CSC-51054-EP - DATA CHALLENGE

Influencer or Observer: Predicting Social Roles

December 11, 2025

Team Name : HAK; Al Hakim Taoufik, Aya Bayna, Khalid Akhmouch

1 Introduction

This report presents our approach to the "Influencer vs Observer" data challenge, which aims to determine whether a user's social role can be inferred from a single tweet. Each data point corresponds to a tweet labeled as originating from either an Influencer (users with highly asymmetric follower networks) or an Observer, who maintains a more reciprocal social structure. Since all explicit network features are removed (such as the number of followers, the engagements of the tweet...), the task relies solely on tweet content and available user metadata.

Our methodology follows a progressive strategy designed to ensure experimental clarity. We first train text-only models using fine-tuned transformer embeddings to evaluate how much linguistic signal alone encodes social roles. We then develop models based on tabular data consisting of informations extracted from the tweet (number of hashtags, number of mentions...) and user activity metadata to isolate the predictive power of user-level attributes. The final step involves embedding the text features using a pre-trained autoencoder. Both modalities' features are then combined and processed by an MLP with multi-head attention, resulting in our most robust final submission.

2 Methods

2.1 Text-only models

See Appendix A.1

In this first stage, we ignored all user metadata and relied exclusively on tweet text to predict whether a tweet comes from an Influencer or an Observer. This served both as a baseline and as a way to quantify the information contained purely in the language used.

We used the tweet text and the binary label as inputs. Then we created a stratified train–validation split to preserve class balance. The text was kept largely in its original Twitter form, including mentions, hashtags, emojis and URLs, since these elements may carry information relevant to the prediction task. We constructed two types of representations.

First, a simple TF-IDF vectorization using word and character n-grams was used to capture lexical and stylistic patterns, trained using a logistic regression classifier, which is a standard choice for high-dimensional sparse text data because it is simple, fast to train and effective with regularization.

Second, we produced contextual embeddings by tokenizing each tweet and fine-tuning a pretrained transformer model : BERTweet-FR [5], designed specifically for tweets in French.

These representations provided both a simple, frequency-based view (focusing on individual word occurrences) and a rich, contextual view of the same input. We found out that despite the natural chaotic nature of tweets, they still carried a lot of useful information for our classification task.

Both approaches were evaluated on the same row-wise validation split, and the best-performing text-only model, the latter one using the pretrained transformer, was retained for the subsequent stages of the pipeline.

2.2 User-feature-only models

See Appendix A.2

In this second stage, we removed the tweet text entirely and relied only on user-related metadata such as profile and activity features, and numerical features extracted from the tweet text (number of hashtags, mentions...). This setup isolates the predictive information contained in user behaviour and attributes, independently of linguistic content.

2.2.1 Data preprocessing

We applied a unified cleaning pipeline to both the training and test sets. From the raw Kaggle files, we retained only user-level metadata and removed all text fields.

User attributes such as activity indicators, profile or location information and client source were cleaned and standardized. We also noticed some missing values due to the nature of the challenge (such as the followers count...). These values were handled through simple imputation or explicit "unknown" categories, and high-cardinality categorical variables were reduced by grouping rare values into an "other" category. Numerical and categorical features were then encoded simply using methods suitable for tree-based model pipelines such as OneHotEncoder. For more details, please refer to the cleaning.py function. [8]

This process yielded a compact tabular dataset containing only user-level features and the binary label, and the same preprocessing was applied to the test set to ensure perfect feature alignment. We will use the same preprocessing pipeline for all the following models.

2.2.2 Models and rationale

Given the tabular nature of the cleaned dataset and the mix of numerical and categorical variables, we focused on gradient-boosted decision tree models. We evaluated a standard boosted-tree model such as XGBoost [2] and a categorical-boosting model such as CatBoost [7], which is specifically designed to handle high-cardinality categorical features without extensive manual encoding.

Tree-based ensembles are well suited for this setting because they capture non-linear interactions between user attributes, handle heterogeneous feature types and scale well to small and medium-sized tabular datasets. We compared individual models trained on all user features and then combined them in a simple ensemble by aggregating their predicted probabilities. The best-performing user-only model in later experiments corresponds to this ensemble.

2.2.3 Hyperparameter tuning and overfitting control

For the boosted-tree models, we tuned only the most influential hyperparameters such as tree depth, number of estimators, learning rate and subsampling rates. We used Optuna [8] to tune the hyperparameters, and the K-Fold method to reduce the variance and thus reducing overfitting without changing the expectation. We finally merged the XGBoost and CatBoost models using a classical ensemble method with the best threshold.

