Kaggle project - CORAVARYOU team

Antoine Arnoult, Gaspard Cotte, Youssef Irhboula, Jules Ravix

February 2021

The problem of looking for meaningful emails among thousands of promotional ones is a well-known issue tackled by all the email service providers. This Kaggle challenge made us create a multi-label classifier that can classify an email into one or more of the eight classes based on the metadata extracted from the email that were given to us. It was the opportunity to grasp how a machine learning problem can be tackled and to confront ourselves to a stimulating online-competition with other teams of the ML course.

1 Feature engineering

1.1 Motivation and intuition behind each feature, new features creation

We tried to get as much use as we can of all the data provided by creating new features that allowed us to exploit more efficiently the existing features.

Firstly, we used directly the numerical features. For our neural network we had to normalize them but it is a particular point.

Then it was time to consider how we were going to use the string features and make the most of them. The idea was to screen information that would be noisy for the classification, and to remove it from the dataset used to train the models. The noise detection was very empirical : we ran trained our models with many feature combinations and checked for which feature combinations we got the best scores with our submissions on the Kaggle plateform. Here is a further explanation of how we dealt with string features :

• Concerning *tld* and *org*, we directly run the *Onehotencoder* algorithm.

• When it comes to *mail_type* we decided to either split on '/' and so creating two new features that we called *mt1* and *mt2* (for "mailtype") (cf our notebook), or to run directly the *Onehotencoder* algorithm with the *mail_type* feature.

• Finally, concerning the *date* feature, it isn't interesting to transform it into a complete disjunctive table (with the *Onhotencoder* algortihm) because each column would probably represent only one line : no mails have been sent at the exact same time, same day hour minute and second. However, we decided to use the *date* feature to create many new different features that were more useful for our models. We thus extracted and created a new feature for the day of the week, the hour, the minute, the second, the month, the year, and the timezone information. We also created teh following features : *w*, *s0* and *hm*, which correspond respectively to : whether the email was sent during the weekend or not (we got that idea by thinking that it's a better indicator than the specific day), whether the email was sent on an exact minute and 0 seconds (intuition : it may detect automatically sent emails), and finally *hm* is a more precise measure of the hour which takes into account the minutes without using directly *mm* feature which is was acting like noise -if used alone.

1.2 Experiments about the impact of various combinations of features on predictive performance

As written in the previous section, after having obtained the new Data-frame by remodeling the string features, we trained our models using different sets of features (corresponding to the columns of the data-frame) in order

	date	org	tld	ccs	bcced	mail_type	images	urls	salutations	designation	chars_in_subject	chars_in_body	mt1	mt2	hh	mm	ss	d	m	y	fus	fus2	hm	s0	w
0	Thu, 13 Jul 2017 08:55:57 +0000	twitter	com	0	0	multipart/alternative	7	56	0	0	67.0	36243	multipart	alternative	8	55	57	Thu	Jul	2017	+0000	None	8.916667	0	1
1	Sun, 30 Sep 2018 14:42:12 +0000	mailer	netflix.com	0	0	multipart/alternative	5	33	0	0	27.0	27015	multipart	alternative	14	42	12	Sun	Sep	2018	+0000	None	14.700000	0	0
2	Mon, 13 Feb 2017 10:47:00 +0530	iiitd	ac.in	0	0	text/plain	0	2	1	0	22.0	788	text	plain	10	47	0	Mon	Feb	2017	+0530	None	10.783333	0	1
3	Thu, 16 Jun 2016 09:56:23 +0000	twitter	com	0	0	multipart/alternative	8	53	0	0	79.0	39504	multipart	alternative	9	56	23	Thu	Jun	2016	+0000	None	9.933333	0	1
4	Mon, 18 Apr 2016 01:51:59 +0530	iiitd	ac.in	0	0	multipart/mixed	0	0	0	0	24.0	178773	multipart	mixed	1	51	59	Mon	Apr	2016	+0530	None	1.850000	0	1

FIGURE 1 – The new Dataframe

to grasp the impact of some features on the precision of our prediction models. We thus observed that minutes or seconds taken alone could be considered as noise (intuition : mails are sent at random moments within a same hour and/or same minute, so minutes or seconds alone don't provide any further information, and using these features could lead to overfitting and so increase the error).
However, some features helped to improve the model : the hour (the hour in a day isn't random and depends on the type of mail), and the FLOAT HOUR *hm* that takes minutes into account is a little more precise and improves a little the prediction.
Some features are interesting but need to be used intelligently not to overfit : the day in the week led to overfitting, but a the simple feature w (cf the previous subsection) which inform whether the day belongs to the weekend improves the model.

In order to select the more informative features, we made several submissions of predictions on the Kaggle plateform using different sets of features, to grasp empirically their importance.

