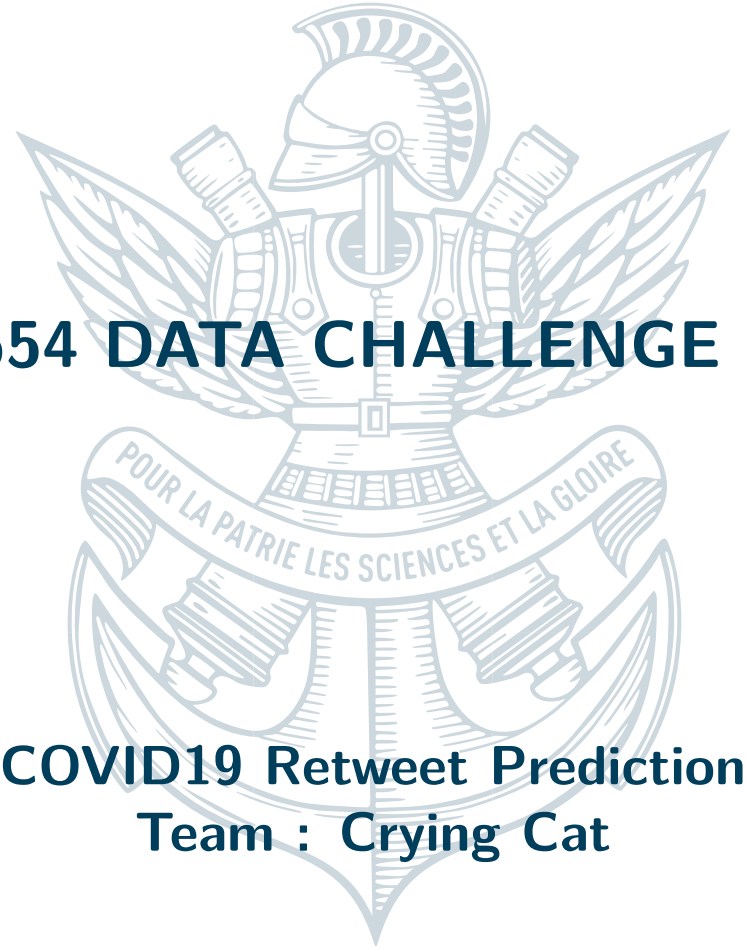


INF554 DATA CHALLENGE 2020

COVID19 Retweet Prediction
Team : Crying Cat

October 4, 2021

Alícia Fortes Machado - alicia.fortes-machado@polytechnique.edu,
Bianca Marin Moreno - bianca.marin-moreno@polytechnique.edu,
Iago Martinelli Lopes - iago.martinelli-lopes@polytechnique.edu

1 FEATURE SELECTION

Our work can be divided in two parts: analysis and adaptation of personal features, and analysis and embedding of text.

Based on [1] our first approach to understand the correlation of our features with the retweet counts consisted on dividing them into intervals $(0, 1, 100, 500, 1000, 5000, 10000, 50000, \geq 10000)$ and making categorical scatterplots and mean value plots for each feature in relation to the interval figs. 2(a) to 2(d). Also, we applied a PCA (figures figs. 3(a) and 3(b)). The projection of our variables on the first two dimensions of PCA implies a strong correlation of `user_followers_count` and `retweet_counts`, while on dimensions 2 and 3 `retweet_counts` has a negative correlation with `user_ statuses_count`. As for the scatterplots and histograms, we see that all features related to user information change considerably with the number of retweets.

For the parameters `url` and `hashtag` we followed the advice of [2] to create a column counting the hashtags and one counting the urls in the tweet instead of dealing only with their text. We also made plots for them figs. 2(e) and 2(f)). One can see that the retweet count variate depending on the number of url and hashtags, which also implies they are important for training.

