A classification system to detect questionable information

Emanuel Tomé
DCC-U.Porto
Porto, Portugal
up200702634@edu.fc.up.pt

Hélder Vieira
DCC-U.Porto
Porto, Portugal
up201503395@edu.fc.up.pt

Klismam Pereira
DCC-U.Porto
Porto, Portugal
up201900035@edu.fc.up.pt

Vânia Guimarães
DCC-U.Porto
Porto, Portugal
up200505287@edu.fc.up.pt

***Abstract*—Misinformation has proliferated rapidly on the internet. The consequences have implications in many areas, including politics. Finding instruments to detect disinformation is urgent. In this work, using tweets retrieved in the 2020 U.S. Presidential election, we developed a machine learning model to detect disinformation and help users evaluate the credibility of tweets.**

***Index Terms*—misinformation, fake news, twitter, social media, natural language processing, machine learning**

I. Introduction and objectives

All dimensions of a country's democracy are effective when its citizens act with knowledge of the facts. In the last U.S. Presidential election, there has been a lot of debate about the influence of misinformation in the vote of citizens, in particular that disseminated by social networks such as twitter. To mitigate information manipulation, politicians have pressured social media platforms to take steps to control the information that is released. In this sense, our project aims to create a tweet classification model that detects if a new tweet is disseminating questionable information, or not. With that purpose, we will study a dataset comprised of 18.000 tweets about the last U.S. Presidential election, which were retrieved before the election was held.

II. Exploratory Data Analysis and pre-processing

The dataset contains 17950 entries and 11 columns/features, which are:

- `id` - tweet id.
- `questionable_domain` - Binary feature, target feature.
- `user_friends_count` - Number of users following the account.
- `user_followers_count` - Number of followers of an account.
- `user_favourites_count` - Number of times a tweet has been liked by Twitter users.
- `user_verified` - Binary variable, true if the user is verified.
- `description` - Text feature, text content of the tweet publication.
- `title` - Text feature, title of the tweet.
- `favourite_count` - Number of times a tweet has been liked by Twitter users.
- `retweet_count` - Number of times a tweet has been retweeted.
- `contains_profanity` - Binary variable. True if contains an offensive or obscene word or phrase.

Among the 17950 entries, 3000 (16.7%) are questionable, and the remaining 14950 (83.3%) belong to the non-questionable class. This difference in proportion denotes an imbalanced dataset.

Since the feature `title` has precisely the same information or a truncated part of the feature `description`, the former was disregarded. The feature `id` was also disregarded since it is just an identifier. The feature `retweet_count` was removed because it has high correlation (Pearson correlation value of 0.82) with the feature `favourite_count`. A total of 113 entries were also removed since they had duplicated text in the `description` variable.

The cleaning and pre-processing of data is a very important step, specially when we deal with unstructured data like texts. A text filtering pipeline (as shown in Figure 1) was designed and executed over the `description` data to allow further feature extraction and the application of text mining and processing techniques. In the pre-processing stage a set of sequential transformations was applied over the corpus, namely:

- Text cleaning: removal of URLs, unicode characters (such as emoticons), digits and spaces.
- Lemmatization.
- Punctuation and stop words removal.

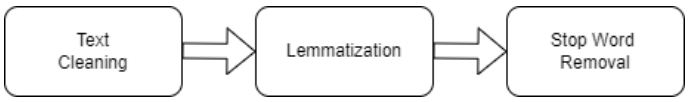
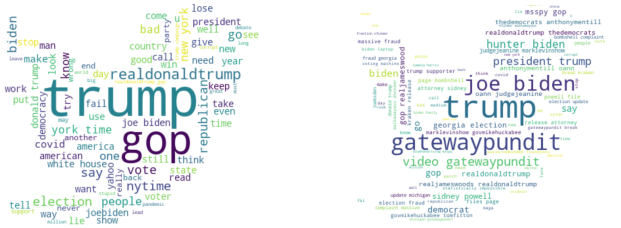


Fig. 1: Pre-processing main steps.

In the exploratory analysis step we looked for words that are more frequent in each class. As shown in figure 2, some words are frequent in both situations, but others, like "gatewaypundit", are very common in questionable tweets and infrequent in non-questionable ones.

