Real-time anomaly detection in the steel industry using Python

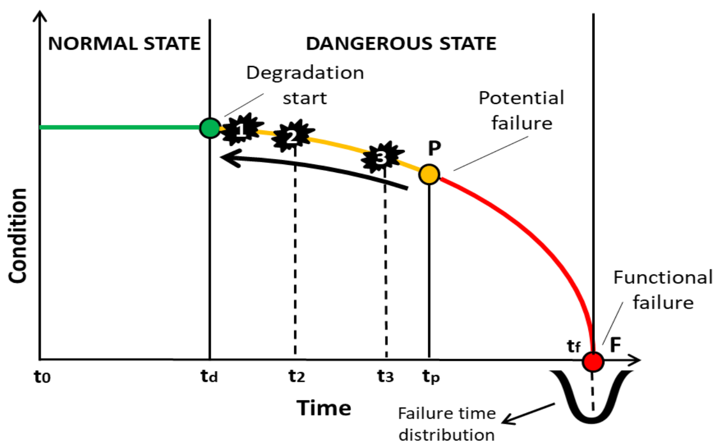
Case Study

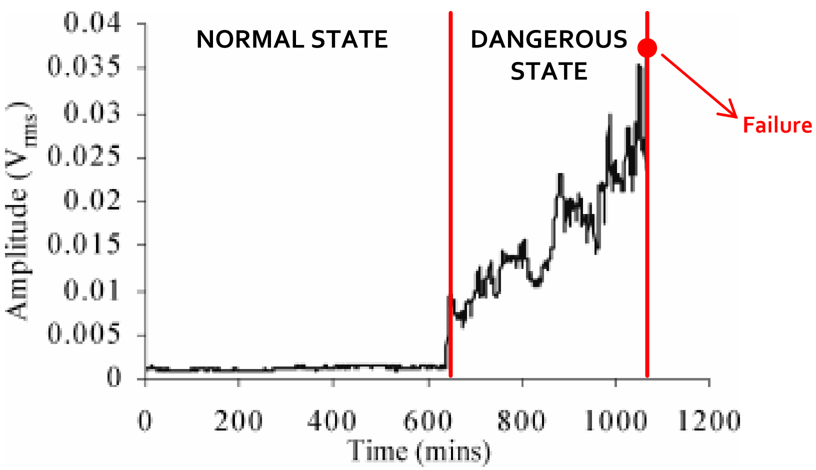
Αλέξανδρος Μπουσδέκης albous@mail.ntua.gr

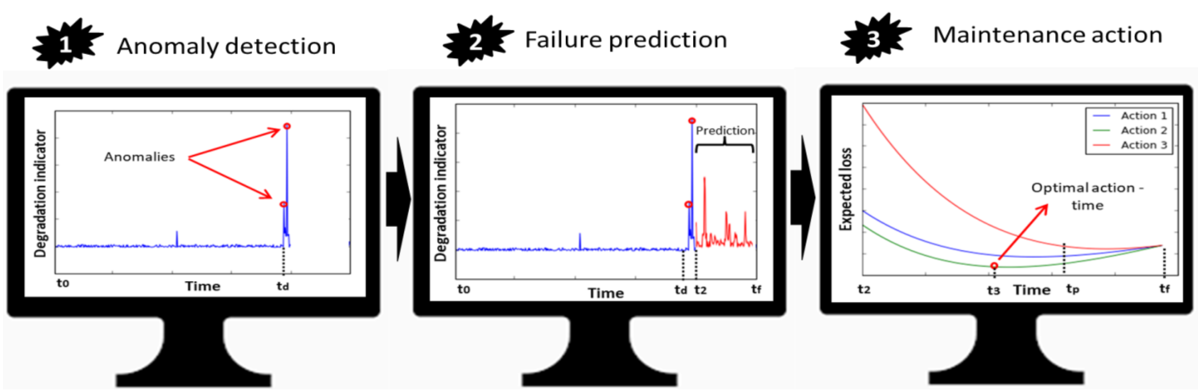
Background

Predictive maintenance

- **Predictive maintenance** uses condition monitoring equipment (e.g. sensors) in order to track the performance of equipment, to detect abnormal behaviour, to predict future failures and to support decision making about proactive actions.



