University of Milan

Faculty of Political, Economic and Social Sciences

To Bot or Not to Bot?

Final Project in the *Natural Language Processing* Course

Julia Maria Wdowinska

Data Science for Economics
II year
Master's Degree
Matriculation Number: 43288A

November 11, 2025

Contents

1 Introduction **1**

2 Data Processing **1**
2.1 Dataset Overview 1
2.2 Feature Engineering 2
2.3 Data Summary 2
2.4 Feature Selection 3
2.5 Scaling 4
2.6 Text Embeddings 4
2.7 Class Imbalance Handling 5

3 Model **5**
3.1 Model Architecture 6
3.2 Model Training and Results 6
3.2.1 Effect of Embedding Variants on Model Performance 6
3.2.2 Impact of Using English-Only Tweets on Model Performance 7
3.2.3 Final Model Selection and Evaluation 8

4 Conclusions **9**

5 Limitations and Future Work **9**

1 Introduction

Social media platforms such as Instagram, Facebook, or X (formerly Twitter) play a central role in modern communication. X, in particular, has become one of the most influential platforms, gradually replacing traditional media as a major source of news and public discussion. As a result, the content shared on X can shape public opinion and influence social debates.

For this reason, it is important to ensure that accounts are authentic and to limit the presence of automated accounts, commonly known as bots. Bots are automated profiles designed to imitate human activity. While some have useful purposes, many are used to spread misinformation, manipulate public opinion, influence elections, or promote extremist and discriminatory content.

As bots become more sophisticated, detecting them reliably remains a difficult task. Although research in this area has advanced, relatively few studies apply a multimodal approach that combines both textual and numerical information. In addition, many existing methods rely on a small number of numerical features, often fewer than ten, and do not fully use the textual data available in user profiles and posts.

A recent study by Arranz-Escudero et al. (2025) tried to address these limitations. Their work introduced a multimodal framework that combined multiple numerical and categorical profile features with textual information extracted from user descriptions and tweets. The authors also used graph-based data reflecting user connections. This approach achieved a 5.48% improvement over the previous best result.

This project aims to replicate and critically evaluate this multimodal methodology, looking closely at its elements and trying to improve where possible.

2 Data Processing

2.1 Dataset Overview

The dataset used in this project was TwiBot-22 (Feng et al., 2023), which is currently the most extensive X dataset available. It is fully labeled (bots vs. humans) and contains various X components such as users, tweets, lists, and hashtags. Working with it was challenging due to its size. It includes 1,000,000 users (860,057 human and 139,943 bot accounts) and 88,217,457 tweets, while the previous version, TwiBot-20, contained only 229,580 users, of which only 11,826 were labeled.

To mitigate the computational overhead, only one chunk of tweets (each approx. 10–11 GB in size) was used. The analysis was then limited to users present in this chunk, along with their corresponding user data.

2.2 Feature Engineering

First, user data was processed to extract features that could help characterize users and distinguish between human and bot accounts. While Arranz-Escudero et al. (2025) introduced 46 features (34 numerical and 12 categorical), in this project 47 numerical and boolean features were created. In addition, as in Arranz-Escudero et al. (2025), the user description was recorded as well. All user-level features are presented in Table 1.

Table 1: User-level Features

Feature	Description
name_length	Number of characters in name
username_length	Number of characters in username
username_name_length_ratio	Ratio of username length to name length
description	User description with URLs, mentions, and emails replaced by tokens
description_length	Number of characters in description
has_name, has_username, has_description, has_url, has_location, has_pinned_tweet	Binary indicators of non-empty fields
has_bot_word_in_name, has_bot_word_in_description	Presence of the word "bot" in name or description
ratio_digits_*	Ratio of digits to total characters in name, username, or description
ratio_special_chars_*	Ratio of special characters (non-alphanumeric) in name, username, or description
name_upper_to_lower_ratio, username_upper_to_lower_ratio	Ratio of uppercase to lowercase letters
name_entropy, username_entropy	Shannon entropy of name and username
username_name_levenshtein	Normalized Levenshtein distance between username and name
description_sentiment	Compound sentiment score of description (VADER)
cashtag_in_description_count, hashtag_in_description_count, mention_in_description_count, url_in_description_count	Count of respective entities in description
is_protected, is_verified	Account protection and verification status
account_age_seconds	Account age in seconds
followers_count, following_count, listed_count, tweet_count	User activity metrics
followers_over_following, double_followers_over_following, following_over_followers, following_over_followers_squared	Ratios of follower and following counts
following_over_total_connections	Ratio of following count to total connections
listed_over_followers, tweets_over_followers, listed_over_tweets	Ratios between listing, tweet, and follower counts
follower_rate, following_rate, listed_rate, tweet_rate	Growth rates per account age

