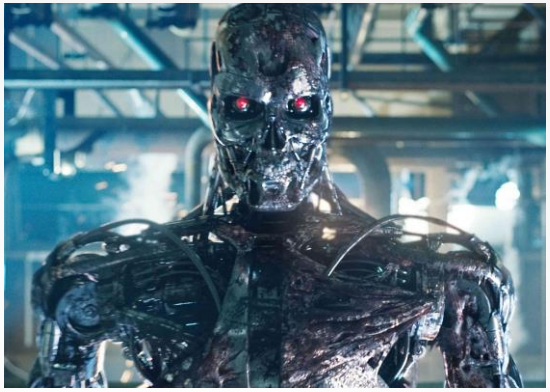


Python in the Enterprise Machine Learning Building Projects

Tomasz Szumlak

WFiIS AGH

30/03/2020, Kraków

At the beginning

- The way you can start to build your own start-to-end projects can be facilitated by tools such, conda, pycharm, git, etc.
- A lot of steps (data pre-processing, feature extraction) can be approached in quite abstract way, thus a set of simple tools can be prepared and shared between different projects
- Key aspect is: always understand your data!
- Elements of statistical data analysis are the key here.

Statistics and ML

- There are many similarities and synergies between statistics and machine learning – understanding them may give you an edge
- The workflows for both these approaches to data analysis are very similar, and differences are somewhat fascinating
- Two main types of problems:
  - **Descriptive way** – data exploration (mean, variance, higher degree moments, frequency, etc.)
  - **Inferential way** – understanding the properties of populations using „small" data samples (experiment)
- So, we model the data using descriptive or inferential statistics – this may lead to uncovering relationships among variables

Statistics and **ML**

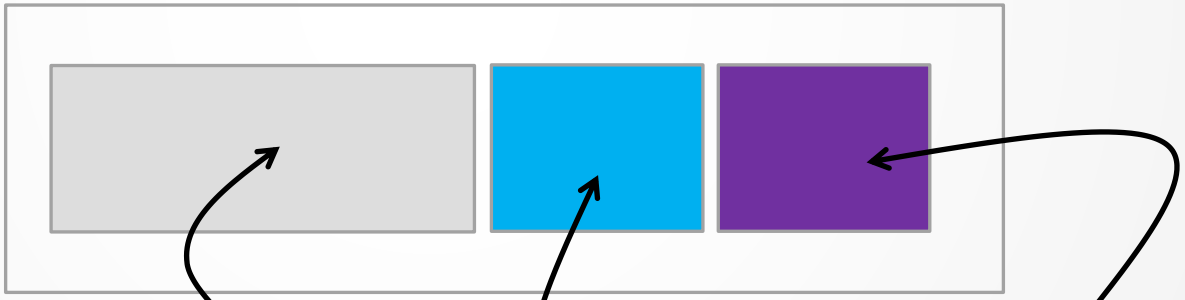
- Completely **new approach** to data modelling (much more flexible and less dependant on a particular data sample – mind overfitting!)
- „Experience" essential to „making a decision" (the knowledge can be accumulated in many steps to improve the power of prediction)
  - Train – predict, re-train (re-tain the previous weights and modify using new data, transfer learning) – predict, re-train
- Detect patterns and create an unknown rule
- The ML ways: supervised, unsupervised and reinforcement (often a mix, if large dimensionality first **reduce the problem**)

But ML is not statistics…

- Must be different, or we would not have ML…
- Need **no rule base** programming (I do not plug equations into code) – learning is a **data driven** process
- No need for assuming some particular model/shape since ML **detects and learns complex patterns** using data
- In case of ML we can only give accuracy of the prediction, **we cannot** in a formal way give neither confidence nor significance (e.g., p-value test)
- Since we do not have tools for testing the output of ML algorithm we must provide a procedure for so called **2-point validation**

But ML is not statistics…

- This usually (2-point validation) works like that:
  - Take the whole data set and divide it **0.5:0.25:0.25**
  - Use the first subset to make training, use the second to validate and tune the hyper-parameters
  - Use the last part to evaluate the results



Initial training (explore data, understand features, transformations)

Validate training **tune hyper-parameters**

Final evaluation – is it any good?