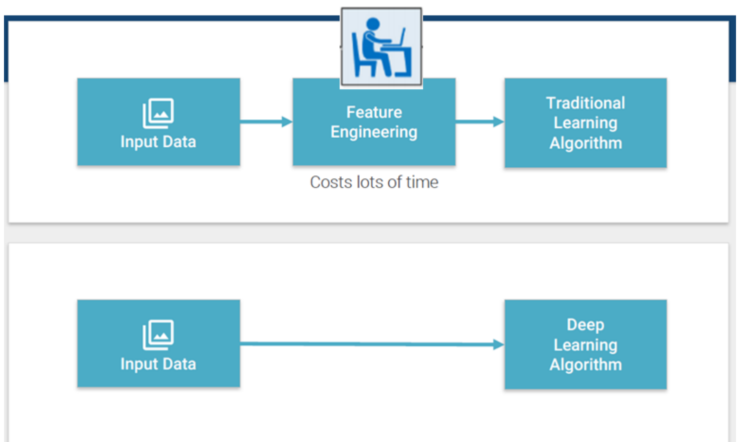




Real-time anomaly detection

Facts

Machine learning requires feature extraction



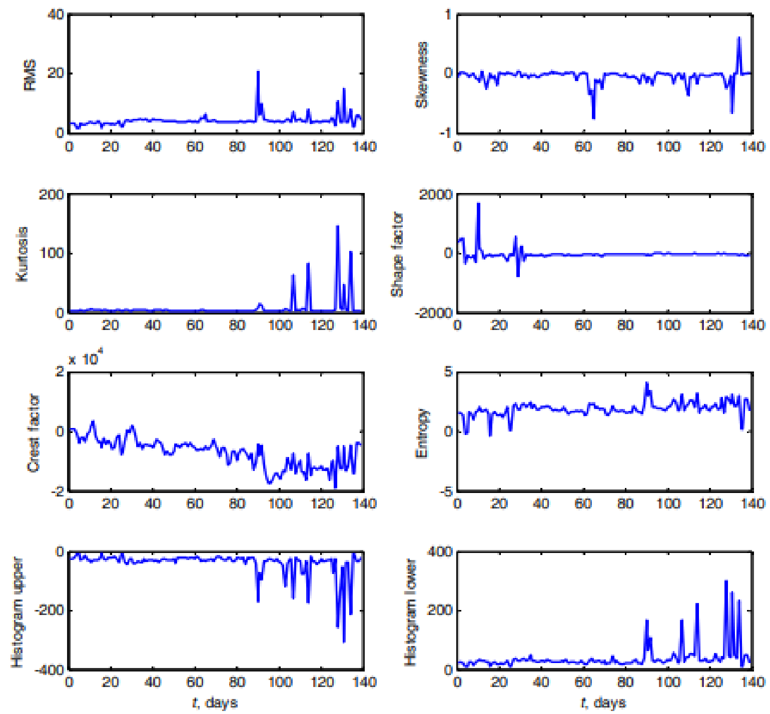
80% of all available data are uncertain

Method of the case study

Extract **time-domain features** → Implement **Bayesian Online Changepoint Detection**

Time-domain features (1/2)

