Multi-Objective Blended Ensemble For Highly Imbalanced Sequence Aware Tweet Engagement Prediction

NICOLÒ FELICIONI, Politecnico di Milano, Italy
ANDREA DONATI, Politecnico di Milano, Italy
LUCA CONTERIO, Politecnico di Milano, Italy
LUCA BARTOCCIONI, Politecnico di Milano, Italy
DAVIDE YI XIAN HU, Politecnico di Milano, Italy
CESARE BERNARDIS, Politecnico di Milano, Italy
MAURIZIO FERRARI DACREMA, Politecnico di Milano, Italy

In this paper we provide a description of the methods we used as team BanaNeverAlone for the ACM RecSys Challenge 2020, organized by Twitter. The challenge addresses the problem of user engagement prediction: the goal is to predict the probability of a user engagement (Like, Reply, Retweet or Retweet with comment), based on a series of past interactions on the Twitter platform. Our proposed solution relies on several features that we extracted from the original dataset, as well as on consolidated models, such as gradient boosting for decision trees and neural networks. The ensemble model, built using blending, and a multi-objective optimization allowed our team to rank in position 4.

CCS Concepts: • **Information systems → Recommender systems**; • **Computing methodologies → Classification and regression trees**; **Neural networks**.

Additional Key Words and Phrases: ACM RecSys Challenge 2020, Recommender Systems, Gradient Boosting for Decision Trees, Neural Networks, Blending

ACM Reference Format:
Nicolò Felicioni, Andrea Donati, Luca Conterio, Luca Bartoccioni, Davide Yi Xian Hu, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2018. Multi-Objective Blended Ensemble For Highly Imbalanced Sequence Aware Tweet Engagement Prediction. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The rise to prominence of web services in the last 20 years (e.g., social networks, e-commerce, news services) has been accompanied by the widespread use of recommender systems to help users navigate the new wealth of options at their disposal. The ACM RecSys Challenge 2020 [2][1] organized by Twitter, focuses on the development of a system able to predict, given a tweet, the likelihood a user will perform a certain action among: Like, Reply, Retweet, and Retweet with comment. The Challenge also takes into account user privacy and requires participants to use frequently changing

[1]http://www.recsyschallenge.com/2020/

training data in which certain tweets are removed. Our team proposes a hybrid solution with a different model for each action type, tuned via multi-objective optimization.

The paper is structured as follows. In Section 2 we provide an overview of the problem analyzing the composition of the dataset and the metrics used in the Challenge. In Section 3 we describe how we split the dataset in order to exploit the temporal information of the samples. In Section 4 we list and describe the models we used as well as the ensembling technique. In Section 5 we describe the feature engineering step and list the most significant features we used. In Section 6 we discuss the experimental choices we made along with the obtained results. Lastly, in Section 7 we draw the conclusions. We publicly release the source code of our final model as well as the relevant documentation[2].

2 PROBLEM FORMULATION

The goal of the ACM RecSys Challenge 2020 is to predict the probability that a user will engage with a given tweet. In particular, four different user actions are possible: Like, Reply, Retweet, and Retweet with comment (we will refer to this class as Comment). Thus, given a $\langle user\_ID, tweet\_ID \rangle$ pair, four different predictions (i.e., likelihoods) should be provided, one for each engagement class.

Twitter provided the participants with a dataset of 160 million tweets and the related user engagements. The provided training data was sampled over one week, while the test data was sampled from the following week. Each sample of the dataset represents a possible engagement between a user and a tweet, which may or may not have occurred. Each possible engagement is associated with several features, among which the user's follower and following count, the tweet text (represented as tokens from the BERT dictionary [5]), the tweet creation timestamp, and a list of links present in the tweet. If the engagement occurred, its timestamp is also available. For the test tweets, the timestamp is not available. In the remainder of the paper we will use *positive samples* to indicate engagements that occurred and *negative samples* to indicate those that did not occur. In order to ensure compliance with privacy laws, samples that were removed from the platform also had to be removed from the dataset. Due to this, several versions of the dataset were released during the challenge. The final training dataset contains about 121 million samples and the final test contains about 12 million samples. The engagement classes in the provided dataset exhibit a strong imbalance. In particular, the Like engagement is balanced while the Reply and Comment classes are strongly unbalanced towards the negative samples. The quota of positive samples over the total number of samples for each engagement class is: 0.434 for the Like class, 0.109 for the Retweet class, 0.025 for the Reply class, 0.007 for the Comment class. Furthermore, 24% of the samples have a *cold user* as engager, i.e., a user who never appeared in the training data as engager both in positive and negative samples.

