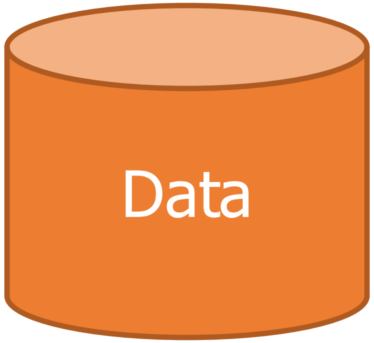
# CSE2525 Preprocessing

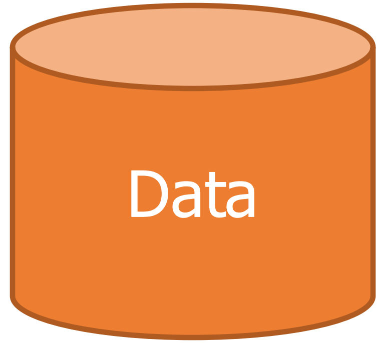
## How to deal with Data

# Schedule

| | Mondays [13:45] | Thursdays [10:45] | Lab schedule |
|---|---|---|---|
| 10 Nov | Intro | Data Processing | *Start Lab 1* |
| 17 Nov | Distances | Dimensionality Reduction | *Implement PCA* |
| 24 Nov | Anomaly Detection | Clustering | *Implement DTW* |
| 1 Dec | Embeddings | Matrix Decomposition | Build pipelines - End lab 1 |
| 8 Dec | **NO LECTURE** | Recommender Systems | *Start Lab 2* |
| 15 Dec | Hashing | Sketching | *Implement NMF* |
| 5 Jan | Text | Discrimination, Bias & Explanations | *Implement MinHash* |
| 12 Jan | Graphs | *Invited Lecture (Manifold Learning)* | Build pipelines - End lab 2 |
| 19 Jan | Exam Preparations | Exam Preparations | |
| 26 Jan | **EXAM** | | |




# What are important properties?




# What are important properties?

most ML needs feature vectors

features can be discrete/continuous

the data distribution

is data missing?

possible imbalance

its SIZE

sparsity

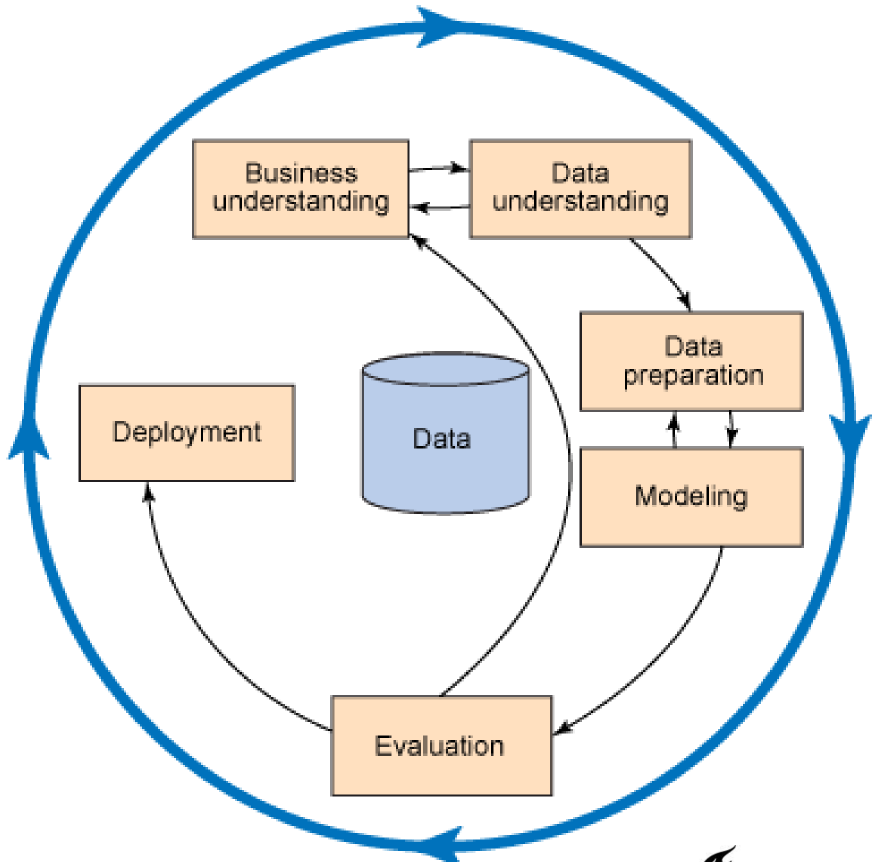
its dimensionality

feature ranges

# Before we begin

- Data can be aggregated in many different ways
  - understand the goal!
  - understand the data!
  - ***before modeling!***
- Key tool: visualization
  - heat maps
  - cross tables
  - pair-wise scatterplots
  - parallel coordinates
  - …

CRISP-DM



# Anomaly detection challenge


Sensor and actuator data
Highly discrete and predictable

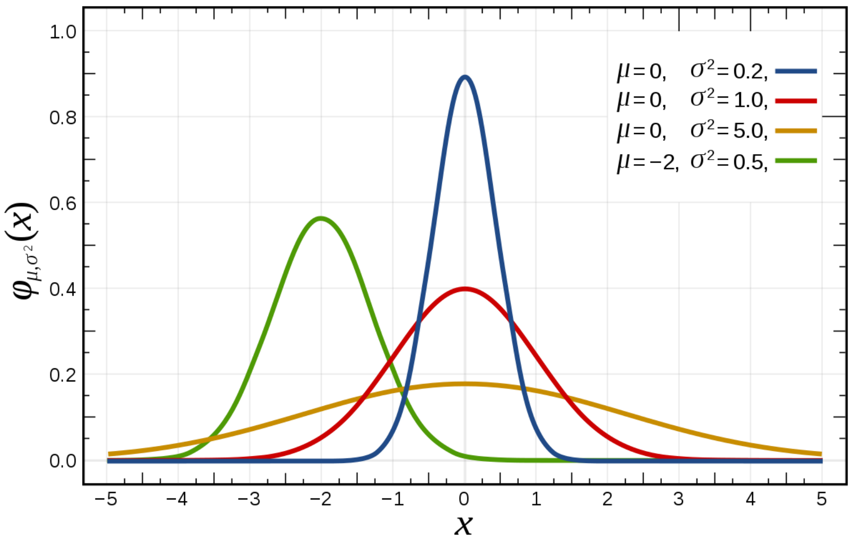
# Understanding the data

- Make plots and tables that show:
  - Features and their distribution
    - *shows what kind of feature processing to use*
  - Dependence between features
    - *shows what kind of feature processing to use*
  - Dependence over time
    - *shows the type of temporal processing to use*
  - The difficulty of the problem
    - *shows what technique might be suitable*

# Features and Distributions

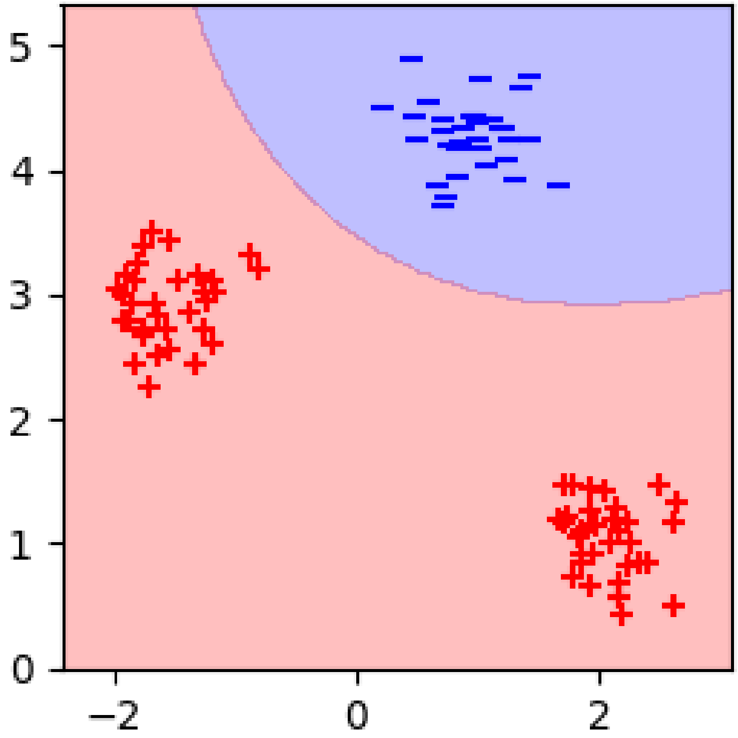
# Gaussian distributed

- Much data is Gaussian distributed
- CLT: the normalized sum of any independent random data tends towards a Gaussian distribution
- Several ML model assume Gaussian distributed data for computational reasons, though most work fine on non-Gaussian data…



# Wrong models can still be useful!

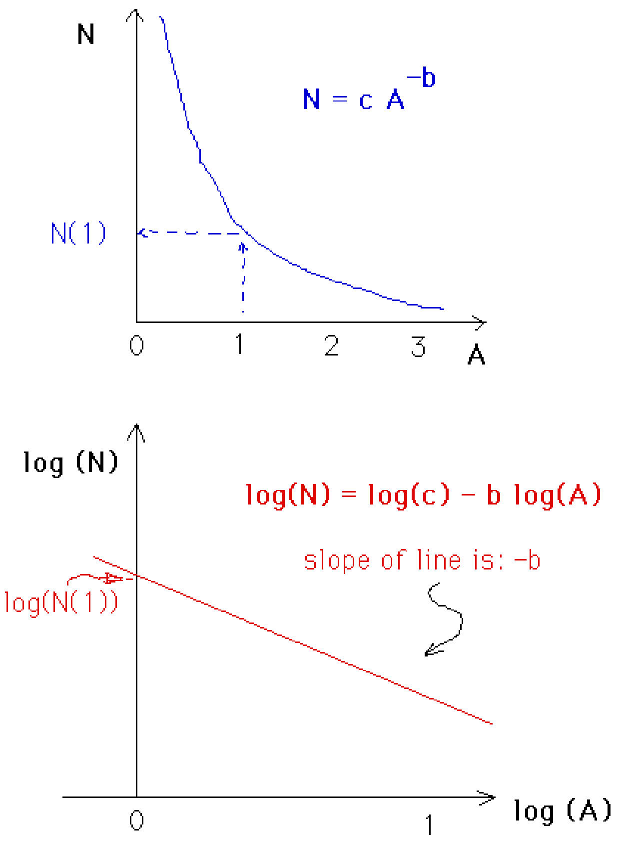
- For instance for Quadratic Discriminant Analysis



# Power laws

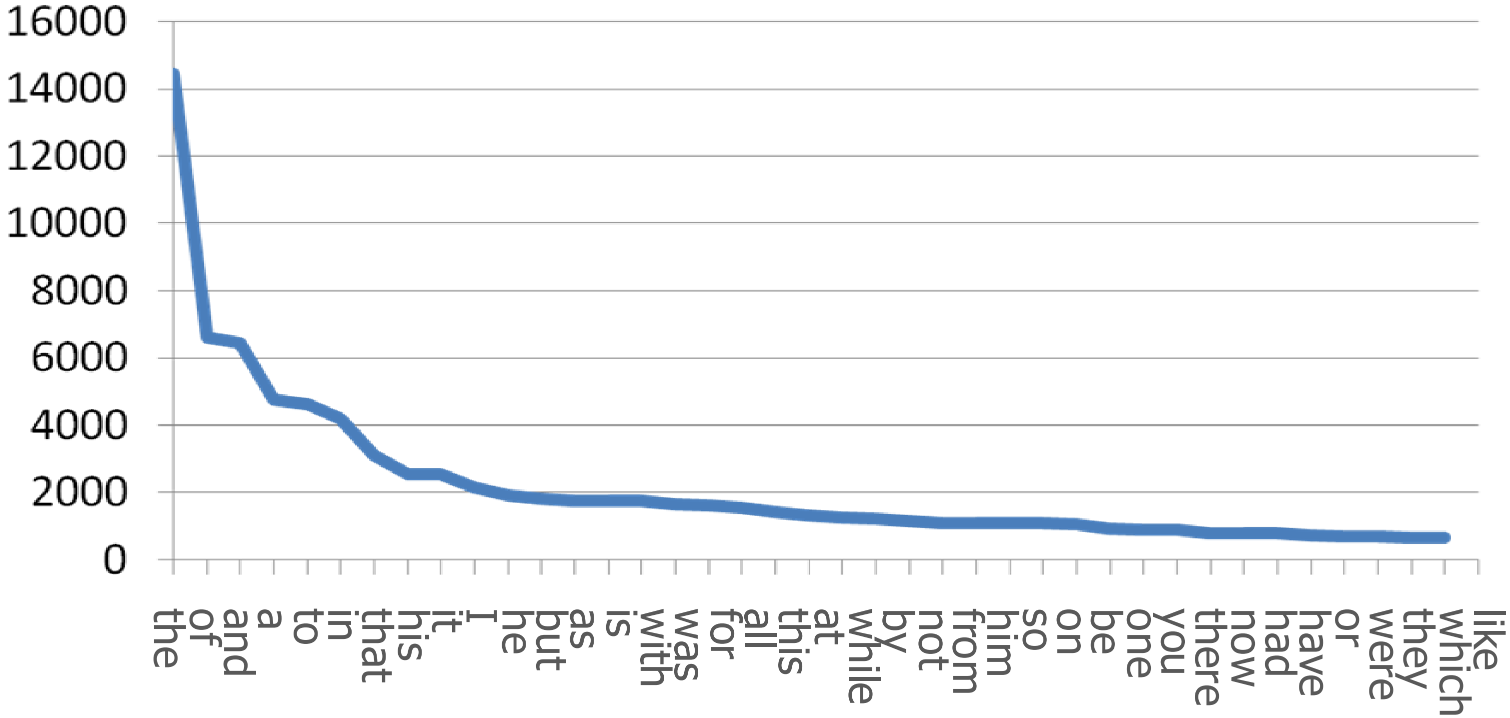
- Real-world data often follows power laws

- A power law is a linear relationship between the logarithms of two variables



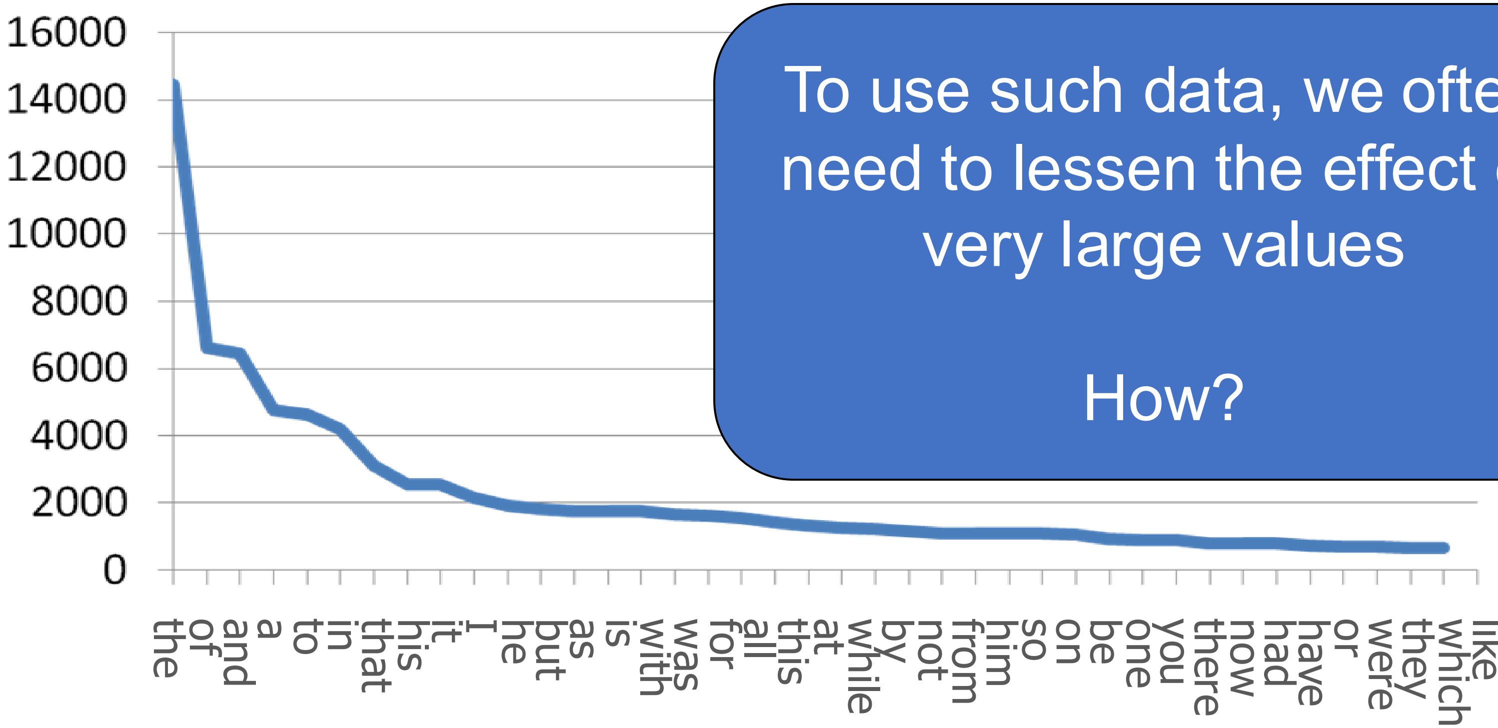
# Power laws: Example

- Order words by frequency in a large number of documents



# Power laws: Example

- Order words by frequency in a large number of documents




To use such data, we often need to lessen the effect of very large values

How?

