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

**On Datasets:**

Pedro added one of the datasets to Github, and Ke-Thia explained how he would have tackled the problem.

From Daniel’s notes:

o\_38 description summary:

- binary classification problem.

- 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 is the same as test schema

We should be careful not to feed “index columns” as features. These are normally the first column in the schema, denoted by “d3mIndex”.

Metric is F1.

Imbalanced class problem:

* only a few instances are “sick”. Most are “negative” => this is why they chose F1
* to address the few instances issue:
  + you can oversample “sick” 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). In neural nets instances can be weighted too.

The key is being able to **detect that the dataset is imbalanced** (this can be seen looking at the target variables and values)

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”

Figure : Doing a hot featurization/vectorization on the input data. Creating new variables is easy. The problem is knowing how to create the relevant ones.

Example of one-hot featurization/vectorization:

If you have categorical features you can transform them to binary variables. For example, the categorical feature on the left (C1) would be transformed to the set of binary features in the right:

**C1** -> **C1\_A** **C1\_B** **C1\_missing**

A -> 1 0 0

B -> 0 1 0

[missing] -> 0 0 1

* + - 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. Overfit may happen if you are doing combinations of columns

Pedro: We could create a set of "strategies" which have the domain knowledge and restrictions for trying to address part of the problem.

* Example: for feature creation: "f.one hot" -> call sci kit primitives a, b and c. If we have a large number of categories. then we can try to explore D and E.

Regularization: how does it work? (see picture below)

* L2 constraints introduce sparsity
* L1: Euclidian 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
* You normally try one then another. Potentially the model is more general if it's more simple.

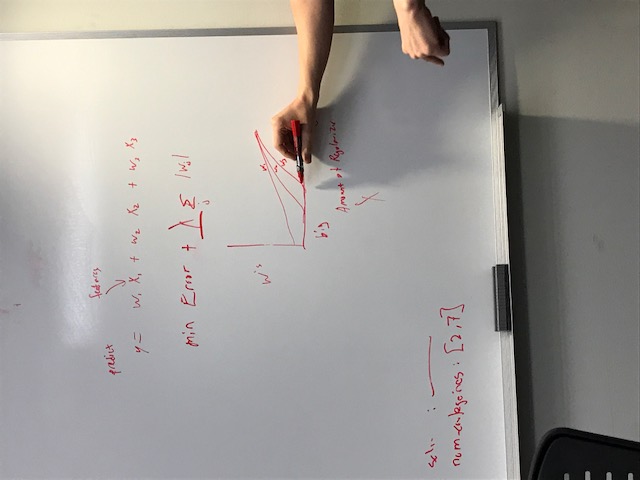


Figure : Regularization (L0, L1 and L2). The key is picking the lambda parameter to reduce the error.

Gregg: look at pairwise correlations to reduce features ahead of time.

* Lasso etc try to apply weights to variables and then try to minimize the error.
* if you increase the amount of regularization, you might overfit.
* Learning the model AND doing feature selection.
* The idea is that feature 1 and 2 together are very good, but 1 and 2 separate are not that good.
* Lasso is L1, but elastic net is L1+L2, which could be better.
* If you want a simpler model, you apply neural nets.

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

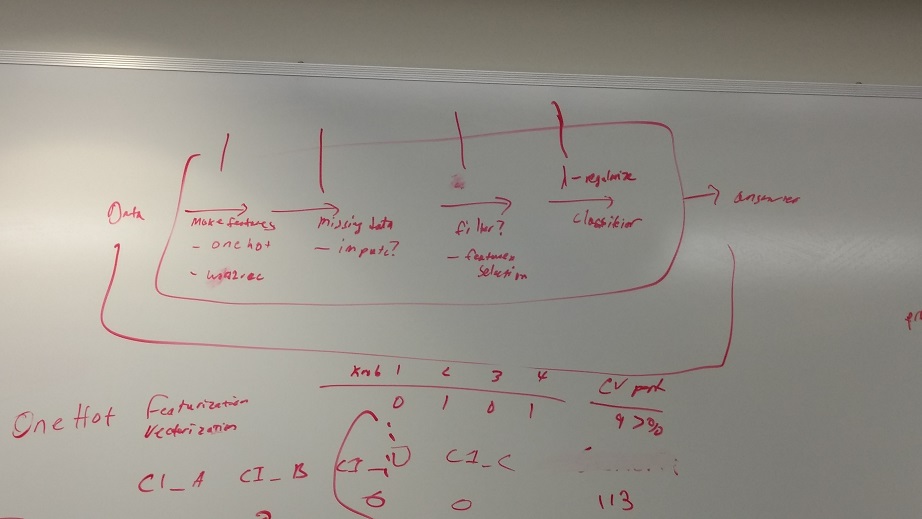


Figure : From Data (left) to an answer (right): make features (e.g., with one-hot, word2vec), address missing data, filter features (feature selection), lambda regularization and then train your classifier. The problem is that different components have different settings. And then have to run everything tuning the hyperparameters. Maybe it makes sense to set up one and then tune the components one by one. This could be a first approach to the problem.

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 parameters, SciKit is very good at setting up defaults automatically. However, you have to be careful in certain cases, as something may go terribly wrong.
* 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