Next, tweet data was processed to extract features at the tweet level, which were then aggregated at the user level and merged with the user data. Seven numerical or boolean features were created, and similarly to Arranz-Escudero et al. (2025), concatenated tweet texts were recorded as well. Since tweets in all languages were considered, only the 10 most recent tweets were used, unlike in Arranz-Escudero et al. (2025), who used 20, because multilingual embedding models accept shorter inputs than English-only models. For the same reason, each tweet text was truncated to 12 tokens.

A change introduced in this project was the addition of non-truncated tweet texts stored in a list. This was done to test whether concatenating all tweets into one long string and embedding it as a single sentence could distort the information. Both ways of gathering tweet text data were later evaluated in terms of their effect on classifier performance. All tweet-level features created are described in Table 2.

Table 2: Tweet-level Features

Feature	Description
top_tweets_truncated_texts_concat	Concatenation of tweet texts, each truncated to 12 tokens
top_tweets_texts	List of tweet texts, no truncation
top_tweets_reply_fraction	Fraction of tweets that are replies to another tweet
top_tweets_num_distinct_langs	Number of distinct languages in tweets
top_tweets_sensitive_fraction	Fraction of tweets that may contain sensitive content
top_tweets_avg_likes	Average number of likes per tweet
top_tweets_avg_quotes	Average number of quotes per tweet
top_tweets_avg_replies	Average number of replies per tweet
top_tweets_avg_retweets	Average number of retweets per tweet

Note: All statistics are computed over the 10 most recent tweets per user.

2.3 Data Summary

The final dataset comprised 314,813 users and 57 features: 44 numeric, 10 boolean, and 3 text features (two of which contained information on tweet texts). Two columns, top_tweets_avg_quotes and top_tweets_avg_replies, had a large proportion of missing values (56.52% each). Given the availability of many other features, these columns were removed rather than imputed.

The dataset was highly imbalanced, with 92.46% of accounts labeled as human and 7.54% labeled as bots. A stratified 90/10 train-test split was applied to preserve this distribution:

- **Training set:** 283,331 users (90% of the dataset)
  - Class 0 (human): 261,979 users (92.46%)
  - Class 1 (bot): 21,352 users (7.54%)
- **Test set:** 31,482 users (10% of the dataset)
  - Class 0 (human): 29,110 users (92.47%)
  - Class 1 (bot): 2,372 users (7.53%)

2.4 Feature Selection

Given the large number of features, some were likely redundant. Therefore, feature selection methods[1] were used to identify the most informative ones. These methods were preferred over dimensionality reduction techniques to keep the features interpretable.

First, pairwise correlations between numeric features were examined (see Figure 1). Six feature pairs were found to be highly correlated (absolute correlation > 0.8), as summarized in Table 3. No features were removed at this stage to avoid arbitrary decisions.

