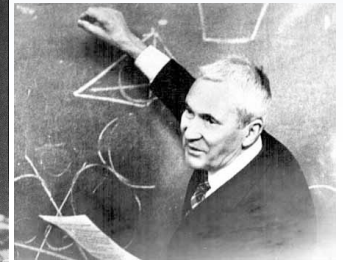
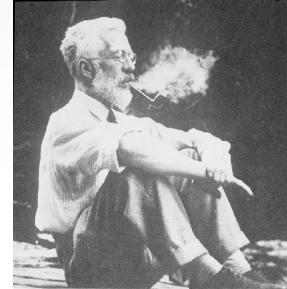
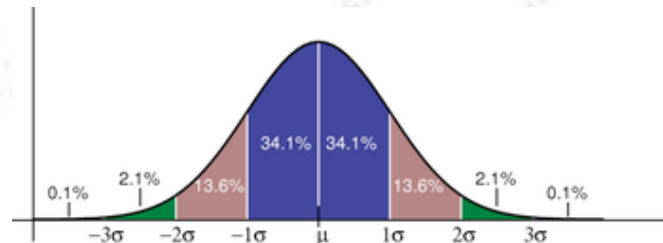


# Applied ML

## PreProcessing of Data



Troels C. Petersen (NBI)



*"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"*

# When data is imperfect

So far, we have looked at “perfect” data, i.e. data without any flaws in it. However, real world datasets are hardly ever “perfect”, but contains flaws that makes preprocessing imperative.

Effects may be (non-exhaustive list):

- NaN-values and "Non-values" (i.e. -9999)
- Wild outliers (i.e. values far outside the typical range)
- Shifts in distributions (i.e. part of data having a different mean/width/etc.)
- Mixture of types (i.e. numerical and text, from something humans filled out)

It is also important to consider, if entries are missing...

1. **Randomly** (in which case there should be no bias from omitting) or
2. **Following some pattern** (in which case there could be problems!).

In case of NaN values, we might simply decide to drop the variable column or entry row, requiring that all variables/entries have reasonable values.

Alternatively, we might insert “imputed” values instead, saving statistics.

# When data is imperfect


Imagine that you get the following data, which contains (many) NaN values. How would you want to treat a dataset like this? Discuss locally and classwide.

ID	City	Degree	Age	Salary	Married ?
1	Lisbon	NaN	25	45,000	0
2	Berlin	Bachelor	25	NaN	1
3	Lisbon	NaN	30	NaN	1
4	Lisbon	Bachelor	30	NaN	1
5	Berlin	Bachelor	18	NaN	0
6	Lisbon	Bachelor	NaN	NaN	0
7	Berlin	Masters	30	NaN	1
8	Berlin	No Degree	NaN	NaN	0
9	Berlin	Masters	25	NaN	1
10	Madrid	Masters	25	NaN	1

# Removing Variables/Columns

A “simple” way to get rid of NaN-values is to drop the column(s) containing NaN values. This works well, when these are few and with a high NaN-fraction.

ID	City	Salary	Married ?
1	Lisbon	45,000	0
2	Berlin	NaN	1
3	Lisbon	NaN	1
4	Lisbon	NaN	1
5	Berlin	NaN	0
6	Lisbon	NaN	0
7	Berlin	NaN	1
8	Berlin	NaN	0
9	Berlin	NaN	1
10	Madrid	NaN	1



ID	City	Married ?
1	Lisbon	0
2	Berlin	1
3	Lisbon	1
4	Lisbon	1
5	Berlin	0
6	Lisbon	0
7	Berlin	1
8	Berlin	0
9	Berlin	1
10	Madrid	1




# Removing Entries/Rows

Alternatively, one might instead decide to remove rows with many NaNs in them, arguing that these are incomplete entries.

This works well, when these seem to be Random NaNs, and when the entries with a high NaN-fraction are few.