Two metrics are used to evaluate the predictions: the Relative Cross-Entropy (RCE), which takes into account the relative improvement from a naive prediction baseline, and the Area Under the Precision-Recall Curve (PRAUC). The two metrics are computed for each engagement class separately. For each metric, the average on the four classes is computed and the submissions ranked. The final score on the leaderboard is calculated as the summation of the rank obtained from the two metrics[3]. Each team can choose two different sets of predictions and the best one will be automatically chosen for the final evaluation.

3 DATA SPLIT

In order to have a local validation data, we further split the provided training data. Reproducing the same splitting protocol used to generate the leaderboard test data, i.e., selecting only the last days of the training data, is not an

[2] https://github.com/MaurizioFD/recsys-challenge-2020-twitter
[3] https://recsys-twitter.com/faq/

advisable strategy. Since the engagements are distributed over a single week, removing the latest engagements from the training data would not properly reproduce the temporal distribution of the test data and would make impossible for the model to use some of the features, such as the day of the week in which a tweet is created. In order to overcome this, we adopted a specifically crafted protocol. The provided training data is split via a random holdout, 88% for the local training data and 12% for the validation. The timestamp of the validation samples is then shifted by one week in the future, to simulate a week long behavior. In the remainder of the paper we will refer to the local training data as simply training data. Moreover, as stated in Section 2, the test data contains a sizable number of cold users, we thus kept this percentage consistent in our validation data by removing some users and their engagements as engagers from the training data, putting them in the validation data.

4 MODELS

The provided dataset, as well as the additional features we developed (see Section 5), mostly consist of well-structured tabular data. For this reason, our models rely strongly on Gradient Boosting for decision trees (GBDT). The only exception are the BERT tokens used to represent the tweet text, which are modeled with a neural network for natural language processing. Finally, all models are combined in a blending ensemble.

4.1 Gradient Boosting for Decision Trees

We used two variations of Gradient Boosting for decision trees:

- **XGBoost:** [3] well known tool which allows to build models using sparse datasets.
- **LightGBM:** [7] which allows for substantially faster training time compared to XGBoost.

4.2 Neural Networks

A significant information source is the tweets' text, provided in the form of token IDs from the BERT tokenizer. BERT is a transformer model introduced by Google [5], particularly effective for natural language processing. We fine-tuned a pre-trained DistilBERT model, a variant of BERT that gives a significant speedup in computation time [8]. The fine-tuning process is necessary for the model to understand the semantics of hashtags, mentions and other domain-specific data that a pre-trained generic version could not capture. In particular, we added two dense layers on top of the pre-trained model (with 128 and 64 neurons respectively). Each of the two hidden layers is followed by a dropout layer, with a high dropout rate (0.5), to counteract overfitting in such a big model. The first dense layer takes as input also a set of other features (50 for Comment and 74 for the other three classes), which are normalized with a Yeo-Johnson transformation to ensure they are normally distributed [9]. The final model architecture is shown in Fig. 1.

4.3 Ensemble

We combined the previously described *base models*: LightGBM, XGBoost and neural networks, with the *blending ensemble* technique, using a new instance of LightGBM. Each base model was trained on the training data and produced predictions on both the validation and test data (Stage 1 of Fig. 2). We enhanced validation and test data with the predictions given by the base models, treating them as new features. We finally trained the ensemble model on the validation data and generated the predictions on the test data (Stage 2 of Fig. 2) in order to produce the submission.

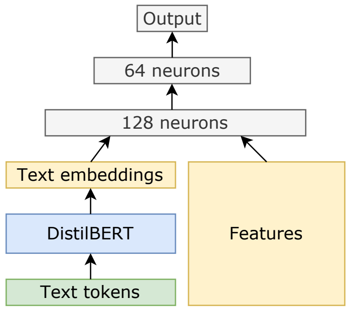

Fig. 1. Structure of the neural network model, using both text embeddings and a set of other features.