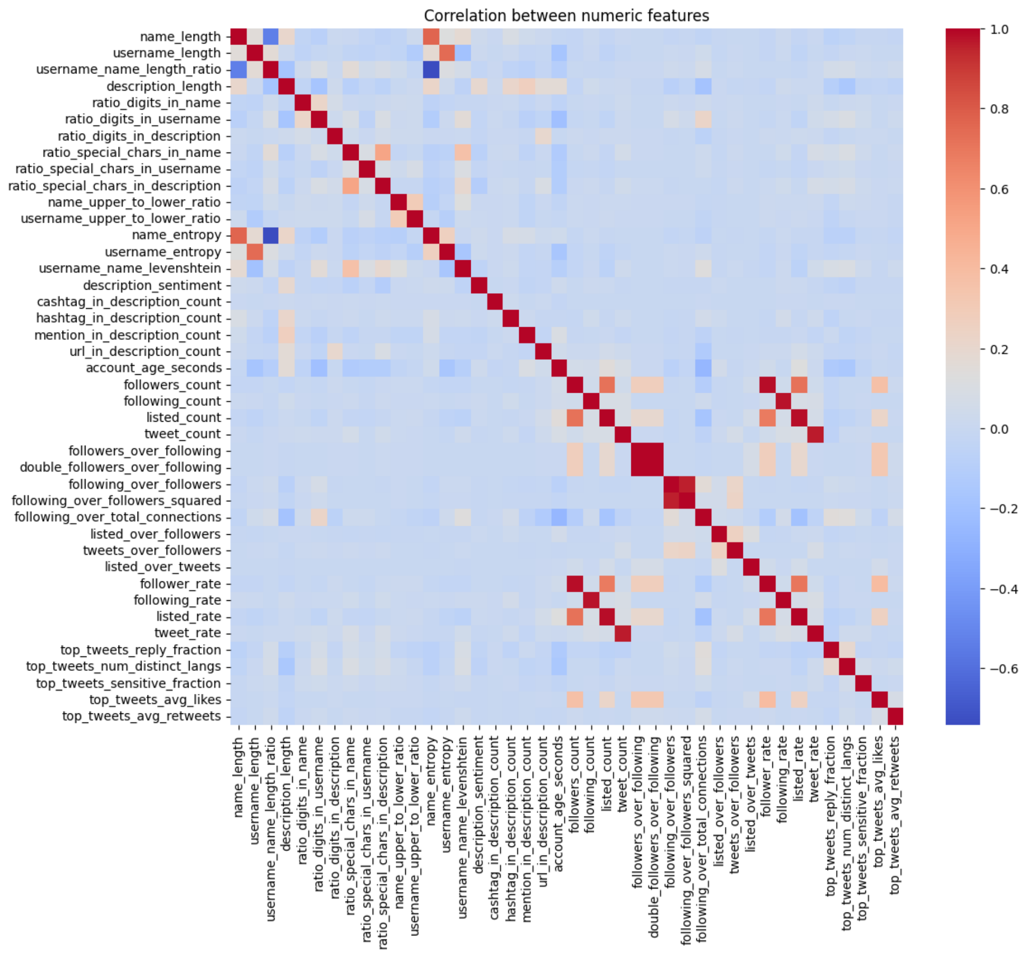

Figure 1: Pairwise Correlations

Next, the variance of numeric and boolean features was examined. Features with variance below 0.01 were considered to have low discriminatory power and were removed. The removed features and their variances are listed in Table 4.

[1]To prevent information leakage, feature selection methods as well as subsequent scaling were fitted exclusively on the training set and then applied to the test set, while oversampling was performed exclusively on the training set, ensuring that the test set remained unseen and could provide an unbiased evaluation of model performance.

Table 3: Highly Correlated Feature Pairs

Feature 1	**Feature 2**	**Absolute Correlation**
followers_count	follower_rate	0.98
following_count	following_rate	0.98
listed_count	listed_rate	0.99
tweet_count	tweet_rate	0.96
followers_over_following	double_followers_over_following	1.00
following_over_followers	following_over_followers_squared	0.96

Table 4: Low-Variance Features

Feature	**Variance**
has_username	0.000000
listed_rate	0.000000
following_rate	0.000000
tweet_rate	0.000000
follower_rate	0.000004
has_name	0.000032
ratio_digits_in_description	0.000829
has_bot_word_in_name	0.000924
ratio_digits_in_name	0.001506
has_bot_word_in_description	0.001853
is_protected	0.001994
ratio_special_chars_username	0.002008
top_tweets_sensitive_fraction	0.004211

