Retweet Prediction

Orso Forghieri, Reda Belhaj-Soulami
orso.forghieri@polytechnique.edu, reda.belhaj-soulami@polytechnique.edu
XGTeam

December 2020

Introduction

In this project, we try to predict the number of retweets using data (including content, hashtags, data about the user, etc.). To do so, we used various machine learning techniques, based on decision trees. Our best models are based either on Random Forest models or on the XGBoost model. In this report, we first explain how we preprocessed the data, then we explain our choice of models. Finally we give the results of our experiments and draw the appropriate conclusions.

1 Preprocessing

1.1 Numerical data

First, we split the data into training data and test data, as well as validation data. The validation data is used for evaluation on Kaggle.

From now on, we will apply two strategies: firstly, select the relevant information from the tweets, and secondly, process it so it can be used by the decision tree. We will use decision trees to model the retweet variable like in [BT16].

id	665773
timestamp	1588324521711
retweet_count	1
user_verified	False
user_statuses_count	1807
user_followers_count	2029
user_friends_count	347
user_mentions	StanfordEMED
urls	twitter.co…
hashtags	COVID19
text	Thank you to all ...

Table 1: Example of a dataframe line

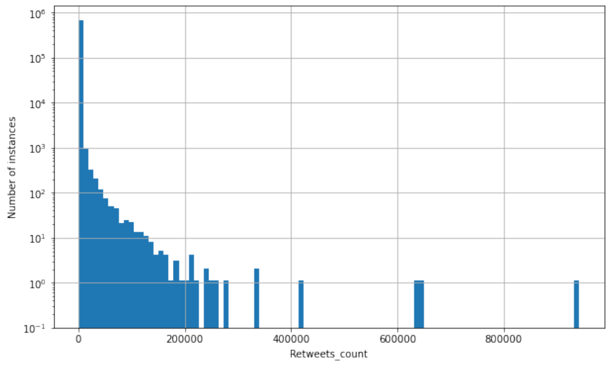

Table 2: Distribution of retweet_count

First, we chose to consider the number of retweets as a discrete variable rather than a continuous one, in order to use classification rather than regression. We considered classes of the form $a_i \leq n \leq a_{i+1}$ with $a_0 = -1$, $a_{11} = +\infty$ and $(a_i)_{1 \leq i \leq 10} = \{0, 5, 10, 30, 100, 300, 1000, 3000, 10000, 30000, 100000\}$

This choice of classes is motivated by the fact that the classes are now fairly balanced, except for the class $0 \leq n \leq 9$ which has a very large number of values. Due to this unbalanced class, a downsampling is necessary. After downsampling, the number of examples of this class is approximately equal to the number of example of the class $10 \leq n \leq 30$ (fig 1, 2). Without using this technique, the model would essentially learn to predict 0 on all instances.

User account data is easily interpretable. The number of followers already gives an important insight. We decided to combine it with the number of friends to form the ratio $\frac{\texttt{user\_followers\_count}}{\texttt{user\_friends\_count}}$ as it models popularity [BT16]. We also use the logarithms of `user_followers_count`, `user_friends_count`, and `user_statuses`, and we use the `user_verified` flag.

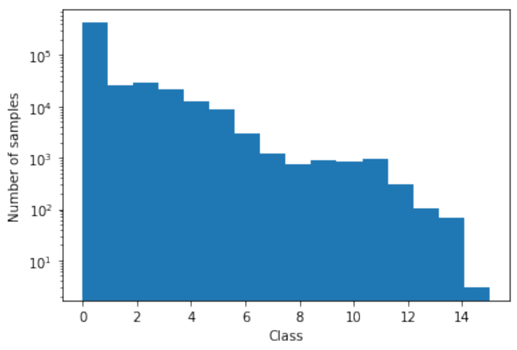

Figure 1: Distribution before downsampling

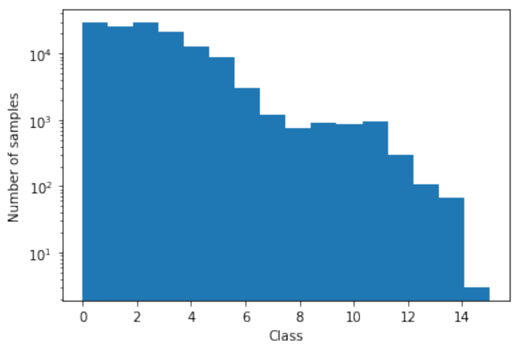

Figure 2: Distribution after downsampling

We also tried to incorporate the time of day variable or the day of the week variable. However, we observed experimentally that these two variables did not significantly improve the performance of the models.