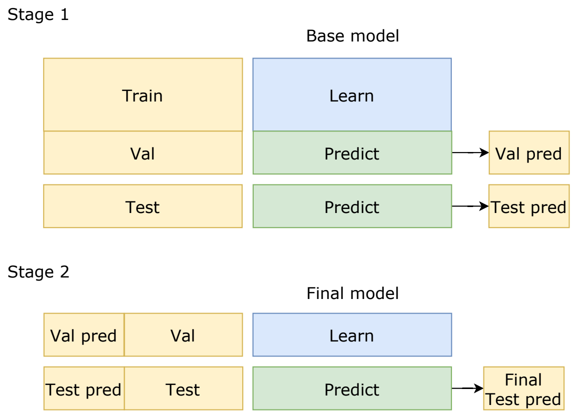

Fig. 2. Stages of the blending ensemble.

5 FEATURES

Feature engineering proved to be very important to achieve competitive results. In addition to the features present in the provided dataset, we developed around 300 new features, half of which were excluded from our final solution because they had limited impact. In the following sections, we describe those we consider to be the most significant.

5.1 User Behavior Modeling

The features belonging to this category aim to represent the user's past behavior. Each of them was built using two different techniques:

Timestamp-aware: Given a user-tweet engagement, the features are computed using all the engagements that occurred previously, according to the tweet creation timestamp. Note that a user is associated to different feature values as time progresses, modeling the sequential engagements pattern.

Cumulative: The features are not sensitive to the sequential patterns and are computed using the whole user history. The training data is split randomly in 20 folds. For each fold, the user behavior features are computed using the engagement data in the *other* 19 folds. We could observe that this approach improved the model robustness.

The user behavior features that we developed can be split in three main subcategories:

Number of Active and Passive Engagements. This group of features is computed for each engagement class and is associated to a specific user. For each sample, there are both the features referred to the engager (active) and the creator (passive). Given a sample, its timestamp and the user role (i.e., creator, engager), the features count the number of engagements of each type in which the user had the same role. The features associated to the role engager will model how likely is the user to interact with tweets, while those for the role creator will represent how popular the user is. The division in engagement types further allows to model the user's interaction style.

Number of Engagements with Language/Hashtag. We create one feature per engagement class, which counts the number of previous engagements the user had with tweets of the same language of the current one. These features provide a way to model the languages a user speaks and therefore which are those the user may understand and engage with in the future with higher probability. A similar feature counts the number of previous engagements with a hashtag and provides a semantic indicator of the user's interests.

User Similarity. We compute a similarity between users by representing them in an undirected graph. Each edge will connect two nodes, i.e., users, if one engaged with a tweet created by the other and will have a weight equal to the number of such engagements. Four features are generated from this representation: the similarity of directly connected users (i.e., 1-hop) and connected by a path of length exactly two (i.e., 2-hop). Both are represented as a binary value, if the users are connected or not, and as the summation of the weights of the edges connecting the two users.

5.2 Tweet Text Features

The text of the tweet was modeled in different ways and used both in the neural network models and in GBDT. For the neural network model we used the text embeddings produced by DistilBERT, as described in Section 4.2. In order to use the text in GBDT models we created some specific features computed from the encoded tweet tokens. The purpose of these features is to summarize some salient text related information in a format that can be used by the GBDT models.

Unique Words Frequency. This feature represents how much a user tends to use repeated words from past tweets when writing a new one. It is computed by concatenating all text the user wrote and calculating the ratio between the number of unique tokens and the total number of tokens. It can be useful in identifying bots and recurrent patterns.

Tweet Topic. We identified some popular topics (COVID-19, K-Pop, sports, etc.) and manually associated each to a list of the most used words. Then, for each tweet we counted the number of words of a certain topic it contains.

6 EXPERIMENTAL EVALUATION

6.1 Hyperparameter Tuning

In order to tune the hyperparameters of the models we performed a Bayesian Optimization, which was proven to be an effective strategy [1, 4, 6], using the scikit-learn library[4].

