ML diary

January 26, 2017

1 1. device model

Notebook: here

1.1 Getting started

First model : - XDGBoost : overfit result : accuracy = 0.93
==> trying RandomForest instead better, less overfit

- increasing the nb folds during hyperpparameters
- removing the features that should not make sense:
  - attribute1
  - min_any and std_any

==> 'reasonable' model, 93% accuracy, but much less overfit

- removing suspicious positives (see dataset exploration)

1.2 Going through features...

- attribute1 : removed
- attribute2 :
- useful as raw feature (max & mean)
- removing it lower perfs : accuracy : 92.4 --> 92.4... so no use
- adding dt_attribute2 ==> 92.4-> 92.77
- addinf dt2_attribute2 ==> 92.7 -> 93.2 (93.5 with calibration)
- attribute3: removed
- attribute4 : with DFT : 93.2=> 93.8
- attribute5 : with DFT : 94.06
- attribute6, 7 : DFT does not bring much : corelation already extracted ? (the feature seems important)
- attribute9: no effet... why ?

feature filtering: no discernable effect feature scaling, PCA => idem
... all of this is reasonable, considering we are trying tree models
testing SVM : gives goood results, still under random forest (0.91 rather than 0.93)
re-testing Gradient boosted trees : still slightly under Randomforest

1.3 TPOT

96.4% accuracy ! (@ 95%, random forest with tuned voting)

Caveat: there seems to be problem of overfitting on most models. The best recommentdation IRL would be to get more data...

2 2. Temporal Model

Notebook: here

2.1 Set up first model:

Windowing groupby: need to adapt the aggregation methods to be commpatible with rolling aggregation

Per device test/train split strategy:

Because the data we aggregate for each time point is very correlated with other timepoint for the same device, we need to make sure there are no selection bias, when building our model.

This is done by splitting the examples per device: a device with examples in the test set cannot also have examples in the train set. This is basicaly assured by using specific splitting startegy provided by sklearn

Each time point with no failure is a negative, that leads to very unbalanced classes: - We use 'f1' scoring method rather than acccuracy, to be less sensible to class imbalance - We subsample negatives, to reduce this imbalance. - To be tested: oversample positives rather than subsample negatives - We extend the positive window by 7 days: every device detected to be failing at most 7 days before a failure, is considered a negative

2.2 Feature engineering

After calibration, we added to the features an averaging lookback window, to use as feature the "last" value observed for an attribute, in addition to its averaged value.

Here, trimmming the useless features seems to have a positive impact on the performances.

2.3 Test other models

The same models are tested: XGboost, Random Forest, and SVM. - best RF : f1=0.47 (but unstable model, some RF perform better than others)

We also test PCA, feature scaling, and feature filtering

3 3. Model bagging

This done at validation time in this notebook

Tried two bagging methods: - consensus: averaged score is taken as final score - vote: the most confident model perform the prediction

Further development: try to use two models in a boosting scheme, rather than a bagging one

```
In [ ]:
```