A general problem was that the dataset is imbalanced (figure 5). The majority of tweets have between 0 and 10 retweets (intervals 0 and 1). Our models all failed to predict large number of retweets (≥ 35000) which enhanced the MAE loss. Trying to solve this problem we decided to apply SMOTE [3], an oversample method that creates new samples for minority classes based on k-nearest neighbours algorithm. However, the losses and accuracy of our models only got worse with this, so we abandoned the idea. We also tried to use random undersampling to do the opposite of SMOTE and remove individuals in popular classes, but again this did not help.

As for the `timestamp` variable, we transformed it into 23 columns representing the hour intervals, and 6 columns corresponding to the week days. Their plots (figure figs. 4(a) and 4(b) annex) suggested that time had little importance to the model, while week days had more. The application of recursive feature elimination to the dataset (RFT) also gave that the hours should be removed for training. Finally, when testing it with our final model we managed to decrease our loss from 154 to 153 by considering only week days and not hours.

The second part concerning the texts analysis posed a particular challenge due to the size of the data text. We first tried to apply a LSTM (Long short-term memory), a RNN adequate to tackle sequences of data. This approach didn't work well considering the architecture proposed in the TP of INF-554, since we weren't capable of reducing the error for the regression. After that, we tried to apply a complete bag of words on the texts, but we had issues with the ram memory demanded.

Finally, we decided to make a feature extraction of the text using tf-idf (term-frequency times inverse document-frequency). First it does a bag of words with a limited number of features. Then it makes a transformation based on the frequency the word appears. The goal is to give less impact to tokens that occur frequently (empirically less informative). We applied this method both to the twitter text and to the hashtags. To diminish the noise we applied a PCA on the columns generated by the text analyses. We left only 66 components out of 100 responsible for explaining 80% of the variance.

We also tried to predict the log of the number of retweets plus one instead of the number of retweets itself. This idea was conceived by approximating the comportment of a retweet as memory-less: if an user sees a tweet, it doesn't matter if it was retweeted n times or not. This means that it has an exponential growth. Thus we tried to apply log and then predict the retweets. This idea worked well for Gradient Boosting method (diminishing the MAE loss on test from 141.14 with no log to 135.92), and also helped performance for simple regressors like Lasso.

Other than that, we used Autoencoders to try to reduce the dimension of the bag of words of the text and hashtag. However, we weren't able to obtain low errors when training the Autoencoder model, so we considered that it was not efficient in this case.

2 MODEL TUNING AND COMPARISION

As mentioned before, one approach was to divide the number of retweets in intervals and approach our problem as a classification problem. Still following [1] we implemented a model with decision tree. As the results were acceptable (mean accuracy of 0.84 and loss of 145 on the test data) we decided to try with a **random forest classifier**. The plot of number of trees vs. losses (figure 7 annex) on the test data made us choose 70 as the number of parameters to avoid overfitting.

Random forest is also good for the interpretability of our dataset. With it we can get a plot of the most important features, as we can see in figure 1.

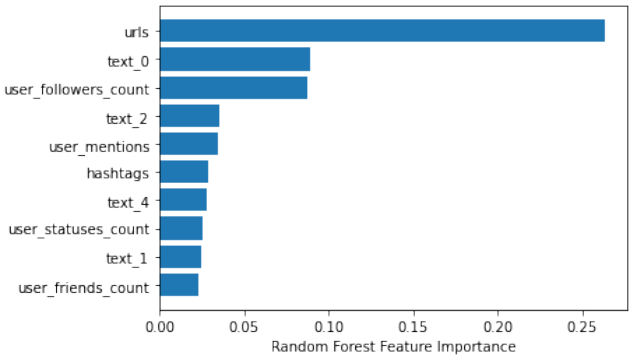


Figure 1: Feature importance by random forest

We can see again that the majority of personal features have a great importance to the model, also some of the text tokens. Apparently having or not urls on the tweet had great importance for the random forest. Again, time is not among the important features.