Hands-on approach

- Below is not a „magic recipe" it is more like **a set of good rules** (may not be possible always to go „exactly like that")
- **Collect data** (aka experiment), can use structured and unstructured sets
- **Pre-processing** – format data accordingly (algorithm dependent), **missing data and outliers** are delicate to handle
- **Data exploration** and **feature engineering** (this is the most time-consuming part of the ML)
- **Train** on training and validation sets
- **Final tests** on evaluation data set
- **Deploy**! May open a can of worms…

Data exploration

```
Python 2.7.13 (default, Dec 15 2017, 16:09:27)
[GCC 6.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> from scipy import stats
>>>
>>> sample = np.array([1,2,5,6,7,8,3,4,9])
>>> # Mean
...
>>> sample_mean = np.mean(sample)
>>> print(" Sample mean: ", round(sample_mean,2))
(' Sample mean: ', 5.0)
>>> # Median
...
>>> sample_median = np.median(sample)
>>> print(" Sample median: ", round(sample_median,2))
(' Sample median: ', 5.0)
>>> print(" Sample median: ", sample_median)
(' Sample median: ', 5.0)
>>> # Mode
...
>>> sample_mode = stats.mode(sample)
>>> print(" Sample mode: ", sample_mode[0][0])
(' Sample mode: ', 1)
```

Testing

- One of the most important tools in every statistician toolkit box
- **Hypothesis testing** – inference regarding a population using results obtained on a sample (parameters and statistics)
- **p-value**: or how likely is the result we observed? Significant and not significant results
- Testing algorithm
  - Define a way to quantitatively describe your statement
  - Experiment
  - Evaluate test statistics
  - Accept or reject the hypo

Testing

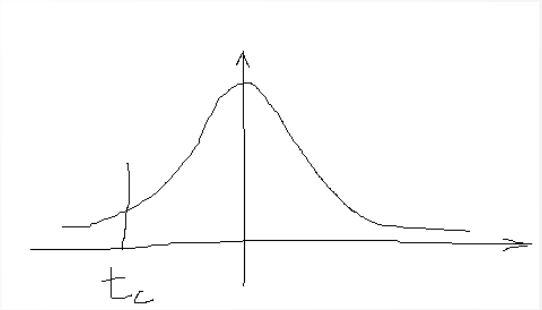
- A pack of chocolate bars have declared weight of 1000 g. To check it a sample of 30 packs were collected. The sample mean and variance were: 990g and 156.25g^2. Test the hypo that the weight is larger than declared given the significance level to be 0.05.

$$H_0\colon \mu_0 \geq 1000g; n = 30, \bar{w} = 990g, s = 12.5g$$

$$t_w = \frac{\bar{w} - \mu_0}{\left(\frac{s}{\sqrt{n}}\right)} = \frac{\bar{w} - \mu_0}{s_{\bar{w}}} = \frac{990 - 1000}{12.5/\sqrt{30}} = -4.38$$

$$t_c(\alpha = 0.05, n = 30) = -1.699$$

$$p = 7.03 \times 10^{-5}$$



Testing

- A pack of chocolate bars have declared weight of 1000 g. To check it a sample of 30 packs were collected. The sample mean and variance were: 990g and 156.25g^2. Test the hypo that the weight is larger than declared given the significance level to be 0.05.

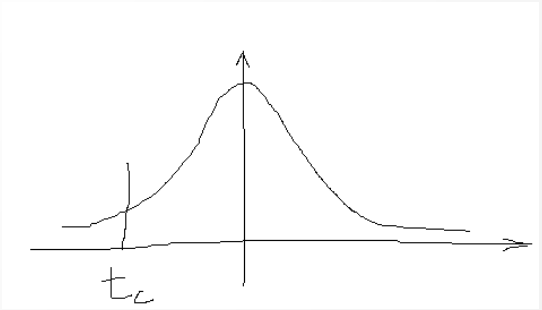
$$H_0: \mu_0 \geq 1000g; n = 30, \bar{w} = 990g, s = 12.5g$$

$$t_w = \frac{\bar{w} - \mu_0}{\left(\frac{s}{\sqrt{n}}\right)} = \frac{\bar{w} - \mu_0}{s_{\bar{w}}} = \frac{990 - 1000}{12.5/\sqrt{30}} = -4.38$$

$$t_c(0.05,30) = -1.699$$

$$p = 7.03 \times 10^{-5}$$

Rejected!