Since the submissions to the Challenge are evaluated with two metrics we developed a custom objective function to be used during the hyperparameter tuning. This choice was further motivated as a way to mitigate two other issues. First, the high class unbalance for engagement type Reply and Comment tended to steer the model to simply predict an engagement probability of zero. Second, an algorithm providing a constant prediction, regardless of its value, is able to achieve an extremely high PRAUC (higher than 0.5) in those classes, but a very low RCE. The effect is especially marked for the Comment class where a constant prediction model achieves a PRAUC of 0.5037, more than 6 times the PRAUC of the winning approaches of 0.0796. In order to address those issues we combined the two metrics and penalized models with high PRAUC but no discriminative power. The objective function is defined as follows:

$$obj(PRAUC, RCE) = \begin{cases} RCE \cdot PRAUC, & \text{if } RCE \geq 0 \\ RCE/PRAUC, & \text{otherwise} \end{cases} \quad (1)$$

This objective function, combined with the stochastic nature of the Bayesian Optimization allowed us to identify two optimal solutions for the Comment class, one of which has a high PRAUC (Comment-P) while the other a high RCE (Comment-R). Fig. 3 visualizes their differences on the prediction distribution. The two Comment models have the

[4] https://scikit-optimize.github.io/stable/auto_examples/bayesian-optimization.html

same mean 0.0076, but a very different standard deviation: 0.0088 for Comment-P and 0.0166 for Comment-R. In this case a higher PRAUC is associated to a greatly reduced variance.

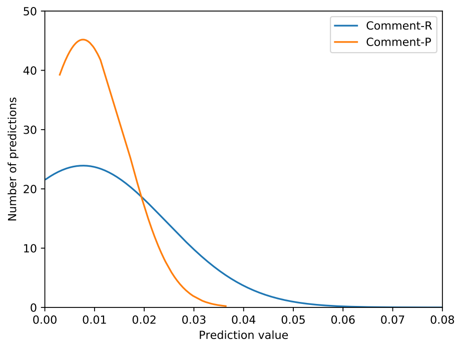

Fig. 3. Distribution of the predictions for the Comment class generated by the two models, one with high PRAUC and the other with high RCE.

PRAUC				
Model	**Like**	**Retweet**	**Reply**	**Comment**
XGBoost	0.7992	0.5391	0.1704	0.0615
LightGBM	0.8201	0.5469	0.1714	0.0650
NN-S1	0.7775	0.4934	0.1428	0.0336
NN-S2	0.7755	0.4889	0.1412	0.0309
NN 4-labels	0.7629	0.4734	0.1185	0.0264

RCE				
Model	**Like**	**Retweet**	**Reply**	**Comment**
XGBoost	28.98	29.75	18.55	11.96
LightGBM	32.86	30.48	18.76	12.39
NN-S1	25.62	25.97	14.40	6.54
NN-S2	25.17	25.68	14.78	3.95
NN 4-labels	23.02	24.45	12.55	5.23

Table 1. Performance of the models on the validation data. NN-S1 and NN-S2 refer to the two instances of the neural network on different subsamples of the training data.

6.2 Performance Analysis

Gradient Boosting for Decision Trees. Throughout the various experiments and hyperparameter settings we tried, LightGBM always provided better predictions than XGBoost. Table 1 shows the performance of the XGBoost and LightGBM models that had the best results on the validation data. We also noticed that the RCE of the GBDT models for the Like class on the test data was about 50% lower than the validation one.

Neural Networks. We explored different variations on how to train the neural network model presented in Section 4.2. An initial solution was to train a network with 4 output neurons, one for each engagement class. This approach allowed us to leverage correlations between labels and save computation time, with respect to training a different model for every single class. However, we observed that further improvements could be obtained by using a mixed strategy. In our final solution we trained a network with two sigmoid outputs for the Like and the Retweet engagements, while we used two separate, single output networks, for the Reply and Comment engagements. The main drawback of the neural network models was that training and computing the predictions were very time consuming. In order to mitigate the computational cost, we independently trained two instances of each network on two small random subsamples of the training data (S1 and S2), each of which was composed by four million samples. This allowed us to parallelize the training process, ensuring that a higher number of training samples could be explored in a smaller amount of time. Table 1 compares the local performance of a neural model with 4 output neurons against the mixed strategy we adopted.

Ensemble. Initially, each ensemble model contained the base models scores for each one of the four labels. Due to the previously discussed discrepancy between validation and test performance of the GBDT models on the Like class, we decided to remove their scores from the ensemble. For the same reason, in the final ensemble for the Like class we opted not to use any score we inferred using the GBDT models. We used, instead, only the neural network scores along with the features described in Section 5. A visualization of the scores and features used in the ensemble for each class can be found in Fig. 4.