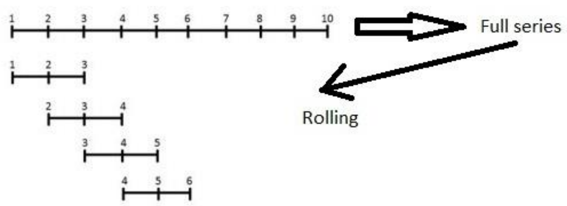
Feature Name	Description: Brief Definition	Formula
RMS	The RMS value increase gradually as fault developed. However, RMS is unable to provide the information of incipient fault stage while it increases with the fault development [11].	$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$
Variance	Variance measures the dispersion of a signal around their reference mean value.	$\mathrm{Var} = \frac{\sum_{i=1}^{N}(x_i - m)^2}{(N-1)\sigma^2}$
Skewness	Skewness quantifies the asymmetry behavior of vibration signal through its probability density function (PDF).	$\mathrm{Sk} = \frac{\sum_{i=1}^{N}(x_i - m)^3}{(N-1)\sigma^3}$
Kurtosis	Kurtosis quantifies the peak value of the PDF. The kurtosis value for normal rolling element bearing is well-recognized as 3.	$\mathrm{Ku} = \frac{\sum_{i=1}^{N}(x_i - m)^4}{(N-1)\sigma^4}$
Shape factor	Shape factor is a value that is affected by an object's shape but is independent of its dimensions [12].	$\mathrm{SF} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}}{\frac{1}{N}\sum_{i=1}^{N}\lvert x_i\rvert}$
Crest factor	Crest factor (CF) calculates how much impact occur during the rolling element and raceway contact. CF is appropriate for "spiky signals" [12].	$\mathrm{CF} = \frac{\max\lvert x_i\rvert}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}}$
Entropy	Entropy, $e(p)$, is a calculation of the uncertainty and randomness of a sampled vibration data. Given a set of probabilities, $(p_1, p_2, \ldots, p_n)$, the entropy can be calculated using the formulas as shown in the right column.	$e(p) = -\sum_{i=1}^{n} p(z_i)\log_2 p(z_i)$



Time-domain features (2/2)

- A rolling feature extraction algorithm on the sensor data set creates another time-series data set including the feature values (instead of the raw data).

- Rolling window:



Longer rolling window sizes tend to yield smoother estimates.

Shorter rolling window sizes are more computationally efficient.

- Kurtosis in Python:

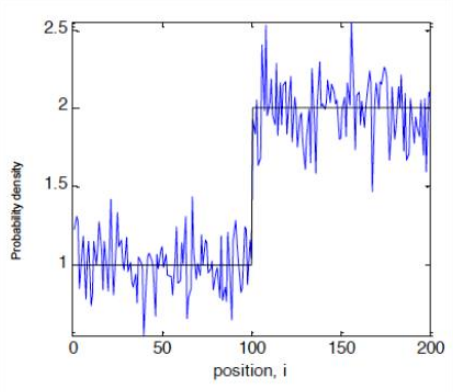
The example below will show a rolling calculation with a window size of four matching the equivalent function call using *scipy.stats*.

```
>>> arr = [1, 2, 3, 4, 999]
>>> fmt = "{0:.6f}"  # limit the printed precision to 6 digits
>>> import scipy.stats
>>> print(fmt.format(scipy.stats.kurtosis(arr[:-1], bias=False)))
-1.200000
>>> print(fmt.format(scipy.stats.kurtosis(arr[1:], bias=False)))
3.999946
>>> s = pd.Series(arr)
>>> s.rolling(4).kurt()
0         NaN
1         NaN
2         NaN
3   -1.200000
4    3.999946
dtype: float64
```

Bayesian Online Changepoint Detection (1/2)

1. Initialize
$$P(r_0) = \tilde{S}(r) \text{ or } P(r_0{=}0) = 1$$
$$\nu_1^{(0)} = \nu_{\text{prior}}$$
$$\chi_1^{(0)} = \chi_{\text{prior}}$$
2. Observe New Datum x_t
3. Evaluate Predictive Probability
$$\pi_t^{(r)} = P(x_t \,|\, \nu_t^{(r)}, \chi_t^{(r)})$$
4. Calculate Growth Probabilities
$$P(r_t{=}r_{t-1}{+}1, x_{1:t}) = P(r_{t-1}, x_{1:t-1})\pi_t^{(r)}(1{-}H(r_{t-1}))$$
5. Calculate Changepoint Probabilities
$$P(r_t{=}0, x_{1:t}) = \sum_{r_{t-1}} P(r_{t-1}, x_{1:t-1})\pi_t^{(r)} H(r_{t-1})$$
6. Calculate Evidence
$$P(x_{1:t}) = \sum_{r_t} P(r_t, x_{1:t})$$
7. Determine Run Length Distribution
$$P(r_t \,|\, x_{1:t}) = P(r_t, x_{1:t})/P(x_{1:t})$$
8. Update Sufficient Statistics
$$\nu_{t+1}^{(0)} = \nu_{\text{prior}}$$
$$\chi_{t+1}^{(0)} = \chi_{\text{prior}}$$
$$\nu_{t+1}^{(r+1)} = \nu_t^{(r)} + 1$$
$$\chi_{t+1}^{(r+1)} = \chi_t^{(r)} + u(x_t)$$
9. Perform Prediction
$$P(x_{t+1} \,|\, x_{1:t}) = \sum_{r_t} P(x_{t+1}|x_t^{(r)}, r_t)P(r_t|x_{1:t})$$
10. Return to Step 2

```python
def step4(d):

    n = len(d)
    dbar = np.mean(d)

    dsbar = np.mean(np.multiply(d,d))

    fac = dsbar-np.square(dbar)

    summ = 0
    summup = []

    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-pos))))

    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
```

