

Scania dataset

Unbalanced dataset and custom objectives.



Overall metrics: 170 Features, 60 000 lines

Train: 60 000 lines, 98.3 % are “neg” or Class 0

Test : 16 000 lines, 97.7% are “neg” or Class 0

7 Histograms : AG, AY, AZ, BA, CN, CS and EE.

100 “Counters” : positive numbers.

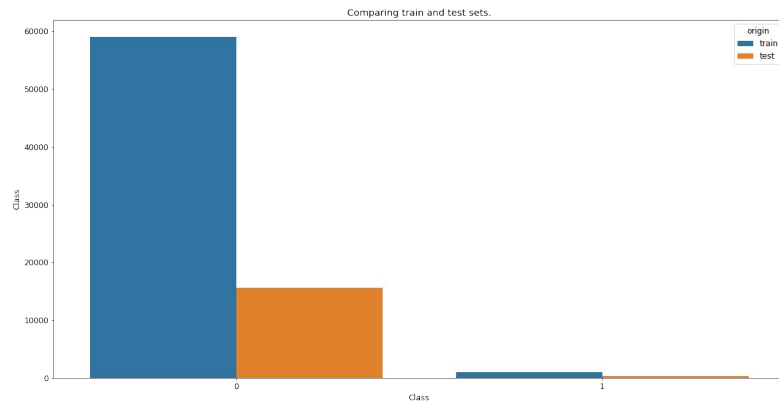
Business objective:

=> Misclassified Class 1: \$500

=> Misclassified Class 0: \$10

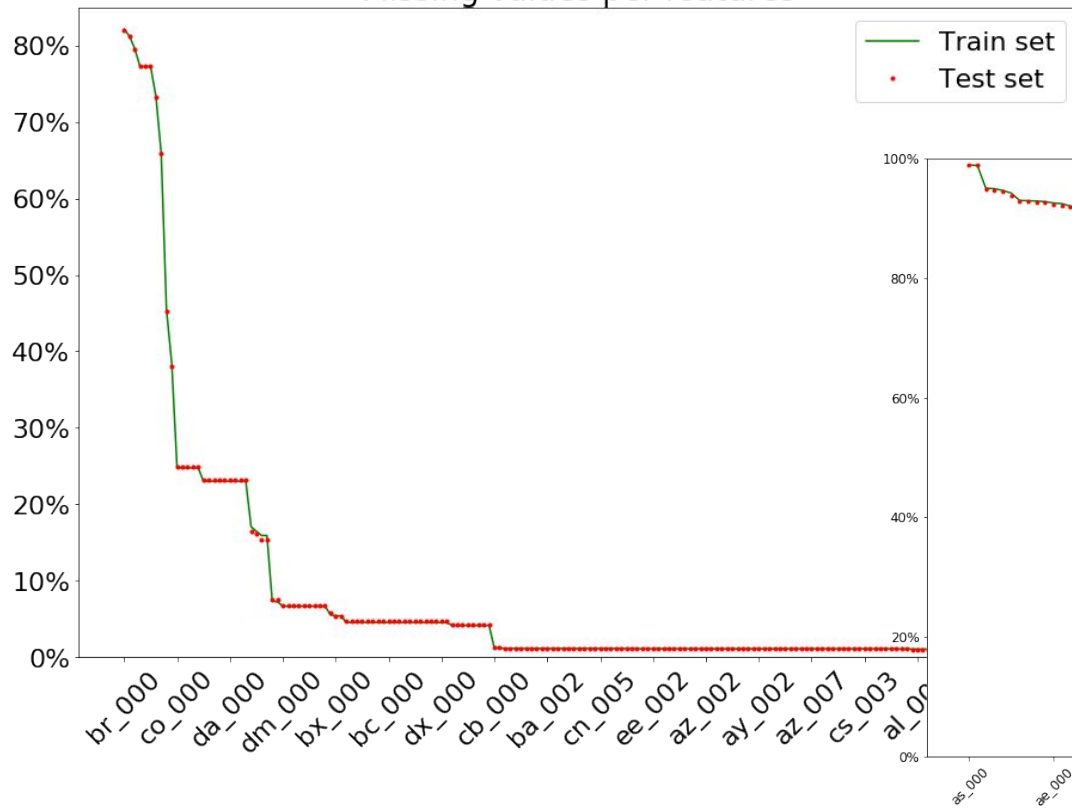
Best published solution \$9 920 cost (lower is better):