Finally, a Gradient Boosting Classifier (GBC) was used to identify the most important numeric and boolean features remaining after variance filtering. The classifier was trained using default hyperparameters:

- $n\_estimators = 100$: number of trees
- $learning\_rate = 0.1$: step size shrinkage to prevent overfitting
- $max\_depth = 3$: maximum depth of each tree
- $subsample = 1.0$: fraction of samples used for each tree
- $loss =$ log_loss: loss function minimized during training
- $max\_features =$ None: number of features considered per split

These settings specify that 100 trees were built with a maximum depth of 3, all samples were used to build each tree, the log loss function was minimized, and all features were considered at each split. Feature importance was calculated as the average reduction in the loss function across all trees. Only features with importance above the median were kept. This approach is more conservative than in Arranz-Escudero et al. (2025), who retained all features with non-zero importance. The selected features and their normalized importance scores are shown in Table 5.

These results are generally consistent with Arranz-Escudero et al. (2025). For example, has_description was the most important feature in their study, whereas it ranked third here. The feature `tweet_count` was the second most important in both studies. Small differences are likely due to this analysis being conducted on a subset of users, while the referenced study considered the full dataset.

2.5 Scaling

Following Arranz-Escudero et al. (2025), all numeric features were standardized to have a mean of 0 and a standard deviation of 1. This step is not needed for tree-based models, but it was justified here because a neural network was used. Scaling the features helps improve both stability and performance for neural networks.

2.6 Text Embeddings

After feature selection and scaling, a numerical representation of textual data was created using pre-trained embedding models. User descriptions were always encoded using *distiluse-base-multilingual-cased-v1*, while tweet texts were encoded using two different models and either as a concatenation or as a list of tweets. This resulted in four embedding versions:

- **v1:** Concatenated tweets encoded using *distiluse-base-multilingual-cased-v1*.

Table 5: Selected Features and Their Importance

Feature	**Importance**
followers_count	0.4999
tweet_count	0.1094
has_description	0.1065
description_length	0.0976
following_count	0.0624
top_tweets_avg_retweets	0.0253
is_verified	0.0218
account_age_seconds	0.0165
username_name_length_ratio	0.0094
ratio_special_chars_in_description	0.0087
ratio_special_chars_in_name	0.0071
hashtag_in_description_count	0.0056
listed_over_followers	0.0043
listed_over_tweets	0.0039
name_length	0.0037
top_tweets_num_distinct_langs	0.0032
top_tweets_avg_likes	0.0023
followers_over_following	0.0021
mention_in_description_count	0.0016
tweets_over_followers	0.0014

- **v2:** Concatenated tweets encoded using *paraphrase-multilingual-MiniLM-L12-v2*.
- **v3:** Tweets encoded separately using *distiluse-base-multilingual-cased-v1*, followed by mean-pooling[2].
- **v4:** Tweets encoded separately using *paraphrase-multilingual-MiniLM-L12-v2*, followed by mean-pooling.

This process was simpler than in Arranz-Escudero et al. (2025), because the embedding models handle tokenization automatically and there is no need to remove stopwords. In fact, removing stopwords can sometimes reduce performance, since the models rely on the full context to generate embeddings. Another difference is that the embedding dimension was 512 instead of 768, corresponding to the output size of the multilingual models used.

2.7 Class Imbalance Handling

To address the substantial class imbalance in the dataset, oversampling was performed using the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002). Each embedding dimension was treated as an additional feature during oversampling. After applying SMOTE, the class distribution became:

- Class 0 (human): 66.67%
- Class 1 (bot): 33.33%

This approach slightly improved model performance and worked better than alternative strategies, such as using a weighted loss function (as in Arranz-Escudero et al., 2025) or applying sampling weights when forming training batches.

The entire data processing pipeline is illustrated in Figure 2.

3 Model

In the study by Arranz-Escudero et al. (2025), the best-performing model was a Relational Graph Convolutional Network (RGCN). This type of model uses the graph structure of a social network such as X, combining information from a user's own features, their connections, the types of these connections, and the features of neighboring users.