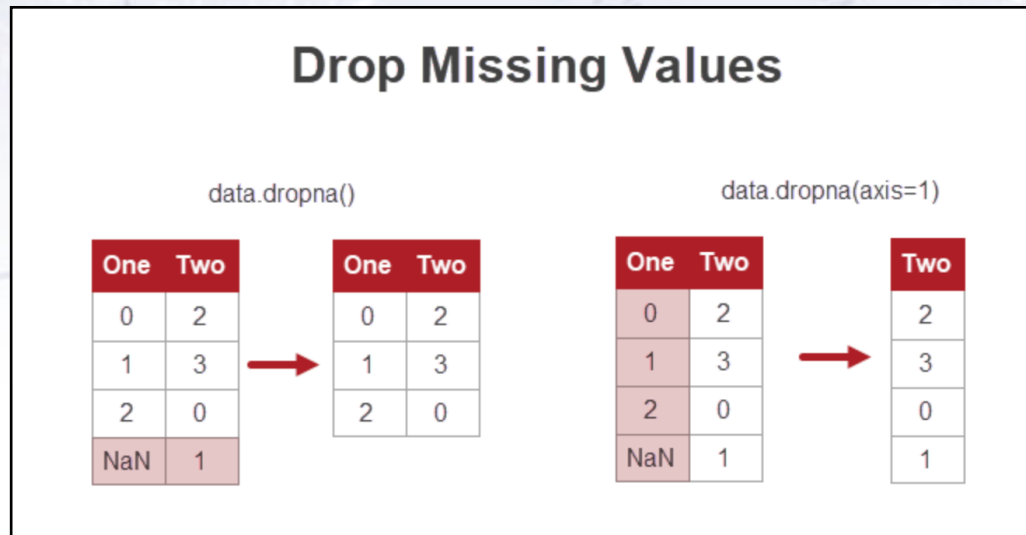
ID	City	Degree	Age	Married ?
1	Lisbon	NaN	25	0
2	Berlin	Bachelor	25	1
3	Lisbon	NaN	30	1
4	Lisbon	Bachelor	30	1
5	Berlin	Bachelor	18	0
6	Lisbon	Bachelor	NaN	0
7	Berlin	Masters	30	1
8	Berlin	No Degree	NaN	0
9	Berlin	Masters	25	1
10	Madrid	Masters	25	1



ID	City	Degree	Age	Married ?
2	Berlin	Bachelor	25	1
4	Lisbon	Bachelor	30	1
5	Berlin	Bachelor	18	0
7	Berlin	Masters	30	1
9	Berlin	Masters	25	1
10	Madrid	Masters	25	1

# When data is imperfect

So, in case of NaN values, we might simply decide to drop the variable column or entry row, requiring that all variables/entries have reasonable values.



Alternatively, we might insert “imputed” values instead, saving statistics:

Original	data.fillna(0)	data.fillna( method = 'bfill')	data.fillna( method = 'ffill')																																								
<table><tr><th>One</th><th>Two</th></tr><tr><td>0</td><td>2</td></tr><tr><td>1</td><td>3</td></tr><tr><td>NaN</td><td>0</td></tr><tr><td>2</td><td>1</td></tr></table>	One	Two	0	2	1	3	NaN	0	2	1	<table><tr><th>One</th><th>Two</th></tr><tr><td>0</td><td>2</td></tr><tr><td>1</td><td>3</td></tr><tr><td>0</td><td>0</td></tr><tr><td>2</td><td>1</td></tr></table>	One	Two	0	2	1	3	0	0	2	1	<table><tr><th>One</th><th>Two</th></tr><tr><td>0</td><td>2</td></tr><tr><td>1</td><td>3</td></tr><tr><td>2</td><td>0</td></tr><tr><td>2</td><td>1</td></tr></table>	One	Two	0	2	1	3	2	0	2	1	<table><tr><th>One</th><th>Two</th></tr><tr><td>0</td><td>2</td></tr><tr><td>1</td><td>3</td></tr><tr><td>1</td><td>0</td></tr><tr><td>2</td><td>1</td></tr></table>	One	Two	0	2	1	3	1	0	2	1
One	Two																																										
0	2																																										
1	3																																										
NaN	0																																										
2	1																																										
One	Two																																										
0	2																																										
1	3																																										
0	0																																										
2	1																																										
One	Two																																										
0	2																																										
1	3																																										
2	0																																										
2	1																																										
One	Two																																										
0	2																																										
1	3																																										
1	0																																										
2	1																																										