This insight is used in the creation of boolean features for the dataset 3, and in the identification of the most frequent mentions in each domain. Among the 30 most common



(a) Unquestionable Cloud (b) Questionable Cloud

Fig. 2: Word clouds of each class.

words in the non-questionable domain we can find mentions of medias, such as @nytimes, @cnn, @foxnews, @yahoo, @nbcnews, @whashingtonpost, @googlenews, which are not included in the top 30 of questionable tweets. This seems to indicate that non-questionable tweets tend be more associated with news from reliable sources. Therefore, the average number of mentions in each tweet can be distinct for each class. The starting point was to create variables with different distributions, according to the class. The questions were focused on the characteristics that could distinguish a formal text from an informal one. The list of variables created can be found in **section IV**.

III. Methods and hyperparameters

A fraction of 80% of the data is used for training. The dataset is shuffled to reduce any dependencies between observations and stratified to maintain the proportion cases in the target variable. The performance of six methods was evaluated with regards to the classification task. The hyperparameters were set using grid search with 10-fold stratified cross-validation optimised for *f1_macro*. The methods and their respective searched hyperparameters spaces are:

- Naive Bayes (NB)
  - no search performed
- Decision Tree (DT)
  - complexity parameter α: [0.0001, 0.001, 0.01, 0.1, 1, 10]
- Random Forest (RF)
  - complexity parameter α: [0.0001, 0.001, 0.01, 0.1, 1, 10]
  - number of estimators: [50, 100, 150, 200, 250, 300]
- Support Vector Machines (SVC)
  - kernel:
    - radial basis function
  - regularization parameter **C**: [0.1, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]
- Multi-layer Perceptrons (MLP)
  - activation function:
    - tanh: $f(x) = \tanh(x)$
    - relu: $f(x) = max(0, x)$
    - identity: $f(x) = x$
    - logistic: $f(x) = \frac{1}{1+e^{-x}}$
  - number of nodes: [10, 25, 50, 100, 150, 200, 250, 300]
  - constant learning rate: [0.001, 0.01, 0.1]
- XGBoost
  - Number of estimators: [50, 100, 150, 200, 250, 300]

The remaining hyperparameters are the default in the Scikit-learn python package [1].

Since the dataset is imbalanced, after feature engineering/extraction and before the training of the models, the train set was randomly undersampled. We also made experiments with the baseline dataset (DS1) without techniques to deal with imbalanced data, with oversampling, and undersampling combined with oversampling. The obtained metrics were similar in all experiments, therefore we kept our analysis only to the undersample approach in order to avoid computational overhead.

Finally, the model performance is evaluated in the test dataset. The metrics computed are *accuracy, f1, precision* and *recall*, and can be seen in Appendix C.

IV. Datasets and extracted features

Four datasets are created and the methods described in the previous section are applied in each of them. In a incremental fashion, the subsequent dataset has all features of the previous dataset, for this reason a continuous improvement of the obtained metrics is expected (see figure 3). The considered datasets are the following:

- DS1 (Baseline): dataset with all the numeric features given in the original dataset.
- DS2: Data set with features based on tweets' content, language and linguistic.
- DS3: Data set with Boolean features created from the most frequent words in questionable tweets.
- DS4: Word embeddings (Word2Vec).

A. Dataset 1 (DS1)

The Dataset 1 (DS1) is used as baseline, where only the non-text features given in the original dataset are considered, namely *user_friends_count*, *user_followers_count*, *user_favourites_count*, *user_verified*, *favorite_count* and *contains_profanity*. The metrics *accuracy, f1, precision, and recall* for all models in the test dataset can be seen in table III, Appendix C.

B. Dataset 2 (DS2)

In this dataset we started by extracting features from the tweet description. The extracted features are envisaged to characterise the tweet's content, language and linguistics [2]. Therefore, the first features that we extracted are simple counts of the contents of the original tweet, namely:

- url_count - Number of urls in a tweet.
- count_tokens_raw - Number of tokens in a tweet.
- hashtag_count - Number of hashtags in a tweet.
- mention_count - Number of mentions in a tweet.
- emojis_count - Number of emojis in a tweet.

Fig. 3: Considered datasets.

- num_chars_raw - Number of characters in a tweet.
- unique_word_count_raw - Number of unique words in a tweet.
- stop_word_count - Number of stop words in a tweet.
- punctuation_count - Number of punctuation characters in a tweet.
- exclamation_marks - Number of exclamation marks in a tweet.
- question_marks - Number of question marks in a tweet.
- person_count - Number of named entities after POS tagging.
- event_count - Number of event entities after POS tagging.
- norp_count - Number nationalities, religion or politic POS taggings.
- capitalized_word_count - Number of capitalized words.