Errors

- Testing is a probabilistic process – the answer is never definite (depends on experiment, significance, etc…)
- When training an algorithm (ML) we need to prepare for the same – not every answer will be perfect!
- In principle errors are related to the fact that we always operate on finite samples – not on populations (regression of information)
- **Type I error** – reject H_0 when it is true (false negative)
- **Type II error** – accept it when it is false (false positive)

Normal distribution

- Why this model is so vital? (Law of Large Numbers, Central Limit Theorem)

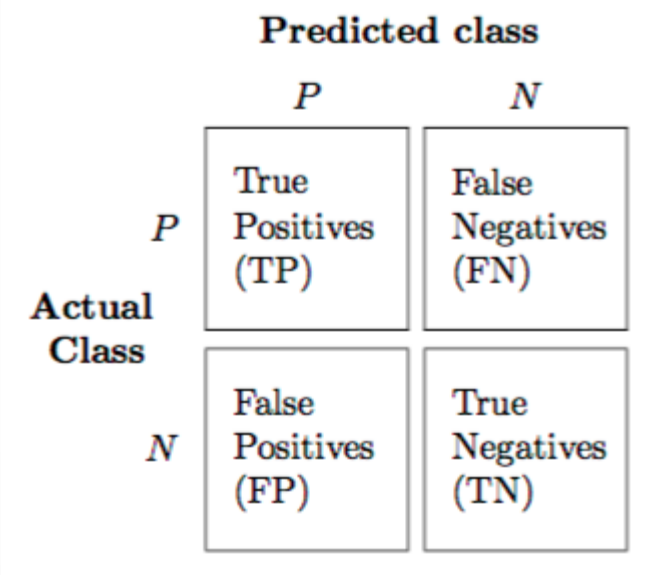
$$\lim_{n\to\infty} P\left(a \leq \frac{S_n - n\mu}{\sigma\sqrt{n}} \leq b\right) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-u^2/2}\, du$$

$$\lim_{n\to\infty} P\left(\left|\frac{S_n}{n} - \mu\right| \geq \epsilon\right) = 0$$