# Imputing with the mode

Instead of removing rows with NaN-values in them, one can decide to rather “impute” values into these, e.g. using the mode (most frequently occurring).

## Most frequent Degree = 'Bachelor'

ID	City	Degree	Married ?
1	Lisbon	NaN	0
2	Berlin	Bachelor	1
3	Lisbon	NaN	1
4	Lisbon	Bachelor	1
5	Berlin	Bachelor	0
6	Lisbon	Bachelor	0
7	Berlin	Masters	1
8	Berlin	No Degree	0
9	Berlin	Masters	1
10	Madrid	Masters	1



ID	City	Degree	Married ?
1	Lisbon	Bachelor	0
2	Berlin	Bachelor	1
3	Lisbon	Bachelor	1
4	Lisbon	Bachelor	1
5	Berlin	Bachelor	0
6	Lisbon	Bachelor	0
7	Berlin	Masters	1
8	Berlin	No Degree	0
9	Berlin	Masters	1
10	Madrid	Masters	1

# Imputing with the mean

Alternatively, one may impute values using the mean (or mode) of that column. This has the advantage, that variables/events are not thrown away.

**Average\_Age = 26.0**

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	NaN	0
7	Berlin	30	1
8	Berlin	NaN	0
9	Berlin	25	1
10	Madrid	25	1



ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	26	0
7	Berlin	30	1
8	Berlin	26	0
9	Berlin	25	1
10	Madrid	25	1



# Alternatives when imputing

Instead of using a fixed number for inputing, one might pick an age randomly, but at the same time “report” (in a new column) that it is an imputed value.

**The two randomly picked Ages are 25 & 30**

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	NaN	0
7	Berlin	30	1
8	Berlin	NaN	0
9	Berlin	25	1
10	Madrid	25	1



ID	City	Age	Age-Imputed	Married ?
1	Lisbon	25	25	0
2	Berlin	25	25	1
3	Lisbon	30	30	1
4	Lisbon	30	30	1
5	Berlin	18	18	0
6	Lisbon	NaN	25	0
7	Berlin	30	30	1
8	Berlin	NaN	30	0
9	Berlin	25	25	1
10	Madrid	25	25	1

# When data is imperfect

Alternatively, one might decide to give the NaN-values a specific value (here -1), which is different from all other values.

## Arbitrary Value = -1

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	NaN	0
7	Berlin	30	1
8	Berlin	NaN	0
9	Berlin	25	1
10	Madrid	25	1



ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	-1	0
7	Berlin	30	1
8	Berlin	-1	0
9	Berlin	25	1
10	Madrid	25	1

# When data is imperfect

For the text case, one might again simply consider making a new column, which indicates if the value given is imputed (no matter what method was used).

**Missing = 1 & Not Missing = 0**

ID	City	Degree	Married ?
1	Lisbon	NaN	0
2	Berlin	Bachelor	1
3	Lisbon	NaN	1
4	Lisbon	Bachelor	1
5	Berlin	Bachelor	0
6	Lisbon	Bachelor	0
7	Berlin	Masters	1
8	Berlin	No Degree	0
9	Berlin	Masters	1
10	Madrid	Masters	1



ID	City	Degree	Missing Indicator	Married ?
1	Lisbon	Bachelor	1	0
2	Berlin	Bachelor	0	1
3	Lisbon	Bachelor	1	1
4	Lisbon	Bachelor	0	1
5	Berlin	Bachelor	0	0
6	Lisbon	Bachelor	0	0
7	Berlin	Masters	0	1
8	Berlin	No Degree	0	0
9	Berlin	Masters	0	1
10	Madrid	Masters	0	1