In this project, a standard Graph Convolutional Network (GCN) was explored. Only a single type of edge was considered, representing a "follows" relationship between users. Furthermore, only edges connecting users within the selected subset were included. The results differed substantially from those in the original paper. This is likely due to the artificial nature of the network created by restricting edges to the subset: users in the subset may have had connections to users outside it, but these were disregarded, which distorted the graph structure.

[2]Mean-pooling means that each tweet in the list was encoded separately and then an average over all tweet embeddings per user was computed.

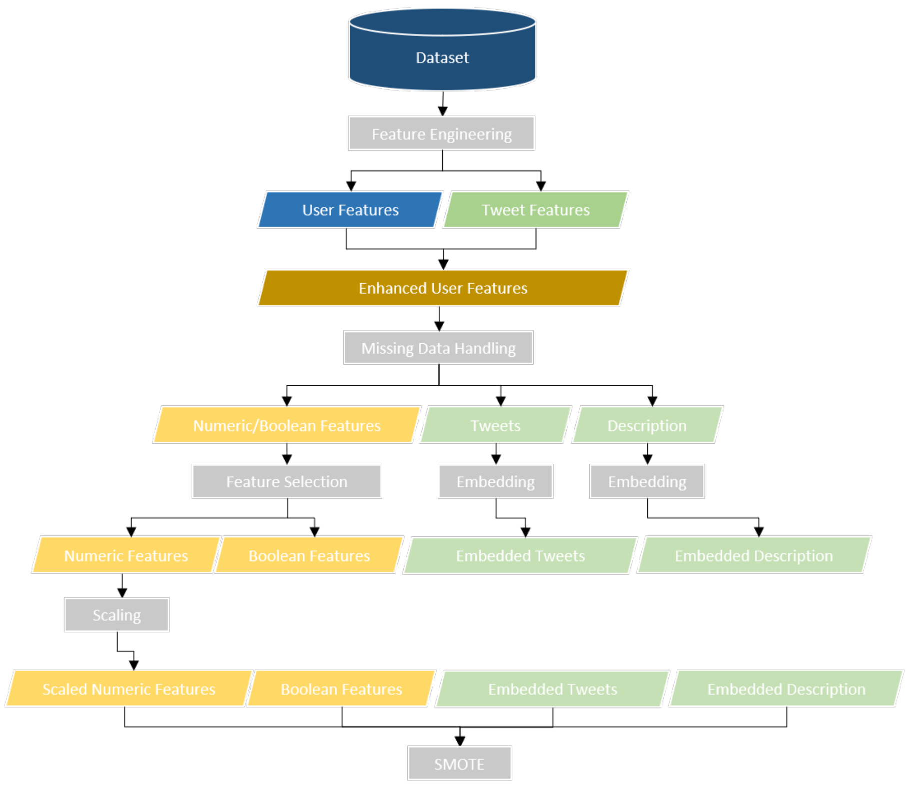

Figure 2: Data Processing Pipeline

3.1 Model Architecture

Consequently, a multi-branch neural network was implemented, following the general architecture of the original paper but without the graph component. The network comprised three branches: one for numeric and boolean features, one for description embeddings, and one for tweet embeddings.

Each branch applied a linear transformation, followed by a Leaky ReLU activation and a dropout layer. The outputs of all branches were then concatenated and passed through another linear transformation, followed by a Leaky ReLU activation, a dropout layer, and a final linear transformation producing a one-dimensional output.

3.2 Model Training and Results

Several experiments were performed to evaluate and optimize the model.

First, the effect of different embedding variants on model performance was assessed. The neural network was trained four times, using the same set of hyperparameters for each run, but with different embeddings.

Next, the impact of using only English-language tweets was examined. All steps from feature engineering to oversampling were repeated, and the neural network was retrained for each of the four embedding variants, with the only difference being the use of English-only embedding models.

Finally, the embedding variant that achieved the highest F1 score, whether all-language or English-only, was selected as the best setting. Hyperparameter optimization was then performed, and the network was retrained using these optimized hyperparameters.