We then created **ratio/compound features**, namely:

- avg_word_length_raw - Word average length in a tweet.
- ratio_capitalized_words - ratio of capitalized words.
- uppercase_ratio - ratio of uppercase characters.
- stop_word_ratio - ration of stopwords.

Further, we also created the following **binary features**:

- contains_repetition - True if tweet contains a character that is followed by at a repetition of itself.
- contains_quote - True if contains quotes.
- starts_with_mention - True if the tweet starts with a mention.
- contains_question_mark - True if contains a question mark.
- contains_exclamation_mark - True if contains a exclamation mark.
- multiple_question_mark - True if contains more than 1 question mark
- multiple_exclamation_mark - True if contains more than 1 exclamation mark
- contains_trust_worthy_source - True if contains at least one of the mentions in the top 30 of more common words in the unquestionable tweets (@nytimes, @cnn, @foxnews, @yahoo, @nbcnews, @whashington-post, @googlenews)

As stated before, these features were created in order to characterise the tweet's content, language and linguistics. Although they do not allow a perfect separation between the classes, they provide some amount of distinction. Therefore, it is expected that their combination leads to a good distinction between the target classes.

Finally, we also created two **Sentiment features**:

- polarity
- subjectivity

Polarity feature indicates the sentiment and its intensity behind each tweet. VADER (Valence Aware Dictionary for Sentiment Reasoning) algorithm from the NLTK python package [3] relies on a dictionary of emotions to decide what the sentiment category of each word, and consequently by summing up the sentiment scores, it assigns a score to the entire sentence. The emotion intensity lies between -1 and 1. The highest value means that the sentence is mostly positive. Negative phrases receive the lower scores. The neutral texts have scores around zero. Each tweet's feeling intensity may influence its attractiveness and create distinction between the target classes. Evaluating our data we can see that there is an overall negative sentiment in the tweets, with greater negativity in non-questionable tweets (-0.319 against -0.258).

The other feature was based on the subjectivity of the text. Factual texts are empirically associated with the sphere of veracity. Nonetheless, an opinion text does not necessarily enter the non-questionable domain. TextBlob, a python library for processing textual data [4], returns a value that lies within [0,1], in terms of subjectivity, which means that the highest value refers to personal opinions and judgements. Analysing our data set, the average scores were similar between each class (0.495 for non-questionable domain and 0.478 for questionable).

Finally, it should be noted that in the end, the features *num_chars_raw*, *unique_word_count_raw*, *ratio_capitalized_words*, *stop_word_ratio*, *contains_question_mark* were removed from the dataset because they have a Pearson correlation coefficient higher than 0.8 with other features. The models' performance is evaluated against the test dataset and the metrics *accuracy, f1, precision, and recall* can be seen in table IV, Appendix C. The obtained metrics for a sequential addition of the four groups of features considered are presented in Appendix A. From that analysis, it is concluded that the highest increase in the performance metrics occurs when the first group of features is added (count features).

C. Dataset 3 (DS3)

The objective of this experiment is to use the most frequently occurring words in questionable tweets as boolean features, complementing the ones used to train the previous models. The intuition is that these features may help algorithms train better models with improved generalisation ability, resulting in increased classification performance. It is assumed that reoccurring words more strongly associated with questionable tweets result in good explanatory features. Moreover, filters based on heuristics were applied to obtain a concise list of frequent words to be used as features. The feature engineering pipeline is described in the following paragraphs.

The support of words in questionable tweets is computed in the training dataset. Words with support lower than the third quantile are filtered out to avoid computational overhead. The fraction of occurrences of words in questionable tweets divided by their total number of occurrences is computed - this proportion represents how strongly associated with questionable tweets a word is, i.e, it is the confidence that the presence of a word leads to the occurrence of a questionable tweet.

Both word support and confidence are essential to select features. If a word has a high support but has low confidence, it is likely not a good predictor for questionable tweets as it also has a high support in non-questionable tweets. The same can be said about words with high confidence but low support, which might lead to features that do not generalise well given the low number of occurrences. Following this argument, words with confidence and support lower than the 0.9 quantiles are filtered out. Finally, some tweets contain the same set of mentions - when a user tags other users in his text using the character @. Mention words that are equally repeated together across tweets generate duplicated boolean features, therefore identical features are removed as the last processing step. The boolean features generated representing the occurrence of the words are: *gatewaypundit, video, sidney, powell, marklevinshow, bombshell, realjameswoods, judgejeanine, tomfitton, govmikehuckabee, marylene, kraken, files, dominion, complaint, exclusive, statistically, leftist, breaking, boom, priot*.

