#### **Getting a dataset**

- A) Read the documentation:
  - What features are needed? What are you trying to predict? Do you need to perform a regression or a classification?
- B) Normalize your data if the explicative variables are of different nature. Make a choice about handling NaNs (missing values): imputation, omission.
- C) Split your dataset into a train, a test and a validation dataset.

  If size of the database is small -> use cross-validation / leave one out

  If size of the database is average -> ~ 60% ~ 20% ~ 20% split

  If size of the database is large -> subsampling for the datasets

Make sure that the statistics (at least mean and std) in the 3 datasets are similar. Percentage of targets in specific classes is also important when performing a classification task.

- D) Select some possible architectures: set your fixed hyperparameters.
- E) Grid/Random/Bayesian search the best combination of hyperparemeters.
- F) Interpret the results. If not good enough, consider altering your earlier choices.

## Feature Engineering: Missing Values

#### **Options:**

Ignore them:

Either the variable or individual observations that have missing values.

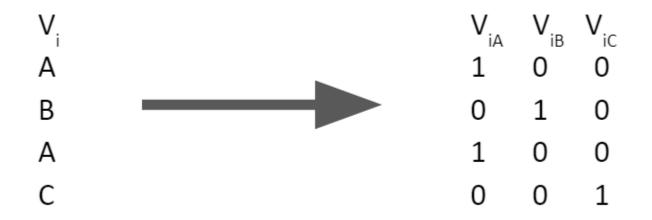
Depends on the number of data you would eliminate and the importance of the variable.

• Impute them:

Either with mean value, median value (of the variable) or with another algorithm (analogs / KNN, DINEOF, ITCOMPSOM...)

#### Feature Engineering: Qualitative Data

Example: Multiple Choice Questionnaires
Transform every choice into a binary variable
(Yes/No)



Skitlearn library can automate this using: sklearn.preprocessing.OneHotEncoder

## Feature Engineering: Ordered Qualitative Data

Example: Multiple choice qualitative answers:

Not at all, a bit, medium, a lot, too much

Give every choice a numerical value

Not at all = 0, a bit = 1, medium = 2, a lot = 3, too much = 4

The values can impact the performance:

Not at all = -3, a bit = 1, medium = 2, a lot = 3, too much = 54

No easy out of the box solution for this.

## Feature Engineering: Cyclical Data

Examples: longitude, day of the year, time of the day

Transform it into two variables: sin and cos functions of the initial variable

ToD 
$$\longrightarrow$$
  $\sin(2\pi*ToD/24)$   $\cos(2\pi*ToD/24)$   
Lon  $\longrightarrow$   $\sin(2\pi*(180+Lon)/360)$   $\cos((2\pi*(180+Lon)/360))$ 

Why we do this? To preserve physical proximity in a numerical sense. I.E. 22:00 - 02:00 numerically is 20 hours of difference... while we know it is only 4 hours of difference. It should be equal to 22:00 - 18:00, which is indeed 4h. With the transformation:

$$\sqrt{(sin(2*\pi*22/24)-(sin(2*\pi*2/24))^2+(cos(2*\pi*22/24)-(cos(2*\pi*2/24))^2}\\ \sqrt{(sin(2*\pi*22/24)-(sin(2*\pi*18/24))^2+(cos(2*\pi*22/24)-(cos(2*\pi*18/24))^2}$$

#### Feature Engineering: Too many Correlated Variables!

They can be reduced using a compression algorithm:

- Principal Component Analysis
- Auto-encoders

Take care to preserve variable importance by applying your compression algorithm on groups of <u>variables expressing</u> <u>the same dynamics</u>.

# Why?

If you have nineteen variables expressing the same dynamic and one that does not, compressing them all together even if you maintain 95% of the variability, the end result will likely compress the on variable that has different information into irrelevance.

## Feature Engineering: Evident non-linear link

If your target has an evident non-linear link with an explanatory variable:

Create a new explanatory variable that linearizes the relationship to the target.

(Example: x is an explanatory variable when your target is  $x^2$ )

Requires a careful analysis of the scatter plots of your variables.

Alternatively, if the distribution of you variable looks like inversible a nonlinear function, linearize it by applying the inverse function. (for example try predicting  $\sqrt{x}$  instead of x, then transform the result)

## **Feature Engineering**

Oversampling

When you try to predict rare events: oversampling examples that correspond to that rare state.