- Recall of 97%
- Precision of 40%

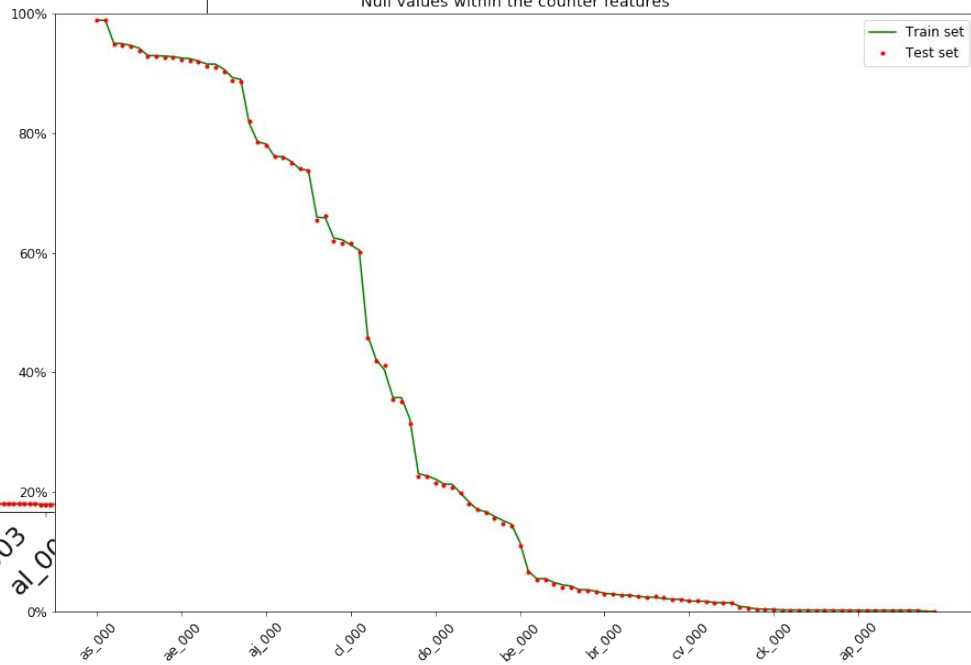


Curated dataset : equivalent missing and null values

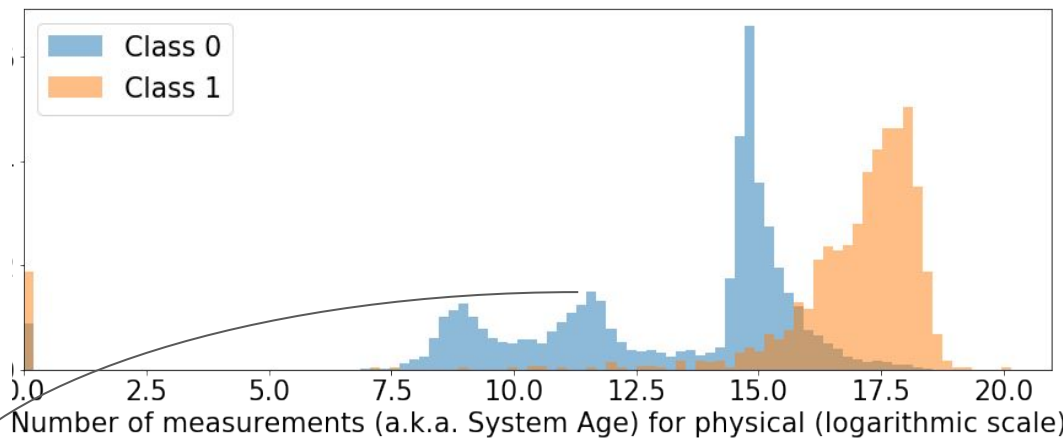
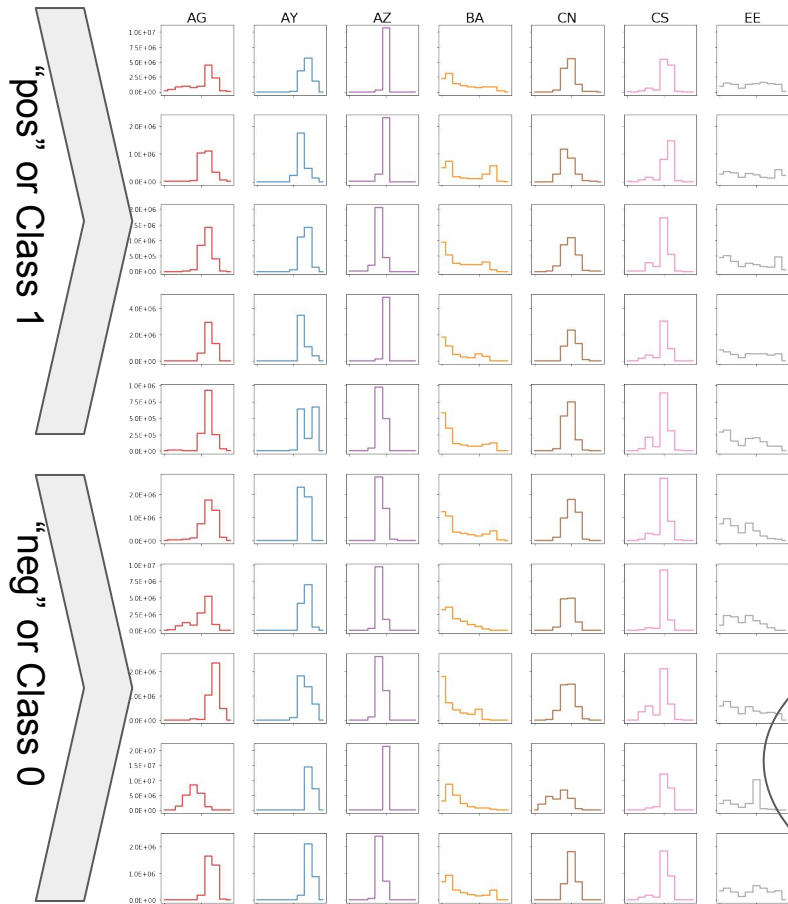
Missing values per features