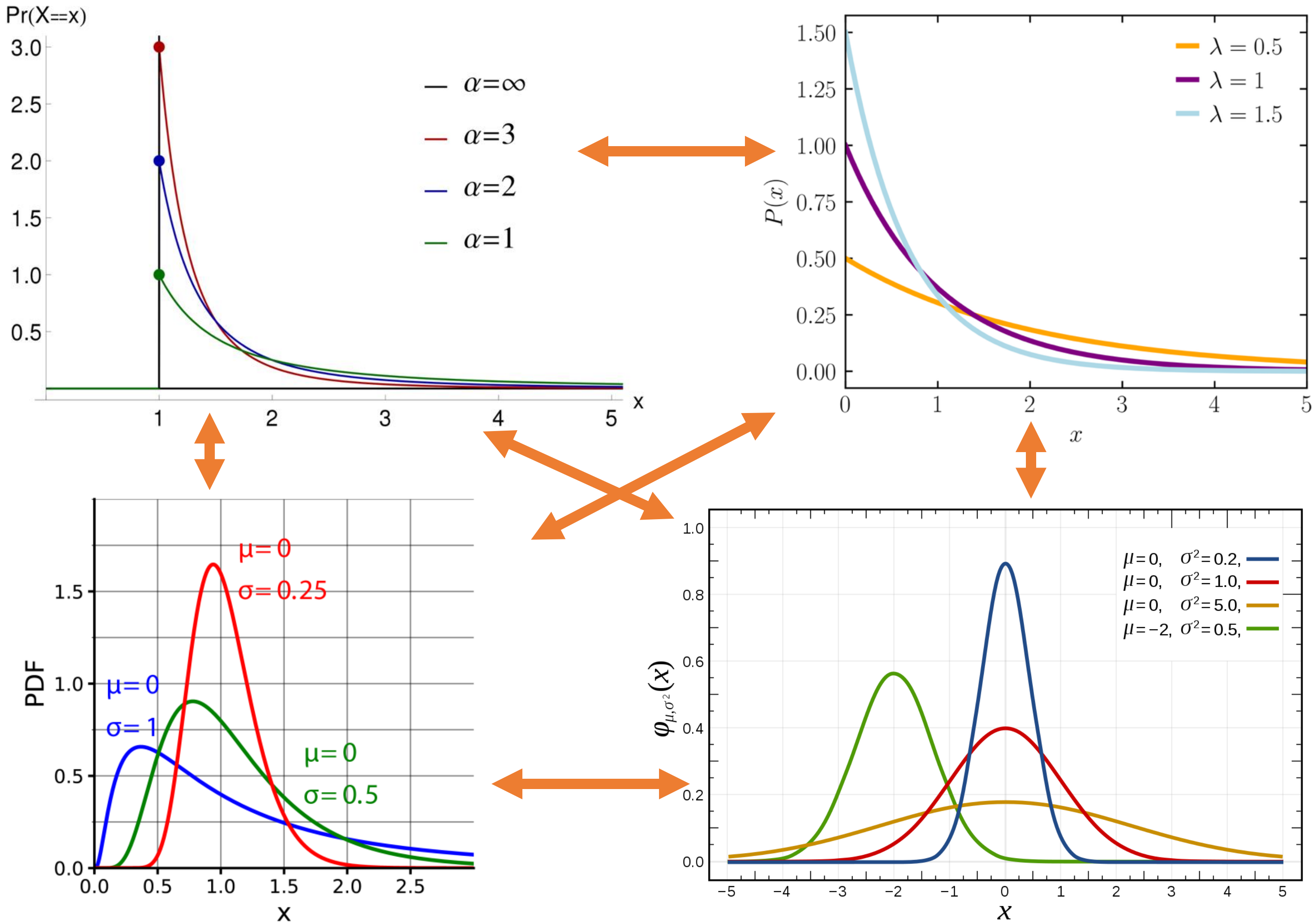
# Power laws: Example

- Order words by frequency in a large number of documents

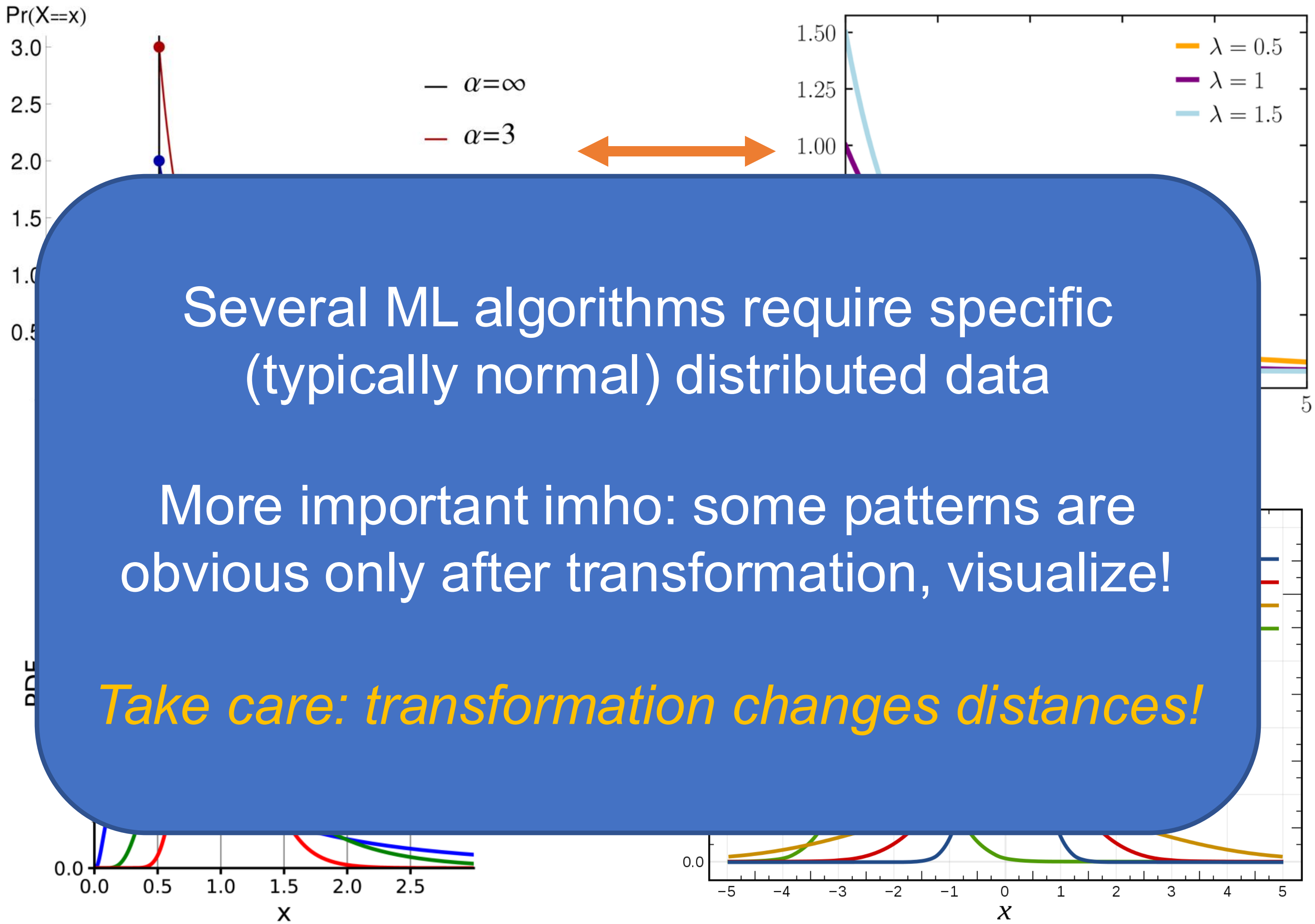

To use such data, we often need to lessen the effect of very large values

Common trick:
***apply log()***

# Why transform the distribution?



# Why transform the distribution?




Several ML algorithms require specific (typically normal) distributed data

More important imho: some patterns are obvious only after transformation, visualize!

*Take care: transformation changes distances!*

# Box-Cox transform

- There exist transformation between several distributions, e.g.
  - normal <-> lognormal
  - Pareto <-> exponential

- We can also try to make any data normally distributed via optimization:
  - Find a λ for $$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

  that maximizes the likelihood = P(data | model), typically using a normal distribution as model

# Box-Cox transform

This is called a power transform, a family of monotonic transformations using power functions

- There exist transformation bet
  - normal <-> lognormal
  - Pareto <-> exponential
- We can also try to make any data normally distributed via optimization:
  - Find a λ for $$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

  that maximizes the likelihood = P(data | model), typically using a normal distribution as model

Box-Cox can model several transforms:

$\lambda$ = 1.00: original data
$\lambda$ = 0.50: square root transformation
$\lambda$ = 0.33: cube root transformation
$\lambda$ = 0.00: log transformation
$\lambda$ = -0.50: inverse square root transformation
$\lambda$ = -1.00: inverse transformation
…

called a power
orm, a family of
ic transformations
power functions

optimization:

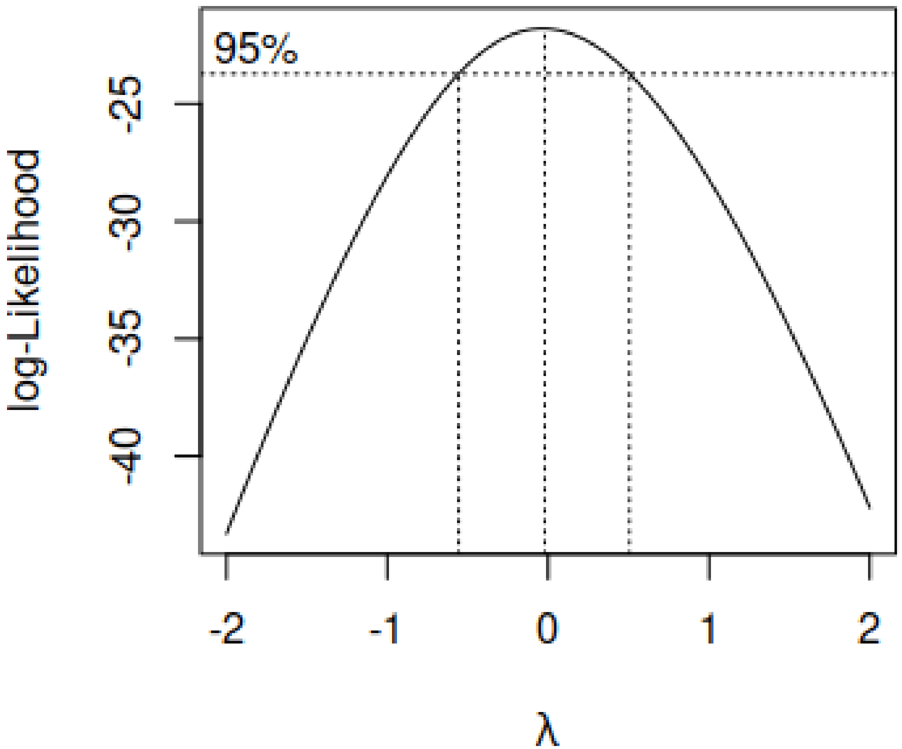
- Find a λ for

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

that maximizes the likelihood = P(data | model), typically using a normal distribution as model

# Box-Cox transform

- How to optimize λ? Try all, pick the maximum



# Quantile transform

- Another way to manipulate data distributions:
  - Compute the rank R(x) (percentile) for data point x
  - Assume a distribution D(), e.g., normal, over a fixed range (0,1)
  - Every point x′ in (0,1) has a rank $R_D(x')$ under D()
  - Map x to x′ such that $R(x) = R_D(x')$
- You can apply this for any distribution, typically normal and uniform are used

# Feature range

- Why can different feature ranges be a problem?

| V1 | V2 |
|---|---|
| 0 | 1000 |
| 2 | 2000 |
| 4 | 1200 |
| 1 | 2800 |
| 2 | 1600 |

# Feature range

- Why can different feature ranges be a problem?
  - Computing distances becomes problematic
  - Feature weights are not very meaningful
  - Loss/errors are problems, especially when squared!

| V1 | V2 |
|---|---|
| 0 | 1000 |
| 2 | 2000 |
| 4 | 1200 |
| 1 | 2800 |
| 2 | 1600 |

# Feature range

- Why can different featu
  - Computing distances b
  - Feature weights are nc
  - Loss/errors are probler

| V1 | V2 |
|---|---|
| 0 | 1000 |
| 2 | 2000 |
| 4 | 1200 |
| 1 | 2800 |
| 2 | 1600 |

Main Solutions

Min-Max Scaling

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Z-score normalization

$$Z = \frac{x - \mu}{\sigma}$$

# Row normalization

- Even after normalizing features, rows have different sizes
  - Row normalization “fixes” this by ensuring every row’s norm is the same (i.e., sums to 1, or using a different norm)

| V1 | V2 |
|---|---|
| 0 | 0 |
| 0.5 | 0.55 |
| 1 | 0.11 |
| 0.25 | 1 |
| 0.5 | 0.33 |

TU Delft

# Row normalization

- Even after normalizing features, rows have different sizes
  - Row normalization "fixes" this by ensuring every row's norm is the same (i.e., sums to 1, or using a different norm)

| V1 | V2 |
|---|---|
| 0 | 0 |
| 0.5 | 0.55 |
| 1 | 0.11 |
| 0.25 | 1 |
| 0.5 | 0.33 |

→

| V1 | V2 |
|---|---|
| 0.5 | 0.5 |
| 0.48 | 0.52 |
| 0.9 | 0.1 |
| 0.2 | 0.8 |
| 0.6 | 0.4 |

# What transforms to perform?

- This depends on two things:
  1. Your data
  2. Your pipeline (classifier/clusterer/…)
- If your algorithm requires data from a normal distribution, try using a power or quantile tranform to make it normal
  - *Visualize the outcome! Does it make sense?*
- Keep in mind that when you compute distances, any such transform can radically change them…
- Also be careful about the interpretation (large may acutally be small)…