Another way to find potential features is to use the lift metric to evaluate the association interest of words and the target labels. This way, we can objectively measure the significance of the rules comparing their quality concerning both questionability labels. Lift is computed for words with support greater than than the quantile 0.998. The words that belong to the top 10 rules according to lift values are (in descending order) *gatewaypundit, joe, ballot, biden, fraud, election, president, see, news* for the questionable category, and *nytimes* associated with the occurrence of non-questionable tweets. The highest lift value is 5.92 for *gatewaypundit* and the lowest is 1.17 for *nytimes*.

To also evaluate association rules with two or more words (antecendents) the apriori method was used. Only rules with questionability as consequent were considered. No rules were found with a support bigger than 0.03 and most rules with a lift bigger than 1 had words already included previously like *gatewaypundit* and *nytimes*. We decided to not include these features in the dataset since they would most likely add little to no value to the model's performance.

The dataset 3 is composed of features created based on the lift metric as well as the ones based on the support and confidence limited by a threshold. Finally, the model performance is evaluated against the test dataset. The metrics *accuracy, f1, precision, and recall* can be seen in table V, Appendix C.

D. Dataset 4 (DS4)

We wondered if the topics of the tweets were a good indicator of the class. We used word embeddings to check if the meaning of the tweets were similar. Word embeddings is a set of techniques which allow us to represent words in the form of a vector. The feature vector represents different aspects of the word. The distribution representation allows capturing semantic similarity of words, in the sense of similar meaning have similar representation.

One of the techniques is Word2Vec, developed by Tomas Mikolov at Google over two papers [5], [6] which is a neural network that is trained to reconstruct contexts of words. The similarity of semantic words is derived by cosine distance between words. There are two different architectures of network. Continuous bag-of-words architecture (CBOW) takes the context of the word, according to the size of window, and from there it predicts the original word. On the other hand, Skip-Gram (SG) architecture uses the current word to predict the surrounding window of context words. Both models were trained to detect the most suitable for our problem. The size of vector space was just 100 in order to capture only the relevant features. The context of word is captured with 5 surrounding words. Additionally, words with total frequency lower than 4 were removed.

Word embeddings are very good for capturing word semantics, but as we are studying phrases, it would be interesting to use document embeddings techniques to learn the semantic of texts. Document embeddings can be considered as vector representations of word embeddings with the addition of paragraph context. Doc2Vec models have two variants. Distributed Memory (DM) model is similar to the Continuous Bag of Word and the Distributed Bag of Words is similar to the Skip-Gram model. The suitability of the method will depend on the data, therefore we test the performance of these four variants with only two algorithms, those who have had better performance, Random Forest and XGBoost, with 5 fold cross-validation. The table II (Appendix B) shows that the results are quite similar between the models and the variants. In that sense, we focused on F1 measure, because it is a harmonic mean of precision and recall. Therefore, the technique selected was Skip-Gram, which present higher performance with both algorithms (see table II, Appendix B).

V. Feature Importance

The final dataset contains 161 features. Random forest model presents the highest overall performance, as seen in Table VI. Trained with DS4, its metrics are *f1_macro=0.85*, *precision=0.76*, and *recall=0.75*. Figure 4 exhibits the 20 features with highest importance according to the mean decrease in impurity. The most important features are the number of capitalized words, followed by the ratio of uppercase letters in a tweet, and *gatewaypundit*, which accounts for the presence of this mention in the tweet. A random forests model trained with the 20 most important features present test metrics *f1_macro=0.70*, *precision=0.66*, and *recall=0.74*. This result implies that the number of features used in a final model can potentially be optimized.