Null values within the counter features

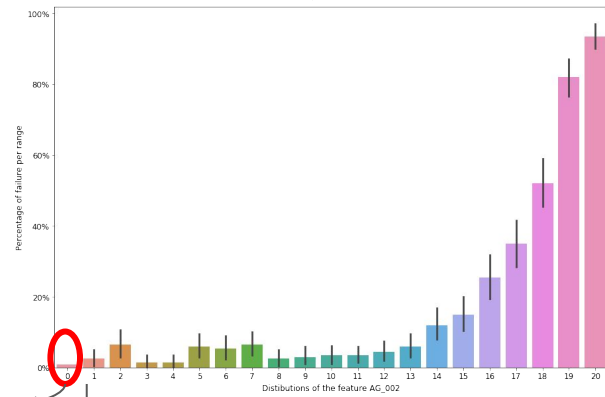
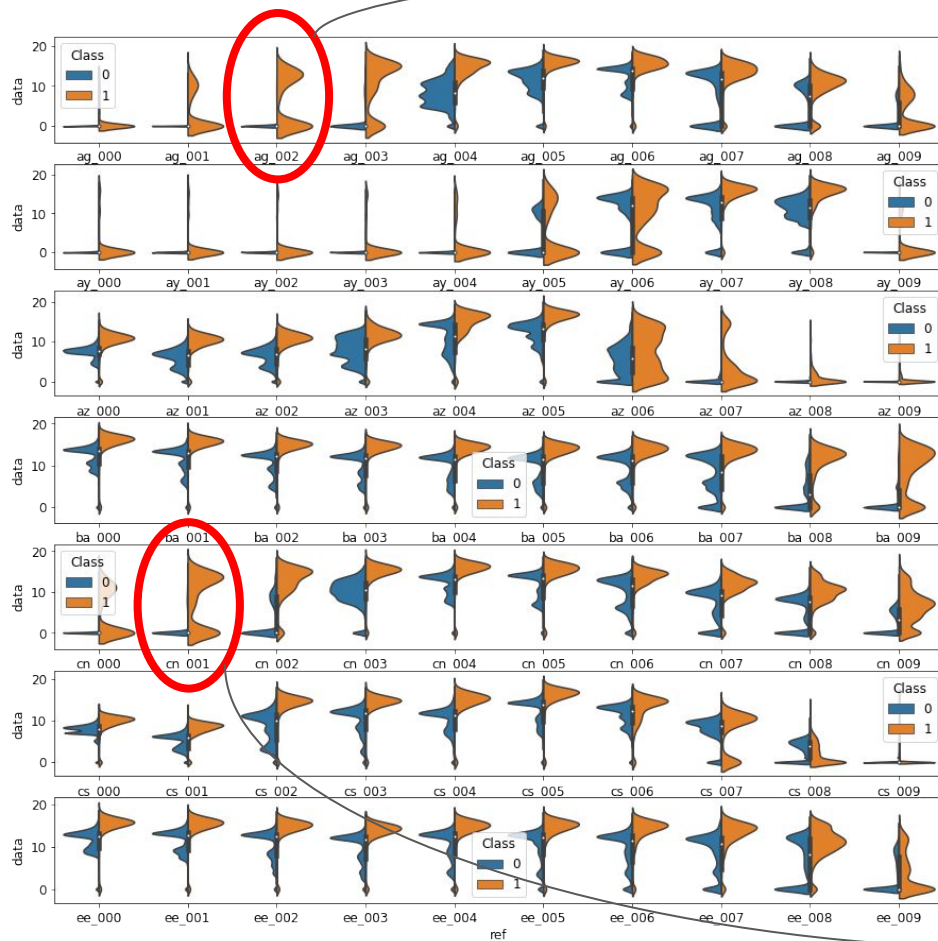


Histograms : system age



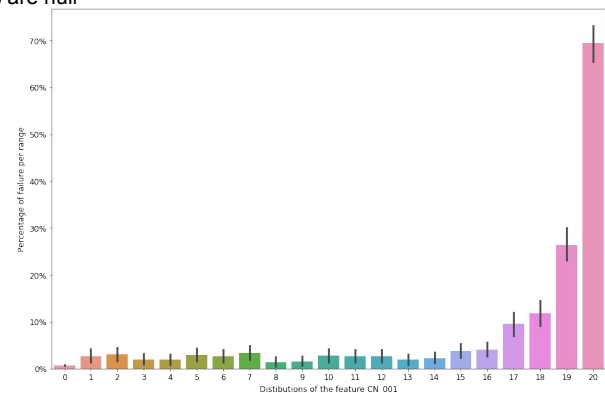
! Surface have been scaled : Class 0 is more than 50 times Class 1