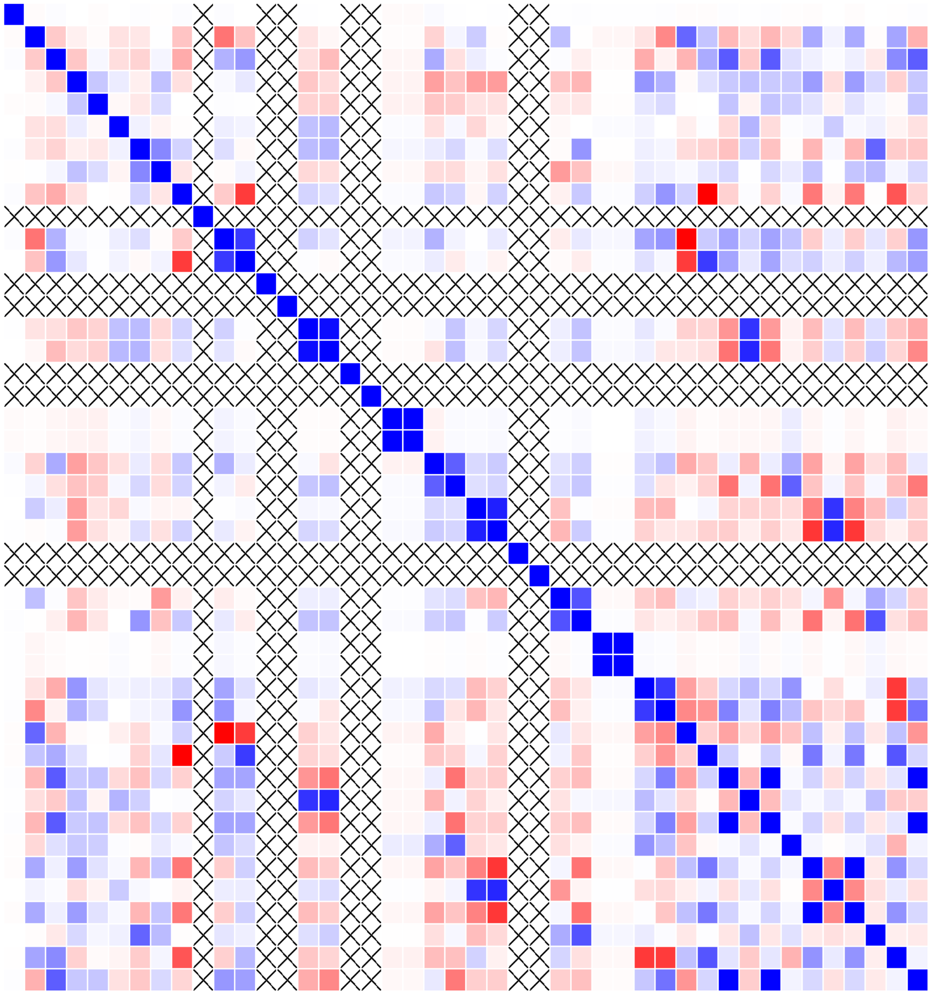
# Let’s look at Lab1 Data

- …

# Dependence between features

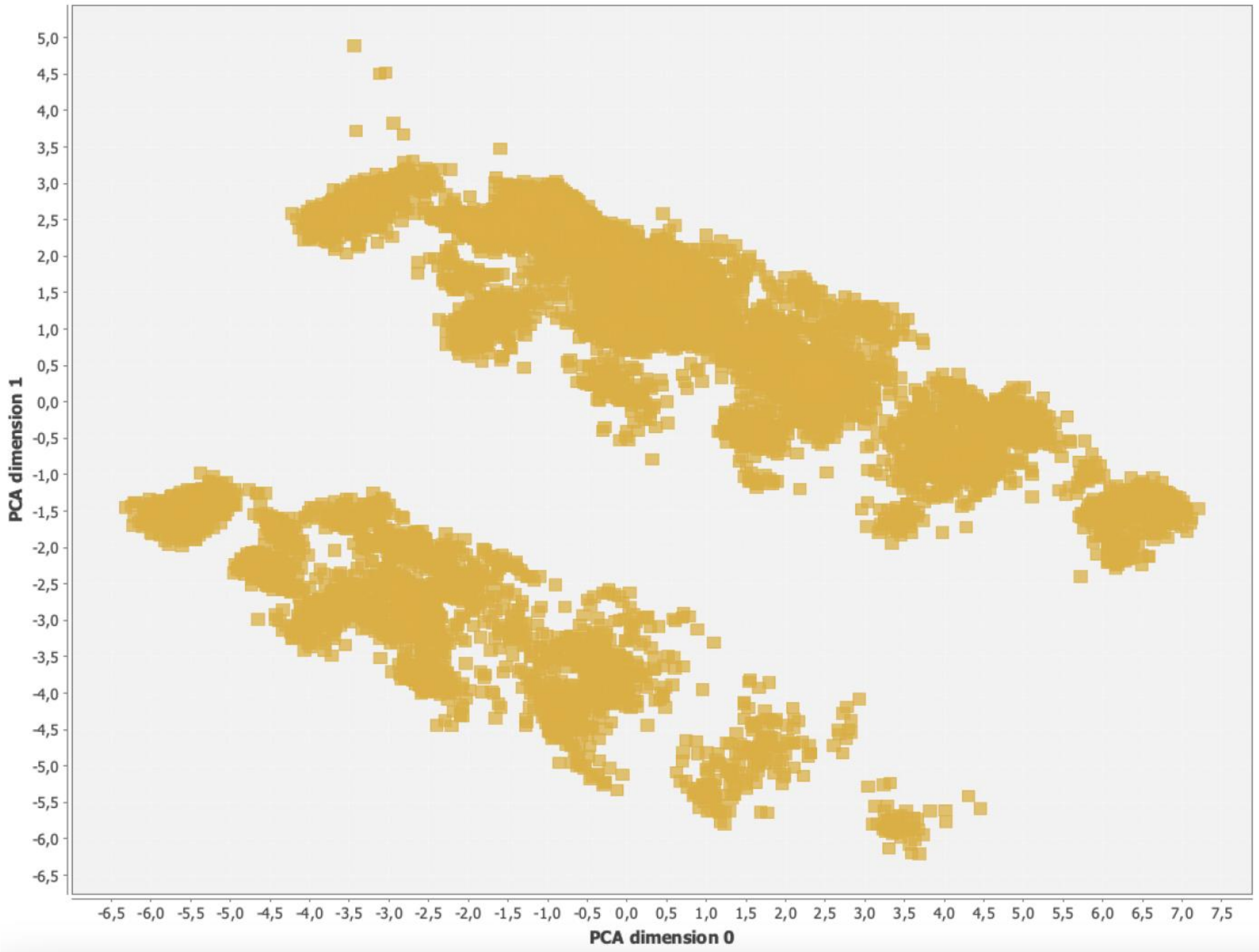
# Correlation between signals

- Red: negative
- Blue: positive

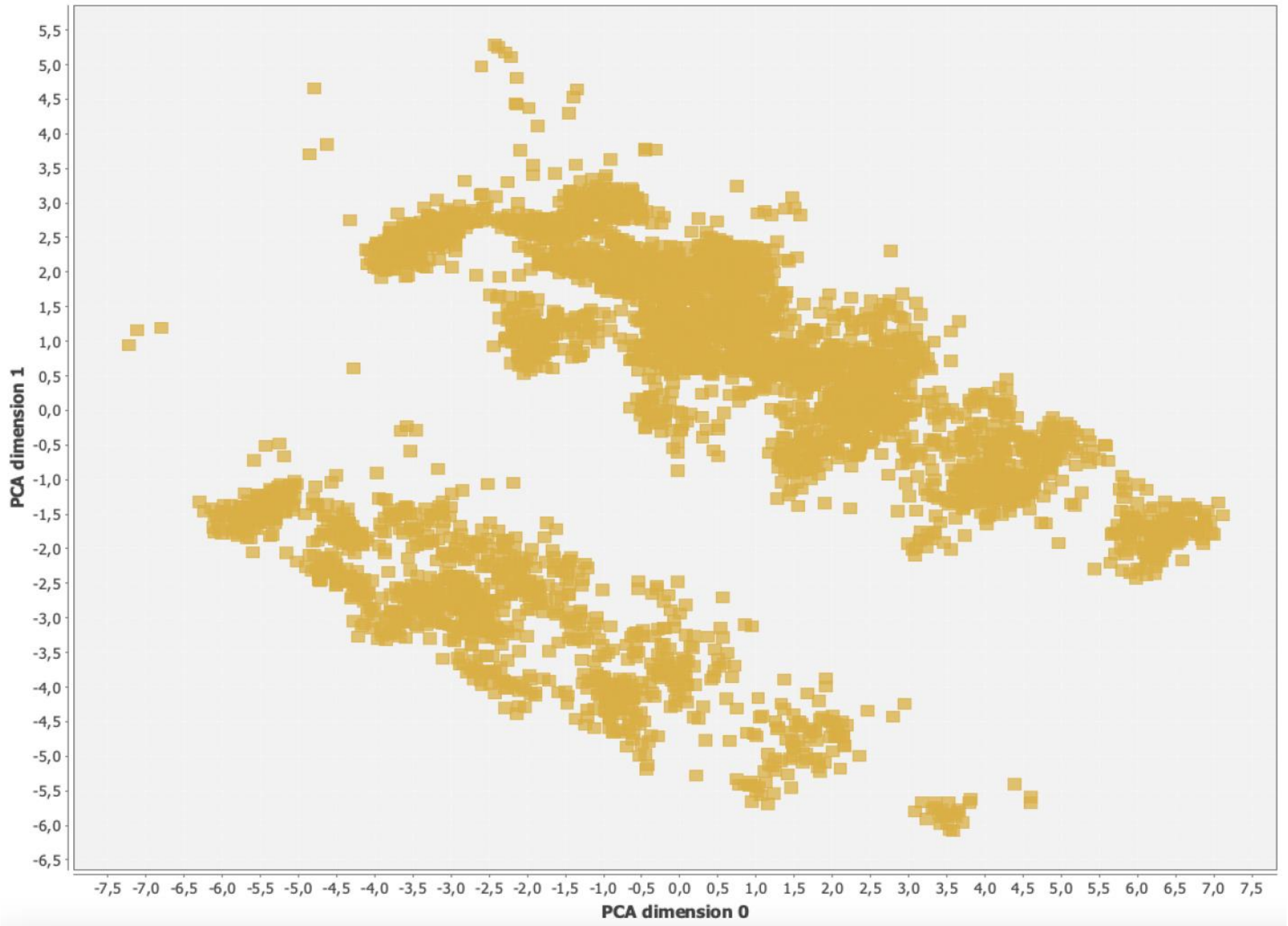


- What does it tell us?

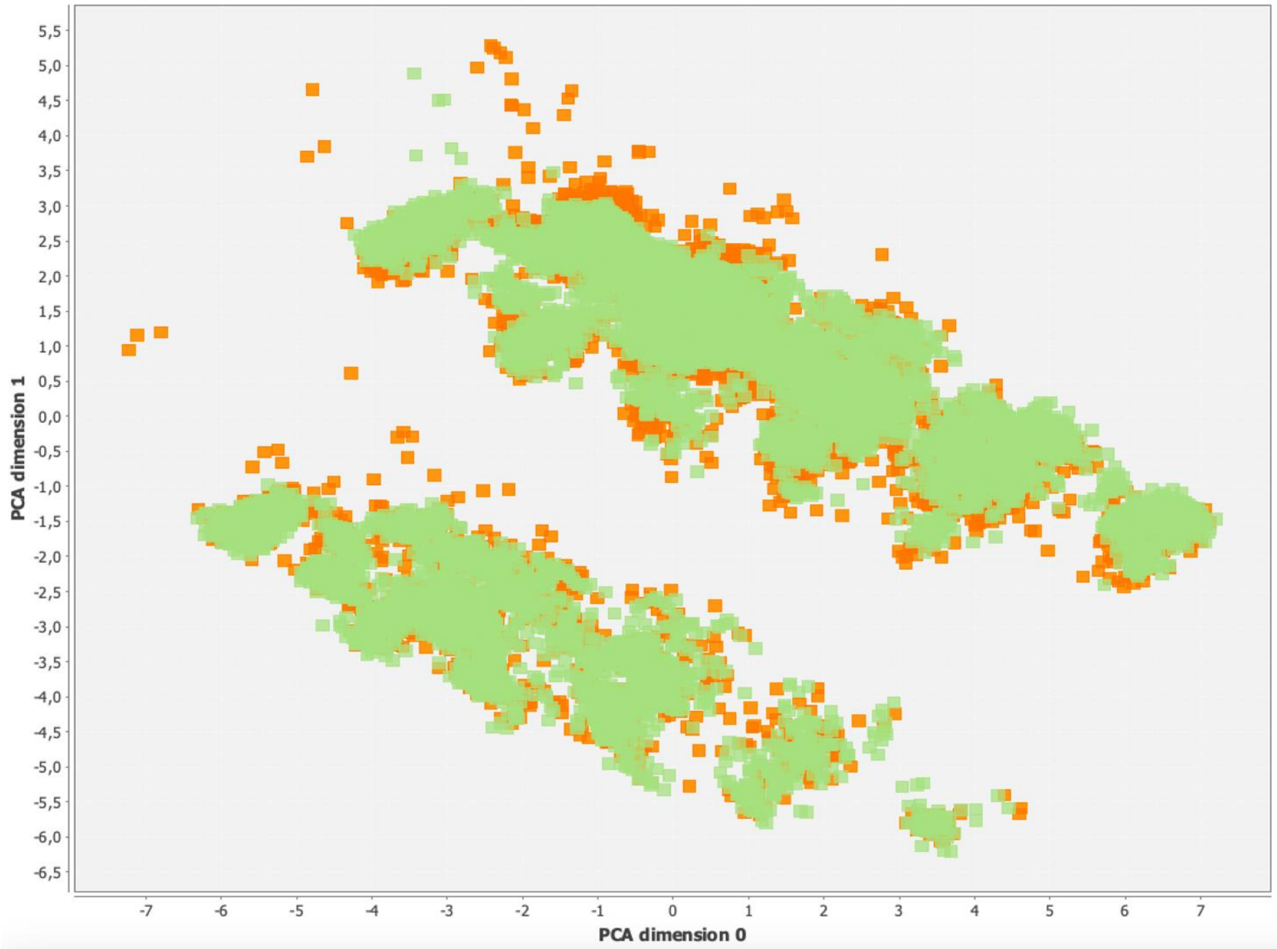
# Embed into two dimensions



# Embed also the test set!



# Notice differences? Are they anomalies?



# Wait a minute… Did I cheat?

- I looked at the test data!

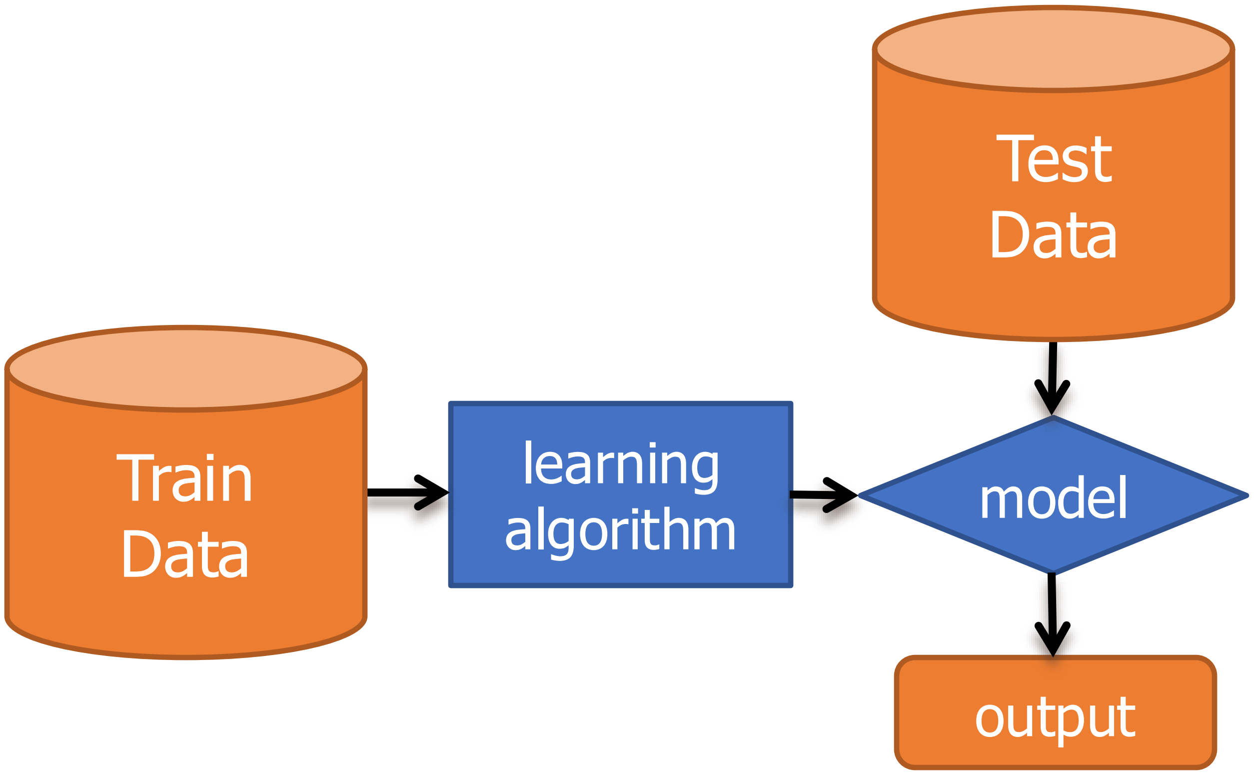
# Wait a minute… Did I cheat?