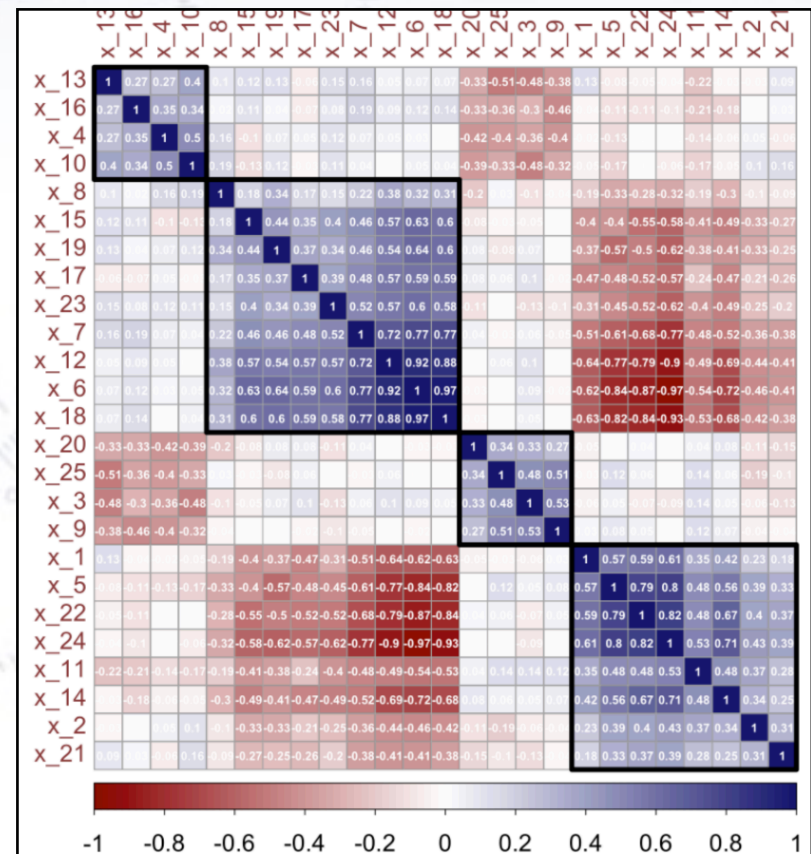
# NaN-values tend to correlate

It is often seen, that several variables have the same source, and thus their NaN occurrence might be correlated with each other.

This can be tested by substituting 0's for numerical values and 1's for NaN values. By considering the correlation matrix of these substitute 0/1 values, one gets a pretty clear picture.

Typically, some entries are 100% correlated, as the source of these variables is shared.