Overview

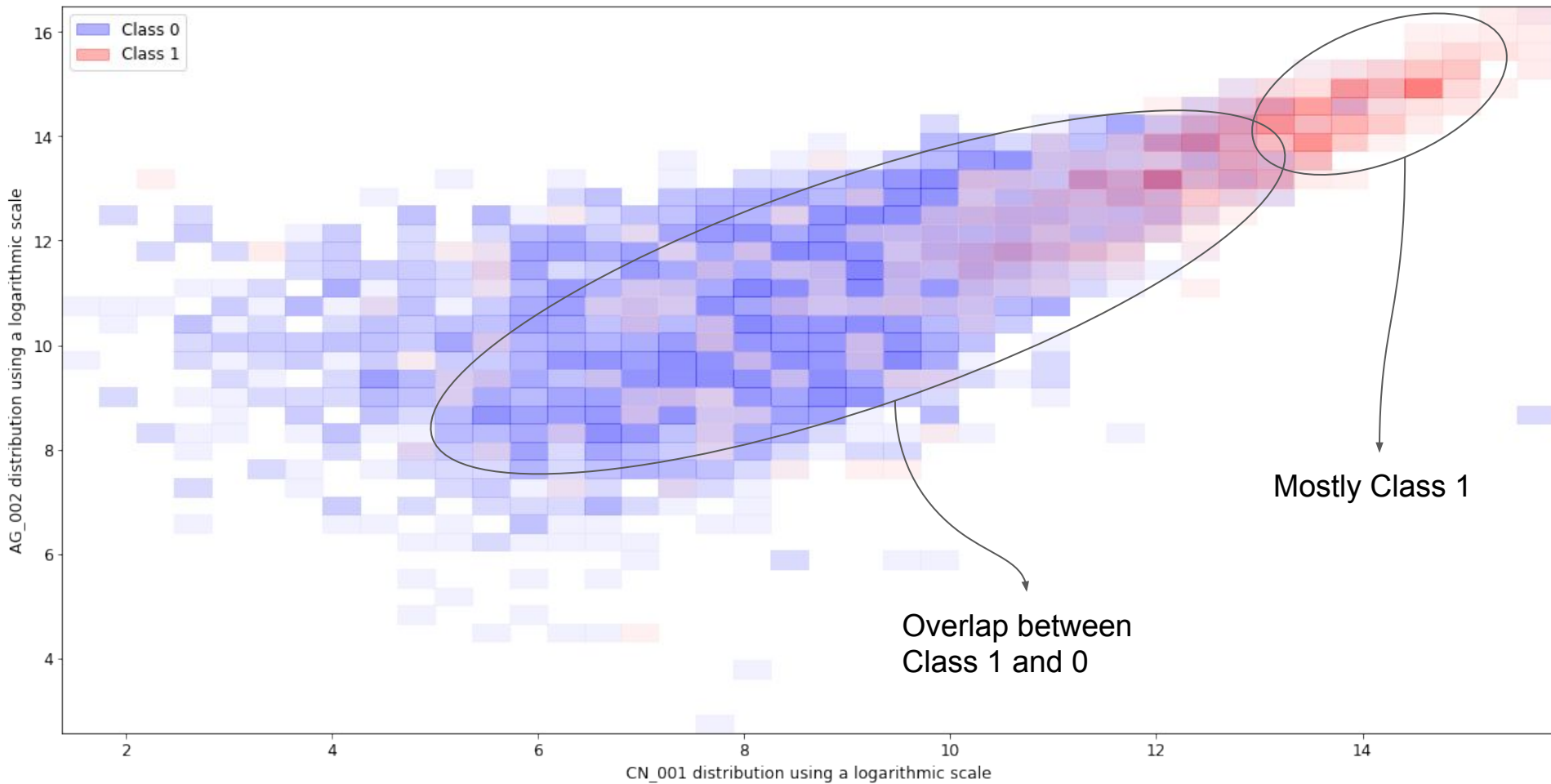


Majority of the measurements are null

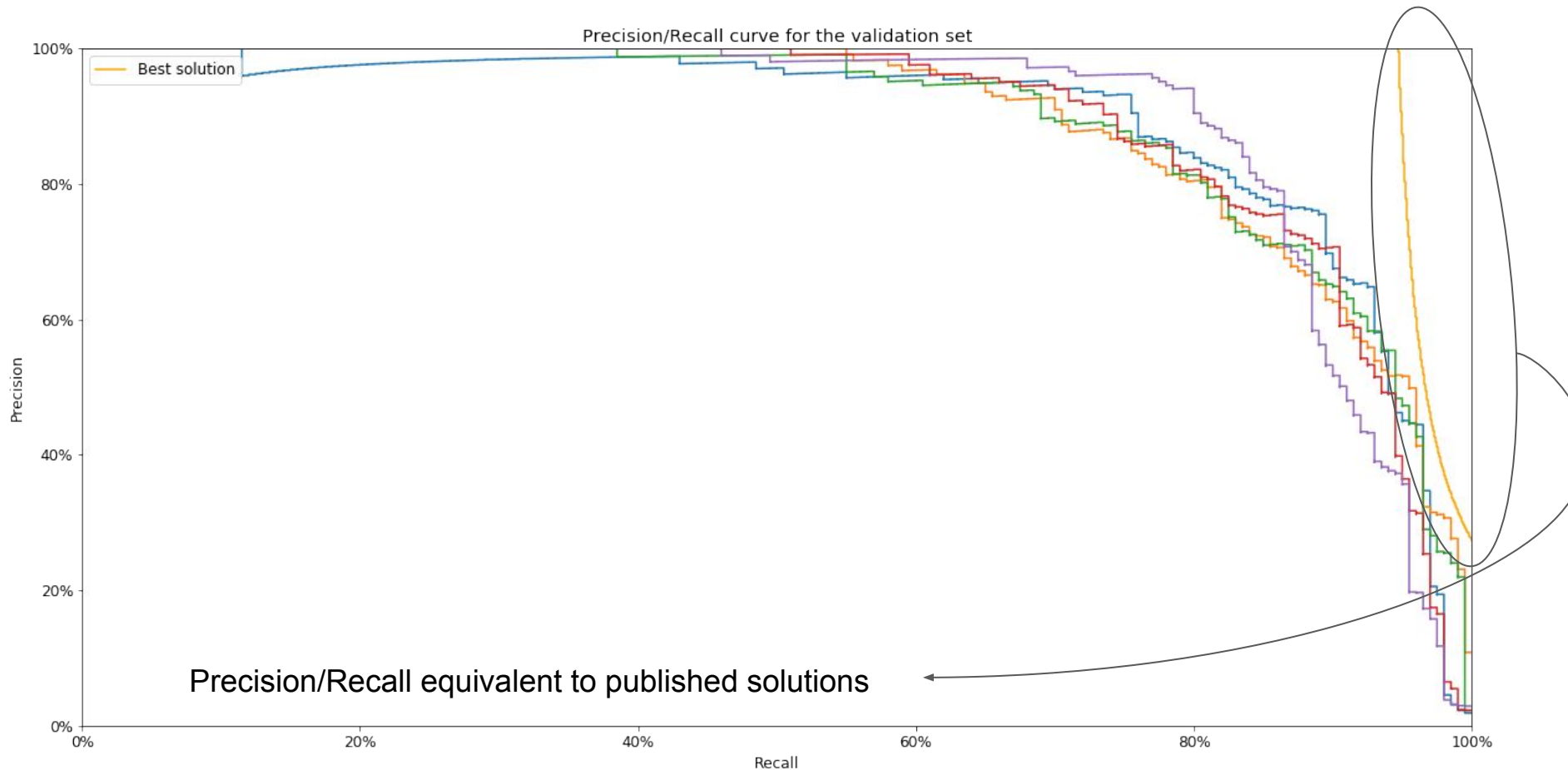
20 buckets of 200 measurements



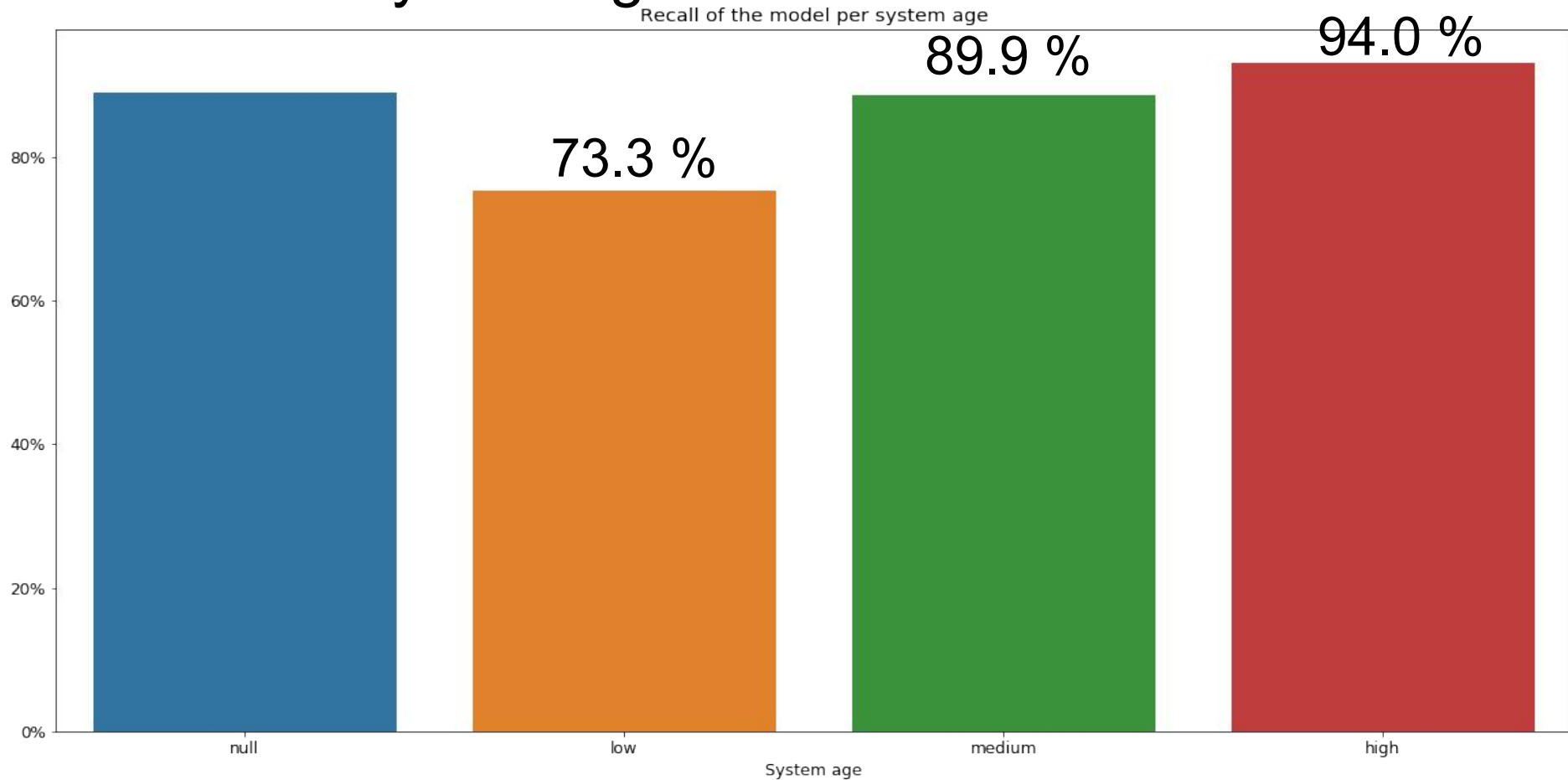
Closer look at AG_002 and CN_001 (log scale)



First lightGBM on the dataset



Recall and system age

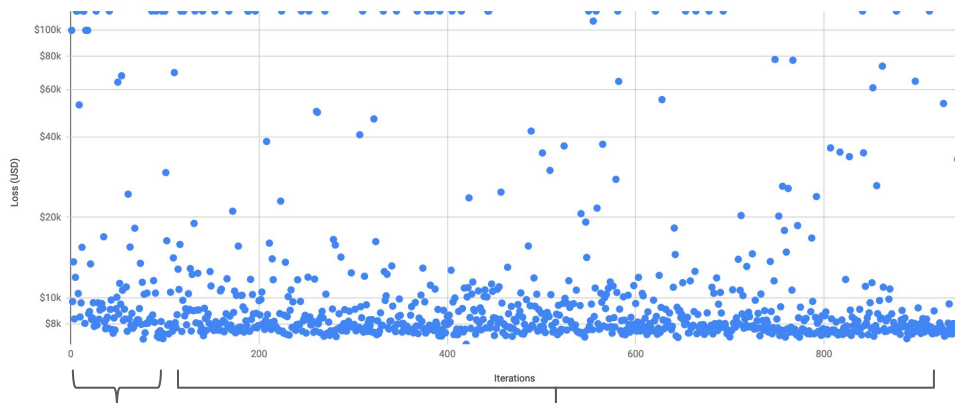


Hyperparameter tuning using hyperopt

Estimator : Tree of Parzen Estimators (TPE)

- High risk of over-fitting
- High dimensional space:
 - Different imbalance dataset settings
 - Unbalance: True
 - Scale_pos_Weight
 - Avoid overfitting :
 - Feature fraction
 - Reg alpha
 - Reg lambda
- Custom function:
 - Custom score : Output actual cost to business on the validation set using variable “threshold” in decision.
 - Custom loss function with parameter “Beta” to increase weight of type II errors.