We had the possibility to submit two different solutions to be evaluated on the final leaderboard. The two submissions we made included the exact same predictions for the Like, Retweet and Reply classes, while they differed in the Comment

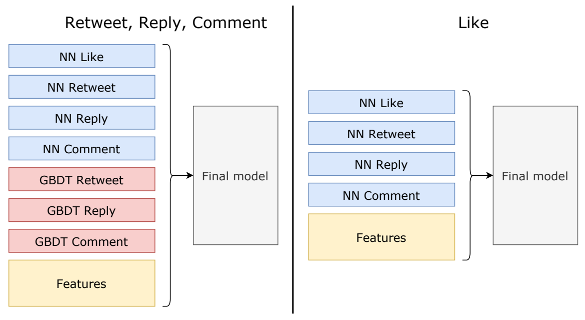

Fig. 4. Final ensemble model for the Like class on the right and for the other three labels on the left. In blue the neural networks scores, in red the GBDT model ones and in yellow the features.

Final Model	**Validation**		**Leaderboard**	
	PRAUC	RCE	PRAUC	RCE
Like	0.8134	31.56	0.7531	21.20
Reply	0.2029	21.29	0.1850	18.71
Retweet	0.5631	32.37	0.5042	27.21
Comment-P	0.1264	9.41	0.1237	8.34
Comment-R	0.0688	13.77	*	*

Table 2. Performance of the final ensemble models on local validation data and on the leaderboard. Since the submission containing Comment-R was not the best one, its PRAUC and RCE values are missing from the final leaderboard.

ones. One submission included the predictions given by the Comment-P ensemble model and the other included those given by the Comment-R ensemble model. The performance on the validation data and the leaderboard test of each final ensemble model is reported in Table 2.

Feature Importance. In order to measure the contribution of the features we used in our final models, we computed the *permutation importance*[5] of each feature. Its value is the ΔRCE between the correct model score and that of a model trained with that feature randomly-shuffled, keeping the others unchanged. The process was repeated 5 times per feature. Higher values indicate more important features. By analyzing the results[6] we noticed that the neural networks scores are highly relevant, which may be because they are the only model that is able to process the text of the tweets. In the case of Comment-P the model relies almost exclusively on the base models scores. Regarding the features we described in Section 5, a feature that was useful for all the final ensemble models is the number of previous passive negative engagements (Section 5.1), reaching an importance of 16.148 in the Like model.

7 CONCLUSION

The goal of the ACM RecSys Challenge 2020 was to predict the probability of an user's engagement to a given tweet. We combined the information captured directly by the features extracted from the dataset with the predictions given by different GBDT and neural network models, trained with features that allowed us to model aspects such as users' behavior or their similarity. We also defined a dedicated objective function to tune our models that allowed us to achieve a reasonable trade-off between a high PRAUC score while maintaining a model with discriminative power in the Comment label predictions. The blending ensemble technique we used provided an improvement with respect to the performance of the single models in both local and leaderboard evaluations, in such a way that we were able to reach the 4th position in the final leaderboard.

8 ACKNOWLEDGEMENT

We would like to thank Prof. Paolo Cremonesi for his support.

[5]https://eli5.readthedocs.io/en/latest/blackbox/permutation_importance.html
[6]The results are available in the online material we release in our Github repository: https://github.com/MaurizioFD/recsys-challenge-2020-twitter

REFERENCES

[1] Sebastiano Antenucci, Simone Boglio, Emanuele Chioso, Ervin Dervishaj, Shuwen Kang, Tommaso Scarlatti, and Maurizio Ferrari Dacrema. 2018. Artist-driven layering and user's behaviour impact on recommendations in a playlist continuation scenario. In *Proceedings of the ACM Recommender Systems Challenge, RecSys Challenge 2018, Vancouver, BC, Canada, October 2, 2018*. ACM, 4:1–4:6. https://doi.org/10.1145/3267471.3267475

[2] Luca Belli, Sofia Ira Ktena, Alykhan Tejani, Alexandre Lung-Yut-Fon, Frank Portman, Xiao Zhu, Yuanpu Xie, Akshay Gupta, Michael M. Bronstein, Amra Delic, Gabriele Sottocornola, Walter Anelli, Nazareno Andrade, Jessie Smith, and Wenzhe Shi. 2020. Privacy-Preserving Recommender Systems Challenge on Twitter's Home Timeline. *CoRR* abs/2004.13715 (2020). arXiv:2004.13715 https://arxiv.org/abs/2004.13715