- I looked at the test data!
- Big question:
  - Are the test data available when learning a model?
  - Some people say yes, some say no … it depends …
- Traditional anomaly detection methods learn only from test data
- Outlier detection aims to find data points that are different
- In the lab, we have test data available, you may use it, but never ever use the test labels during model training

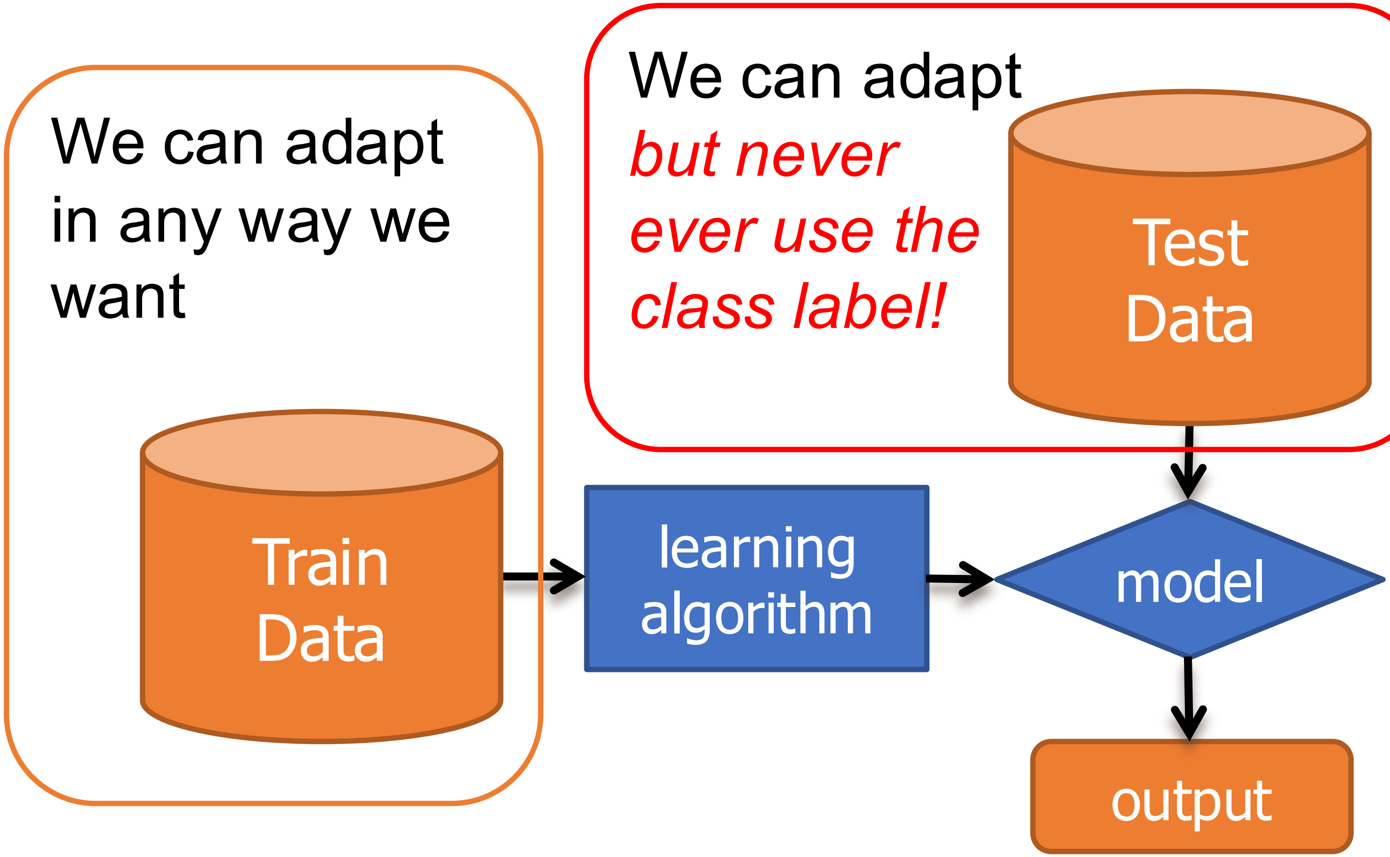
# Wait a minute… Did I cheat?

- I looked at the test data!
- Big question:
  - Are the test data available when learning a model?
  - Some people say yes, some say no … it depends …
- Traditional anomaly detection m
- Outlier detection aims to find d
- In the lab, we have test data a
  never ever use the test labels d

1. Team up
2. Investigate data
3. Transform if needed
4. Learn models
5. Detect deviations
6. Raise alarms




What can we modify?



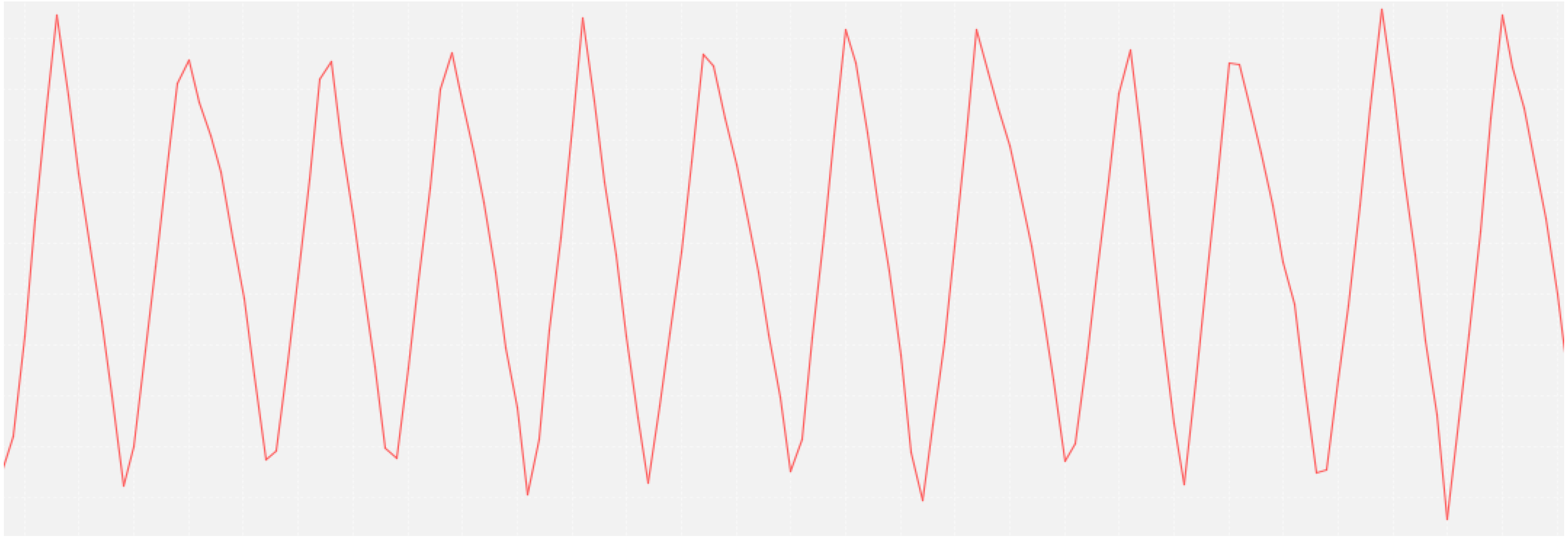

What can we modify?

# Understanding the data

- Make plots and tables that show:
  - Features and their distribution
    - *shows what kind of feature processing to use*
  - Dependence between features
    - *shows what kind of feature processing to use*
  - Dependence over time
    - *shows the type of temporal processing to use*
  - The difficulty of the problem
    - *shows what technique might be suitable*

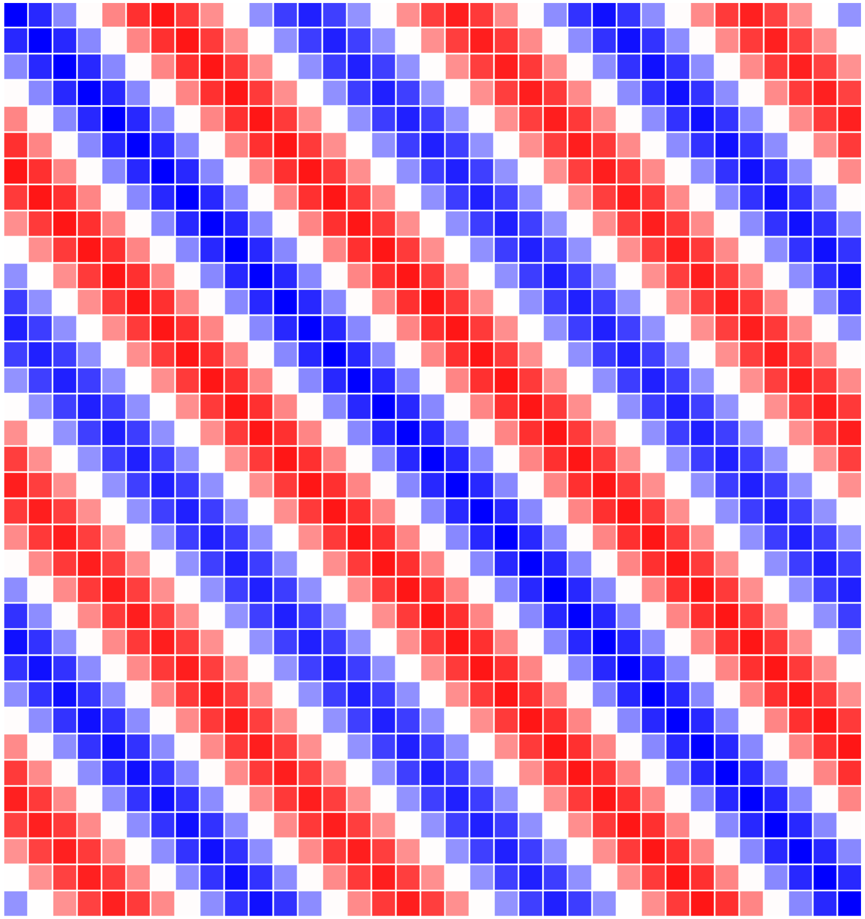
# Temporal dependence

# One signal plotted over time



- Shows what?

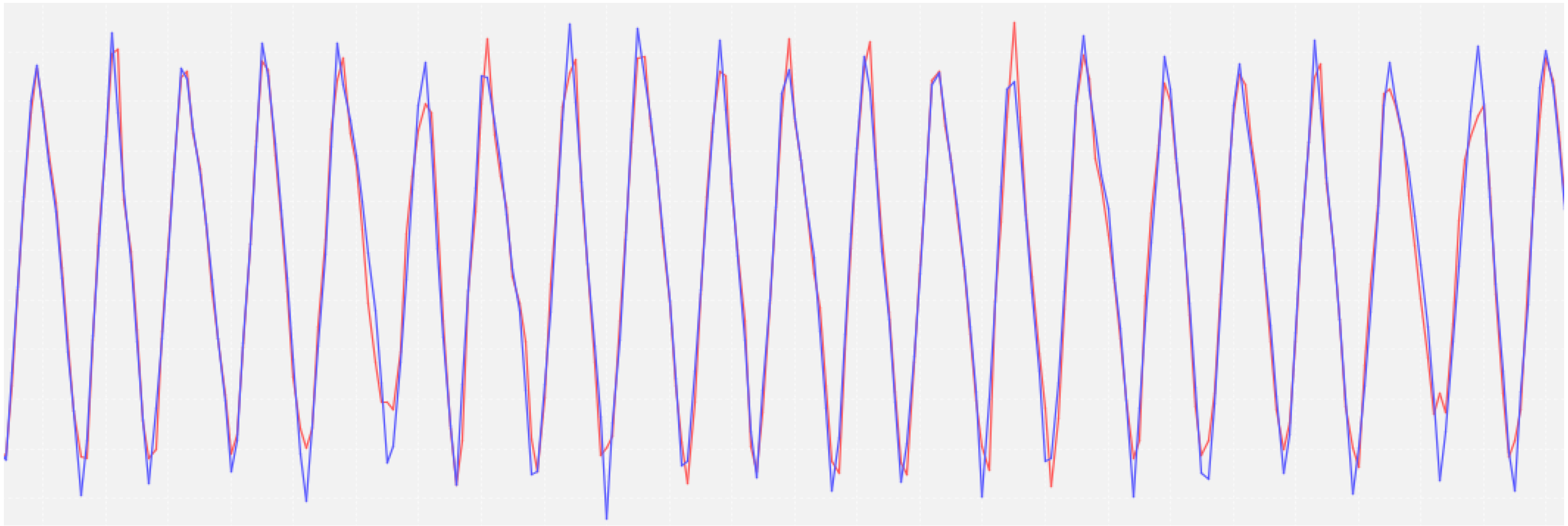
# A signal's self-correlation



- Red: negative
- Blue: positive

- What to conclude?

# Predict using linear regression



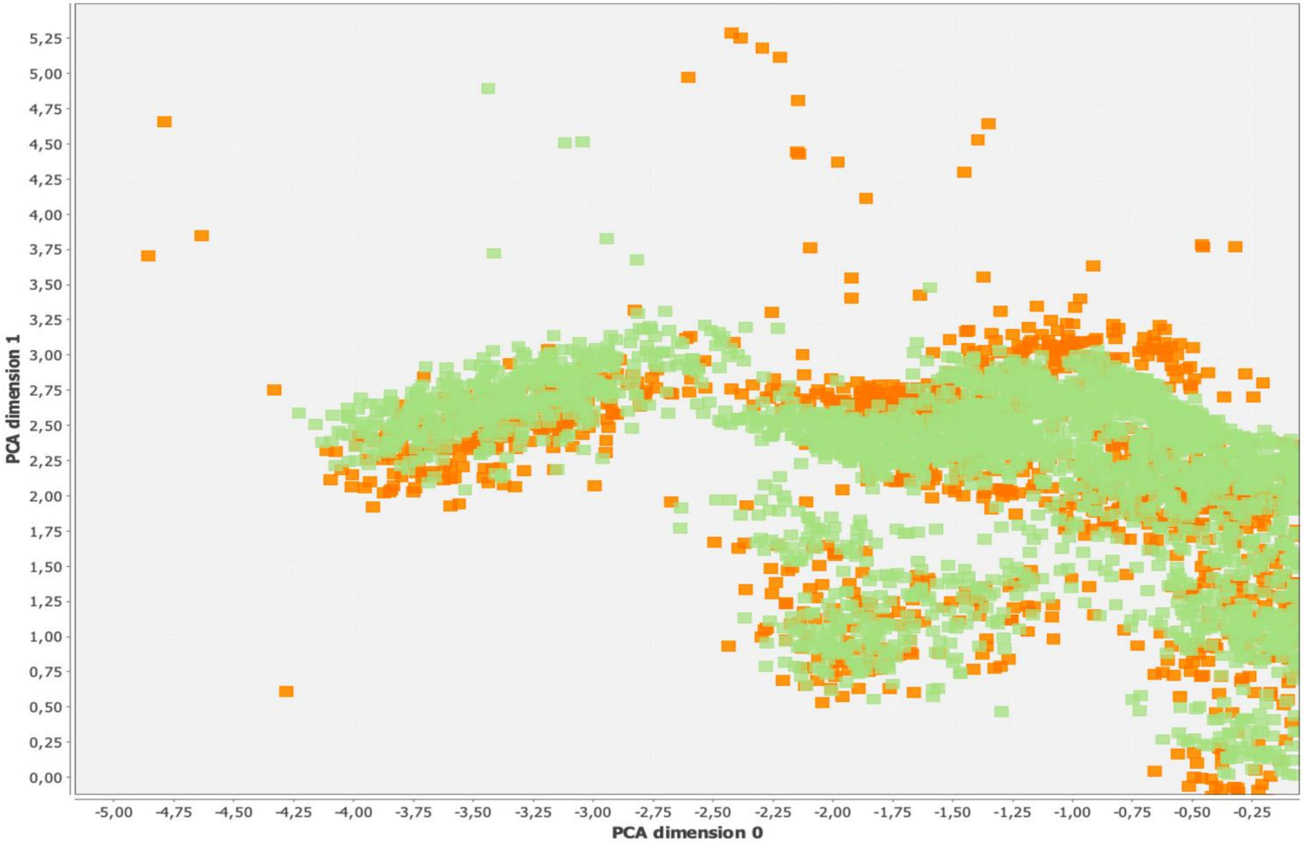
- Seems not too hard…

# Let's look at Lab1 Data

- ...