Evolution of the loss during the Hyperparameter optimisation (5 fold cross-validation)



Rapid improvement during first 100 iterations

Variability remained high but still gradual improvements

Very important for end-result, 2nd optimisation on reduce space for “threshold” and “beta” only was required

Ref :

- <https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html#deal-with-over-fitting>
- <https://github.com/hyperopt/hyperopt>

LightGBM and RandomForest

LightGBM:

- No feature engineering and missing value handling.
- Likely to overfit
- High correlation between hyper-parameters
- Required custom functions to address business objective
- Very fast (#20 sec per training)

	Prediction 0	Prediction 1
Class 0	15248	377
Class 1	12	363

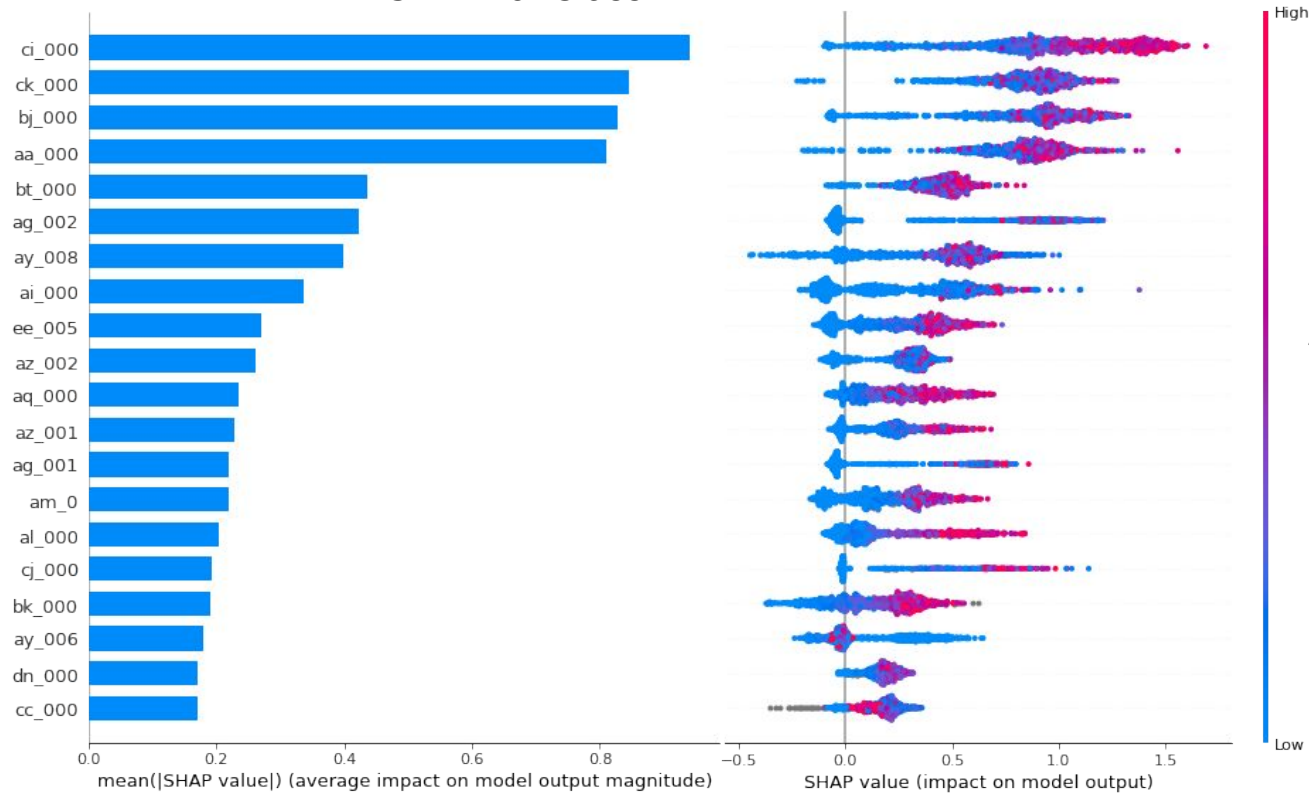
RandomForest:

- Missing values handling via “SoftImpute”
- Little to no over-fitting
- Hyper-parameter tuning was simple (but long)
- Customer scorer to select right model, but not need for further customization
- Slow (#25 minutes per training)

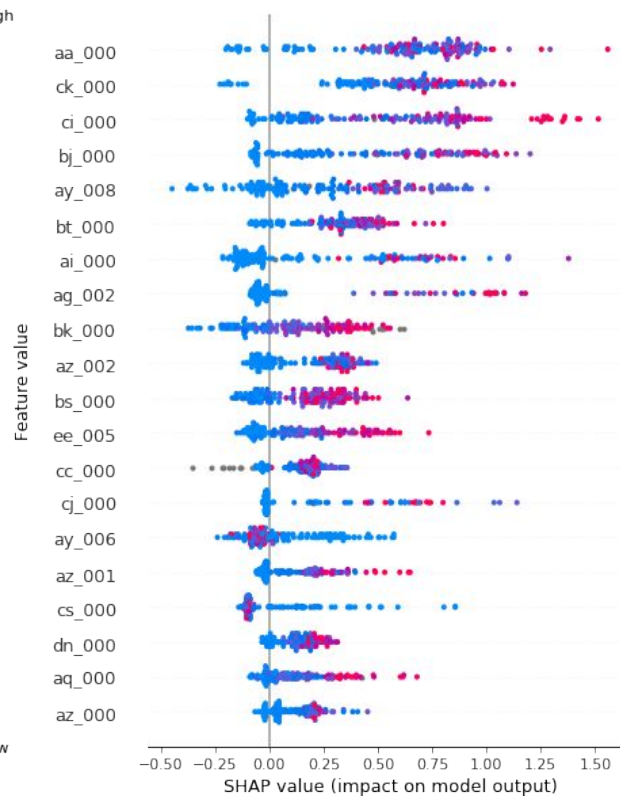
	Prediction 0	Prediction 1
Class 0	15083	542
Class 1	9	366

