Predicting Pipeline Failures

In collaboration with Veolia Research & Innovation

Thomas Bordier, Christophe Lanternier

Spring 2017

Table of contents

Data presentation

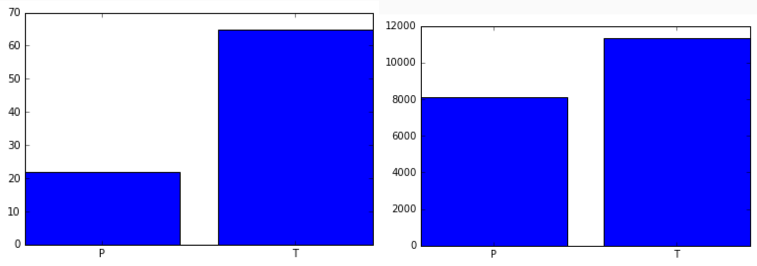
Data presentation

	Feat1	Feat2	Feat3	Feat4	Length	Year Constr.	Year Last Fail.
1	T	IAB	-0.209841	C	3.5972	2001	NaN
2	T	U	2.184992	M	4.0547	1965	NaN

Table 1: Sample from the raw dataset

- Low dimensionality
- No obvious invariance
- Highly unbalanced dataset: **1/200** pipeline fails.

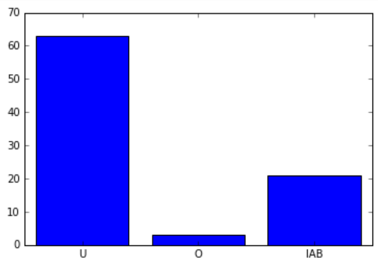
Basic Data Observations



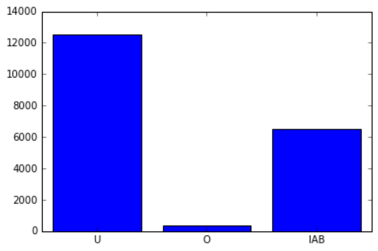
(a) Broken Pipes (b) All Pipes

Figure 1: Feature 1 observations

Basic Data Observations



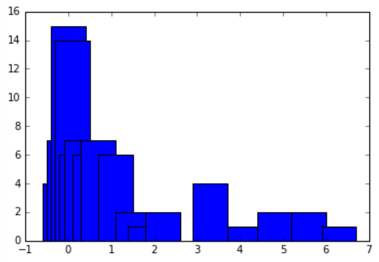
(a) Broken Pipes



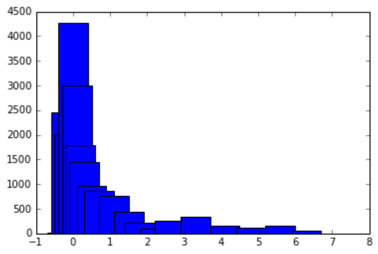
(b) All Pipes

Figure 2: Feature 2 observations

Basic Data Observations

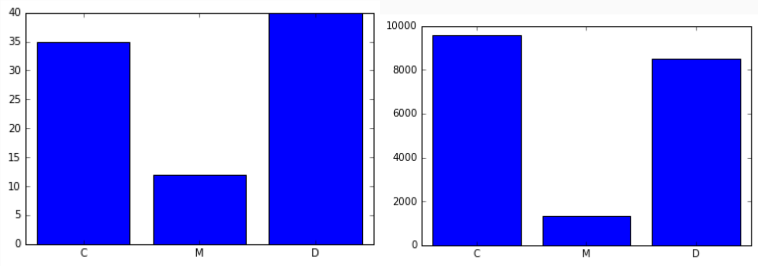


(a) Broken Pipes



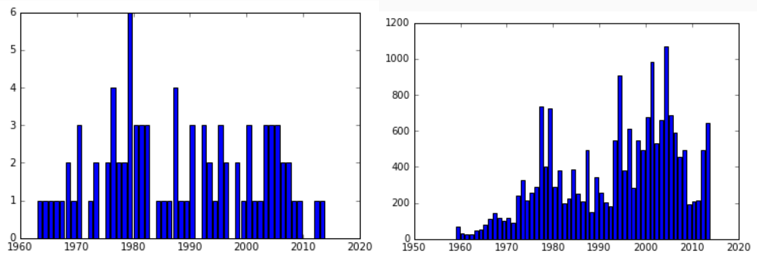
(b) All Pipes

Figure 3: Feature 3 observations



(a) Broken Pipes (b) All Pipes

Figure 4: Feature 4 observations



(a) Broken Pipes

(b) All Pipes

Figure 5: Pipe Length