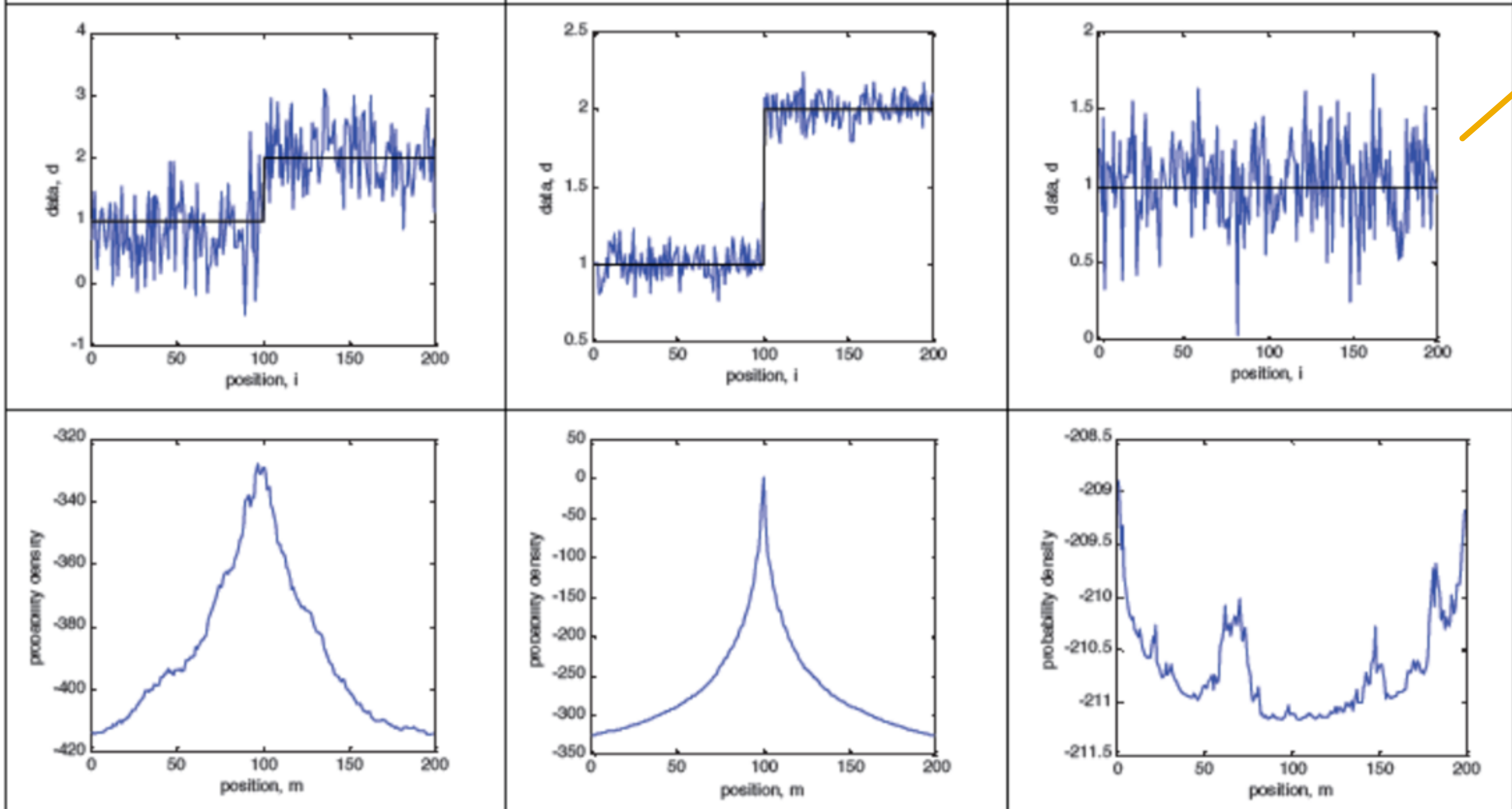


http://gregorygundersen.com/blog/2019/08/13/bocd/ https://arxiv.org/abs/0710.3742

Bayesian Online Changepoint Detection (2/2)

CP = Changepoint location(s), Noise= S.D. of noise component. Log posterior probability plots from the single changepoint algorithm are shown below each simulated data plot.