Understanding the model drivers

SHAP for Class 1



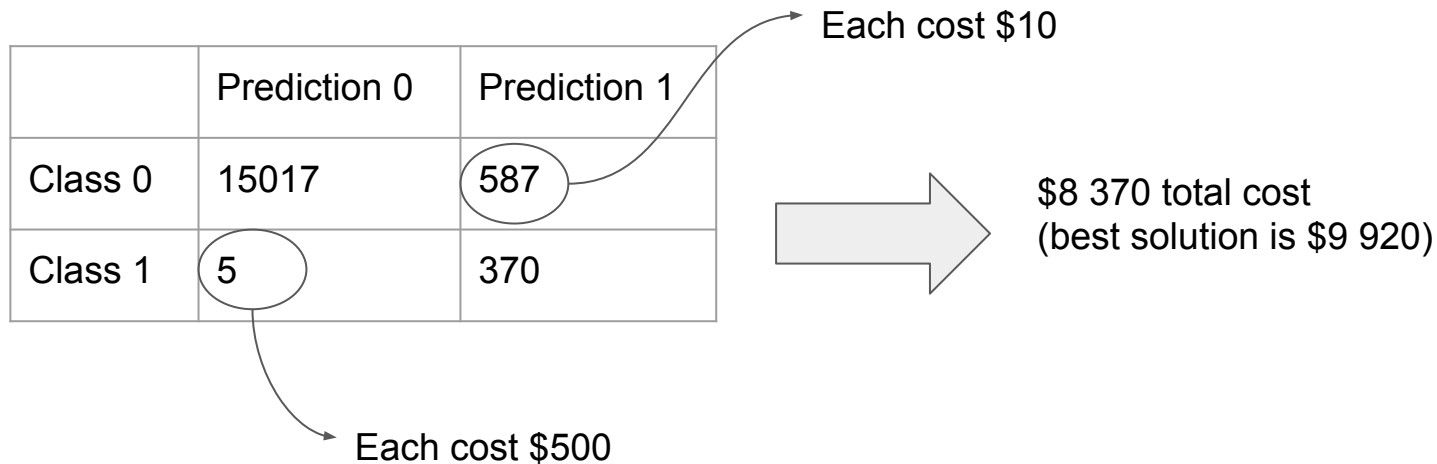
SHAP summary for misclassified



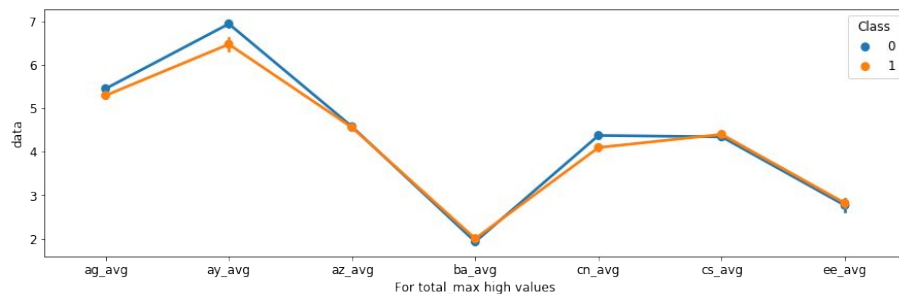
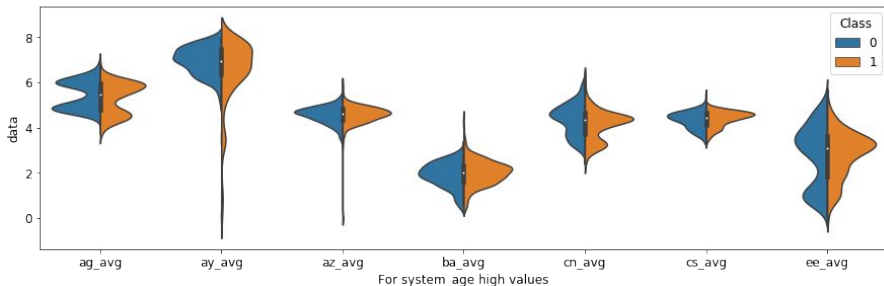
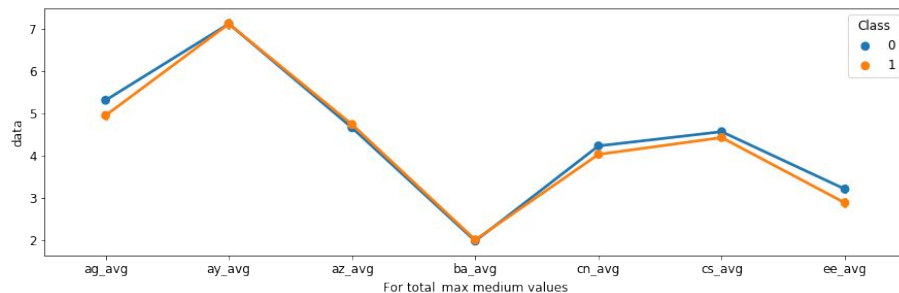
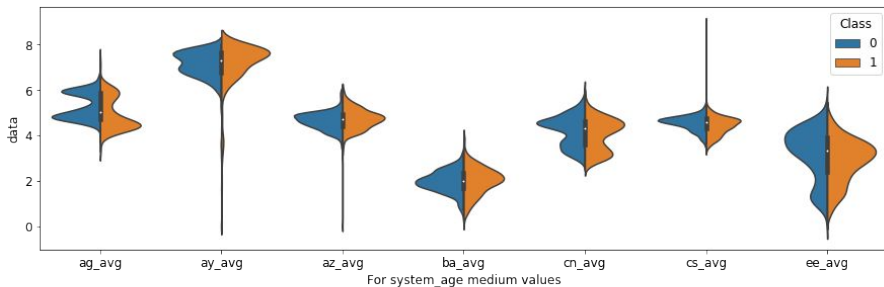
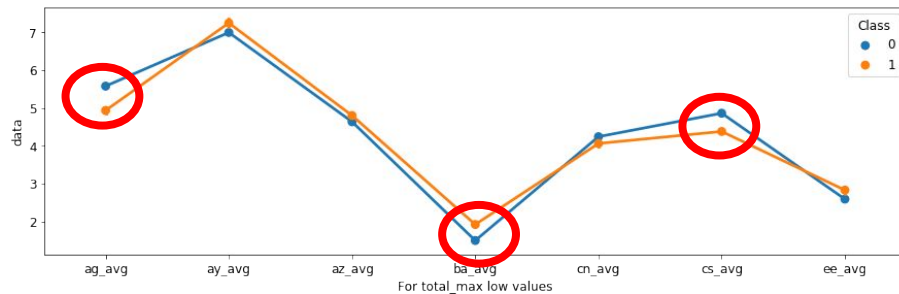
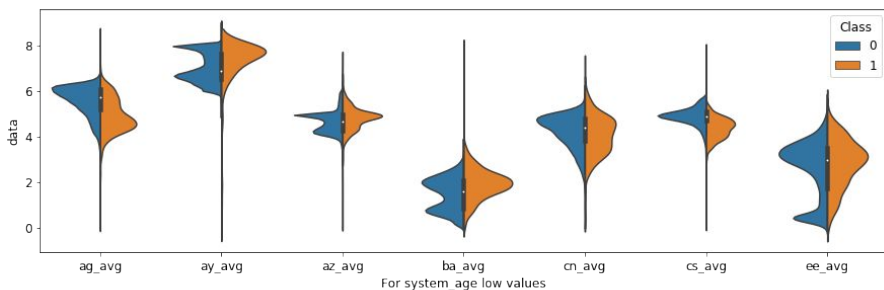
Model Stacking ?

