**Notes from DSBox project meeting, 20 April 2017**

**On Datasets:**

From Daniel’s notes:

o\_38

- binary classification.

- The problem is not clearly stated in the description file. The Problem description contains as well all the information of the dataset.

- Thyroid disease records. The problem is to identify if a given patient is sick (class 1) or negative (class 2).

- The csv for the training data is as is, without pointers to raw data

- The dataset combines integer, categorical and float variables.

**How would we analyze O38**

In this dataset, train schema not the same as test schema

We should be careful not to feed “index columns” as features

Metric is F1:

Imbalanced class problem:

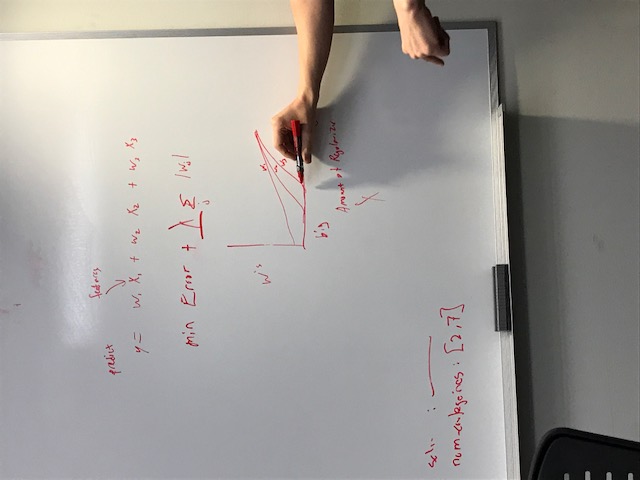
* only a few instances are negative => this is why they chose F1
* to address the few instances issue:
  + you can oversample negative instances
  + you can add weights to the instances (eg in SciLearn you can add weight in the SVM. You can put also class weight, based on all the instances that have that class in multi-class problems)

Missing values:

* Looking at the training data, lots missing
  + If few instances have missing data, you can throw them away
    - Here every row is missing something
  + Not missing at random: lots of instances missing the same features (based on doctor not doing a test)
    - Hard to tell this for a random dataset
    - There is a true/false column that says whether you took the test
      * Pedro: you could throw away the values and just use true/false
    - May add a column that says “missing test”
      * Whether something is missing may be useful, to do imputation
  + To deal with missing values, there are several techniques:
    - Use a modeler that can deal with that, eg Naïve Bayes
    - Or you can fill out missing values:
      * Do imputation (SciKit has a function for that)
      * Non-negative matrix factorization, as in recommender systems where you predict a value based on other values
      * Give missing values a special value, eg if gender is missing use “x”
      * Do a column per value and a column for “missing”, the values for all these columns are 1 or 0
        + This is called “one hot featurization/vectorization”
    - Pedro: can combine columns (eg multiply)
      * SVM if you give it a polynomial kernel will do this
      * Pedro: SciKit has a function to do this
  + Feature selection vs feature reduction
  + 3,000 instances, few attributes
  + lasso plus regularization – there is feature selection and regularization at the same time
* if score in training is much better than in testing, then you are overfitting, so you can include regularization

Regularization (see picture below)

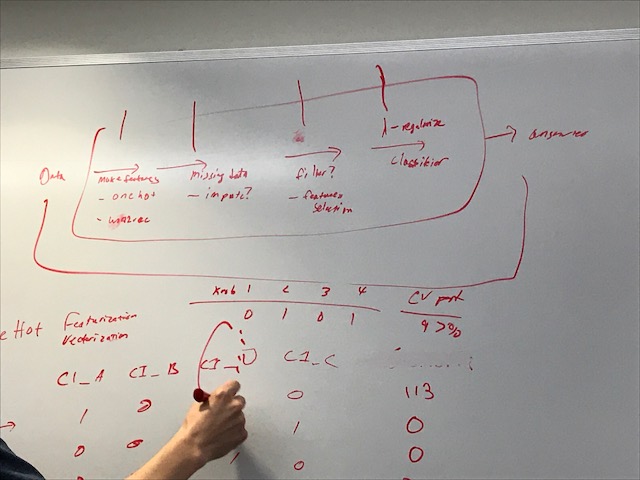
* L1: eucledian distance
* L0 is hard to optimize, that’s why people prefer L1
* Sparsity drives some of the weights to 0
* People like L1 because it encourages sparsity
* Better do regularization than feature selection



For text, it works better to use word to vec and do regularization than doing TF/IDF and correlation.

There is a canonical pipeline with several typical steps (see picture below)

* There are a lot of constraints, eg should not do featurization if we use certain types of classifiers
* Each step has several parameters



Hyperparameter search: How do you search for best param values for the params in that pipeline?

* Grid search (there is a function in SciKit)
  + You would wrap around grid search around the typical steps
* You may use default values for most params, SciKit is very good at setting up defaults automatically.
* A common strategy is to pick defaults, and then vary one parameter at a time
  + This avoids having to try to search all parameters at the same time
* To find best lambda, you wrap around the grid search