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *CoRR* abs/1603.02754 (2016). arXiv:1603.02754 http://arxiv.org/abs/1603.02754

[4] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk (Eds.). ACM, 101–109. https://doi.org/10.1145/3298689.3347058

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[6] Edoardo D'Amico, Giovanni Gabbolini, Daniele Montesi, Matteo Moreschini, Federico Parroni, Federico Piccinini, Alberto Rossettini, Alessio Russo Introito, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2019. Leveraging Laziness, Browsing-Pattern Aware Stacked Models for Sequential Accommodation Learning to Rank. In *Proceedings of the Workshop on ACM Recommender Systems Challenge* (Copenhagen, Denmark) *(RecSys Challenge '19)*. Association for Computing Machinery, New York, NY, USA, Article 7, 5 pages. https://doi.org/10.1145/3359555.3359563

[7] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3146–3154. http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree

[8] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[9] In-Kwon Yeo and Richard Johnson. 2000. A new family of power transformations to improve normality or symmetry. *Biometrika* 87 (12 2000). https://doi.org/10.1093/biomet/87.4.954

A APPENDIX: FEATURE IMPORTANCE

Table 3 reports the results of the permutation importance. The table is composed by 6 columns: the first one is the name of the feature and the other 5 report the ΔRCE obtained by the final ensemble models when the feature is permutated.

We reported the main feature categories, that are:

- Basic features, which are the features provided by the organizers of the Challenge.
- Content based features, which are the features we extracted from the dataset.
- Number of engagements received by a creator, which represents a user behavior feature.
- Neural network predictions, which are the predictions generated by the Neural network models.
- LightGBM predictions, which are the predictions generated by the LightGBM models.
- XGBoost predictions, which are the predictions generated by the XGboost models.

Feature name	Retweet	Reply	Comment R	Comment P	Like
Basic features					
Engager following count	0.057	0.046	0.052	0.000	1.681
Engager follower count	0.091	0.061	0.066	0.000	0.827
Creator following count	0.027	0.041	0.02	0.000	1.148
Creator follower count	0.031	0.163	0.025	0.000	1.182
Creator follows engager	0.019	0.093	0.002	0.000	1.568
Content based features					
Text unique tokens	0.032	0.032	0.023	0.000	0.179
Text token counts	0.071	0.049	0.015	0.000	0.168
Text sport words count	0.006	0.003	0.005	0.000	0.017
Text kpop words count	0.003	0.005	0.000	0.000	0.010
Text covid words count	0.005	0.000	0.006	0.000	0.001
Number of engagements received by a creator					
Retweet engagements	0.078	0.021	0.032	0.000	0.800
Reply engagements	0.020	0.030	0.010	0.000	0.533
Positive engagements	0.018	0.024	0.006	0.000	0.812
Negative engagements	0.084	0.044	0.034	0.000	16.148
Like engagements	0.034	0.026	0.009	0.000	10.097
Comment engagements	0.019	0.005	0.031	0.000	0.035
Neural network predictions					
NN comment S1	0.118	0.032	0.312	0.048	0.048
NN comment S2	0.109	0.050	0.985	1.687	0.178
NN reply S1	0.040	0.980	0.138	0.023	0.088
NN reply S2	0.045	4.944	0.017	0.000	0.162
NN retweet S1	0.635	0.129	0.061	0.288	0.096
NN retweet S2	9.247	0.046	0.730	0.006	0.245
NN like S1	0.088	0.123	0.063	0.000	1.353
NN like S2	0.158	0.406	0.050	0.000	12.331
LightGBM predictions					
LGBM comment	0.000	0.003	0.811	5.625	-
LGBM reply	0.012	2.795	0.010	0.000	-
LGBM retweet	12.677	0.671	1.095	0.001	-
XGBoost predictions					
XGB comment	0.010	0.016	0.042	0.484	-
XGB reply	0.006	0.113	0.005	0.000	-
XGB retweet	0.221	0.181	0.647	0.000	-

Table 3. Permutation importance of the most relevant features in the ensemble model.