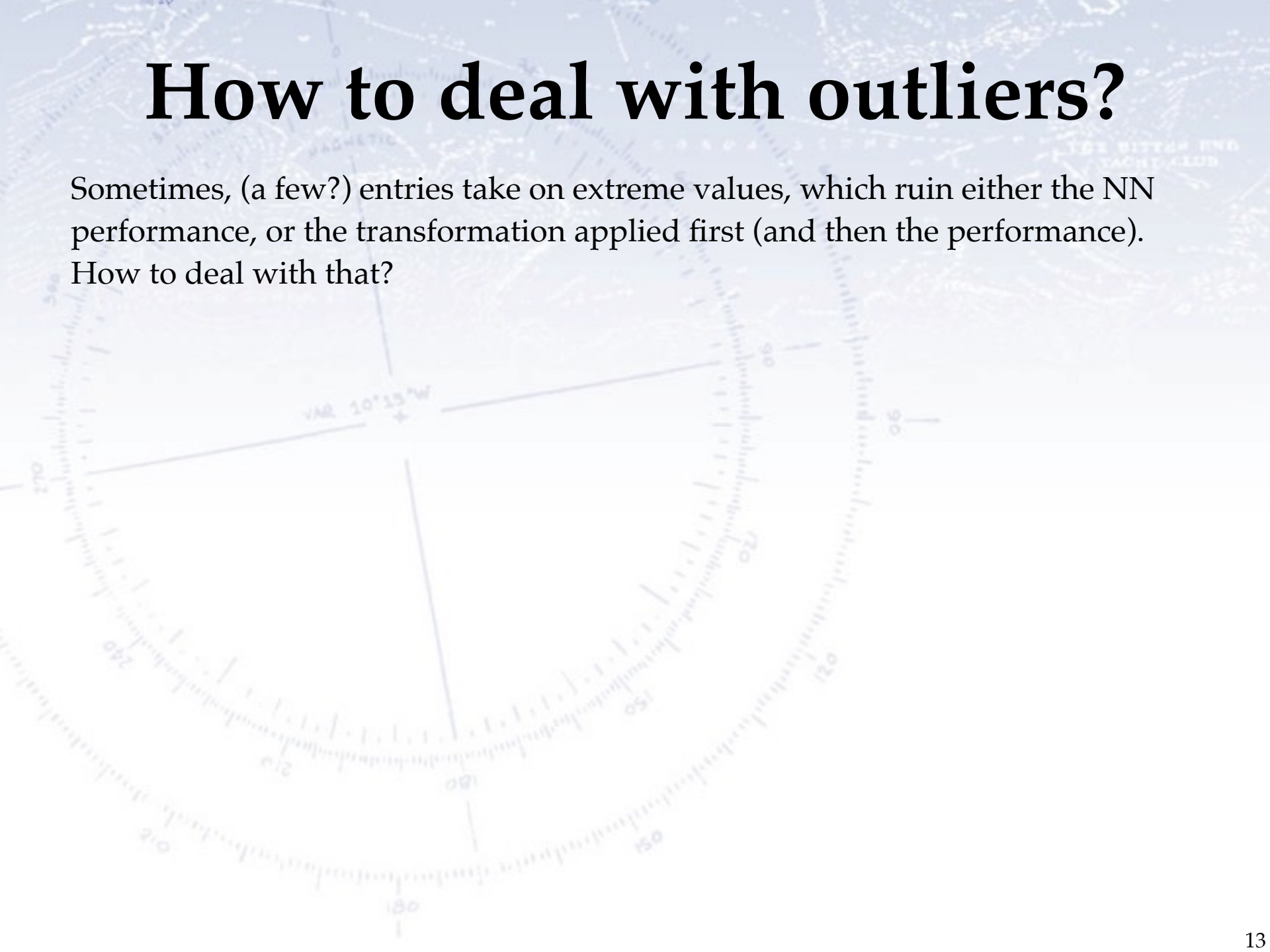
Based on this information, one can better decide how to deal with these variables.





