PREDICTION OF RETWEETS

INF 554

December 13, 2020

Julien BERGEROT, julien.bergerot@polytechnique.edu
Alexis Groshenry, alexis.groshenry@polytechnique.edu
Capucine Leroux, capucine.leroux@polytechnique.edu
Team name : CapAlJul

CONTENTS

1 DATA PREPROCESSING

We first cleaned our data : transforming the boolean values into integers, centering and reducing numerical values, and then worked more specifically on the data to extract relevant features. We used this article [1] as a guideline, separating our features in three categories.

1.1 Text features

1.1.1 • Occurrences of URLs, hashtags, mentions and specific vocabulary

We transformed the input fields to keep boolean values telling if the tweet contains URLs, hashtags and mentions and also a field counting the number of those.

Added to these key features, we also found out that the presence of some specific vocabulary boosts the number of retweets. Thus, we looked at the occurrences of retweet vocabulary, upper words and exclamation points. On the same idea, the vocabulary naming a specific place, person or organisation has also a boosting effect. We checked if a tweet contained such words using the "en_core_web_sm" model from the Spacy library [2].

1.1.2 • Sentiment level

Whether a tweet is very positive or negative has an important impact on its popularity, particularly in the Covid crisis. That is why we added a sentiment score to our data (0 = very negative, 1 = very positive). To do so, we trained a model on the Sentiment140 database [3] that recollects 1.6 million tweets and their sentiment score.

Our first idea was to classify the tweets between positive and negative with a linear SVM. We had to preprocess the text, cleaning it by removing urls and none-text characters, building a vocabulary dictionary with the dataset, and vectorizing the text in an array counting the occurrences of each word in the dictionary. The two main problems of this solution are that we only get a binary answer (positive or negative), which does not take neutral feeling into account. And the second problem is that all words are uncorrelated with each other. Therefore, if the word "happy" is strongly associated with a positive score, the sentence "I am not happy" will be scored positive.

To overcome these issues, we decided to build a RNN model instead. It required more processing. First, to improve the cleaning process, we used a Tweet tokenizer that is specially designed for tweets. We also replaced common abbreviations and removed mentions and urls again. But one important step we added was to embed the words with GloVe to pass on a tensor to the model which takes the meaning of the word into account. We then trained our RNN model and added the sentiment score feature to our data.

1.2 User features

1.2.1 • Followers, friends and statuses

According to the article [1], the user is far more important than the content of the tweet to predict the number of retweets. Unfortunately, our data doesn't give much information on the user. We used the number of friends, followers and statuses, and we centered and averaged this data.

1.2.2 • Outliers

After running our first model, we noticed that users with very large amounts of followers behaved as outliers in our model, hence a prediction not very accurate about them. Indeed, we noticed that some users had, after preprocessing, values such as the number of followers, way larger compared to others. We looked at such tweets and realized that their numbers of retweets were very high but the model did not manage to predict it. We therefore decided to force the prediction on such tweets to predefined values, based on the mean. Our model would train on the whole data but the final result for said tweets would be forced afterwards, if the model predicted less than our threshold value.

1.3 Time features

We finally transformed the timestamp to extract the month, the weekday and the hour. It is commonly known that some time slots in a day tend to ease a high number of retweets. To identify them, we plotted the distribution of tweets in a day for given number of retweets. We identified peaks and troughs in the distribution and thus put a boolean to indicate if the tweets was published on a strong or weak time of the day, or neither of them. We could have also extracted if the tweet was posted during a holiday, however, since all the tweets were posted around the same week in both the train and the evaluation set, it did not make much difference.