1.2 Tweet and text processing

The text content of the tweet contains important information, as it sometimes includes mentions, an URL, hashtags or emojis.

Before attempting to use NLP techniques, we first tried simpler ways to deal with this kind of data. Using the training data, we built a list of the n_h most used hashtags, as well as a list of the most mentioned n_s websites. For a given tweet, we then used one-hot encoding : we construct a vector of size n_h (or n_s) with a 1 in the k^{th} component if hashtag (or website) k is mentioned in the tweet. However, using a vector encoding in a decision tree is very penalizing. Thus, we decided to convert these vectors with values in $\{0,1\}^{n_h}$ to binary encoded natural integers. As for the hashtags, we also build a list of hashtags related to covid: "COVID19", "coronavirus", etc. We also used one-hot encoding : we added new variables equal to 1 if the tweet contains covid-related hashtags.

A first NLP approach consisted in using word embeddings. We used Word2Vec-style models [Mik+13], implemented and pre-trained with the library Spacy [HM17]. Word2Vec models transform a word to a vector. In our case, each word is transformed to a 96-dimensional vector. We will not explain in this report how the underlying model is trained. The model aims to capture semantic similarities between words : if two words are semantically close, their vector representations should be close (in L^2 norm) too. Training is done using a very large corpus of texts in English. The word2vec model outputs 96 new vectors for each word. A first simplification is to use the "bag-of-words" paradigm : each tweet is represented as the mean of the vector representation of the words forming the tweet. A second simplification was to use PCA in order to reduce the number of NLP features from 96 down to 5 features only.

We are also processing emojis to analyse the text and leverage sentiment analysis [SA17; BHP11] and this provides a small decrease of error. This analysis consists of doing a sentiment analysis by classifying the tweet as 1 or 2 if the sentiment of the emojis in the tweet is positive, 0 if neutral, -1 or -2 if negative.

2 Models that we considered

2.1 Tree-based models

The choice of decision trees for this type of problem was motivated by the distribution of retweets count and the features [BT16; HJ20]. These types of models make it easy to manage unbalanced datasets and require less data processing. They are divided into two categories: decision trees and random forests (a set of independent decision trees which output is aggregated to form a result). This second method is much more effective than the first because it avoids some of the problems known from simple trees (overfitting, large trees, selection of predictors, etc.). However, a very efficient construction of decision trees is provided by XGBoost [CG16], which makes these trees comparable to random forests with less depth.

Thus, the depth of the tree built by XGBoost must meet two constraints: avoid overfitting (with a depth that is too large) and be expressive enough to properly analyse the data. Therefore, it is necessary to minimize the number of new columns coming from the same feature, and combine vectors (from hashtags for example) into one single column.

2.2 Logistic Regression

Similarly to previously presented models, this logistic regression model is a 10-class classifier, taking as an input the same features as the other models and classifying into one of the 10 classes.

2.3 Neural Networks

We used two types of neural networks. The first one is a regressor network, which has an output of size 1, representing the predicted number of retweets. It is trained with the MAE loss. The second one is a classifier network, that has an output of size 10. It is trained using a cross-entropy loss, better suited to classification problems. We used smaller networks with only one hidden layer (of size 10) in order to be able to train them quickly. The networks were implemented using PyTorch, and trained for 50 epochs.

3 Results

3.1 NLP features

As shown in figure 4 and 3, although the neural network classification training seems to work better when the NLP features are added (Fig. 3), it is not the case during testing. In the case of XGBoost, Figures 6 and 7 seem to show the same idea. This is probably due to the fact that the word embeddings provided by the NLP model do not generalize well for new data in this task. Our models perform generally better when the data does not include the NLP features.