Noise=0.5, CP = 100	Noise=0.1, CP = 100	Noise=0.3, No CPs



The most common case in manufacturing

Case study in the steel industry

Test cases

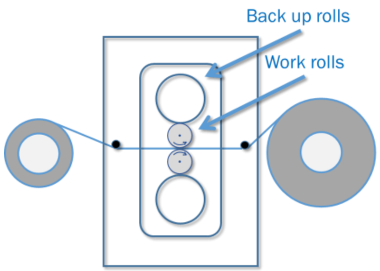
Datasets with sensor measurements during the whole lifetime of the equipment, i.e. from installation until a failure mode or time-based replacement.

1. Bayesian Online Changepoint Detection on **raw sensor data**
2. Bayesian Online Changepoint Detection on the **Kurtosis feature**

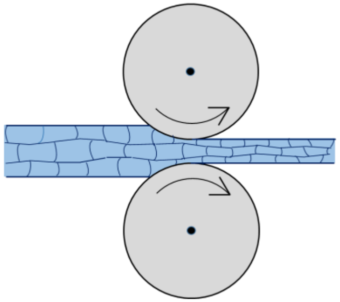
Steel industry

M. J. MAILLIS

Roll Mill Stand



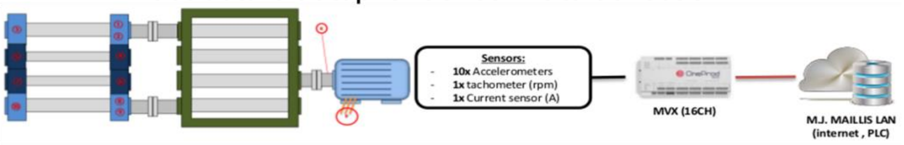
Deforming and Reducing the Grain Size



Raw material

Cold rolling mill

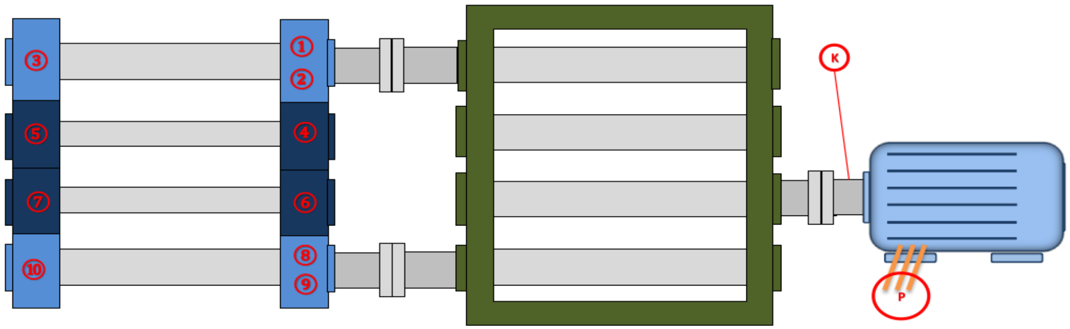
Infrastructure Setup for Sensor Data Collection