$$S_n = X_1 + X_2 + \cdots + X_n (n = 1, 2, \ldots)$$

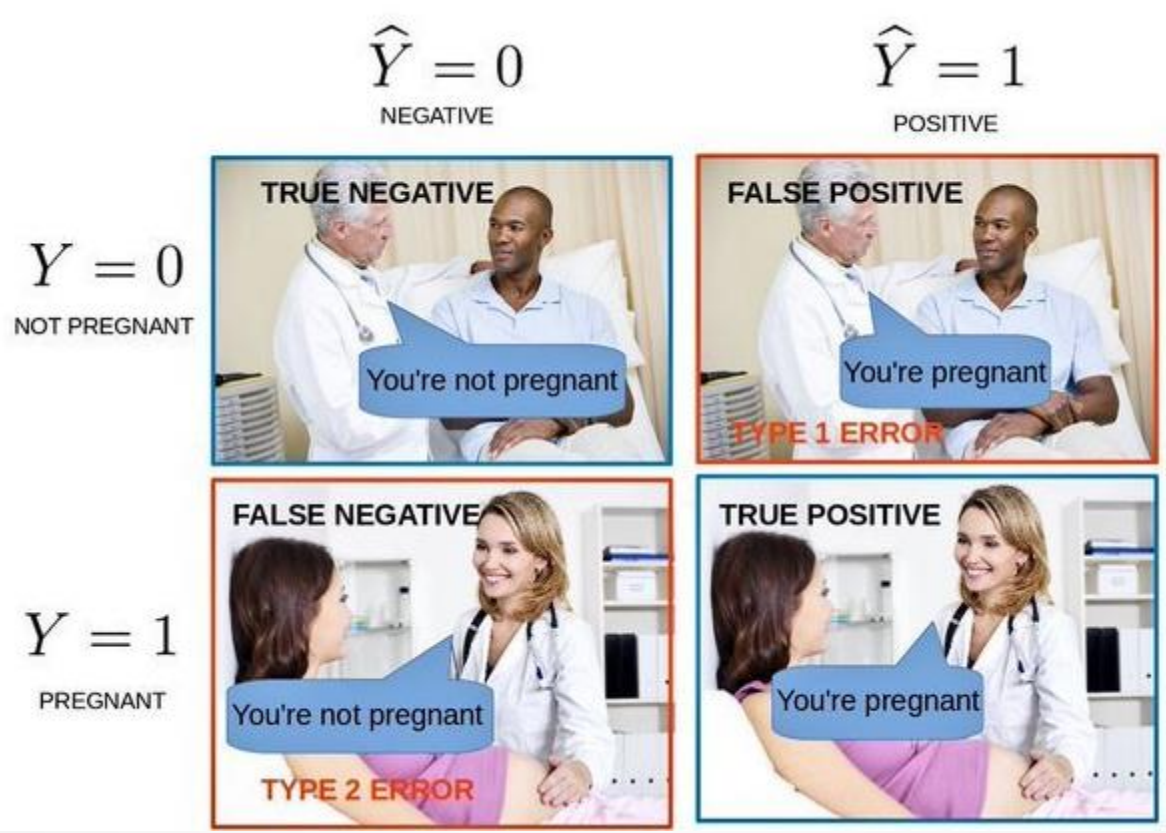
Confusion Matrix

- This is an important tool to asses the quality of our trained model



- True positive (TP) – predicted = actual
- True negative (TN) – predicted = actual
- False negative (FN) – predicted ≠ actual
- False positive (FP) – predicted ≠ actual

Confusion Matrix



Confusion Matrix

- Base on the CM we can provide some measures to asses the quality (quantitative!)
- **Precision** (P): or how are we sensitive to true positive hits (how often our signal is predicted correctly)

$$P = \frac{TP}{TP + FP}$$

- **Recall** (R): sensitivity (true positive rate), what fraction of true signal was predicted as signal

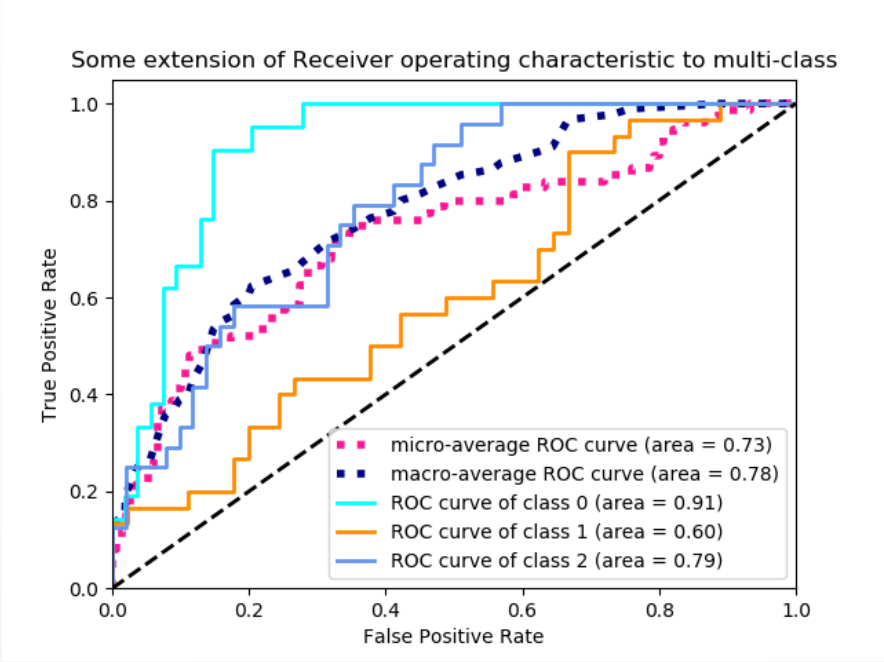
$$R = \frac{TP}{TP + FN}$$

- **F1 score**: (harmonic mean of the precision and recall)
- **Specificity** (S): what fraction of noise was predicted as noise

$$S = \frac{TN}{TN + FP}$$

ROC

- Receiver operating characteristic curve
- It expresses the dependence of TP events rate versus FP events rate (aka 1-S)
- This basically shows how kind of a trade-off we are willing to accept – the measure of goodness is area under ROC - A_{ROC}



NOTE, this is just a random plot…

R-squared

- This is a convenient measure to check how our model performs in explaining the variation in our data sample
- In other words – how well our model minimises the variance compare to calculating just a simple mean
- Can be negative – if our model is not useful at all…

$$\bar{y} = \frac{1}{n}\sum_{i/1}^{n} y_i \qquad ss_{Tot} = \sum_{i/1}^{n} (y_i - \bar{y})^2$$