Overfitting was also controlled through early stopping on the validation set, shallow trees, small learning rates and subsampling, all of which reduce model complexity and improve generalization. This user-feature-only stage established a strong tabular baseline complementary to the text-based models.

2.3 Combining text and user features

See Appendix A.3

2.3.1 Early fusion

In the final stage, we combined tweet content and user metadata within a unified modeling framework. We explored two strategies. The first, early fusion, merged text-based and user-level features into a single tabular representation used to train a boosted-tree model. The second, late fusion, combined the predicted probabilities of the best text-only and user-only models. This comparison allowed us to evaluate whether a single joint model or a modular ensemble best leveraged the complementary information provided by both modalities.

For early fusion, we reused the preprocessing pipelines from the text-only and user-only methods. Text features were obtained through the TF-IDF or embedding representations from Method 1, while user attributes were cleaned and encoded as in Method 2. Both feature blocks were then aligned and merged into a single feature matrix with consistent indexing for the train and test sets.

We then trained a single gradient-boosted tree model (XGBoost) on the merged text and user feature matrix, allowing the model to learn interactions between linguistic patterns and user characteristics within a unified representation.

Finally, we tuned the main XGBoost hyperparameters using Optuna [3], searching over tree depth, learning rate, number of estimators and regularization terms, and selecting the configuration that maximized validation accuracy. Overfitting was controlled through early stopping, moderate tree depths and small learning rates.

2.3.2 Late fusion

For late fusion, no feature merging was required. Each model received the preprocessing associated with its modality, and only the predicted probabilities from the text-only and user-only models were combined. However, this time we switched to "BERTweet-base" [6] for the tweet, we also added the user description to the embedding lot, as it showed more promising results.

This ensured full consistency with the previous stages and avoided introducing additional preprocessing choices. [4]

We then combined the predicted probabilities of the best text-only and user-only models through a weighted averaging scheme, with the fusion weights selected on the validation split. This approach allows each model to specialize on its own modality and provides robustness when one source of information is noisy.

For the late-fusion model, tuning was minimal since both component models had already been optimized independently. Only the fusion weights combining their predicted probabilities were adjusted on the validation set. Because the search space was extremely small, the risk of overfitting at this stage remained limited.

Late fusion consistently outperformed the early-fusion approach on the validation set and was therefore retained as the final model.

2.4 Using state-of-the-art models

2.4.1 Tabular Deep Learning

To try and use the recent SOTA models available, we tried TabNet [9] on the metadata-only, because we saw promising results using the gradient boosting models, and we wanted to profit off the non-linearity and the performances of deep learning on complex models. However, it wasn't as promising, as it gave us an accuracy of 80% with a very slow learning, lackluster compared to the gradient-boosting models.

2.4.2 Attention mechanism

Finally, we tried to use multi-headed attention layers in the MLP we used for an early fusion model that mixed textual data after embedding using "BERTweet-base" and tabular data. It was really promising. However, it overfitted very badly, even if we used a user split to guarantee no data leakage, and even if we used dropout, loosen up the gradient descent, and froze the transformer at the first epoch... It still overfitted. However, if we had to continue on this project, this seems the most promising path.

3 Results

Model	Validation Accuracy
TF-IDF + Logistic Regression	0.625
Fine-tuned Transformer	0.836
User-only: XGBoost	0.833
User-only: CatBoost	0.837
User-only Ensemble	0.839
Early Fusion	0.838
Late Fusion (Final Model)	**0.840**

Table 1: Validation accuracy of all evaluated models.

4 Discussion and Limitations

Our results show that tweet content and user metadata capture complementary aspects of the Influencer versus Observer distinction, and that combining both modalities yields the best performance. However, the study has several limitations. The dataset provides only three to five tweets per user, which prevents modeling user-level behavior over time. The tweets were also quite noisy, with sometimes very little information there. Metadata fields can be noisy or incomplete.

Finally, transformer fine-tuning is computationally expensive, and boosted-tree models remain sensitive to preprocessing choices. These limitations suggest that multi-tweet aggregation and richer user-level features could further improve performance.