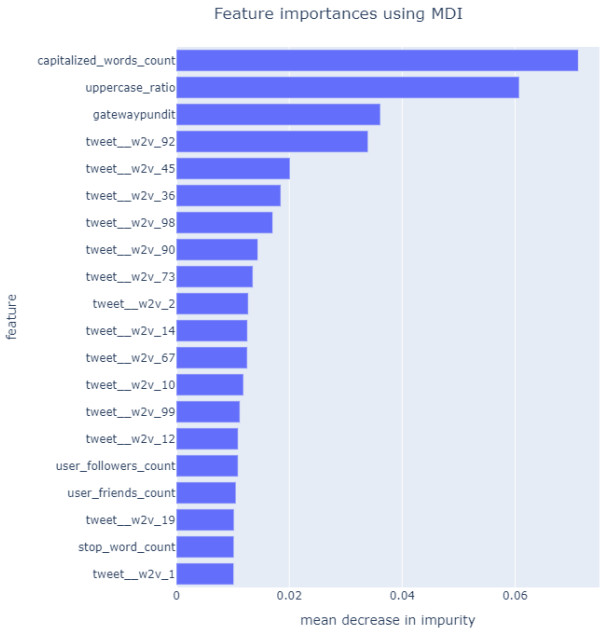


Fig. 4: Feature Importance. Output limited to 20 features.

VI. Results

The metrics for the best model across each of the datasets are presented in Figure 5. Overall, the model trained with Random Forest presents superior performance in all metrics, and for all dataset iterations.

Overall, models' performances increase in each metric across all dataset iterations, as shown in the tables of Appendix C. In contrast, Random Forest presents a small decrease in *recall* with the addition of features from DS2, and Gaussian Naive Bayes *recall* diminishes over iterations. Indeed, *recall* shows lowest improvement over the different datasets and models. Overall, the highest performance increase occur with the addition of features from DS2 (Table IV), specially regarding precision. All models present improvements in the precision metric with new features, and while Gaussian Naive Bayes presents the highest precision in DS4 (Table VI), it also presents a clear trade-off with Recall. This is reflected in the *f1 macro* metric, for which Random Forests has the highest value, and Naive Bayes the lowest.

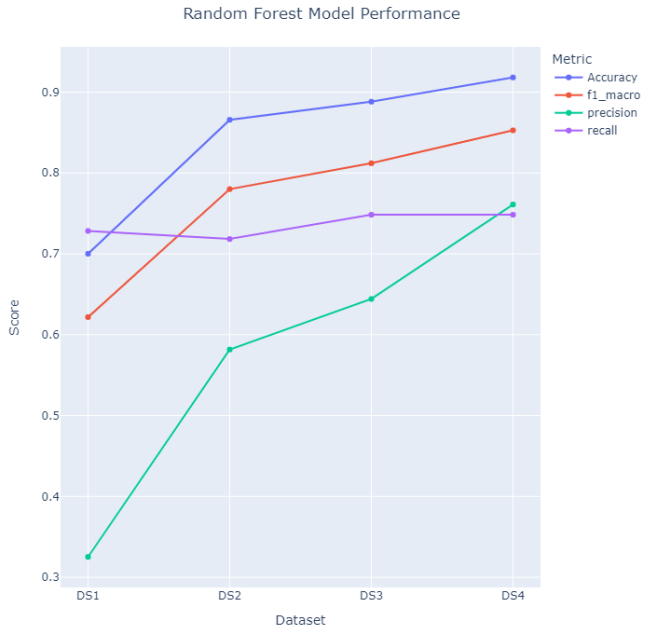


Fig. 5: f1-macro, precision, recall and accuracy for the best model for each dataset (RF).

In general similar performance was obtained using SVC, XGBoost and MLP (see Appendix C). The poorer performance metrics were obtained by Decision Trees and Gaussian Naive Bayes.

VII. Conclusions

In this work, the content of tweets of the last U.S. presidential election and its users were analysed in order to create a machine learning model to detect questionable tweets. We applied linguist, context, sentiment metrics, and other data mining tools to extract relevant information.

Some interesting results were found. Questionable tweets authors do not have many followers; however, the number of friends is higher compared to non-questionable tweets. Moreover, capitalised words and the use of stop words are more frequent in unreliable tweets. Mentions to specific accounts, such as *gatewaypundit*, proved to be suitable explanatory variables for questionable tweets, while other rules based on words' presence were not as important. These variables had a higher contribution to the identification of classes.

Nevertheless, future studies might use different weights for false positives and false negatives, optimise the number of features without performance loss, and study online stream-based solutions. The performance achieved is satisfactory, although the issue of disinformation is delicate and needs robust and ethical models. Therefore, researchers must continue to construct useful and credible tools.

REFERENCES

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] S. Helmstetter and H. Paulheim, "Collecting a large scale dataset for classifying fake news tweets usingweak supervision," *Future Internet*, vol. 13, no. 5, 2021.