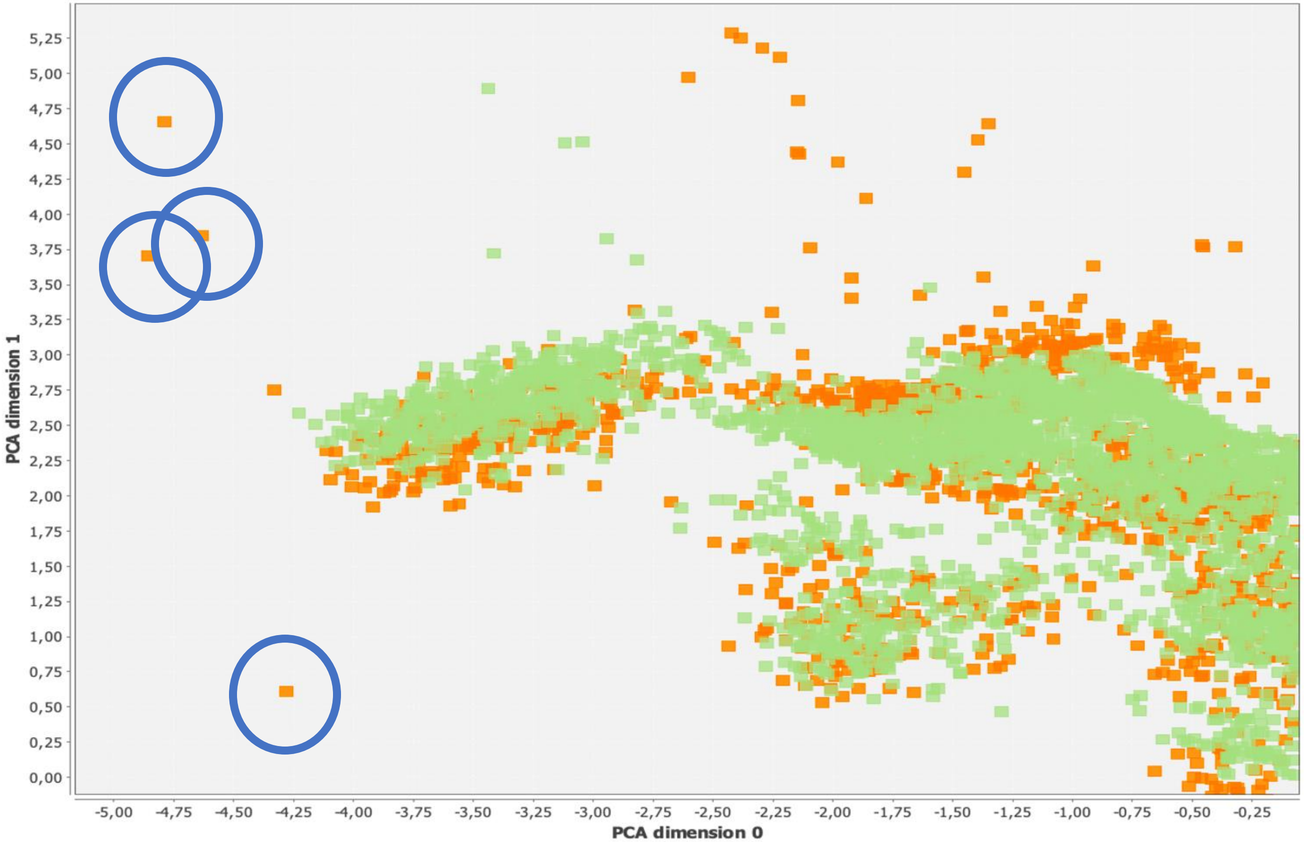
# Point Anomalies

- An individual data instance is anomalous w.r.t. the data



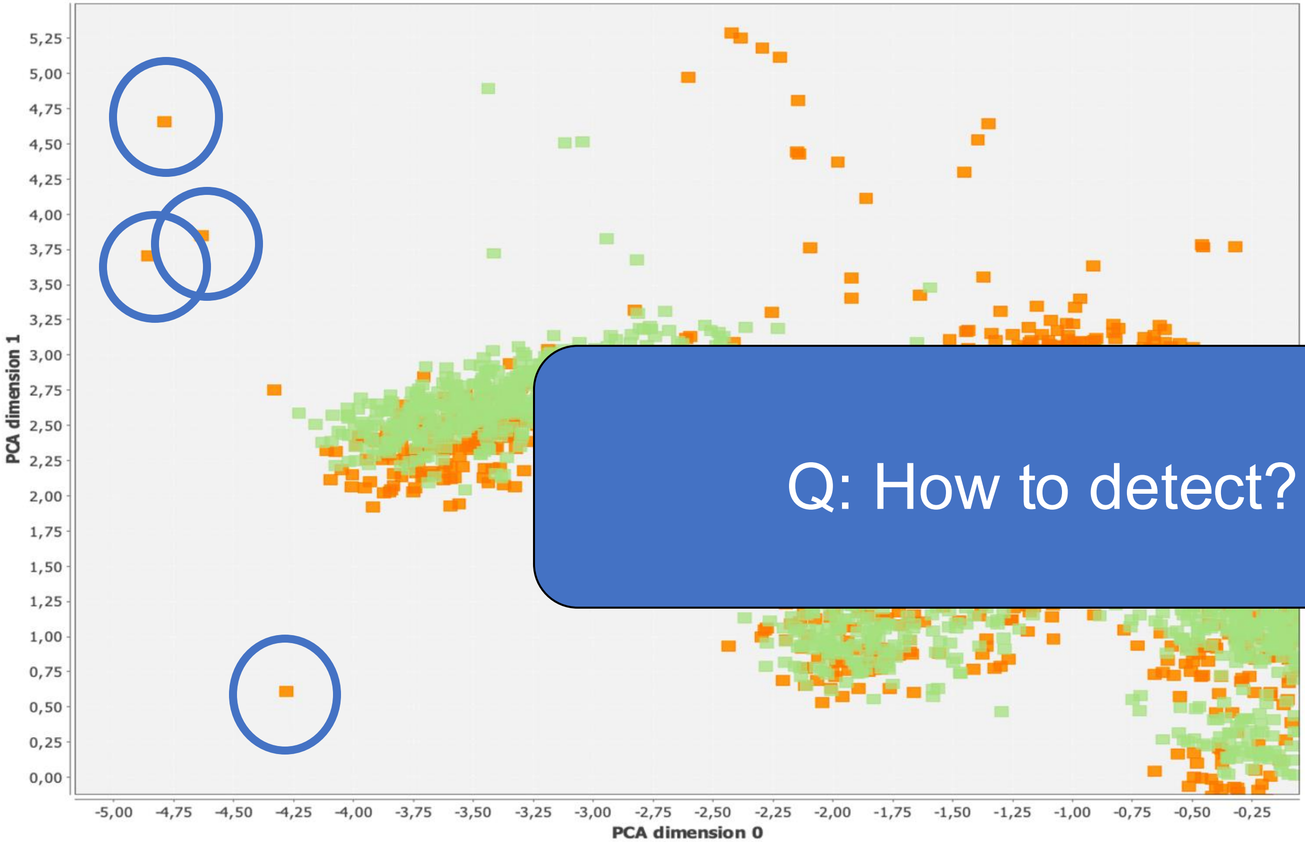
# Point Anomalies

- An individual data instance is anomalous w.r.t. the data



# Point Anomalies

- An individual data instance is anomalous w.r.t. the data



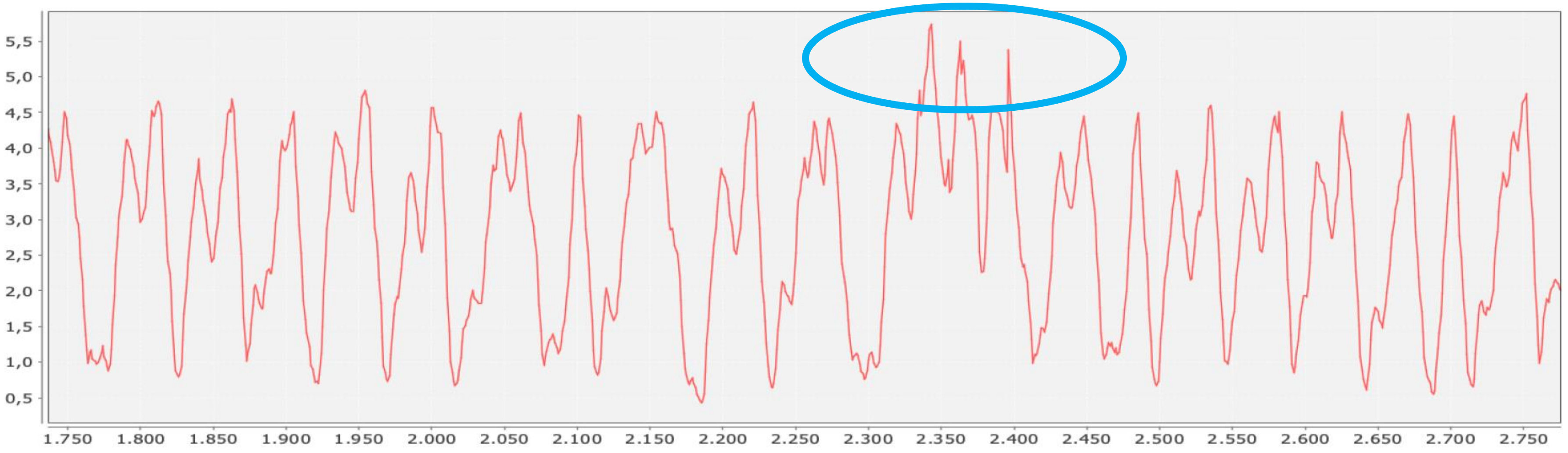

Q: How to detect?

# Detecting point anomalies

- Classification
  - learn from unlabeled data (one-class classification)
  - tests whether the point lies in an empty part of the input space
- Distance-based
  - tests whether the distance to the nearest neighbors is normal
- Data reconstruction
  - tests whether each feature value is normal given all other feature values
- In Lab 1, you will implement distance-based anomaly detection, and anomaly detection based on data reconstruction
  - more details coming weeks…

# Contextual Anomalies

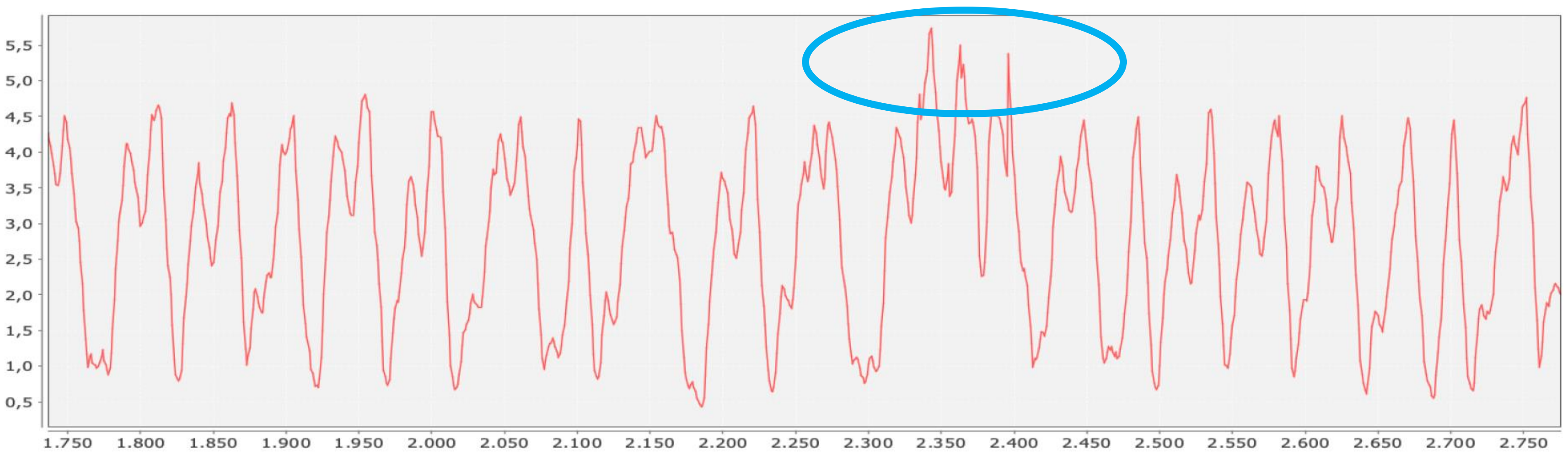
- An individual data instance is anomalous within a context
- *Context are other data points!*
  - surrounding data in time or space, but can be in other features




- Are these contextual anomalies?

# Contextual Anomalies

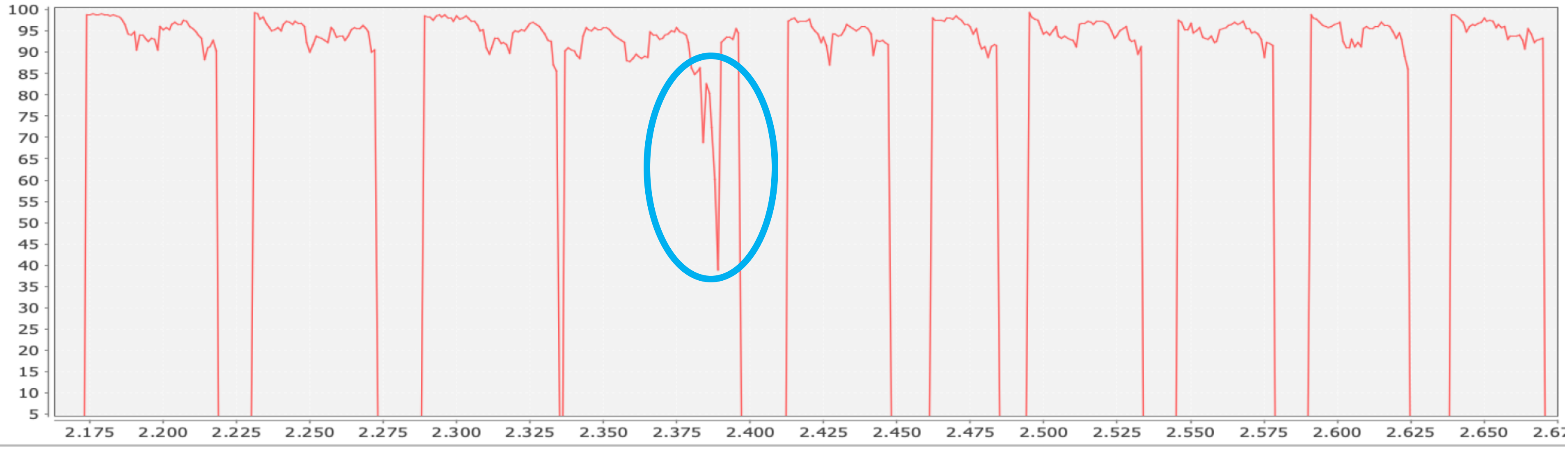
- An individual data instance is anomalous within a context
- *Context are other data points!*
  - surrounding data in time or space, but can be in other features