3.2 Use of classification instead of regression

As for the neural networks, figure 5 shows that solving the regression problem seems to be better, as it achieves better results at test time. However, we have found the opposite result with XGBoost, as our best model is a classification model.

3.3 Choice of tree model

By plotting XGBoost's test loss w.r.t the max depth parameter (fig 7), we see that the optimal depth parameter is 5 as it does well on the test set. Using a larger value would result in overfitting. Doing the same with Random Forests 9 gives a max depth of 17.

These two models achieve similar performance on the test set. However, the random forest model is a rather large model, so we chose to prefer the XGBoost model.

3.4 Our best model

Finally, table 3 shows the final comparison between our models. It shows that our best model is XGBoost, when trained on data that does not uses NLP features.

Conclusion

Looking at the number of retweets, we found that the distribution was largely unbalanced, and that we had few examples of tweets with a large number of retweets. We then had to set up a model that was able to model the retweet_count value and taking into account a random aspect of the retweet number, something that a decision tree seems to do well. It was therefore necessary to integrate the popularity of the tweet (content), the number of people likely to see it (number of followers) as well as the people to whom it is addressed (mention of people, hashtags). We also considered other, more traditional models based on logistic regression or neural networks which did not exceed the performance of the decision tree. Finally, we concluded that the most difficult task lies in text analysis and language processing. Indeed, after considering the numerical data, interpreting the text must be the most explanatory data of the number of retweets. However, the use of sentiment analysis, top-word vectorizer or the use of Word2Vec did not lead to a significant improvement in the result. Finally, more work on language processing would allow using the semantic sense of the tweet to better predict the number of retweet.

4 Figures, Tables

	Training	Test
Logistic Regression	148,87	147,87
NN classifier - using text		145,31
NN classifier - no text		145,05
NN regressor - using text	144,54	143,99
XGBoost - using text	128,45	140,02
XGBoost - no text	135,73	139,23

Table 3: Final comparison results (MAE)

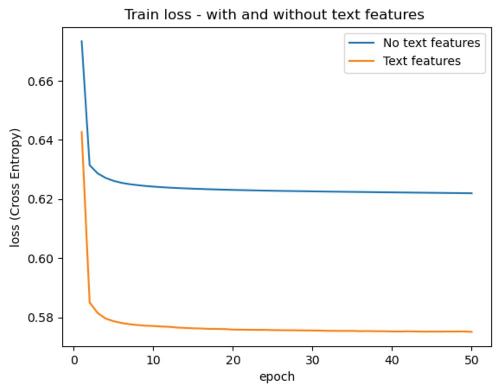

Figure 3: Neural Network Classifier training loss (text and no text)

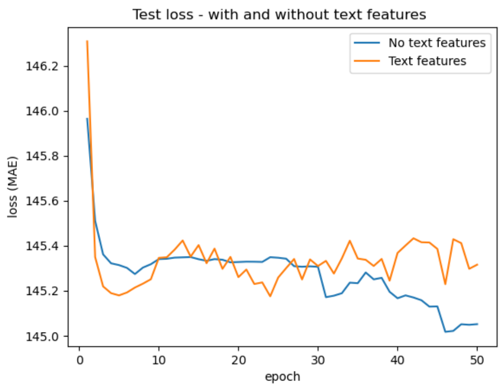

Figure 4: Neural Network Classifier test loss (text and no text)