We also tried a **random forest regression** to use it with the total number of retweets. It took a tremendous time using MAE as loss, but had a reasonable time when using MSE. In order to tune the hyperparemeters we used 'GridSearchCV' from ScikitLearn that tests different parameters and return the best result (figure 8(a)).

Another regression model we used was **k-nearest neighbors**. Usually KNN models deal with non linearity well. Setting the good k is important to avoid overfitting. However, it

can be a slow method to evaluate on longer datasets, and it lacks interpretability. The final performance on our dataset was worse than the all zero prediction as you can see below.

k	3	5	7
MAE loss	233.90	231.82	232.10

Due to its worst performance comparing with other methods and the time of computation we did not spend long time trying to make it better.

The other regression models we tried were the **Passive Agressive algorithm** (PA) that we discovered on the article [4]. Although they used PA to classification, we found it would be interesting to tackle our problem with it. We set the parameter $C = 0.01$ (the default value is 1, but due to our dataset we found that decreasing C corresponded to more regularisation). We also tried the method **Gradient Boosting** commonly used on kaggle competitions and we also tuned its hyperparameters (figure 8(b)). It produces a model as an ensemble of weak prediction models that in our case are decision trees. Finally, we also tried the **Lasso** regressor.

Below we have the table comparing the MAE loss obtained with these models set to their best tuning parameters considering our dataset.

Method	Lasso	Gradient Boosting	Gradient Boosting with log	PA	Regression Tree
Loss MAE	265.02	141.14	135.92	150.87	242.16

The same article [4] also tries to tackle the retweet classification problem by training a model for each time interval separately and joining them at the end. As mentioned above, we found that time did not have great importance, so we tried the same approach on our best neural network model but with the weekdays. Results did not seem to get better.

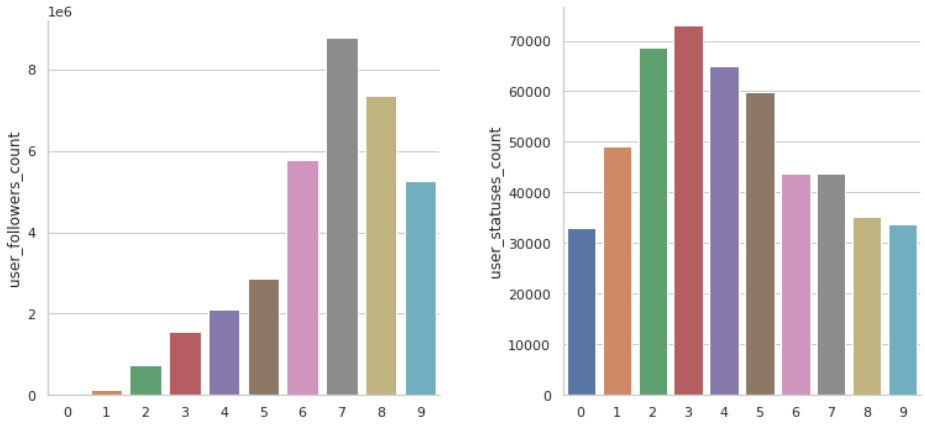
Despite all our tentative using other models, the ones that gave us the lowest losses at kaggle were **Neural Networks**. We implemented them using PyTorch and we experimented with different number of layers and neurons. We used Adam optimiser with the L1 loss, decaying learning rate (the advantage of this is that it can help the algorithm to converge faster and more accurately) and 10 epochs. To justify our choice of epochs we plotted the validation and train loss per epoch on the graph of figure 6.

The neural network giving the best prediction had the following structure.