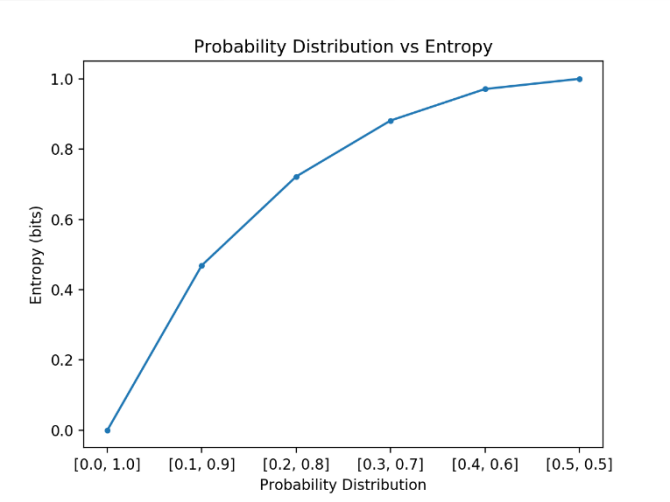
$$ss_{Res} = \sum_{i/1}^{n} (y_i - f_i)^2 \, , f_i - \text{model}$$

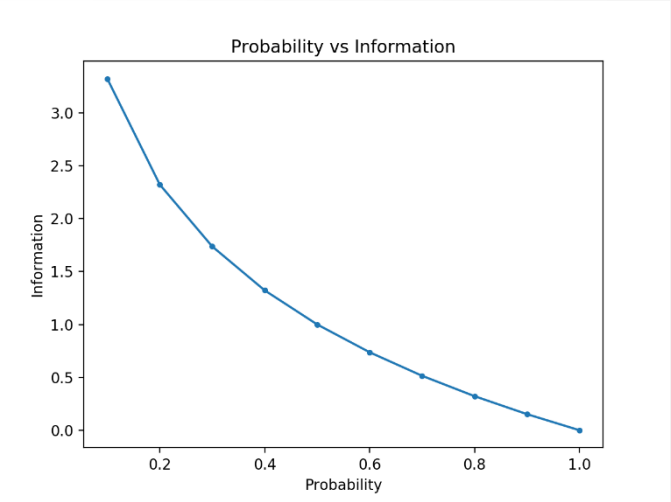
$$R^2 = 1 - \frac{ss_{Res}}{ss_{Tot}}$$

Entropy

- Not the physics (information) one but close of course
- For completely homogenous sample the entropy (that is measure of chaos) is zero
- If sample is equally divided the entropy is max
- Vital for tree-based models

$$S = -p_1 \cdot log_2 p_1 - p_2 \cdot log_2 p_2 - \cdots - p_i \cdot log_2 p_i$$





Entropy

- Simple example with coins…
- **We have two coins** – fair and loaded (P(H)=1/3, P(T)=2/3)

$$S_{FC} = -\frac{1}{2} log_2 \frac{1}{2} - \frac{1}{2} log_2 \frac{1}{2} = 1\ [bit]$$

$$S_{LC} = -\frac{1}{3} log_2 \frac{1}{3} - \frac{2}{3} log_2 \frac{2}{3} = 0.91\ [bit]$$

- In decision tree algorithm the predictor (in variable) with the most variation will be considered nearest to the root node (so called **greedy mode** classification)
- In our example – the **lower value** of entropy is **more desirable** since it gives more predictive power (better discrimination of classes)

Information gain

- If our algorithm is suppose to make good classification (again tree-based is a good example) in each step we expect the entropy to decrease (partitioning given a „question" in each node)
- We start with something mixed and we divide until we get as homogenous state as possible
- In each step can choose the variable with the max information gain (greedy approach)