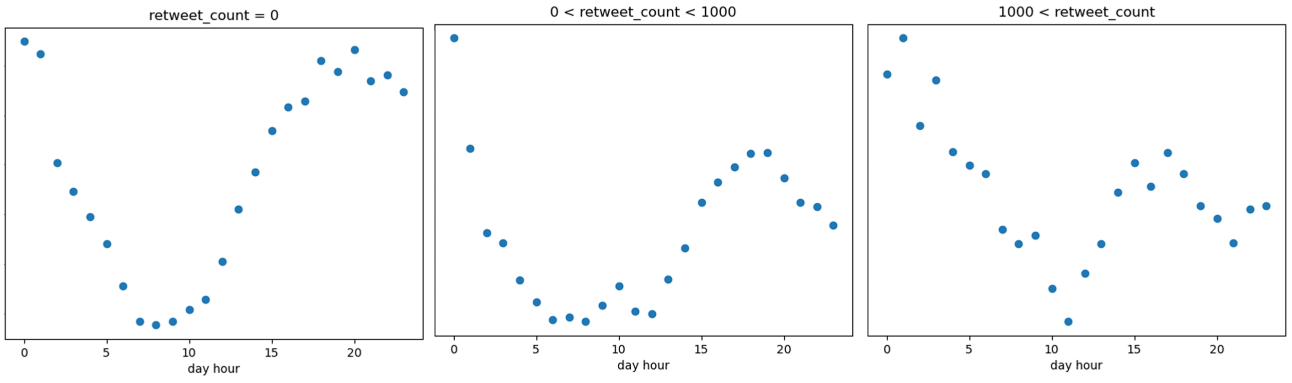


Figure 1: Time distribution of tweets for given retweet number

2 PREDICTION MODEL

2.1 First naive model

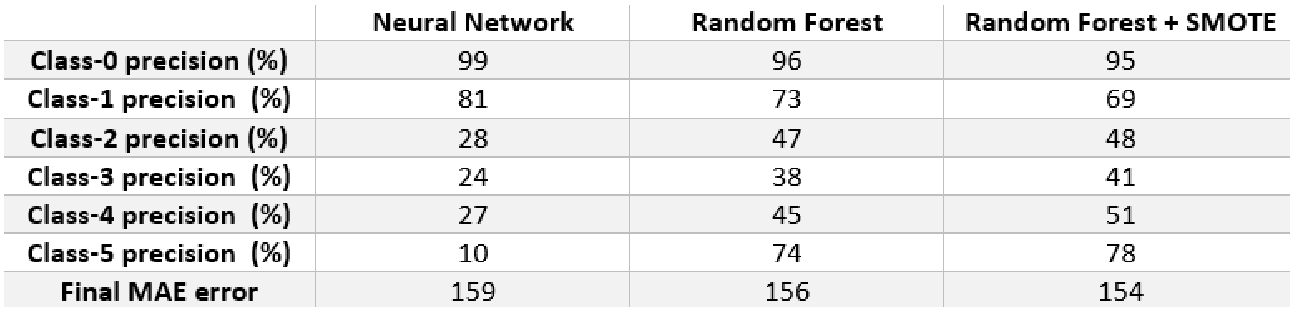
As our first approach, we decided to launch a unique model, using only the numerical input fields. We designed a basic three layers neural network and used the ReLU activation function for each dense layer. We used *keras* implementation and trained this model in a few minutes. Our first try led to a 159 MAE error and we considered it as the reference value to improve.

To that extent, we wanted to add more text features and therefore combined our numerical treatment with a text treatment. For each tweet's text, we deleted stop words and, for each final word, embedded it via GloVe, so as to have a link with familiar words. We then fed all this to the neural network to train. We had LSTM cells for the text and then concatenate the result with the other features to dense layers. We thought this would boost our score, but it did not since we still got a 159 MAE error.

2.2 Classifier

When we were training our first models, we noticed that many of them only predicted 0 values, despite having reduced the number of these tweets in the training set. To tackle this, we decided to firstly classify the data before the actual prediction. We started by defining three classes: 0 , $<$ 1000 and the rest. When we later found this paper [1], it supported our claims and we increased the number of classes to 6 on the same principle as before so as to minimize the error caused by misclassification.

We tried different classifiers, starting with a basic neural network with a soft-max activation function but we obtained a much better precision using a Random Forest classifier. Finally, we used the SMOTE method presented in [1] to create artificial data in large retweets classes and get a balanced training dataset. This allowed us to improve the precision for these classes since they were responsible for the most important part of the error. The following table compares the performance of each classifier.