Front view of rollers

Rear view of rollers

Sensor infrastructure



Front view of rollers

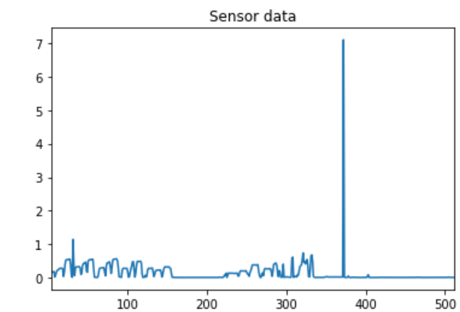
Rear view of rollers

Sensor ID	Measurement point	Sensor direction	Sensor Type
1	Upper backup roll – DE side	Vertical	Accelerometer
2	Upper backup roll – DE side	Axial	Accelerometer
3	Upper backup roll – NDE side	Vertical	Accelerometer
4	Upper working roll – DE side	Reverse horizontal	Accelerometer
5	Upper working roll – NDE side	Horizontal	Accelerometer
6	Down working roll – DE side	Reverse horizontal	Accelerometer
7	Down working roll – NDE side	Horizontal	Accelerometer
8	Down backup roll – DE side	Vertical	Accelerometer
9	Down backup roll – DE side	Axial	Accelerometer
10	Down backup roll – NDE side	Vertical	Accelerometer