3.2.1 Effect of Embedding Variants on Model Performance

Training was performed using mini-batches of 32 samples with a dropout rate of 0.5. The hidden dimension of the neural network was set to 128. The Binary Cross-Entropy (BCE) loss function, implemented with an integrated sigmoid layer for numerical stability, was used. The Adam optimizer was applied with a learning rate of 1×10^{-3} and a weight decay of 1×10^{-5}. Predictions were binarized using a threshold of 0.5.

The network was trained for 100 epochs for each embedding variant. The epoch that achieved the highest F1 score for each variant is reported in Table 6.

The highest F1 score was achieved by the third embedding variant, with 43.72%. The highest recall was observed for the fourth variant, 50.97%, which was associated with the second best F1 score of 43.3%. This indicates that the approach introduced in this project, where each tweet is embedded separately without truncation and then mean-pooling is applied, slightly outperformed the method of Arranz-Escudero et al. (2025), where tweet texts were concatenated into a single string before embedding.

Table 6: Performance Across Embedding Variants

Variant	Best F1	Epoch at Best F1	Test Accuracy	Test Precision	Test Recall
v1	0.4314	30	0.9112	0.4166	0.4473
v2	0.4263	8	0.9076	0.4004	0.4557
v3	0.4372	43	0.9069	0.4016	0.4798
v4	0.4330	68	0.8994	0.3764	0.5097

3.2.2 Impact of Using English-Only Tweets on Model Performance

A question arose as to whether restricting tweets to English only would improve model performance. To investigate this, the network was trained and evaluated using embeddings derived exclusively from English tweets.

Because English-only embedding models support longer token sequences, the 20 most recent tweets (instead of 10) were used to compute tweet-level statistics. The feature top_tweets_num_distinct_langs was omitted, since all users had tweets in a single language after filtering. The resulting dataset included 278,227 users and 56 features. As before, the features top_tweets_avg_quotes and top_tweets_avg_replies contained a large proportion of missing values (59.39% each) and were removed. The class distribution remained imbalanced, with 92.16% human accounts and 7.84% bots. A stratified 90/10 train-test split was applied to preserve this ratio:

- **Training set:** 250,404 users (90% of the dataset)
  - Class 0 (human): 230,765 users (92.16%)
  - Class 1 (bot): 19,639 users (7.84%)
- **Test set:** 27,823 users (10% of the dataset)
  - Class 0 (human): 25,641 users (92.16%)
  - Class 1 (bot): 2,182 users (7.84%)

Variance thresholding and feature selection using a Gradient Boosting Classifier (GBC) were performed with the same hyperparameters as before. The selected features and their importance scores are shown in Table 7.

Table 7: Selected Features and Their Importance (English-Only Tweets)

Feature	Importance
followers_count	0.4971
description_length	0.1402
tweet_count	0.1036
following_count	0.0678
has_description	0.0631
top_tweets_avg_retweets	0.0303
is_verified	0.0230
account_age_seconds	0.0217
username_name_length_ratio	0.0106
hashtag_in_description_count	0.0058
name_length	0.0055
listed_over_followers	0.0050
double_followers_over_following	0.0033
listed_over_tweets	0.0030
ratio_special_chars_in_description	0.0027
top_tweets_avg_likes	0.0022
ratio_special_chars_in_name	0.0021
tweets_over_followers	0.0019
top_tweets_reply_fraction	0.0016

Numeric features were standardized to have zero mean and unit variance. User descriptions were embedded using *all-MiniLM-L6-v2*, while tweets were embedded using the same four approaches as before but with English-only models, yielding the following variants:

- **v1:** Concatenated tweets encoded using *all-MiniLM-L6-v2*.
- **v2:** Concatenated tweets encoded using *paraphrase-MiniLM-L6-v2*.
- **v3:** Tweets encoded separately using *all-MiniLM-L6-v2*, followed by mean-pooling.
- **v4:** Tweets encoded separately using *paraphrase-MiniLM-L6-v2*, followed by mean-pooling.

Class imbalance was addressed using SMOTE, resulting in 66.67% human and 33.33% bot accounts.