How to work with an unbalanced data set?

Main issues

There are two main issues to be tackled:

- Re-establish balance in the dataset
- Detect and prevent over-fitting

detecting and preventing over-fitting

Detecting over-fitting:

- Implementation of a randomized train/test split
- Observation of mean scores and variance over 20 experiences

Preventing over-fitting:

- Focus on regularized models (Adaboost with low number of estimators, Logistic Regression with high regularization)

These precautions allowed a big jump in performance and ranking.

Reestablishing balance

We considered two ways of doing so:

- Simple duplication
- Synthetic Minority Over-sampling TEchnique [1]

SMOTE

Intuitive comprehension of the SMOTE algorithm:

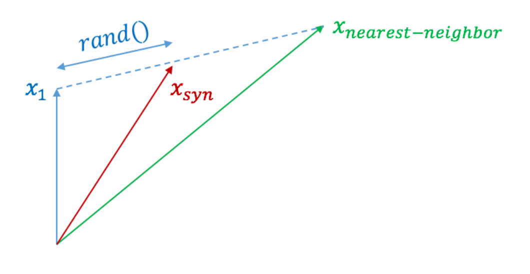


Figure 6: SMOTE explained

Performance comparison

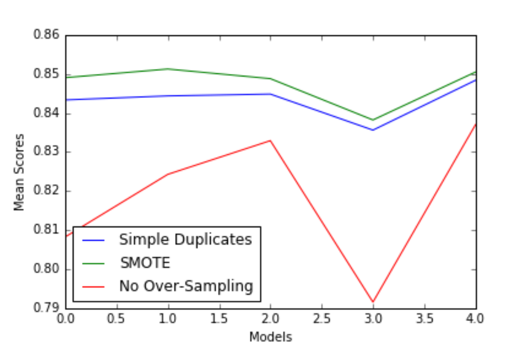


Figure 7: Mean score for 3, 5, 7, 10 and 20 estimators on Adaboost

Performance comparison

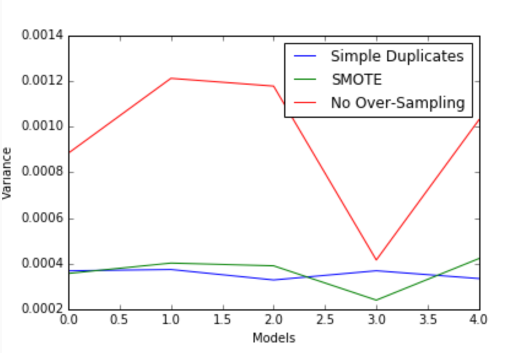


Figure 8: Variance for 3, 5, 7, 10 and 20 estimators on Adaboost

Data Representation

How to go beyond basic features?