- No, they are out of normal range, thus point-anomalies

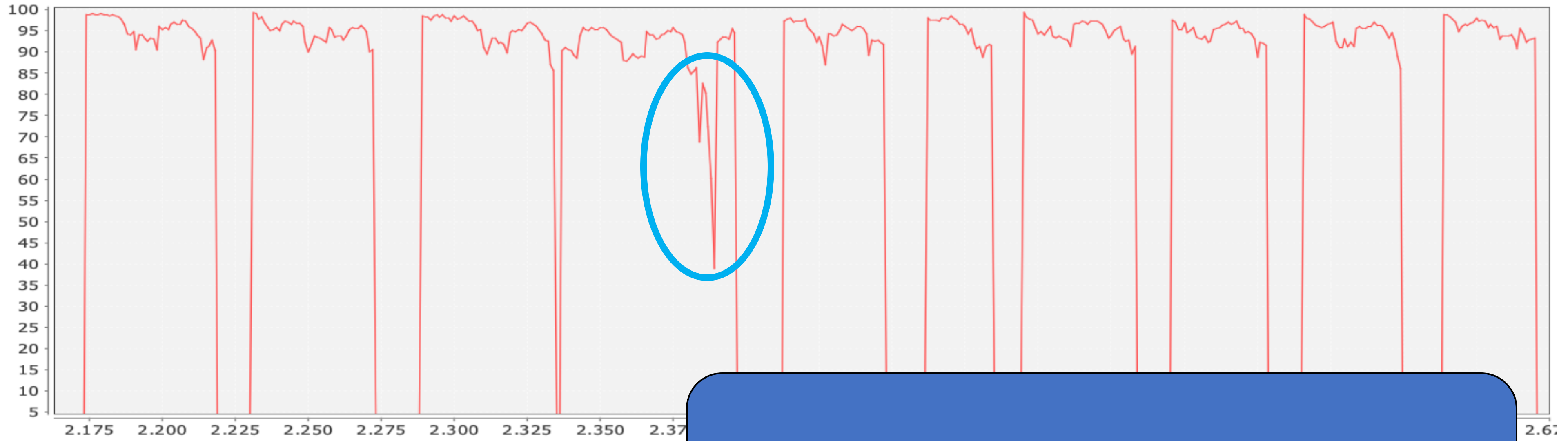
# Contextual anomalies

- *Normal data points, but strange given the surrounding data*



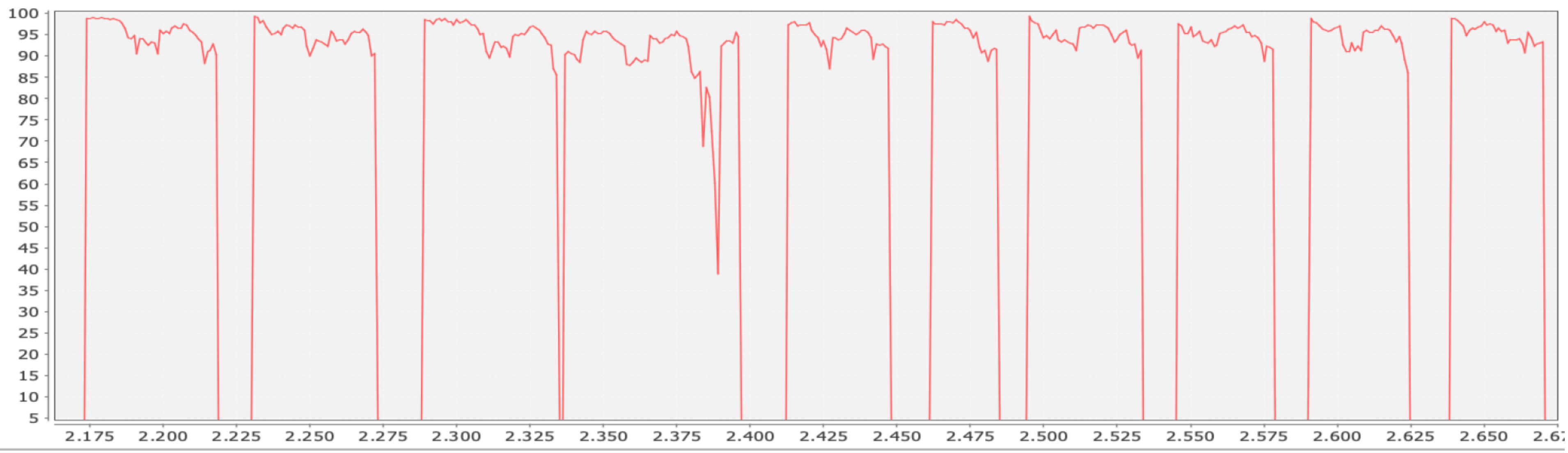
# Contextual anomalies

- *Normal data points, but strange given the surrounding data*



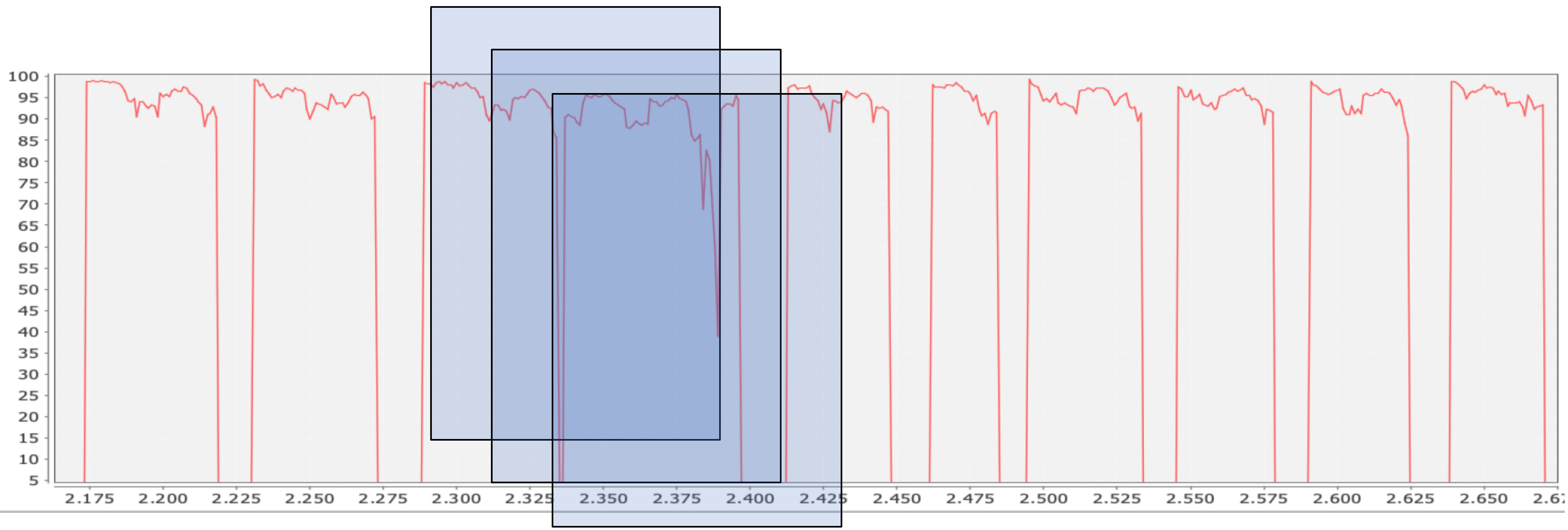

Q: How to detect?

# What to use as context C?




TU Delft

# What to use as context C?




The typical approach is to use sliding windows, and predict $x_t$ from $x_{t-n}$ … $x_{t-1}$

# Does this detect the anomaly?

- I used sliding windows of length 5, predicting the last point using linear regression

# Compute the residual

- Use sliding windows from training data to learn a model f for predicting the next value:

$$y_k = f(y_{k-1}) + \epsilon$$

- Compute the expected next value

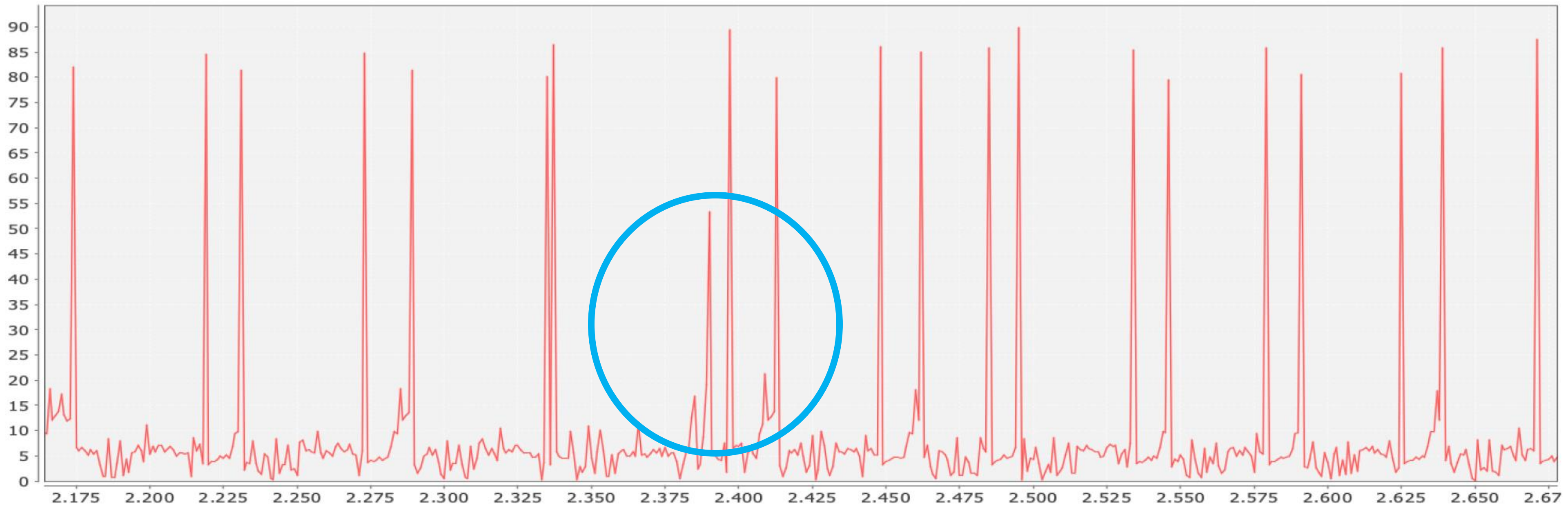
$$\hat{y}_{k|k-1} = f(y_{k-1})$$

- Evaluate the residual using the real next value

$$r_k = y_k - \hat{y}_{k|k-1}$$

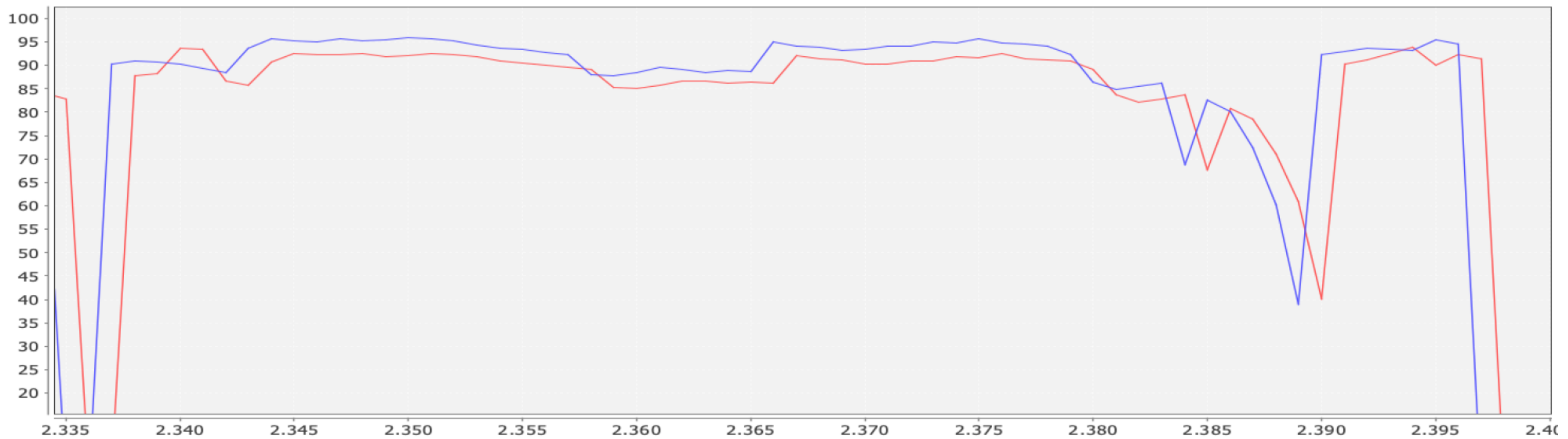
- Use a decision threshold, typical:
  - 2 or 3 times the standard error
  - or simply sort on residual error and return largest ones

# The residual



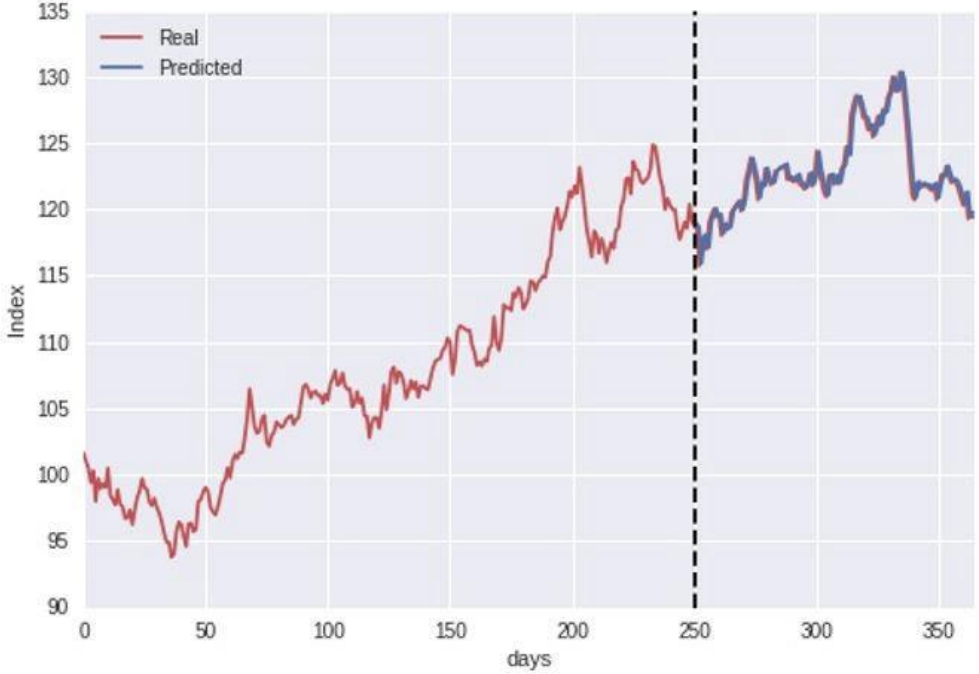

- No threshold would detect this, or give many false positives

# Zoomed in



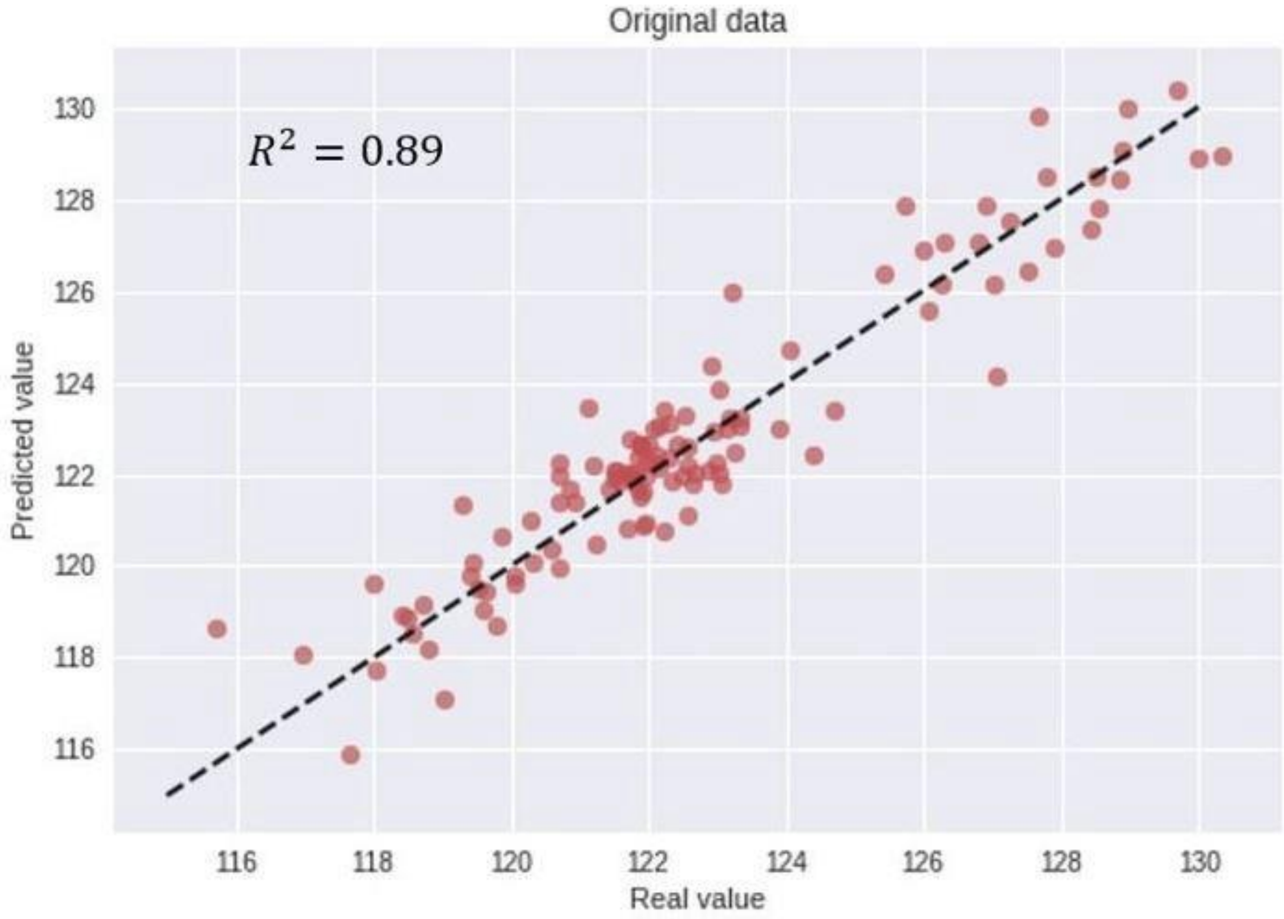

What is going on?