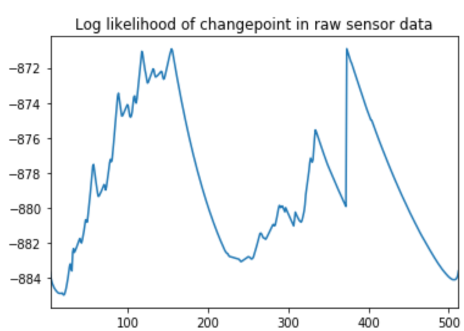
Example

Bayesian Online Changepoint Detection on raw sensor data

Raw sensor data

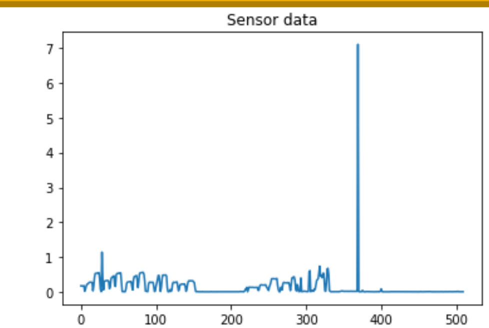


Changepoint detection

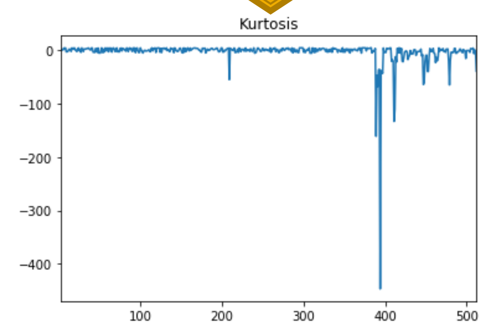


Bayesian Online Changepoint Detection on the Kurtosis feature

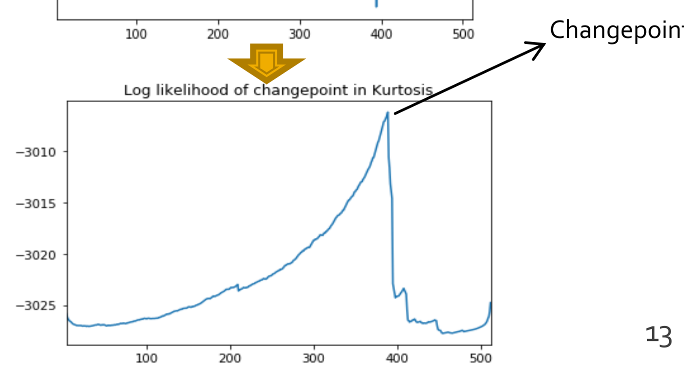
Raw sensor data



Kurtosis extraction



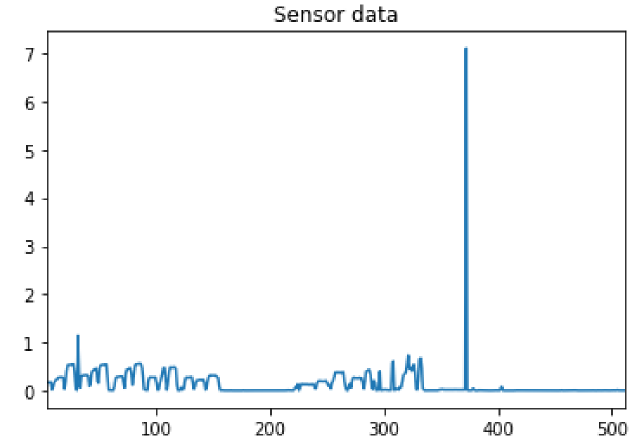
Changepoint detection

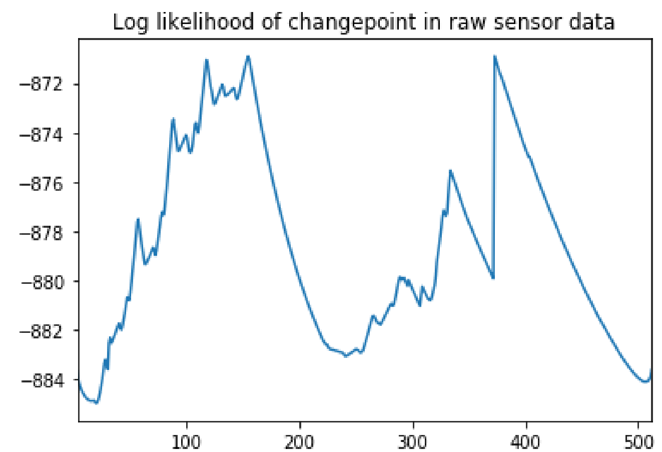


Implementation

Test case 1: Bayesian Online Changepoint Detection on raw sensor data

```python
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)
    summ = 0
    summup = []

    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-pos))))

    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

#Read the data from csv
measurements = []
time = []
i = 3
with open('20_06_2019_09.00_17.15_skasimo_Upper_backup_roll_NDE_side_vertical.csv', '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
measurements_series = pd.Series(measurements,index=time)
measurements_series.plot(title='Sensor data')

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





Explanation of the Python code (1/2)

1

```python
import csv
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import math
```

Import the required Python libraries

2

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

    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-pos))))

    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
```

Define the function of Bayesian Online Changepoint Detection.

*It takes as input a list with numbers (**d**).*

Explanation of the Python code (2/2)

3

```
#Read the data from csv
measurements = []
time = []
i = 3
with open('20_06_2019_09.00_17.15_skasimo_Upper_backup_roll_NDE_side_vertical.csv', 'r') as f:
    reader = csv.reader(f)
    for row in reader:
        measurements.append(float(row[1]))
        time.append(i)
        i = i + 1
```

Read the sensor data from the csv file

4

```
#Plot sensor data
measurements_series = pd.Series(measurements,index=time)
measurements_series.plot(title='Sensor data')
```

Plot the raw sensor data.

5

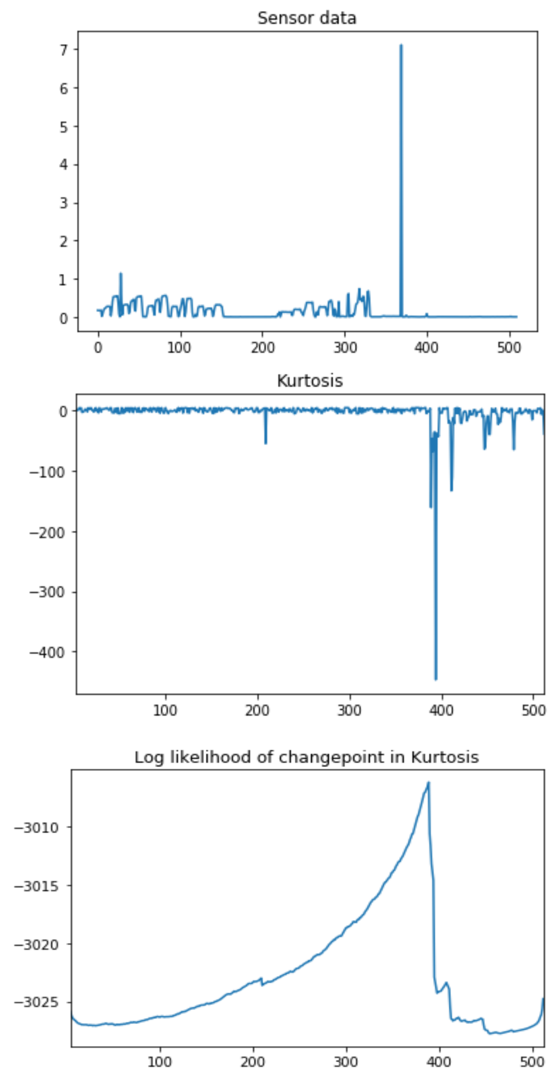
```
#Anomaly detection with online Bayesian changepoint detection
step_like = changepoint(measurements)
step_series = pd.Series(step_like[0],index=time[1:])
plt.figure();
step_series.plot(title='Log likelihood of changepoint in raw sensor data')
```