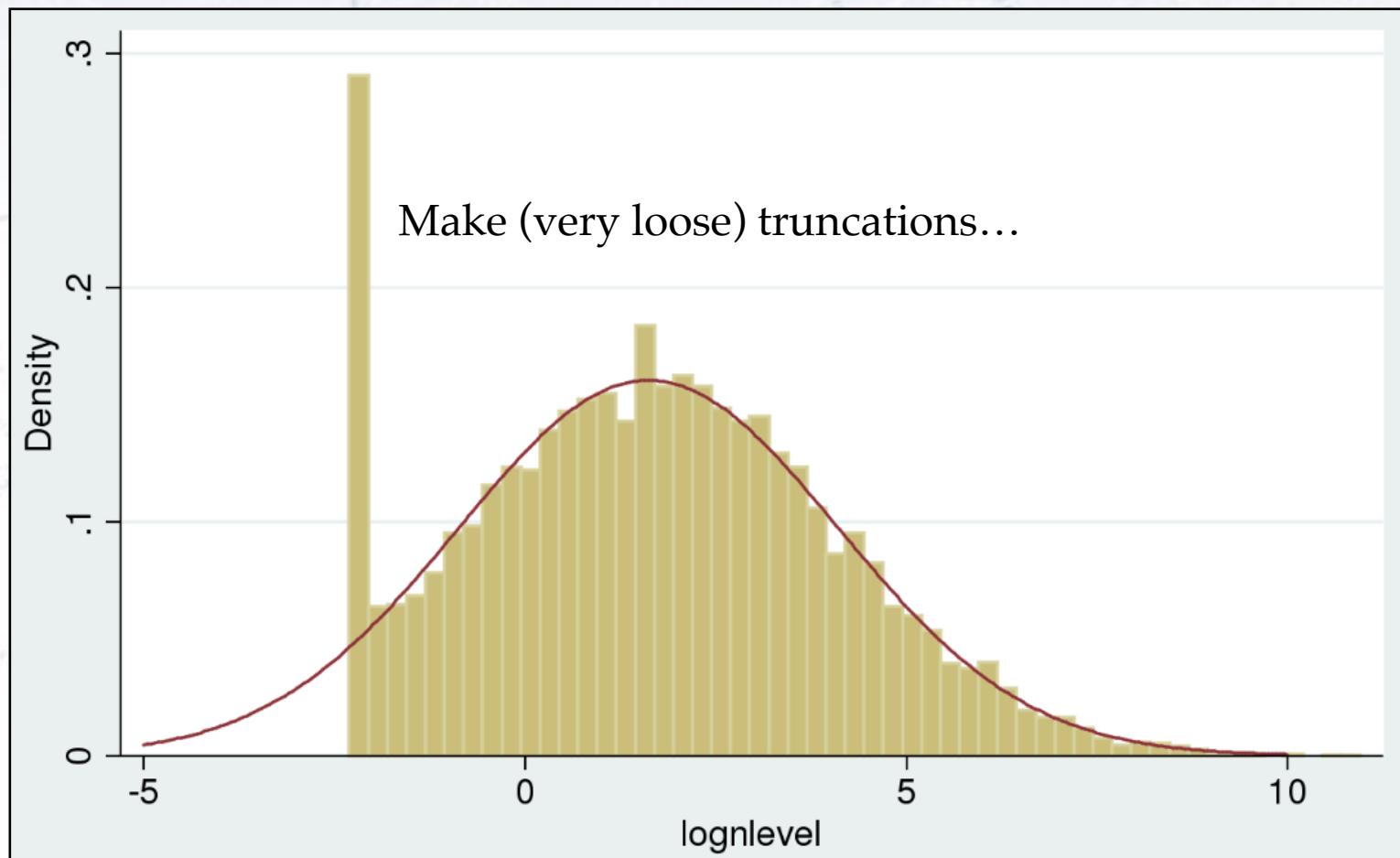
# How to deal with outliers?

Sometimes, (a few?) entries take on extreme values, which ruin either the NN performance, or the transformation applied first (and then the performance).  
How to deal with that?



# How to deal with outliers?

Sometimes, (a few?) entries take on extreme values, which ruin either the NN performance, or the transformation applied first (and then the performance).  
How to deal with that?



# Mixture of types

At other times, types may be mixed. An example could be from a form, where people are asked “How long time did it last?”. The answers:  
2 minutes, 4.5h, about a second, 4:07:32, donno, all day, between four and 5 min.

In that case, there is nothing else to do, than to run through the bitter (large) number of unique cases, and get more than 95% working, possibly leaving the rest.

# Conclusions

No matter what you plan to do with data, my first advice is always:

## Print & Plot

This is your first assurance, that you even remotely know what the data contains, and your first guard against nasty surprises.

Also, working with others (from know-nothings to domain experts) you will be required to show the input, and assuring that it is valid and makes sense.

Remember also to do so in your final projects...