$$I_{Gain} = S_{Parent} - \sum w_i \cdot S_{i(Child)}$$

$$w_i = \frac{n_i}{\sum n_i}, n_k - \text{events in k} - \text{child node}$$

Model variance and bias

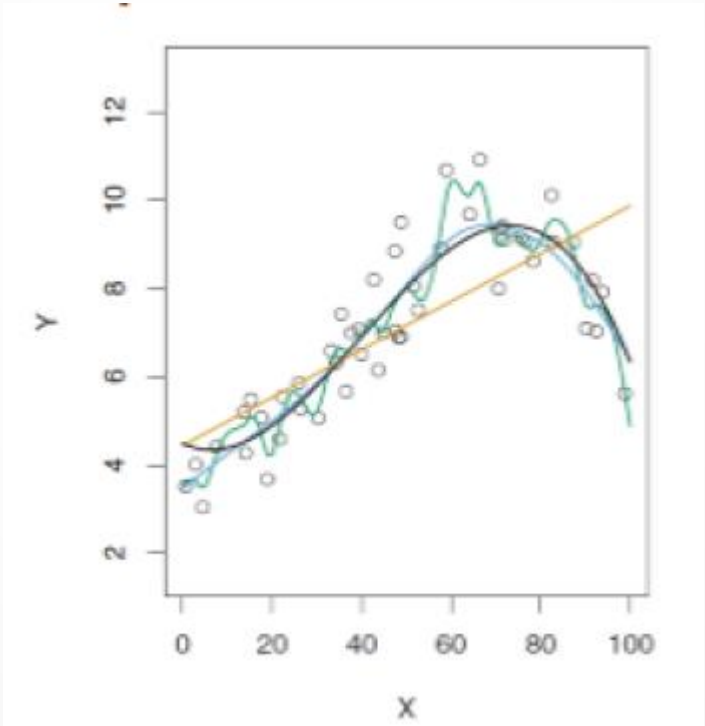
- When evaluating the overall performance of the model we need to take into account:
  - Model bias
  - Model variance
  - Intrinsic noise of data sample

$$E\left[\left(\vec{y}_0 - \hat{f}(\vec{x}_0)\right)^2\right] \propto Var\left(\hat{f}(\vec{x}_0)\right) + E\left[\left(\vec{y}_0 - \hat{f}(\vec{x}_0)\right)\right]^2 + Var(\epsilon)$$

- High bias – problem with the learning algorithm (result in underfitting, wrong detection of patterns)
- High variance – too much sensitivity (result in overfitting)
- Find right balance, **understand your data**!

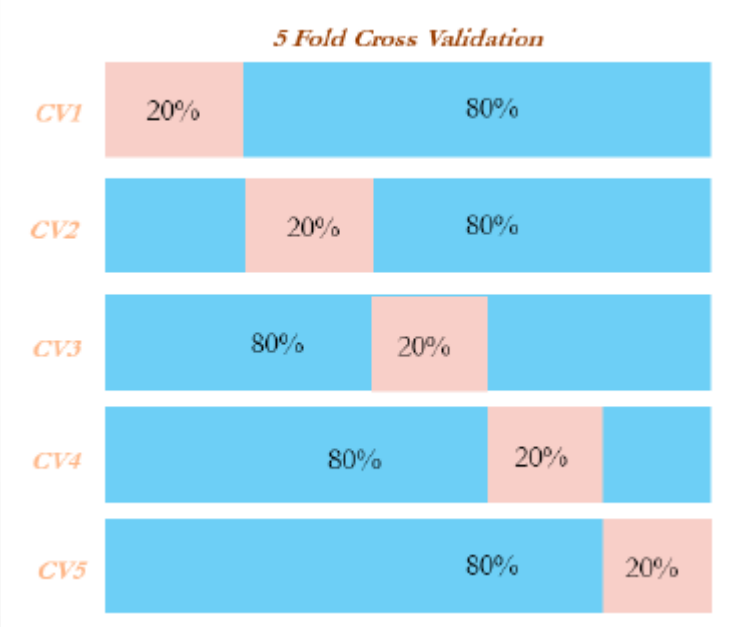
Model variance and bias

- Balancing variance and bias is a kind of art – something like Type I and Type II errors…