[3] S. Bird, E. Klen, and E. Loper, *Natural Language Processing with Python*, o'reilly m ed., 2009.

[4] S. Loria, "textblob documentation," *Release 0.15*, vol. 2, 2018.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

APPENDIXES

A. Phased results for Dataset 2

In dataset 2 a total of 29 features were extracted, divided across four main groups (count features, ratio/compound features, binary features and sentiment features). In order to evaluate the impact of each group of features, performance metrics were obtained for the sequential addition of each group. In Table I are presented the obtained performance metrics with the sequential addition of the four group of features. One can notice that the higher increase of performance happens when the count features are added. When the other groups of features are added, the increase in the performance metrics is residual. Indeed, the performance metrics slightly decrease for the RF, SVC and MLP models when the sentiment features are added.

TABLE I: DS2 Test performance for phased addition of features.

Dataset	Metric	RF	XGB	SVC	MLP	DT	GNB
+ Count	Accuracy	0.857	0.818	0.856	0.828	0.819	0.390
	f1_macro	0.770	0.734	0.762	0.735	0.720	0.384
	precision	0.556	0.475	0.560	0.493	0.472	0.199
	recall	0.725	0.763	0.675	0.698	0.668	0.868
+ Ratio	Accuracy	0.863	0.832	0.866	0.832	0.818	0.423
	f1_macro	0.778	0.745	0.777	0.745	0.724	0.412
	precision	0.573	0.499	0.587	0.501	0.473	0.207
	recall	0.725	0.738	0.693	0.733	0.695	0.862
+ Binary	Accuracy	0.869	0.832	0.856	0.843	0.828	0.452
	f1_macro	0.785	0.749	0.767	0.753	0.732	0.437
	precision	0.588	0.501	0.557	0.525	0.491	0.216
	recall	0.727	0.763	0.705	0.713	0.683	0.857
+ Sent.	Accuracy	0.866	0.836	0.849	0.830	0.834	0.455
	f1_macro	0.780	0.751	0.760	0.740	0.738	0.439
	precision	0.582	0.509	0.540	0.496	0.505	0.217
	recall	0.718	0.743	0.710	0.717	0.682	0.857

B. Performance of W2V and D2V

TABLE II: Performance of W2V and D2V.

Learner	Precision	Recall	F1-Measure
Random Forest - SG	0.957	0.518	0.672
XGBoost - SG	0.914	0.600	0.724
Random Forest - CBOW	0.957	0.514	0.669
XGBoost - CBOW	0.923	0.596	0.722
Random Forest - DM	0.962	0.339	0.501
XGBoost - DM	0.855	0.511	0.639
Random Forest - DBOW	0.964	0.499	0.658
XGBoost - DBOW	0.902	0.581	0.707

C. Results

TABLE III: DS1 Test performance.

Metric	RF	XGB	SVC	MLP	DT	GNB
Accuracy	0.700	0.675	0.668	0.658	0.647	0.312
f1_macro	0.622	0.599	0.545	0.551	0.571	0.312
precision	0.325	0.302	0.237	0.247	0.275	0.188
recall	0.728	0.710	0.437	0.505	0.673	0.935

TABLE IV: DS2 Test performance.

Metric	RF	XGB	SVC	MLP	DT	GNB
Accuracy	0.866	0.836	0.849	0.830	0.834	0.455
f1_macro	0.780	0.751	0.760	0.740	0.738	0.439
precision	0.582	0.509	0.540	0.496	0.505	0.217
recall	0.718	0.743	0.710	0.717	0.682	0.857

TABLE V: DS3 Test performance.

Metric	RF	XGB	SVC	MLP	DT	GNB
Accuracy	0.888	0.864	0.873	0.865	0.835	0.904
f1_macro	0.812	0.789	0.791	0.782	0.744	0.792
precision	0.644	0.568	0.601	0.578	0.508	0.866
recall	0.748	0.797	0.732	0.738	0.708	0.507

TABLE VI: DS4 Test performance.

Metric	RF	XGB	SVC	MLP	DT	GNB
Accuracy	0.918	0.883	0.898	0.885	0.810	0.905
f1_macro	0.853	0.814	0.831	0.813	0.722	0.795
precision	0.761	0.612	0.664	0.626	0.460	0.872
recall	0.748	0.822	0.797	0.787	0.737	0.512