The same neural network hyperparameters as before were used: batch size of 32, dropout rate of 0.5, Binary Cross-Entropy loss without class weighting, Adam optimizer with learning rate 1×10^{-3} and weight decay 1×10^{-5}, and a prediction threshold of 0.5. Each network was trained for 100 epochs, and the best-performing epoch (by F1 score) for each embedding variant was recorded:

Table 8: Performance Across Embedding Variants (English-Only Tweets)

Variant	**Best F1**	**Epoch at Best F1**	**Test Accuracy**	**Test Precision**	**Test Recall**
v1	0.4125	6	0.8920	0.3597	0.4835
v2	0.4052	2	0.8902	0.3522	0.4771
v3	0.4149	7	0.8917	0.3601	0.4895
v4	0.4187	62	0.8988	0.3811	0.4647

The highest F1 score (41.87%) was achieved with embedding variant *v4*. The highest recall (48.95%) occurred with *v3*, which also produced the second highest F1 score (41.49%). As previously, embedding tweets separately and then performing mean pooling yielded better results. Overall, however, F1 scores were slightly lower than those obtained using tweets in all languages. This may be due to the smaller dataset size for English-only tweets (278,227 versus 314,813 users). Nevertheless, these results suggest that filtering tweets by language is not strictly necessary for bot detection, allowing all tweets to be used and reducing preprocessing time.

3.2.3 Final Model Selection and Evaluation

The embedding variant that achieved the best performance in terms of F1 score was the all-languages variant, *v3*, and it was selected for final model tuning.

Instead of using a separate training, validation, and test split as in Arranz-Escudero et al. (2025), a 5-fold cross-validation[3] was performed on the training set. In each of the five iterations, 80% of the training data was used for training and the remaining 20% for validation. The parameter grid for tuning was as follows:

```
param_grid = {
    "batch_size": [16, 32, 64],
    "hidden_dim": [64, 128, 256],
    "dropout": [0.3, 0.5, 0.7],
    "lr": [1e-2, 1e-3, 5e-4],
    "weight_decay": [0, 1e-5, 1e-4]
}
```

To reduce computation time, only 10 random combinations out of 243 were tested. For each combination and each iteration, the network was trained for 50 epochs with early stopping if the validation F1 did not improve for 3 consecutive epochs. The highest F1 score per iteration was recorded, and the mean of these scores across iterations was computed. This mean F1 score was used to compare different parameter combinations. The results are summarized in Table 9.

Table 9: Cross-Validation Results

batch_size	hidden_dim	dropout	lr	weight_decay	mean_best_f1
32	64	0.5	0.0005	0.0001	0.7960
32	256	0.5	0.01	0.0001	0.6992
64	256	0.5	0.0005	0	0.8742
64	64	0.3	0.001	0.0001	0.8082
16	256	0.5	0.01	0.0005	0.6894
64	256	0.7	0.01	0	0.7182
64	64	0.3	0.01	0.0001	0.7254
64	128	0.5	0.001	0	0.8407
16	128	0.5	0.001	0.0001	0.7973
64	128	0.5	0.001	0.0001	0.8252

The highest mean F1 score (87.42%) was achieved with batch size 64, hidden dimension 256, dropout 0.5, learning rate 0.0005, and weight decay 0. After selecting this combination, the network was retrained on the entire training set for 100 epochs and evaluated on the test set. The best performing epoch in terms of F1 was:

- Best F1: 0.4310
- Epoch at best F1: 15

[3]Cross-validation provides a more robust estimate of model performance by reducing the variance associated with a single validation split.

- Test Accuracy: 0.9136
- Test Precision: 0.4277
- Test Recall: 0.4342

Surprisingly, this F1 score was slightly lower than the one achieved with the previous parameters without hyperparameter tuning.

4 Conclusions

In this project, a multimodal approach to X bot detection was implemented and evaluated using a subset of the TwiBot-22 dataset (Feng et al., 2023). A large number of numerical and boolean user features were carefully curated and enriched with additional features extracted from users' most recent tweets. The most informative features were selected through variance thresholding and Gradient Boosting Classifier-based selection, and numerical features were scaled. Textual data, including user descriptions and tweet texts, was embedded using pre-trained models. Tweets were represented in two ways, either as truncated tweets concatenated into a string or as non-truncated tweets gathered in a list. Combined with two different models for tweets and a single model for descriptions, this produced four embedding variants. Significant class imbalance, approximately 7% bots, was addressed using SMOTE.