Stacking the Model using an “OR” logic as it is critical not to “miss” a Class 1:

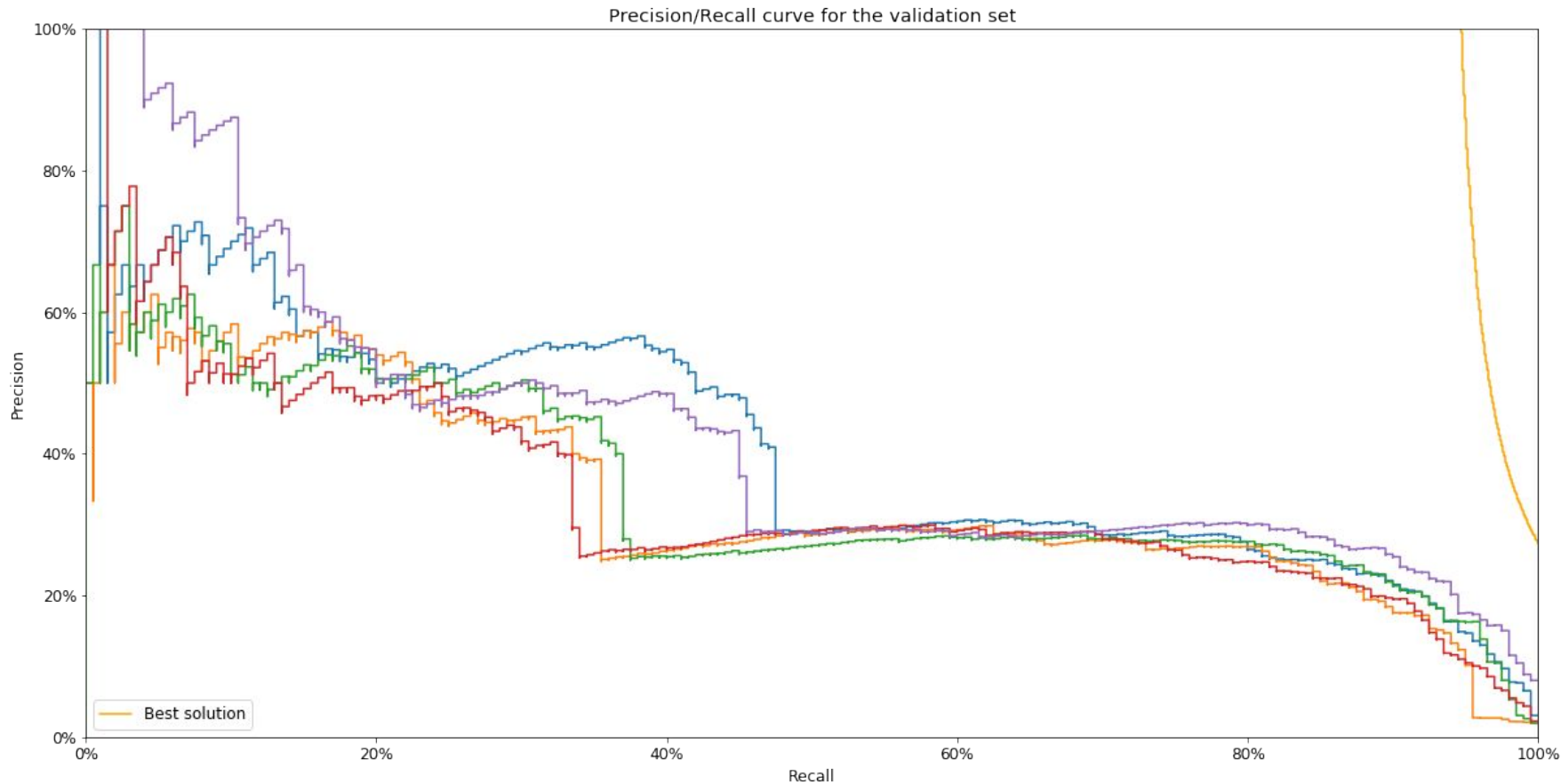
- Number of positive from lightGBM: 758
- Number of positive from Random Forest: 925
- Number of positive from using both model: 978



Further feature engineering for linear model



Logistic regression after scaling, PCA, missing values handling.



Conclusion

- State of the art classification can be done with little feature engineering, but would need to be confirmed on more data.
- SHAP graphs allows us to understand the features that are driving the classification and therefore potential explanation of their origin (with domain expertise).
- Classification seems to required some non-linear step (KNN to complete missing values for examples).

Improving classification:

- Get more data on failure of “younger” system to increase detection.
- Dig deeper in “Counter” features to do feature engineering.