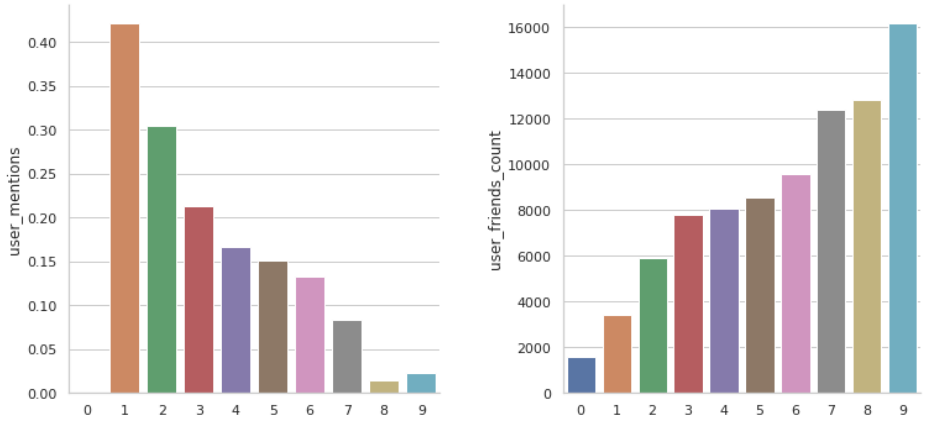
Layer	In	Out	Activation function
1	119	70	relu
2	70	35	relu
3	35	15	relu
4	15	5	relu
5	5	1	none

To prevent overfitting we tried to use layers of batch normalisation (that adds a little bit of noise to the data), and than we tried dropout layers. They did not change the results considerably. For the majority of models we standardised the training data and used the same scale with the test data to have meaningful results.