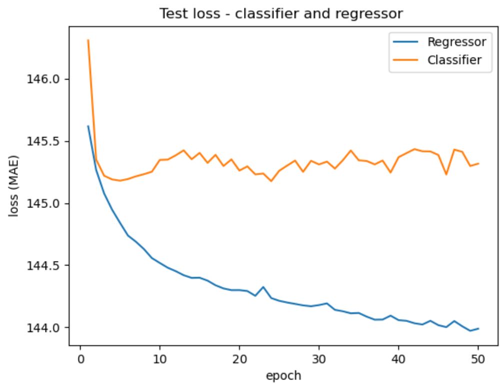

Figure 5: Neural Network test loss : classification and regression

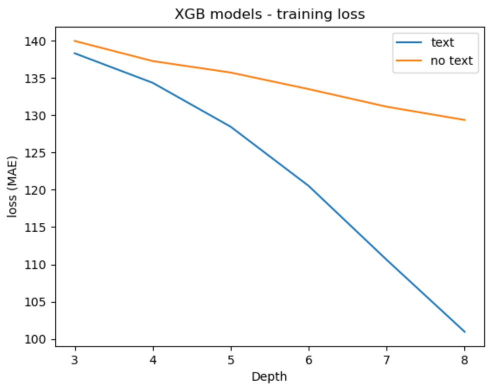

Figure 6: XGBoost training loss (text and no text)

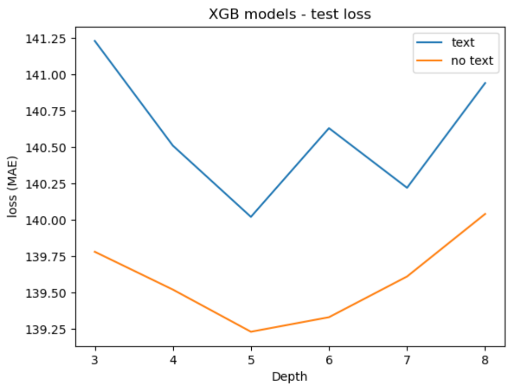

Figure 7: XGBoost test loss (text and no text)

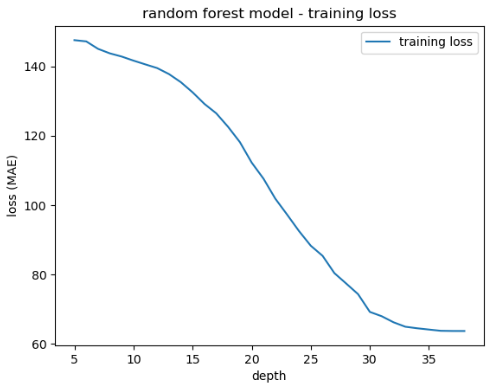

Figure 8: Random forest training loss

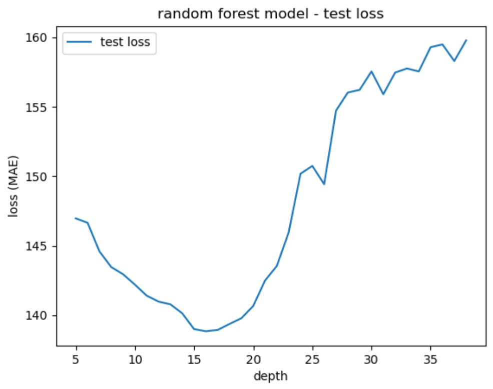

Figure 9: Random Forest test loss

References

[BHP11] Albert Bifet, Geoffrey Holmes, and Bernhard Pfahringer. “Moa-tweetreader: real-time analysis in twitter streaming data”. In: *International conference on discovery science*. Springer. 2011, pages 46–60.

[BT16] Hendra Bunyamin and Tomas Tunys. “A comparison of retweet prediction approaches: the superiority of Random Forest learning method”. In: *Telkonika (Telecommun Comput Electron Control)* 14.3 (2016), pages 1052–1058.

[CG16] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pages 785–794.

[HJ20] Daniel Hain and Roman Jurowetzki. “Introduction to Rare-Event Predictive Modeling for Inferential Statisticians–A Hands-On Application in the Prediction of Breakthrough Patents”. In: *arXiv preprint arXiv:2003.13441* (2020).

[HM17] Matthew Honnibal and Ines Montani. “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing”. To appear. 2017.

[Mik+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems*. Edited by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Volume 26. Curran Associates, Inc., 2013, pages 3111–3119. URL: `https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf`.

[SA17] Mohammed Shiha and Serkan Ayvaz. “The effects of emoji in sentiment analysis”. In: *Int. J. Comput. Electr. Eng.(IJCEE.)* 9.1 (2017), pages 360–369.