# Pitfall: predictions



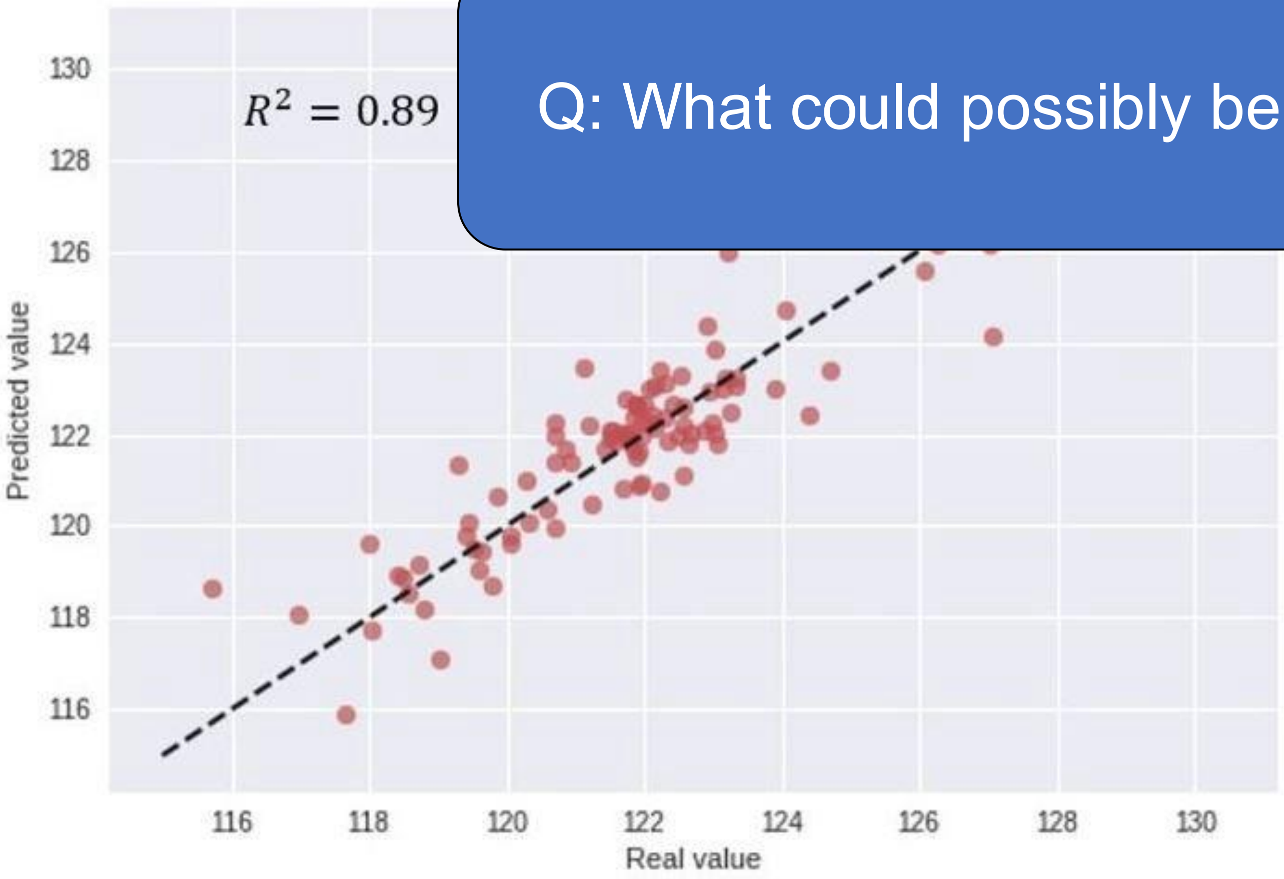

from https://www.kdnuggets.com/2019/05/machine-learning-time-series-forecasting.html

# Pitfall: predictions




from https://www.kdnuggets.com/2019/05/machine-learning-time-series-forecasting.html

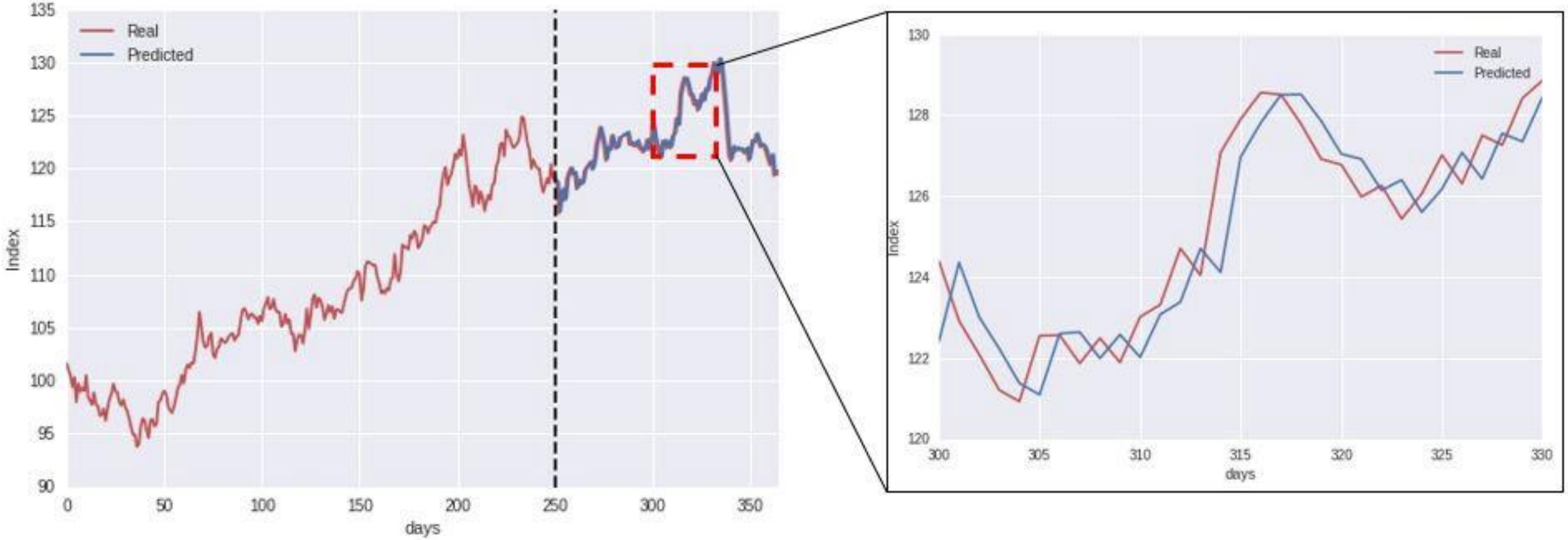
# Pitfall: predictions




Q: What could possibly be wrong?

from https://www.kdnuggets.com/2019/05/machine-learning-time-series-forecasting.html

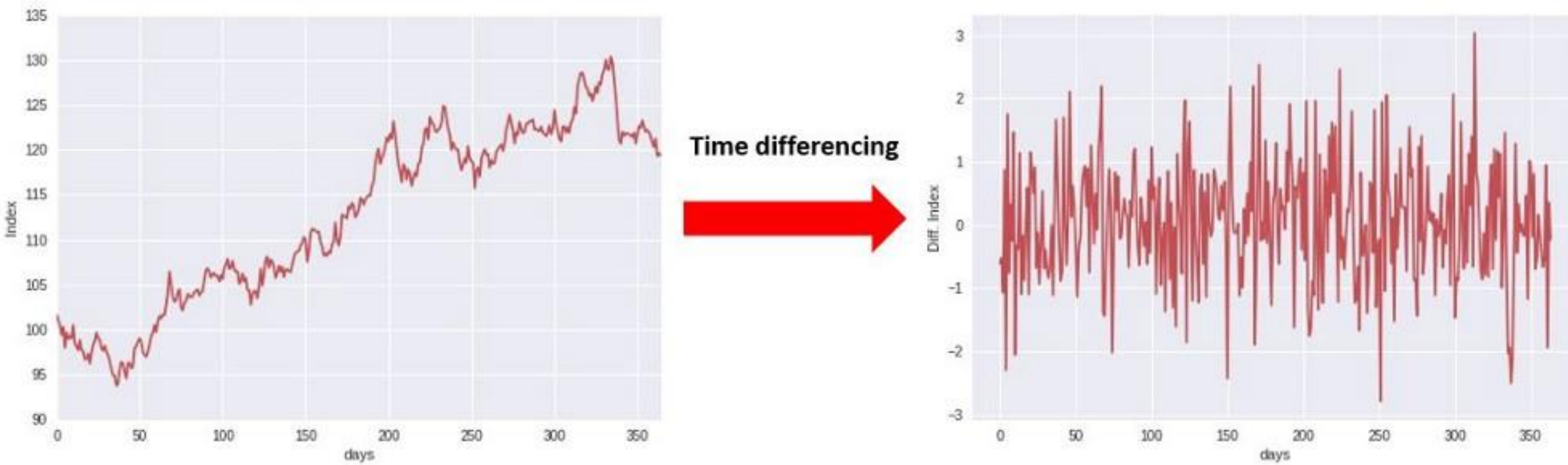
# Results: zoomed in – persistance!



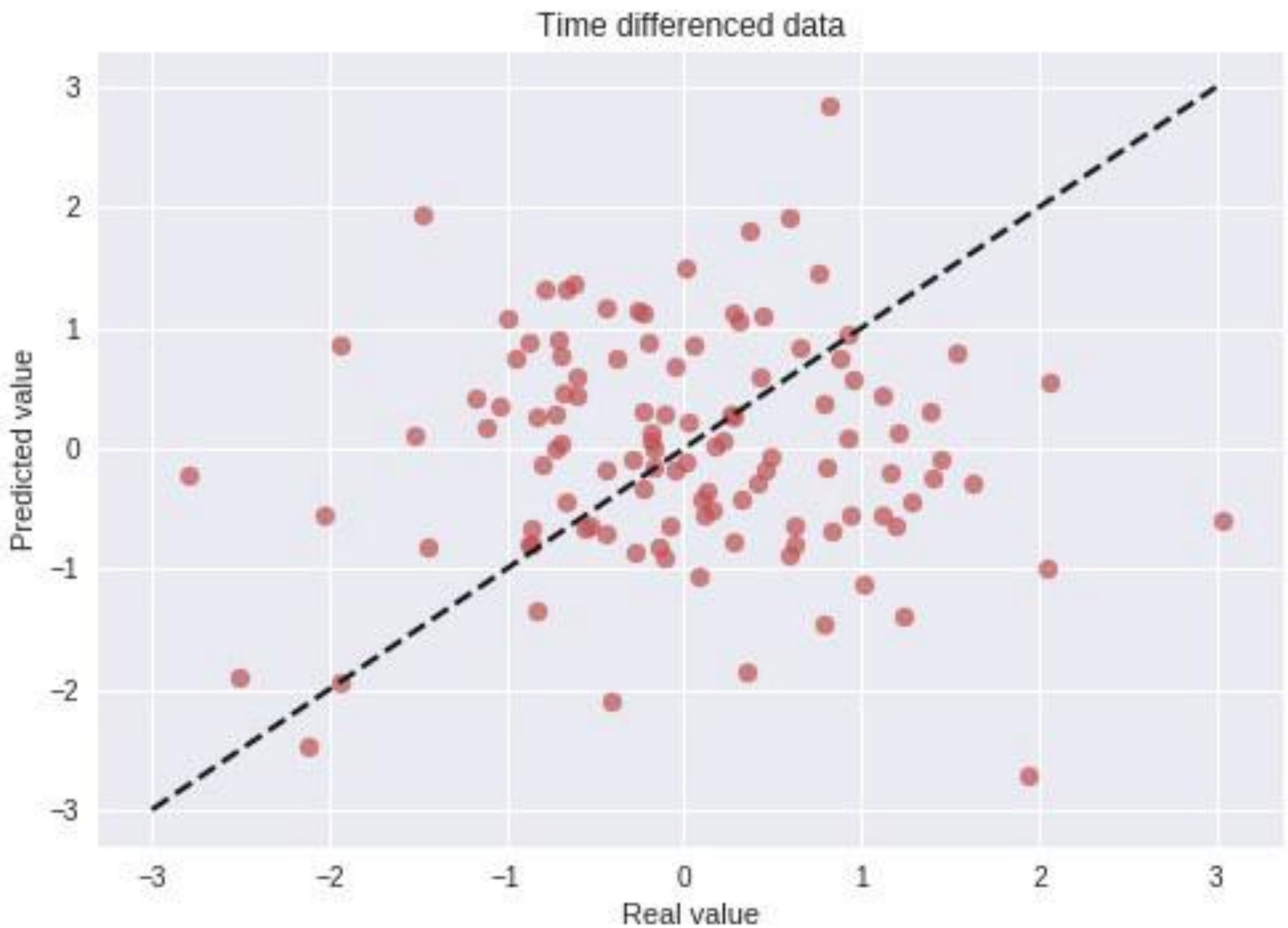

from https://www.kdnuggets.com/2019/05/machine-learning-time-series-forecasting.html

# Pitfall: temporal correlation

- $x_t$ is quite likely close to $x_{t-1}$
- Solution:
  - Differencing - $x_t := x_t - x_{t-1}$