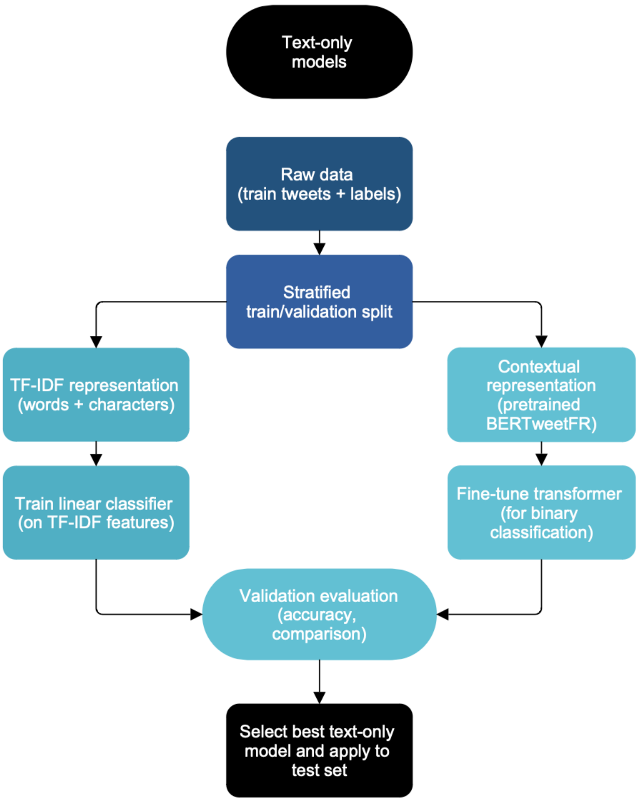
5 Conclusion

We evaluated a range of models based on tweet text, user metadata and their combination to predict whether a tweet originates from an Influencer or an Observer. Both modalities provided meaningful but distinct information, and the late-fusion ensemble consistently achieved the best validation performance. These results highlight the value of combining linguistic and behavioural cues. Future work could incorporate richer contextual signals to further improve performance.

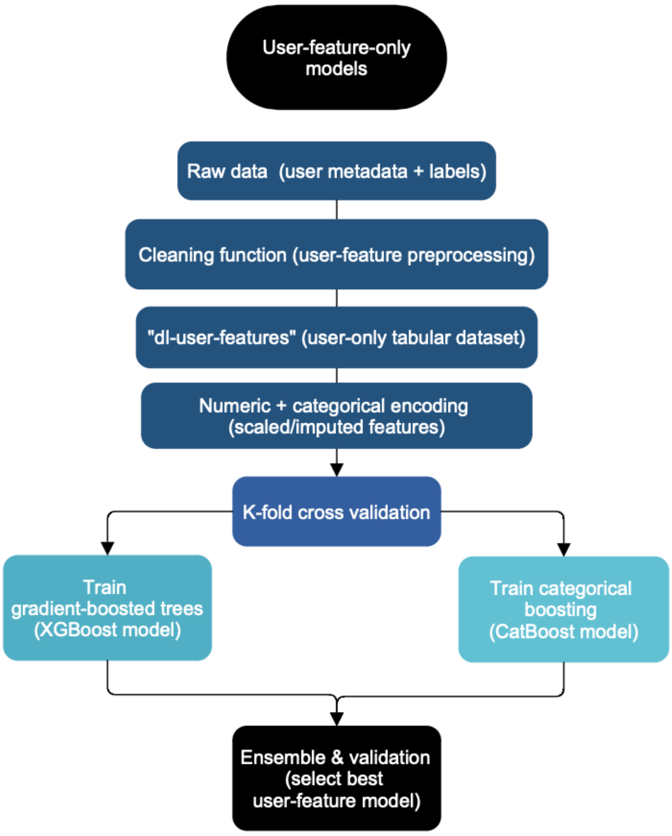
We learned a lot of techniques and methods of deep learning and machine learning that we will surely use later in our careers and projects.