	F3	Length	Year Const.	Year Last Obs.	P	T	IAB	O	U	C	D	Dr	M
1	-0.001506	0.000494	0.003946	-0.007229	0	1	1	0	0	1	0	0	0
2	0.015677	0.000557	0.014094	-0.007229	0	1	0	0	1	0	0	0	1

Table 2: Basic preprocessing

- **Idea 1:** non-linear combinations
- **Idea 2:** neural network embeddings

Non-linear combinations

Implement binary operations such as:

- AND
- OR

For pairs and triplets for categorical variables

	F3	Length	Year Const.	Year Last Obs.	P	T	IAB	O	U	C	D	Dr	M
1	-0.001506	0.000494	0.003946	-0.007229	0	1	1	0	0	1	0	0	0
2	0.015677	0.000557	0.014094	-0.007229	0	1	0	0	1	0	0	0	1

Table 3: Basic preprocessing

Non-linear combinations

Metrics	Raw data	Pair 'and'	Pair 'or'	Pair 'and' + 'or'	Triple 'and' + 'or'
Mean AUC Score	0.840	0.860	0.857	0.860	0.865
Variance AUC Score	4.38×10^{-4}	4.33×10^{-4}	7.95×10^{-4}	4.21×10^{-4}	$6.50 \times 10—4$

Table 4: Evolution of our estimated score with binary operations, with AdaBoost

Careful

Curse of dimensionality when too many combinations (13 features to 900 features approx.)

Neural Network feature

- We gave neural network generated features, a try, without much success.

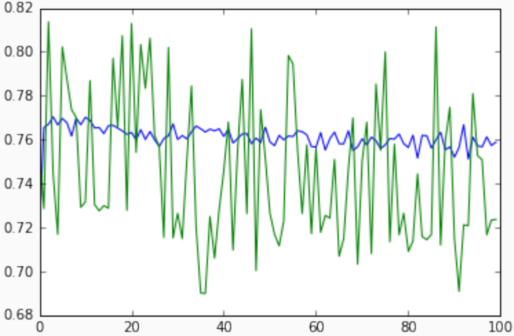


Figure 9: Evolution of accuracy with epochs

Classification Algorithms used

Classification algorithms

- Logistic Regression
- Decision Trees
- SVM
- Neural Networks
- AdaBoost

Focus on best performing algorithms

- Logistic Regression
- AdaBoost

Logistic Regression

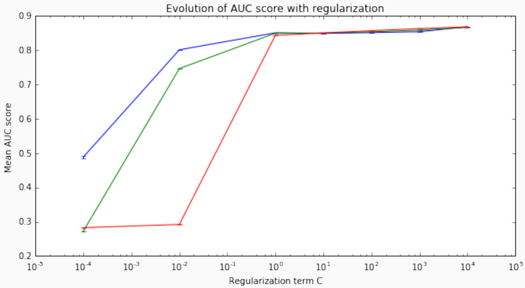


Figure 10: Evolution of score with penalization

Logistic Regression with $L1$ penalty

Feat	F3	Length	Years Last. Fail.	TC and	TDr or	IABM or	ODr or	OM or	POC or	TIABDr or	TODr or	TUD and
Importance	-69	125	402	-1	-1	-6	-1	-6	-1	-1	-1	-1

Table 5: Feature importance in model

Interpretability

$L1$ penalty leads to sparsity

This method leads to a score around 0.76 locally, 0.74 on the website.

Adaboost basics

- Model update:

$$C_m(x_i) = C_{(m-1)}(x_i) + \alpha_m k_m(x_i)$$

- Weight update and loss function:

$$E = \sum_{i=1}^{N} w_i^{(m)} e^{-y_i \alpha_m k_m(x_i)} \text{ with } w_i^{(m)} = e^{-y_i C_{m-1}(x_i)}$$

Conclusion

Our performance depended on:

- Effective **over-fitting control**
- Effective data **augmentation**
- Effective data **representation**
- **Optimization** of hyperparameters for AdaBoost algorithm

Final Performance

Ranked 2^{nd} - Final Score : 0.8845

Thank you for attention, any questions?

- Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research 16, 321–357*, 2002.
- Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences , 55(1):119-139*, August 1997.
- Robert E. Schapire. *Explaining Adaboost*
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, *An Introduction to Statistical Learning*, 2013.
- Jerome H. Friedman, Greedy Function Approximation: A Gradient Boosting Machine. *IMS 1999 Reitz Lectures*. 1999