Regarding the model, a Graph Convolutional Network was initially explored but did not yield satisfactory results. A multibranch neural network without the graph component was therefore preferred. The impact of different embedding variants on performance was evaluated, and variant *v3* achieved the best F1, 43.72%, followed by variant *v4* with 43.3%. This showed that encoding tweets separately and then applying mean pooling produces better results than concatenating tweets into a single string, as in Arranz-Escudero et al. (2025). The effect of using only English-language tweets was also assessed. After filtering for English-only tweets and repeating all processing steps, results were slightly worse than those obtained with all-language tweets, with the best F1 at 41.87%, but variants *v3* and *v4* remained superior to variants *v1* and *v2*. Finally, the best embedding variant, all languages variant *v3*, was selected, hyperparameters were fine-tuned via 5-fold cross-validation, and the network was retrained on the full training set. The results were slightly lower than those obtained using the initially chosen parameters, which was somewhat surprising.

An important observation is the large discrepancy between F1 scores on validation sets (68.94%–87.42%) during cross-validation and those on the test set (40.52%–43.72%). Validation scores were close to or exceeded the results reported by Arranz-Escudero et al. (2025), whereas test scores were considerably lower. This suggests that the results in Arranz-Escudero et al. (2025) may be overly optimistic and potentially influenced by information leakage, as their processing steps are applied before splitting the data. In contrast, in this study, the data was first split into training and test sets, and all processing steps were fitted on the training set and then applied to the test set, ensuring that the test set remained independent and unbiased.

5 Limitations and Future Work

Despite the careful design of this study, several limitations should be noted.

Only a subset of the TwiBot-22 dataset was used, which restricted the network structure and may have limited the model's ability to leverage relational information. Consequently, a multibranch neural network was used instead of a graph-based model, which may have reduced performance by ignoring connections between users.

Feature and text representation choices also imposed limitations. Feature selection, while effective, may have overlooked interactions between features that are weak individually but informative in combination. Text embeddings involved truncation or mean pooling, potentially losing information from users with many tweets. Moreover, the choice of embedding models may have limited the richness of textual representations.

Finally, class imbalance and evaluation constraints should be considered. Although SMOTE mitigated imbalance, synthetic oversampling may not fully capture the true distribution of bot behavior. Discrepancies between cross-validation and test set performance indicate some sensitivity to data splits, and no external datasets were used to assess generalization.

These limitations highlight opportunities for future work, such as using the full TwiBot-22 dataset, exploring graph-based architectures, incorporating more sophisticated feature interactions, experimenting with alternative embedding strategies, and evaluating the approach on additional datasets to improve generalization and robustness.

AI Usage Disclaimer

This work was assisted by AI tools (mainly OpenAI's GPT-5, both free and paid versions) for research guidance, including understanding concepts such as class imbalance handling and graph neural networks, for refining and drafting code to process data or display statistics, and for improving language and clarity in the final report. The AI was used solely as a support tool; all research decisions, analyses, interpretations, and conclusions are my own. I remain fully responsible for the content, accuracy, and claims presented in this project.

References

Arranz-Escudero, O., Quijano-Sanchez, L., and Liberatore, F. (2025). Enhancing misinformation countermeasures: A multimodal approach to Twitter bot detection. *Social Network Analysis and Mining*, 15(1):26.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Feng, S., Tan, Z., Wan, H., Wang, N., Chen, Z., Zhang, B., Zheng, Q., Zhang, W., Lei, Z., Yang, S., Feng, X., Zhang, Q., Wang, H., Liu, Y., Bai, Y., Wang, H., Cai, Z., Wang, Y., Zheng, L., Ma, Z., Li, J., and Luo, M. (2023). TwiBot-22: Towards graph-based Twitter bot detection. *arXiv*. https://arxiv.org/abs/2206.04564.