```
detect.analyze("Datasets/Replacement1.csv", 4)
```



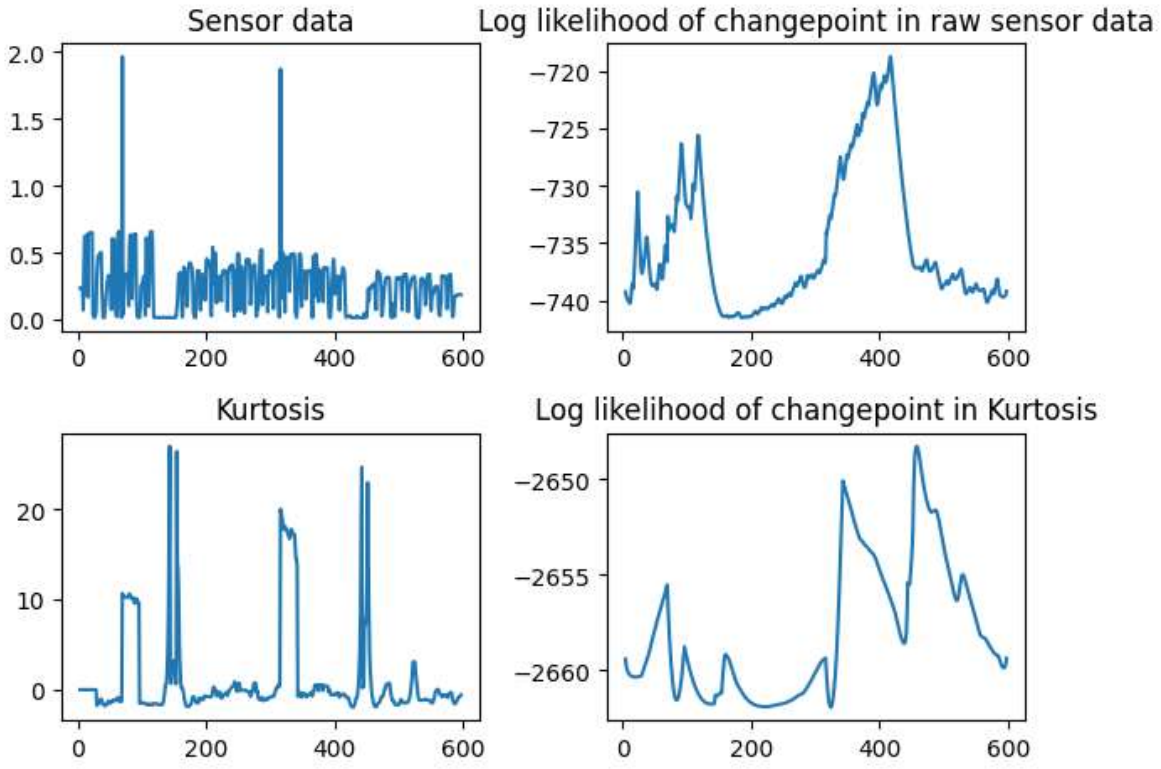

```
Changepoint time: 413
Alert at time: 0
```

In [37]:
```
detect.analyze("Datasets/Replacement1.csv", 6)
```




```
Changepoint time: 413
Alert at time: 65
```