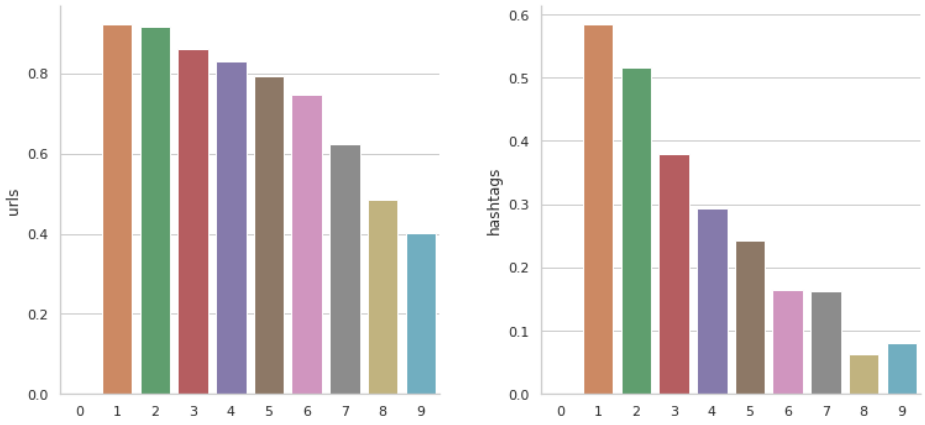
A
ANNEXE : IMAGES



(a) Mean number followers per interval (b) Mean number statuses per interval

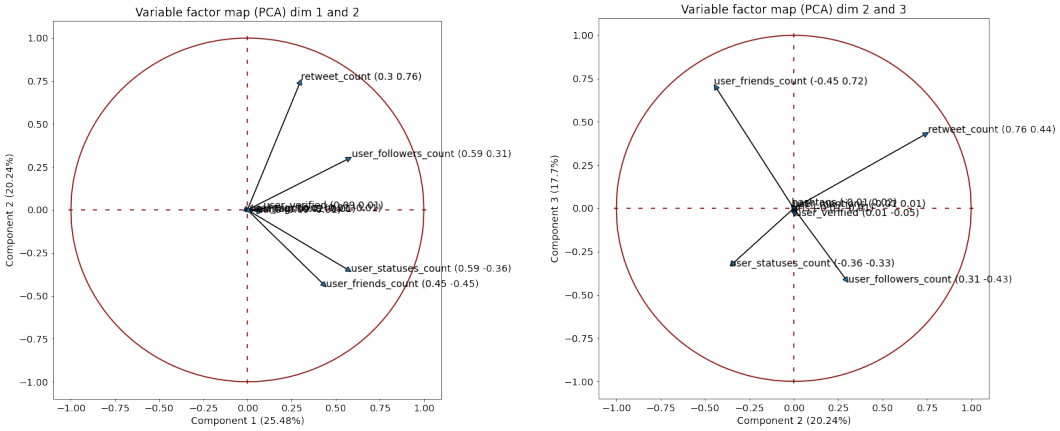


(c) Mean number mentions per interval (d) Mean number friends per interval



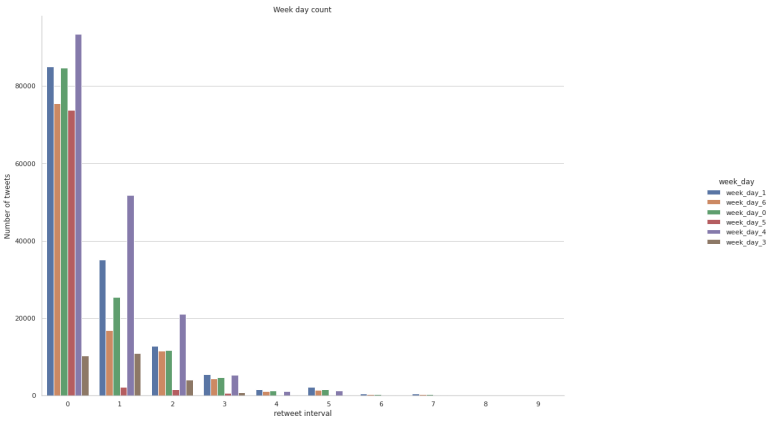
(e) Mean number urls per interval (f) Mean number hashtags per interval

Figure 2: Numerical features mean values per interval

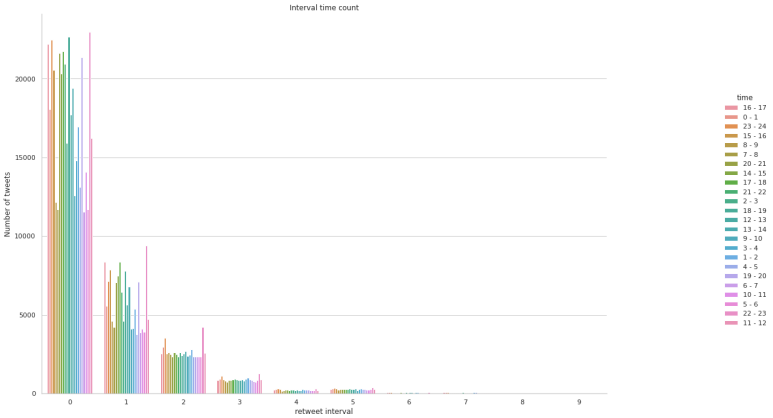


(a) PCA numerical features dim 1 and 2 (b) PCA numerical features dim 2 and 3

Figure 3: PCA of numerical features



(a) Number of tweets per day per interval



(b) Number of tweets per time per interval

Figure 4: Analysis of day and time features

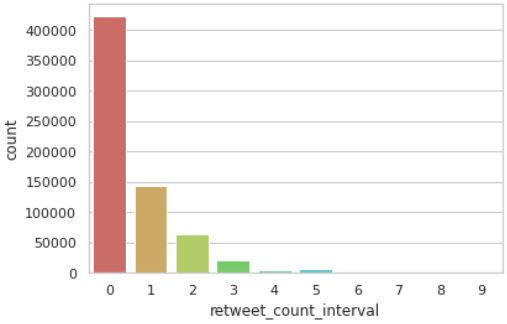


Figure 5: Number of tweets on each retweet category. Most of tweets have between 0 and 10 retweets.