k-fold x-validation

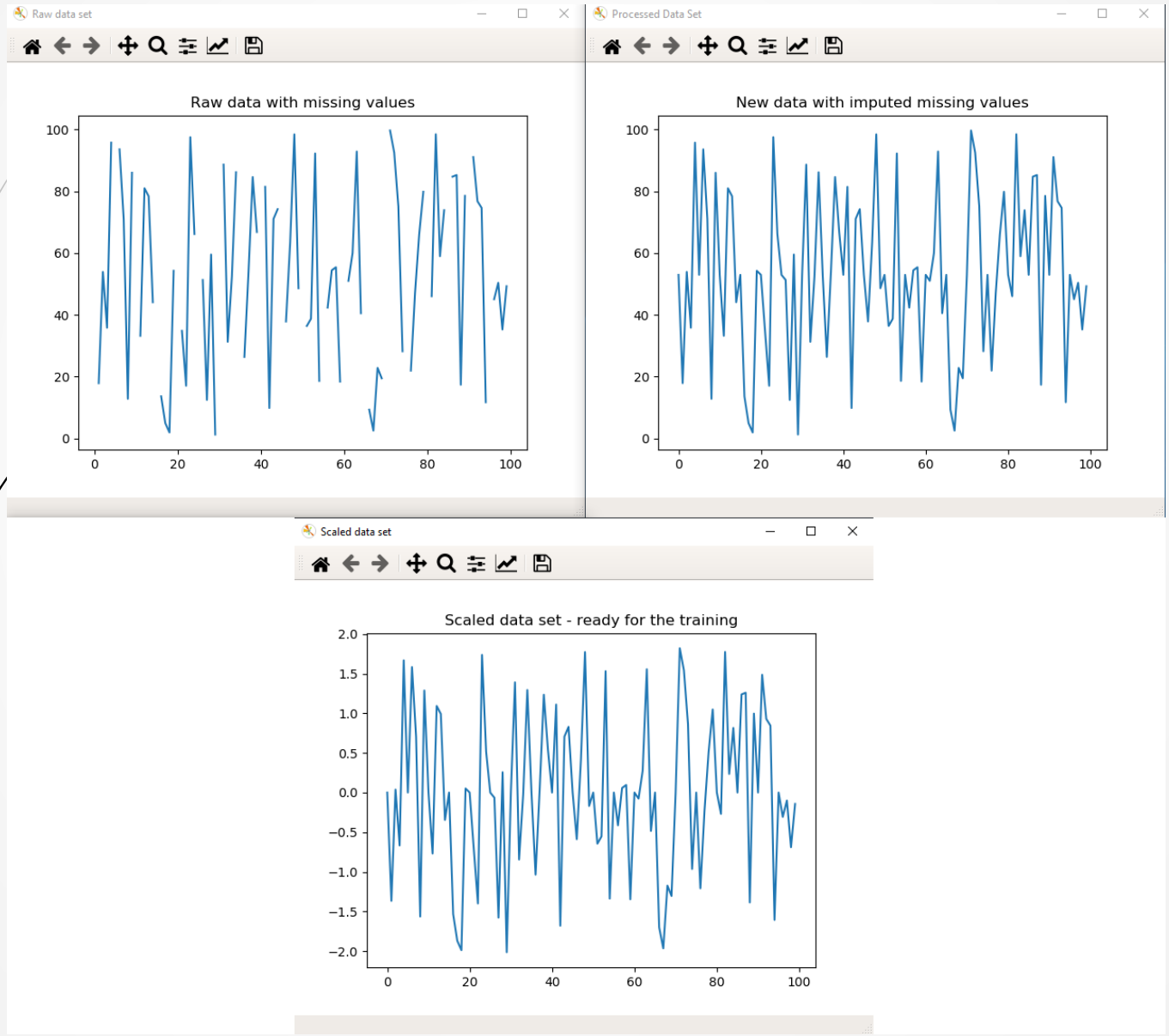
- One of the best way to perform very robust training
- It may happen that (due to bad luck) some interesting features of the data will not go into the training sample
- No easy way out… unless…



Simple example

```
# generic script for data exploration - add missing values
# with a strategy NAN for missing and and mean for the values
# to be imputed

import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer as Imputer
from matplotlib import pyplot as plt

data = np.load('sample.npy')

# Plot raw data.
plt.figure('Raw data set')
plt.title('Raw data with missing values')
plt.plot(data)

# Impute missing values.
imputer = Imputer()
data = imputer.fit_transform(data)

plt.figure('Processed Data Set')
plt.title('New data with imputed missing values')
plt.plot(data)

# Scale data.
scaler = StandardScaler()
data = scaler.fit_transform(data)

plt.figure('Scaled data set')
plt.title('Scaled data set - ready for the training')
plt.plot(data)

plt.show()
```

Simple example



Summary

- **This is just the beginning!**
- Next time you will need to analyse data – thins about machine learning approach
- But also, sometimes cigar is just a cigar, so if we have a case fitting well into standard data analysis techniques just use them