In [39]:
```
detect.analyze("Datasets/Replacement1.csv", 27)
```

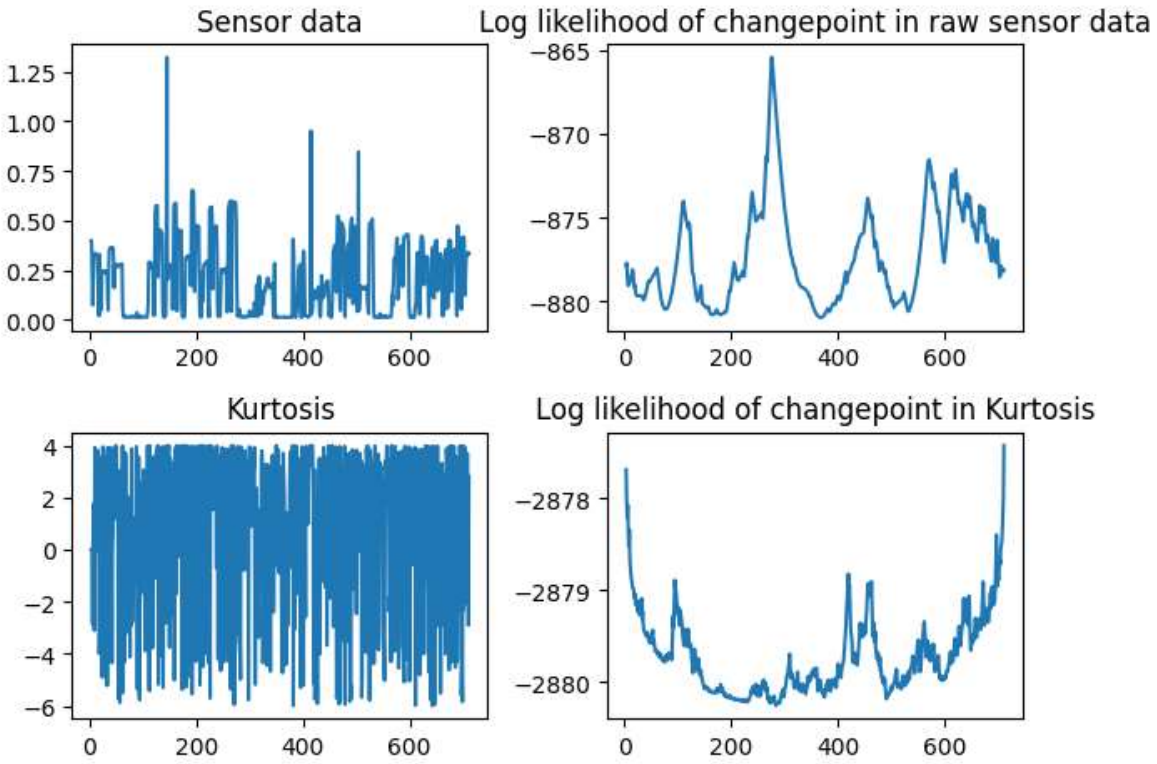



```
Changepoint time: 413
Alert at time: 454
```

The prediction we can make doesn't help us because this is scheduled maintenance and as we can see from the kustosis it doesn't detect the patterns, is either too early or too late compare to tha sensor data. Which means we are changing parts at time where we do not need it. In test case 1, we can see the changepoint and the alert is fired immediately.

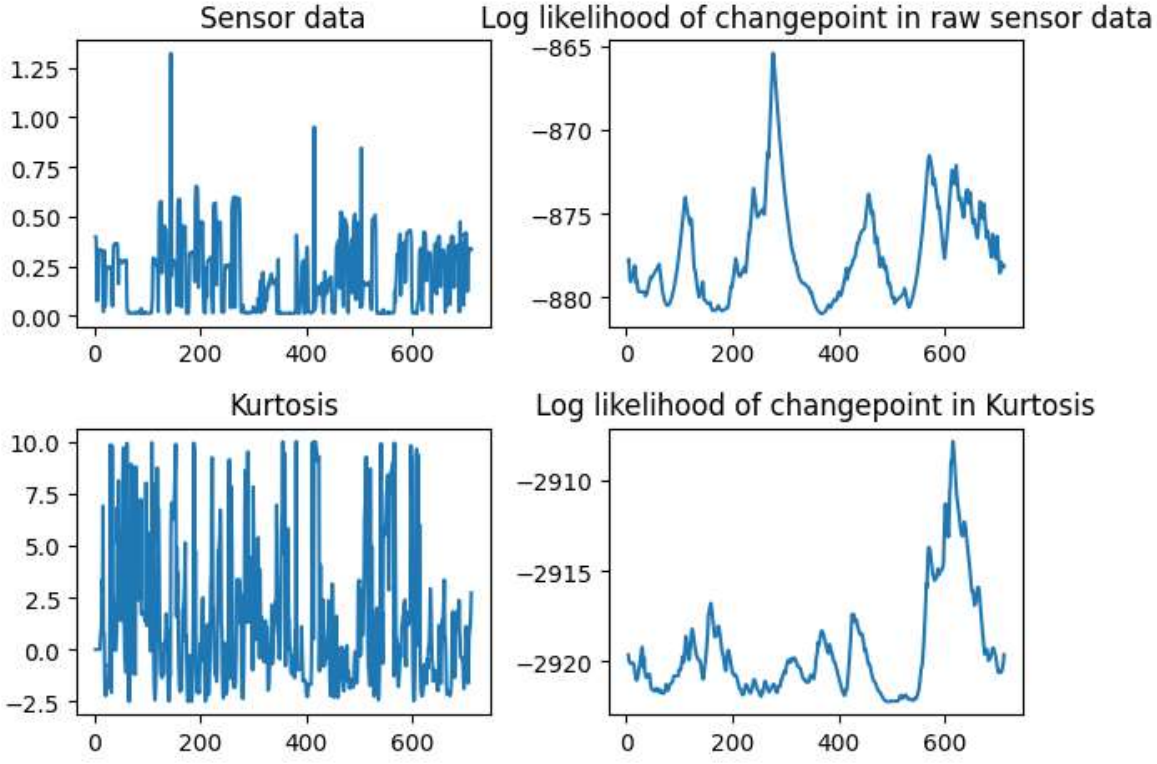
Dataset Num2:

In [42]:
```
detect.analyze("Datasets/Replacement2.csv", 4)
```




```
Changepoint time: 272
Alert at time: 707
```

In [43]:
```
detect.analyze("Datasets/Replacement2.csv", 10)
```




```
Changepoint time: 272
Alert at time: 611
```

Same applies for dataset num 2. Test case 1 gives us the a changepoint while test case num2 is not helpful.

Final Conclusion

During our analysis, the Bayesian Online Changepoint Detection algorithm was utilized on both raw sensor data and the kurtosis feature. Our goal was to detect changing points that corresponded to specific failure modes in machine parts. We carried out an experiment where we tested different rolling window sizes in order to gain a better understanding of their influence on the detection performance.

By using the Bayesian Online Changepoint Detection on the kurtosis feature, we gained valuable insights into the fluctuations in the sensor data. Employing larger rolling window sizes allowed for more refined measurements of kurtosis, capturing overarching patterns in its behavior. On the other hand, using smaller rolling window sizes enabled us to catch more precise changes or fluctuations in kurtosis, potentially signaling specific failure modes.

One should carefully consider the trade-off between being sensitive to local changes and the amount of computational resources needed. Striking a balance between the two could lead to the most ideal solution, potentially by experimenting with different window sizes, taking into account the unique needs of the dataset and the goals of the analysis.

This experiment's key findings will highly inform our approach to monitoring machine health and detecting failures. This valuable information will ultimately lead to the improvement of our maintenance strategies, resulting in increased reliability and enhanced efficiency.

Appendix

Code used in this experiment

In [21]:
```
# %load BayesianOnlineChangepointDetection.py
import csv
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import math

def changepoint(d):
    n = len(d)
    dbar = np.mean(d)
    dsbar = np.mean(np.multiply(d, d))
    fac = dsbar - np.square(dbar)
    summup = []
    summ = 0

    for z in range(n):
        summ+=d[z]
        summup.append(summ)

    y = []
    for m in range(n - 1):
        pos = m + 1
        mscale = 4 * (pos) * (n - pos)
        Q = summup[m] - (summ - summup[m])
```

```
        U = -np.square(dbar * (n - 2 * pos) + Q) / float(mscale) + fac
        y.append(-(n / float(2) - 1) * math.log(n*U/2) - 0.5*math.log((pos * (n

    z, zz = np.max(y), np.argmax(y)
    mean1 = sum(d[:zz+1])/float(len(d[:zz+1]))
    mean2 = sum(d[(zz+1):n])/float(n-1-zz)

    return y, zz, mean1, mean2

def analyze(str, window_size):

    # Read data from CSV file
    measurements = []
    time = []
    i=3

    with open(str, 'r') as f:
        reader = csv.reader(f)
        for row in reader:
            measurements.append(float(row[1]))
            time.append(i)
            i=i+1

    # Plot sensor data
    plt.subplot(2, 2, 1)
    measurements_series = pd.Series(measurements, index=time)
    measurements_series.plot(title='Sensor data')

    # Anomaly detection with online Bayesian changepoint detection
    plt.subplot(2, 2, 2)
    step_like = changepoint(measurements)
    step_series = pd.Series(step_like[0], index=time[1:])
    step_series.plot(title='Log likelihood of changepoint in raw sensor data')

    window = window_size
    s = pd.Series(measurements)
    kur = s.rolling(window).kurt()

    k = []
    for x in range(len(kur)):
        if x > window - 2:
            k.append(kur[x])
        if x < window - 1:
            kur[x] = 0
            k.append(kur[x])

    plt.subplot(2, 2, 3)
    k_series = pd.Series(k, index=time)
    k_series.plot(title='Kurtosis')

    plt.subplot(2, 2, 4)
    step_like = changepoint(k)
    step_series = pd.Series(step_like[0], index=time[1:])
    step_series.plot(title='Log likelihood of changepoint in Kurtosis')

    plt.tight_layout()
    plt.show()

    a, anomaly_time, b, c = changepoint(k)
    a, point_time, b, c = changepoint(measurements)
```

```
    print('Changepoint time:', point_time)
    print('Alert at time:', anomaly_time)
```