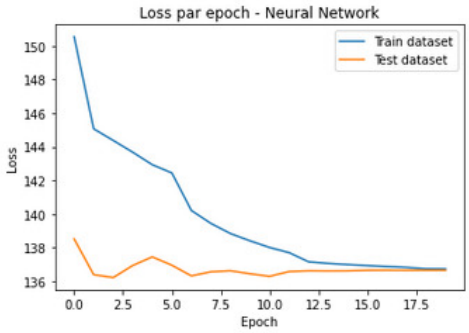


Figure 6:
Neural network loss per epoch. We can see that 10 epochs is the best to avoid overfitting and diminish our loss.

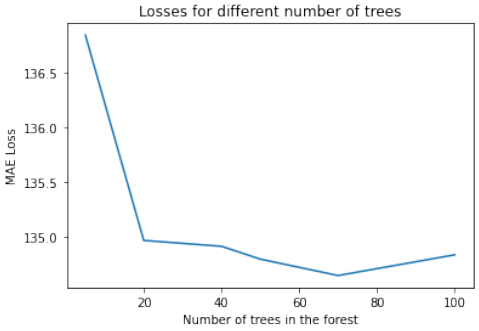
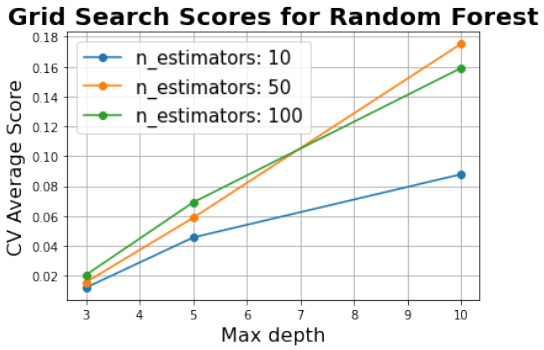
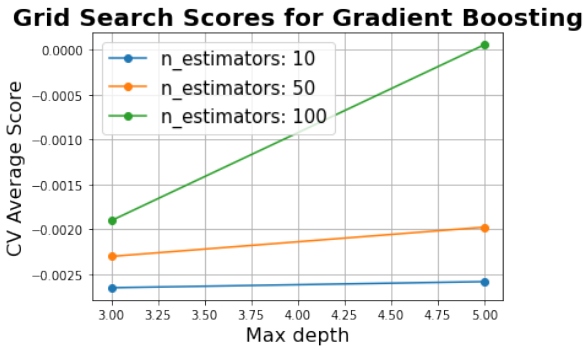


Figure 7:
Random forest classifier losses for different number of trees. We see that 70 is the best number to avoid overfitting and diminish our loss.



(a) Grid search for Random Forest Regressor. The other parameters used to produce this graph were: 0.1 for learning_rate, 3 for min_samples_split and 10 for max_features.



(b) Grid search for Gradient Boosting Regressor. The other parameters used to produce this graph were: 15 for max_features and 3 for min_samples_split.

Figure 8: Grid search for different methods [5]. We tried 36 and 48 different combinations of parameters for both Random Forest and Gradient Boosting Regressor, respectively. We show the plots of Cross-Validation Average Score against the maximum depth that could be explored.

REFERENCES

[1] Irene Paoli Imad Zaza Paolo Nesi, Gianni Pantaleo : Assessing the retweet proneness of tweets: predictive models for retweeting. *Multimedia Tools and Applications*, 77(20), 2018.

[2] Bongwon Suh, Lichan Hong, Peter Pirolli et H. Ed Chi : Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. *SocialCom/PASSAT*, pages 177–184, 2010.

[3] Francisco Herrera Vijay Chawla Nitesh V. Chawla Alberto Fernández, Salvador García : Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of Artificial Inteligence Research*, 61.

[4] M. Osborne S. Petrovic et V. Lavrenko : Rt to win! predicting message propagation in twitter. *The AAAI Press*, 2011.

[5] Will Koehrsen : Hyperparameter tuning the random forest in python. 2018.