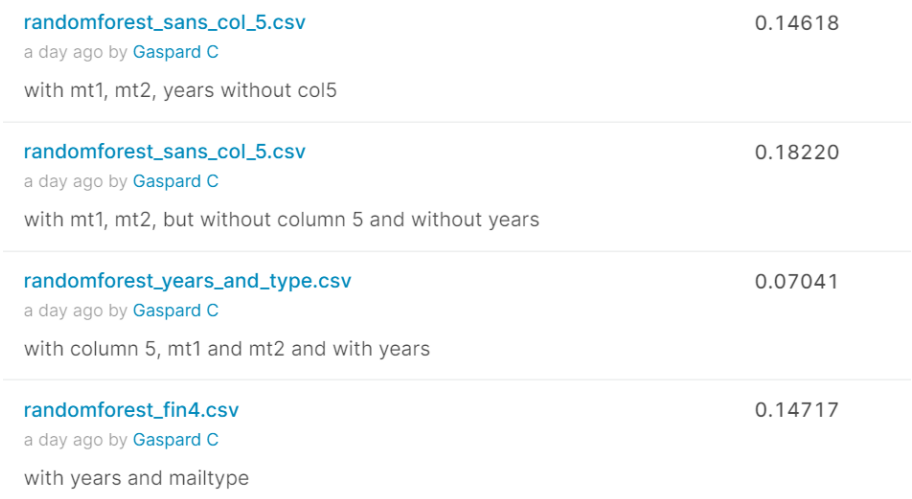

FIGURE 2 – Example of "empirical" feature selection on the Kaggle plateform

When it comes to dimension reduction : PCA increased the prediction error without significantly reducing the time complexity of our algorithms, so we didn't use it at the end. We also changed the CSV file containing the dataset with excel manipulations, in order to delete all the ORGS and TLDS that appeared less than 10 times which reduced a lot the number of columns created by the ONEHOTENCODER algorithm. It didn't affect at all the results of the RANDOMFORESTCLASSIFIER, though reducing a lot its execution time.

2 Model tuning and comparison

After the work of feature selection and data preprocessing, we had to chose a training algorithm for our model prediction. In reality, we combined both works and made our predictions become more refined by crossing them.
We compared mutliple classifiers thanks to two main indicators : their cross-validated performance and their score

on the test set given by the Kaggle submissions.
Note that for SVMs, logistic regression and AdaBoost, we had to structure the label data differently because of the shape of the data the algorithms tolerate. We created a list of dimension 1 containing the label information. For these algorithms, the one-hot encoding method caused some problems so we used mainly numeric features, but it implied a loss of information. On the contrary, the one-hot encoding method and all the data preprocessing that we explained in the first section worked well with RandomForest, Decision Trees and K nearest neighbors.

Multiple classifiers comparison			
Classifier	Cross Validation scores(CVS), k=4	Mean(CVS)	Mean Log Loss (Kaggle submission)
Logistic regression	[0.27253 0.28675 0.27687 0.17626]	0.24	/
K nearest neighbors	[0.34351 0.33706 0.33514 0.34526]	0.34	/
Adaboost	[0.37598 0.37315 0.37487 0.35040]	0.37	/
Neural Network	[0.82435, 0.82385, 0.82990, 0.82898]	0.82	0.39
Decision tree	[0.81992 0.82012 0.82617 0.8242]	0.82	0.12
RandomForest	[0.82788 0.82597 0.82667 0.83341]	0.83	0.062

With regards to these results (cf table) we decided to consider only the Decision Tree and the Random Forest algorithms for the remaining part of our work and to improve our prediction results. We didn't even submit predictions for the first three classifiers of the table because they got non conclusive Cross Validation Scores (which can be explained by the fact that we had to discard string features to make the algortihm run).

We then submitted the prediction results given by Decision Tree and Random Forest classifiers -after running the models with different sets of features -as explained in the first section- and parameters, which allowed us to increase our model complexities so that the test error decreases until it increases again which was a sign of overfitting. These experiments allowed us to choose the best features and parameters that allowed us to minimise the error (with regards to our data pre-processing and feature selection) without overfitting.

Note that we tested a few other models but we abandoned them because they either didn't converge or gave very unsatisfying scores. Those models were : SVM, and ridge regression, logistic regression, KNN.

Conclusion : this challenge was a very motivating competition that made us practice machine learning in a concrete way !

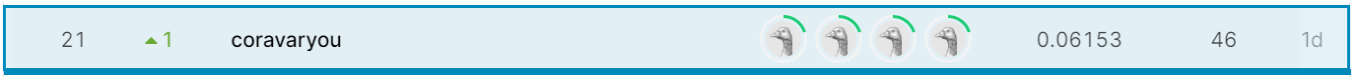

FIGURE 3 – Our final position