*Apply the Bayesian Online Changepoint Detection function (**changepoint**) and plot the results.*

*The input is the raw sensor measurements (**measurements**).*

Test case 2: Bayesian Online Changepoint Detection on the Kurtosis feature

```python
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)
    summ = 0
    summup = []

    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-pos))))

    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

#Read the data from csv
measurements = []
time = []
i = 3
with open('20_06_2019_09.00_17.15_skasimo_Upper_backup_roll_NDE_side_vertical.csv', '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.plot(measurements)
#plt.ylabel('Upper_backup_roll_NDE_side_vertical')
plt.title('Sensor data')
plt.show()

window = 4
#Calculate kurtosis
s = pd.Series(measurements)
kur = s.rolling(window).kurt()

#Remove null values
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])

#Anomaly detection with online Bayesian changepoint detection
k_series = pd.Series(k, index=time)
k_series.plot(title='Kurtosis')
step_like = changepoint(k)
step_series = pd.Series(step_like[0], index=time[1:])
plt.figure();
step_series.plot(title='Log likelihood of changepoint in Kurtosis')

#Generate warning
a, time, b, c = changepoint(k)
anomaly_time = time
print('Alert at time:', anomaly_time)
```



Alert at time: 385

Explanation of the Python code (1/3)

1

```
import csv
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import math
```

Import the required Python libraries

2

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

    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-pos))))

    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
```

Define the function of Bayesian Online Changepoint Detection.

*It takes as input a list with numbers (**d**).*

Explanation of the Python code (2/3)

3

```
#Read the data from csv
measurements = []
time = []
i = 3
with open('20_06_2019_09.00_17.15_skasimo_Upper_backup_roll_NDE_side_vertical.csv', 'r') as f:
    reader = csv.reader(f)
    for row in reader:
        measurements.append(float(row[1]))
        time.append(i)
        i = i + 1
```

Read the sensor data from the csv file

4

```
#Plot sensor data
plt.plot(measurements)
#plt.ylabel('Upper_backup_roll_NDE_side_vertical')
plt.title('Sensor data')
plt.show()
```

Plot the raw sensor data.

5

```
window = 4
#Calculate kurtosis
s = pd.Series(measurements)
kur = s.rolling(window).kurt()

#Remove null values
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])
```

*Define the window size of the rolling kurtosis (**window**).*

Derive kurtosis dataset.

Remove the null values.

Explanation of the Python code (3/3)

5

```
#Anomaly detection with online Bayesian changepoint detection
k_series = pd.Series(k, index=time)
k_series.plot(title='Kurtosis')
```

Plot the kurtosis data.

6

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

*Apply the Bayesian Online Changepoint Detection function (**changepoint**) and plot the results.*

*The input is the kurtosis data (**k**).*

7

```
#Generate warning
a, time, b, c = changepoint(k)
anomaly_time = time
print('Alert at time:', anomaly_time)
```

Estimate the time of changepoint.

Experiments

Experiments for the exercise

- Execute test cases 1 & 2 for different datasets (i.e. corresponding to different failure modes)

```
#Read the data from csv
measurements = []
time = []
i = 3
with open('20_06_2019_09.00_17.15_skasimo_Upper_backup_roll_NDE_side_vertical.csv', 'r') as f:
    reader = csv.reader(f)
    for row in reader:
        measurements.append(float(row[1]))
        time.append(i)
        i = i + 1
```

- For each dataset, execute test case 2 for different kurtosis windows

```
window = 4
#Calculate kurtosis
s = pd.Series(measurements)
kur = s.rolling(window).kurt()

#Remove null values
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])
```

- Compare and discuss the results

Thank you