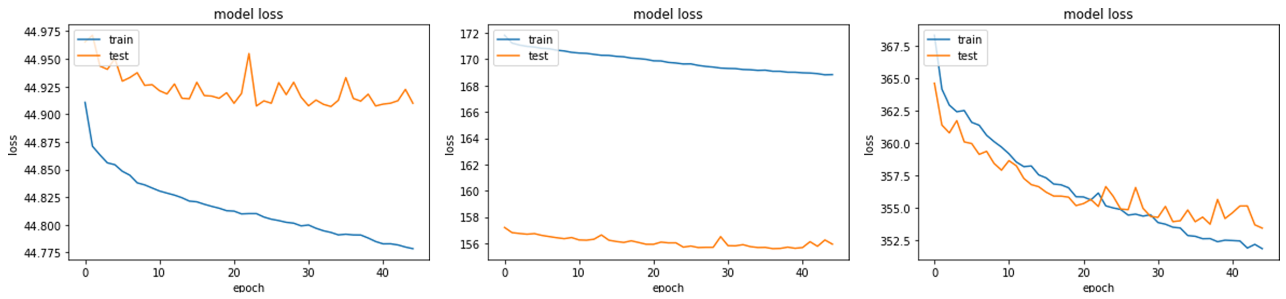
	Neural Network	Random Forest	Random Forest + SMOTE
Class-0 precision (%)	99	96	95
Class-1 precision (%)	81	73	69
Class-2 precision (%)	28	47	48
Class-3 precision (%)	24	38	41
Class-4 precision (%)	27	45	51
Class-5 precision (%)	10	74	78
Final MAE error	159	156	154

Figure 2: Performance of the classifiers

2.3 Regression

For each of the previous class identified, we trained a neural network, with the same architecture as in the first naive model, on the tweets that were classified in the corresponding class (except for the 0-retweet class for which we automatically predict 0).

We prevented over-fitting by training on 80% of the data and testing on the 20% remaining and by choosing the number of epochs according to the learning curves.



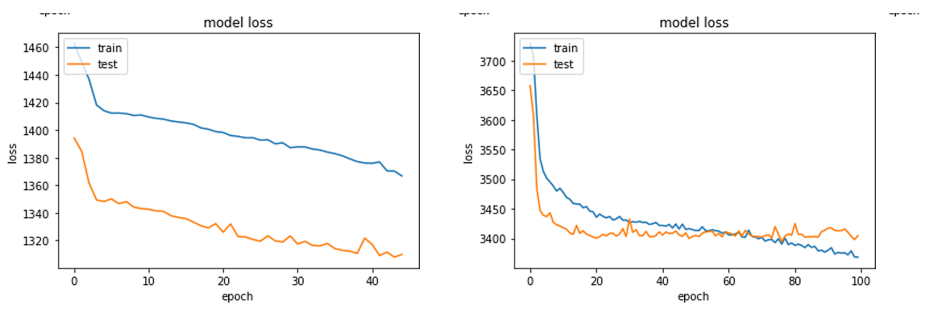


Figure 3: Learning curves for the 5 neural networks

On the previous learning curves, we can see that the model has difficulties to properly learn. Thus we tried other methods notably a Random Forest regressor and a Gradient Boosting regressor but they did worse on the prediction, so we decided to keep the neural network model.

2.4 Handling of outliers

As discussed in this *section*, some tweets were outliers. From the moment the classifier did not classify them in the strongest class, our error would be very high. To counter that, we decided to train our neural network on all the other tweets. Therefore, we had better results for these "normal" tweets. Regarding the outliers, we tried to train a model to predict them, but there was not enough data for training. We decided to force them with a default value, related to how far from the normal data they are. For instance, tweets with a "user_followers_count" value higher than 47, get a forced prediction of 40000.

We tried it for the first and our error dropped down to 150. We wanted to push this process further, but did not manage to recreate this low error ever again. During the last day, we tried different approaches to outliers, and the error got even higher. So we decided to drop this idea, still not knowing exactly how we reach this result.

REFERENCES

[1] T. B. N. Hoang, J. Mothea. *Predicting information diffusion on Twitter – Analysis of predictive features*. J. Comput. Sci, 2017.

[2] https://spacy.io/models/en

[3] https://www.kaggle.com/kazanova/sentiment140