A Model architectures

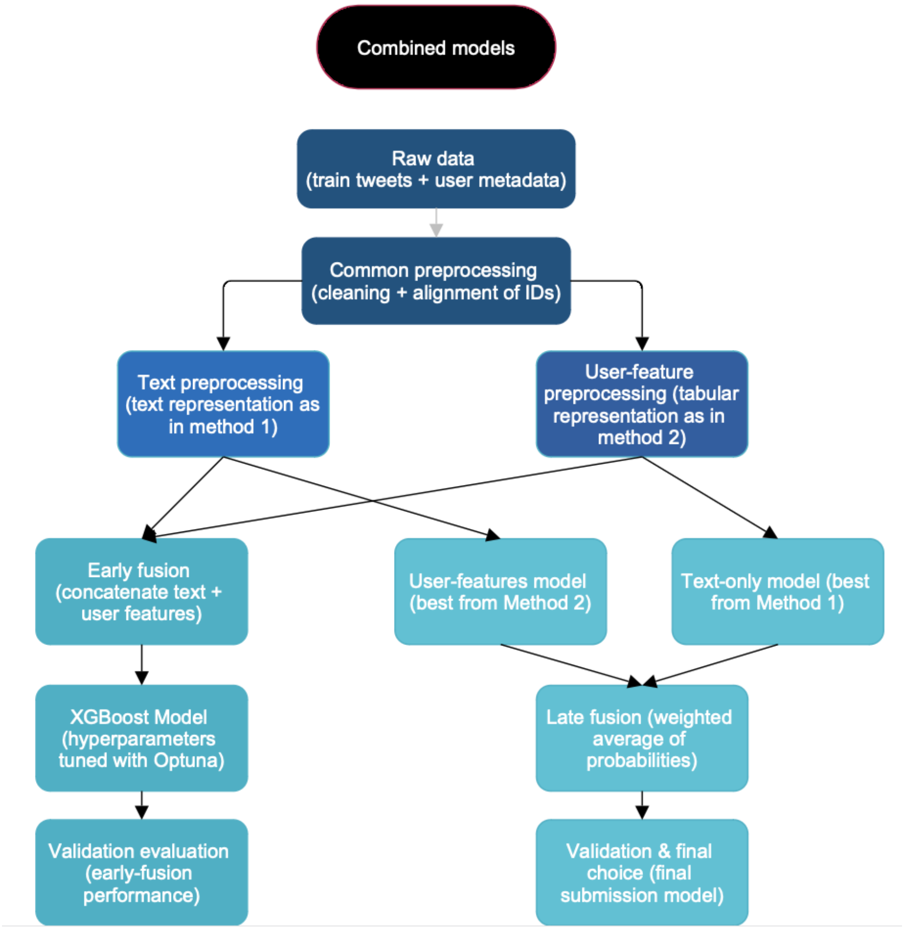
A.1 Method 1 - Text-only


Text-only models
Raw data (train tweets + labels)
Stratified train/validation split
TF-IDF representation (words + characters)
Contextual representation (pretrained BERTweetFR)
Train linear classifier (on TF-IDF features)
Fine-tune transformer (for binary classification)
Validation evaluation (accuracy, comparison)
Select best text-only model and apply to test set

A.2 Method 2 - User-features-only


User-feature-only models
Raw data (user metadata + labels)
Cleaning function (user-feature preprocessing)
"dl-user-features" (user-only tabular dataset)
Numeric + categorical encoding (scaled/imputed features)
K-fold cross validation
Train gradient-boosted trees (XGBoost model)
Train categorical boosting (CatBoost model)
Ensemble & validation (select best user-feature model)

A.3 Method 3 - Combining text and user features


Combined models
Raw data (train tweets + user metadata)
Common preprocessing (cleaning + alignment of IDs)
Text preprocessing (text representation as in method 1)
User-feature preprocessing (tabular representation as in method 2)
Early fusion (concatenate text + user features)
User-features model (best from Method 2)
Text-only model (best from Method 1)
XGBoost Model (hyperparameters tuned with Optuna)
Late fusion (weighted average of probabilities)
Validation evaluation (early-fusion performance)
Validation & final choice (final submission model)

References

[1] Nguyen, D. Q., Vu, T., Nguyen, A. T. *BERTweet: A pre-trained language model for English Tweets.* EMNLP, 2020.

[2] Chen, T., Guestrin, C. *XGBoost: A Scalable Tree Boosting System.* KDD, 2016.

[3] Akiba, T. et al. *Optuna: A Next-generation Hyperparameter Optimization Framework.* KDD, 2019.

[4] Bonnier, T. et al. *Revisiting Multimodal Transformers for Tabular Data with Text Fields.* Findings of ACL, 2024.

[5] http://nlp.polytechnique.fr/bertweetfr

[6] https://huggingface.co/vinai/bertweet-base

[7] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., Gulin, A. *CatBoost: Unbiased boosting with categorical features.* Advances in Neural Information Processing Systems (NeurIPS 2018).

[8] Preprocessing Function : cleaning.py

[9] medium.com/@kdk199604/tabnet-a-deep-learning-breakthrough-for-tabular-data-bcd39c47a81c