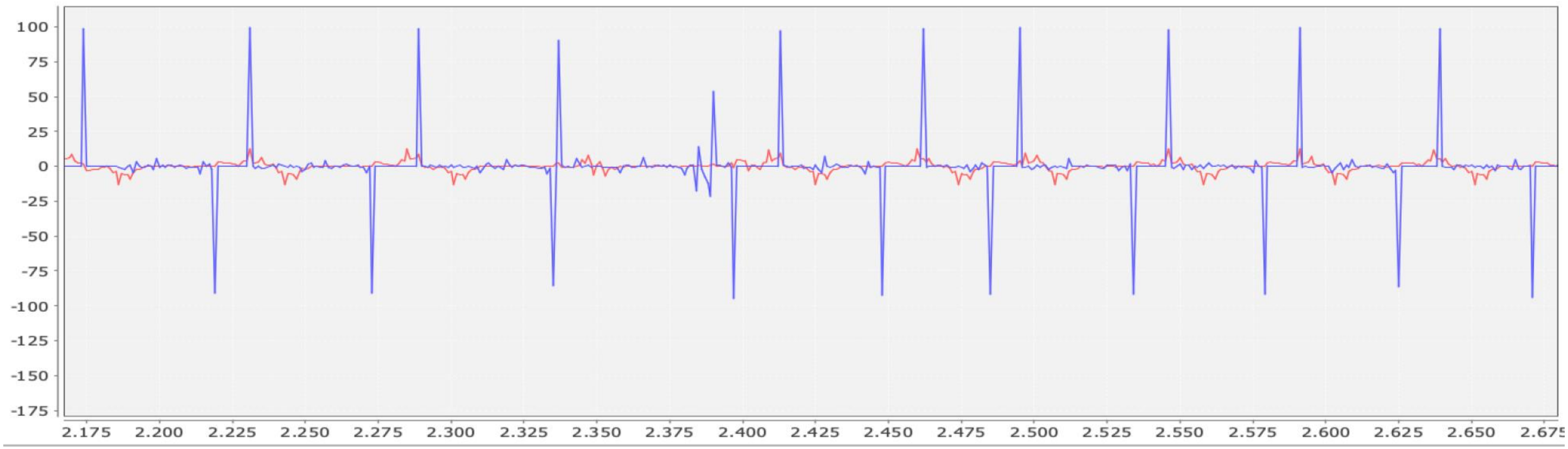


# Results: not better than random (in fact, the data are random!)



# So, what about our data?

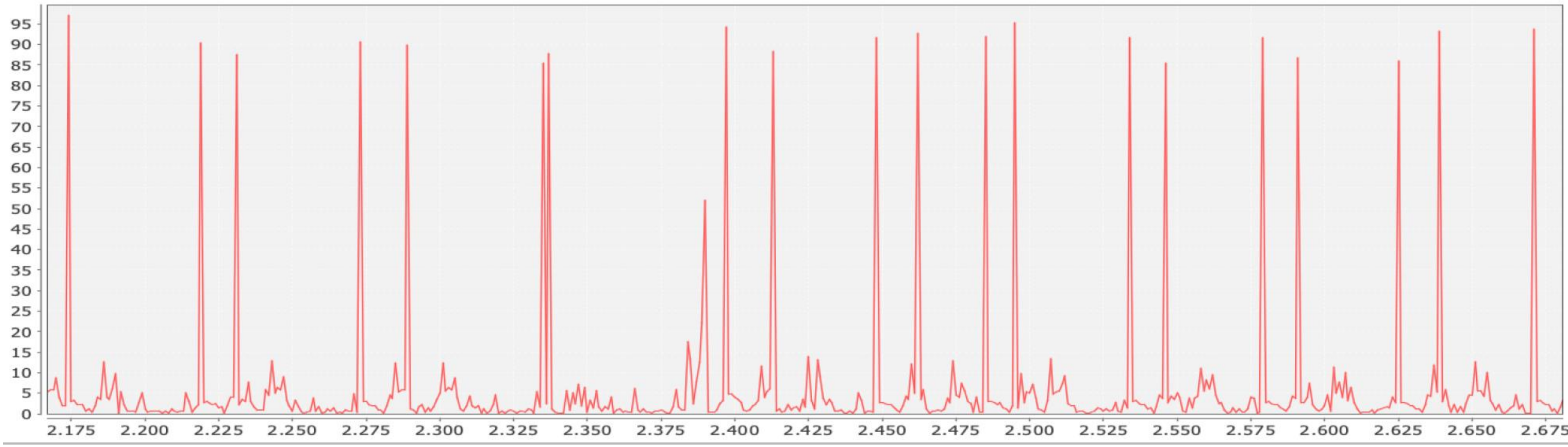
- Set sliding window a bit longer, length 20




- Although regular, predicting when a peak occurs is hard

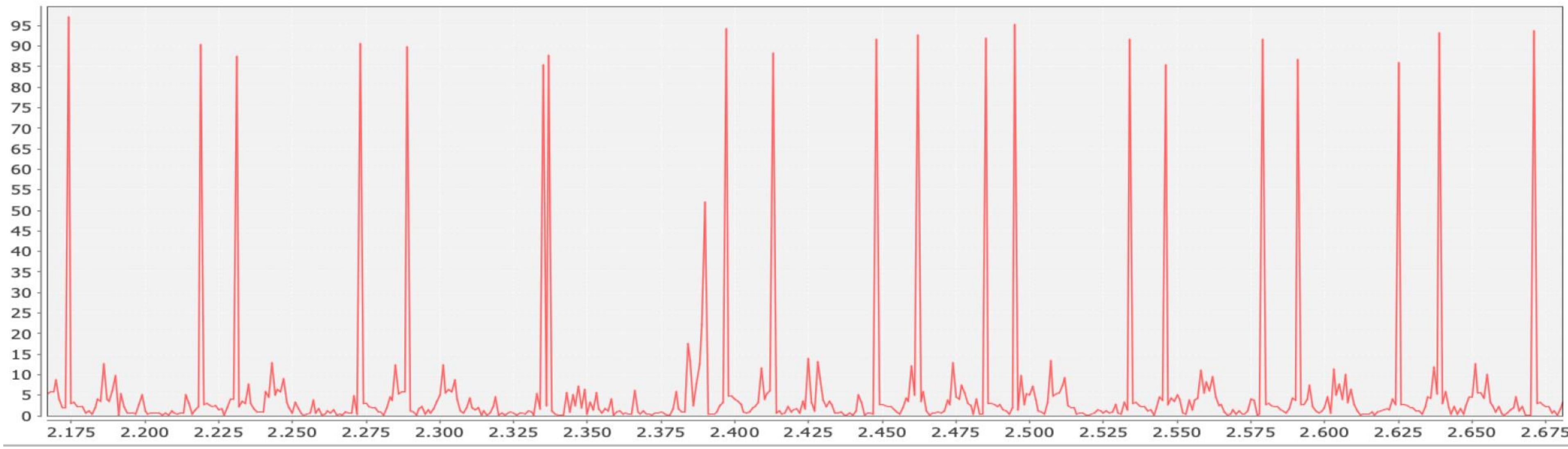
# So, what about our data?

- This shows the absolute errors



# So, what about our data?

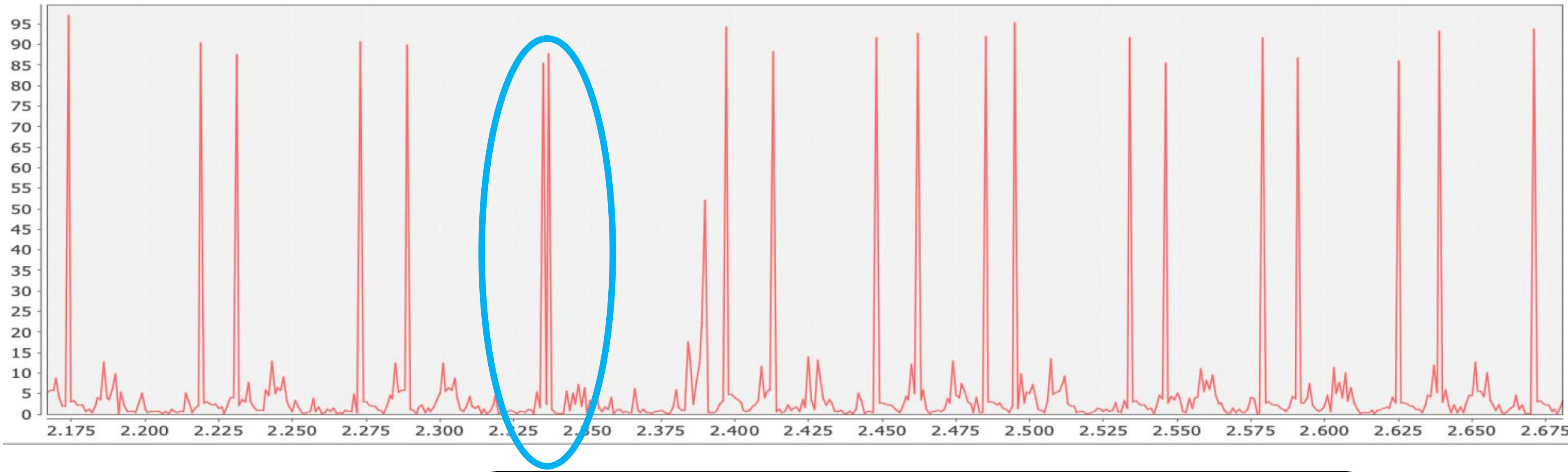
- This shows the absolute errors




Notice anything?

# So, what about our data?

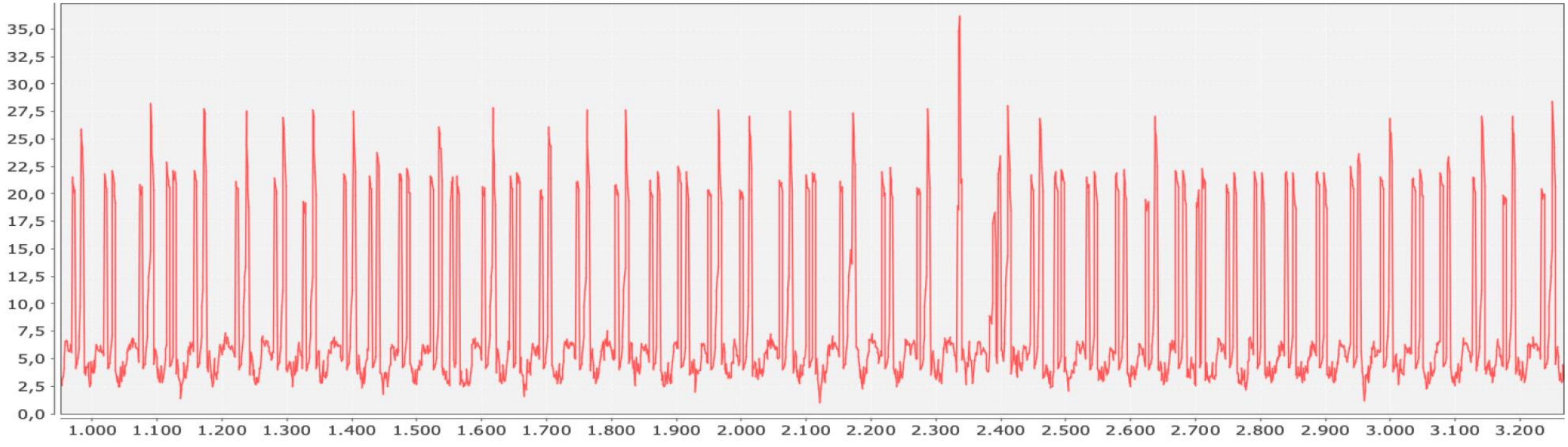
- This shows the absolute errors




These peaks are very near in time…

# Some postprocessing

- Let's plot the mean error over a larger window, say size 5
- Remove differencing, as this seemed not useful (Keep it simple)




- Simple linear regression can still be used to detect this anomaly!

# The initial data mining process

1. Visualize your data!
2. Try simple approaches. Is the problem difficult?

3. Look for something your system should be able to do
4. Look for problems in your setup, analyze it, fix if needed
5. Look for patterns, properties, features that could help
6. Add them to your system if helpful, but keep it simple!
7. Does it work? Goto 3
8. Does it not? Goto 4

9. Try and try again until satisfied, and hope it generalizes

# The initial data mining process

1. Visualize your data!
2. Try simple approaches. Is the problem difficult?

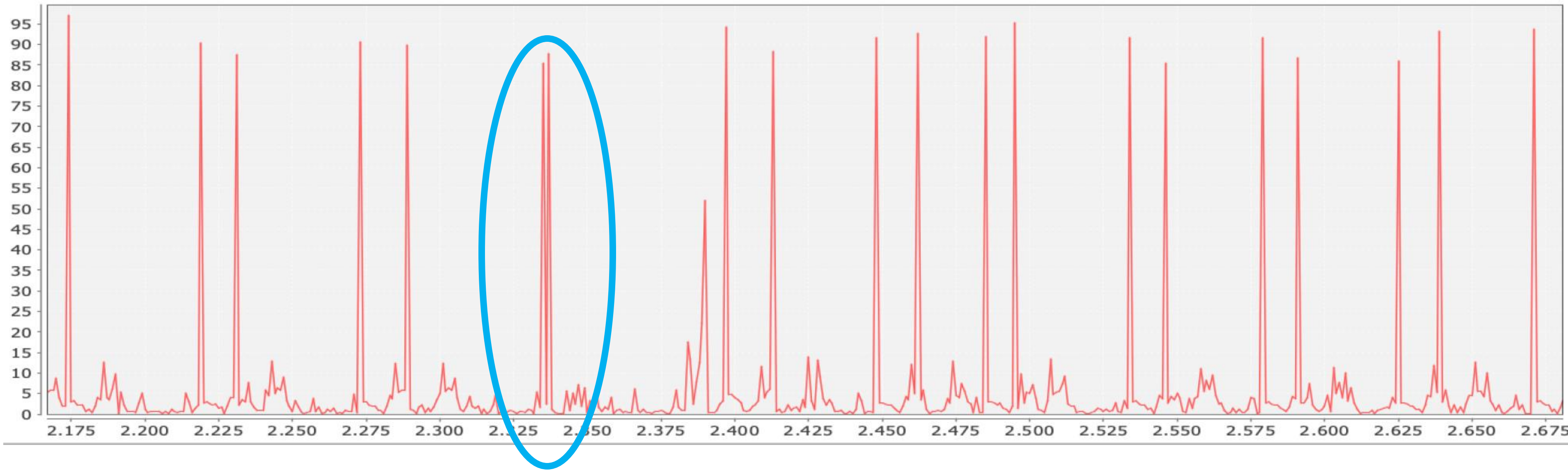
3. Look for something your system should be able to do
4. Look for problems in your setup, analyze it, fix if needed
5. Look for patterns, properties, features that could help
6. Add them to your system if helpful, but keep it simple!
7. Does it work? Goto 3
8. Does it not? Goto 4

9. Try and try again until satisfied, and hope it generalizes

This is your task for the first lab session in every assignment!

# Collective anomalies

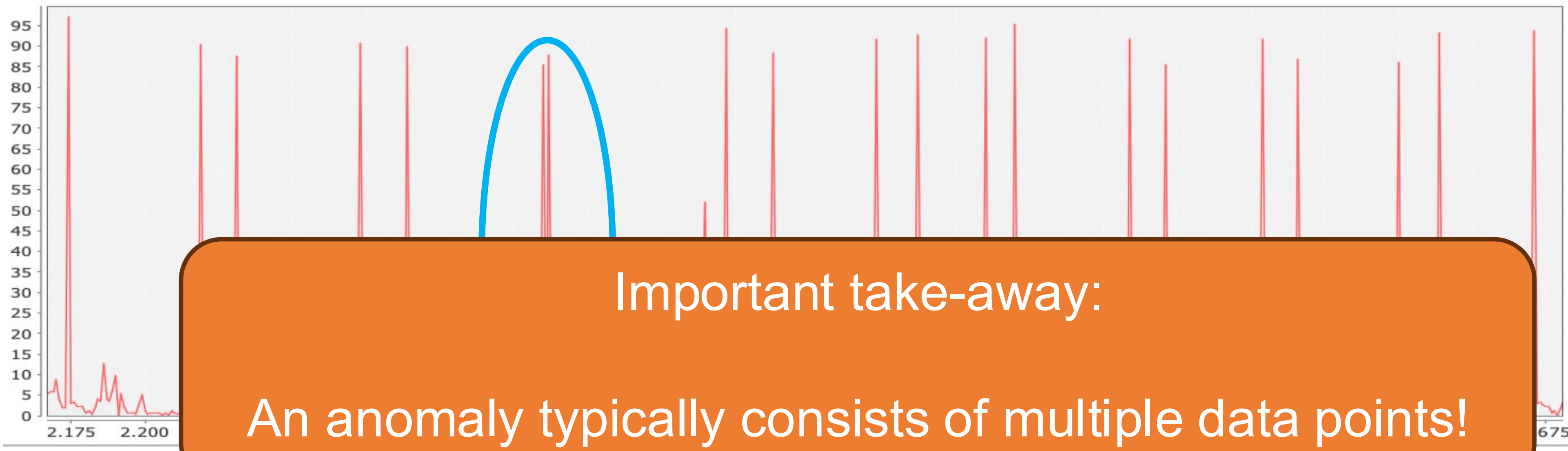
- Detection based on multiple data points -> collective




- In this case, we use post-processing, but we can also add pre-processing features such as moving averages to detect such anomalies

# Collective anomalies

- Detection based on multiple data points -> collective




Important take-away:

An anomaly typically consists of multiple data points!

- In this case, we use post-processing, but we can also add pre-processing features such as moving averages to detect such anomalies

# Three kinds of anomalies

- Point anomalies:
  - *an individual strange data points*
- Contextual anomalies:
  - *a data point that is strange given a set of data points as context*
- Collective anomalies:
  - *a set of data points that together are strange*