University of Pisa

Data Mining and Machine Learning

Project

Deep Fake Tweet Detection

Bruchi Caterina
Bruni Davide

A.Y. 2022-2023

Abstract

Social media has been in our everyday life for several years and we know that they could be used to manipulate and alter the public opinion. This is possible with and without bots, i.e., computer programs that control a fake social media account as a legitimate human user would do: by “liking”, sharing and posting old or new media which could be real.
Moreover, the generation and sharing of deepfake multimedia over social media, tricking people into believing that they are human-generated, have already caused distress in several fields (such as politics and science).
Deepfake social media texts can already be found, though there is still no misuse episode on them, anyway in this work we want to discern between Tweets written by real human user and deepFake Tweets.
In the following pages we’ll compare our work with ”TweepFake: About detecting deepfake tweets”[1], the work that inspired us, and we’ll try to extend their work with more recent Tweets and we’ll try to understand if the results obtained by them are the best available.

Contents

Chapter 1

Data Pre-Processing

We obtained our dataset from the authors of the following paper "TweepFake: About detecting deepfake tweets"[1].
We could directly take their dataset which contained these features

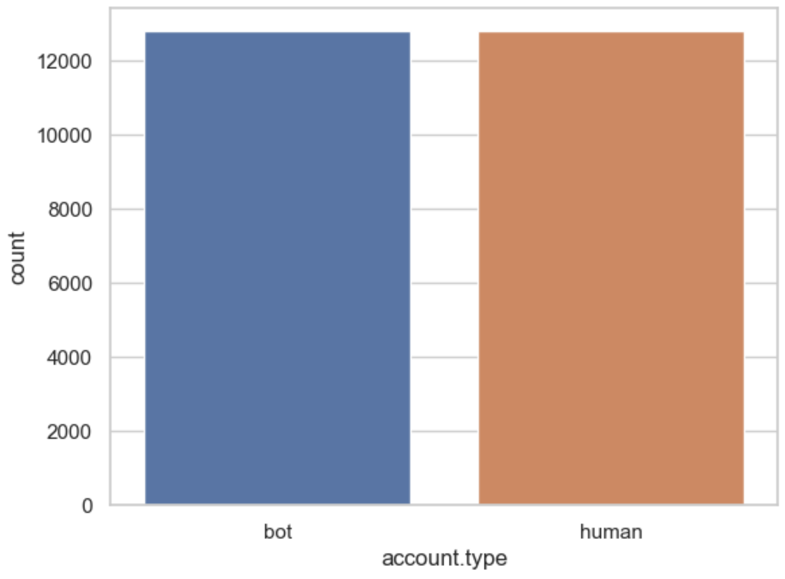
- *user id*
  The username of the twitter account
- *status id*
  The tweet id
- *text*
  The tweet: a message of at most 280 characters, which can contain hashtags and links
- *screen name*
  This is the username of tweet's author
- *screen name anonymized*
  The username is replaced by a string composed by the account type and an incremental number
- *account type*
  The typology of the generative method used to write the tweet ('human', 'gpt-2', 'rnn' or 'others')
- *class type*
  This is the label used to classify a tweet as 'human' or 'bot' generated

For our classification task we actually needed only the text of the tweet and the account type just distinguishing if it was a bot or a human. Instead the paper also wanted to recognize the type of IA used to replicate the behaviour of the human account, so we just consider all the other types of bots as a unique category

The dataset was originally available already divided into train, test and validation set, so we had to unite them all together being careful not to have any duplicates in values or indexes.

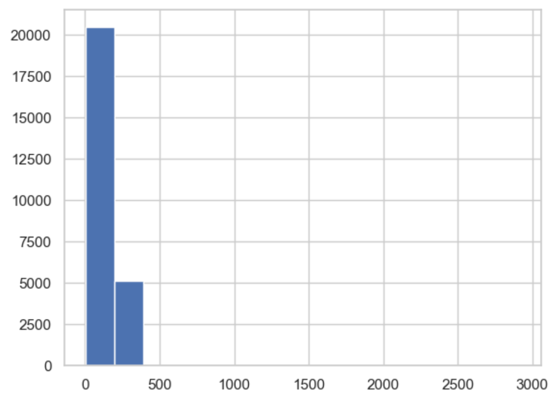
1.1 Dataset Analysis

Joining all the records together we had a total of 25572 instances, since the dataset was already used it was already balanced and the null values were already removed as we can see from the plot below.



But they don't come from an equal number of accounts, since some fake accounts imitate the same human profile, so in the end we obtained 23 bots and 17 human accounts.

We also checked the tweet length and verified that around 4/5 of the total tweets are below 250 characters long.



1.2 Data Cleaning

1.2.1 Duplicate removal

We had two other situations to fix:

1. We have discrepancy of unique text count and the unique status_id count. It means that we have different Tweets with the same id,this must be a mistake since each tweet has its own ID.

2. We have discrepancy between the number of unique text and the size of dataset. We have to check if this situation is due to the duplicated id or for other reasons.

Before investigating further we checked on the paper and found out that the dataset wasn't originally balanced and it was fixed by oversampling so that might be a possible explanation for what we just found out.

Duplicated IDs

Concerning the duplicated IDs we noticed some of them were actually double and related to the same user. Also the tweet text was identical too but they weren't detected as such because of some deprecated characters, so we first removed them manually and then proceeded to clean such charachters and check again for duplicates (this procedure will be explained in detail in the following paragraph)

Duplicated Tweet content

There were 2 couples of identical tweets, for each pair one belonged to the human and the other to the associated bot. In this case they didn't have

the same status id, they were different tweets, maybe the IA bot algorithm produced a perfectly identical output.
Since they are only 4 (2 human generated and 2 bot generated) on a total of about 25000 tweets, we decided to drop them.

1.2.2 Deprecated Unicode Characters

In particular we found out that some tweets had badly memorized the quoting marks and the apostrophe, and they were always substituted with the two following glyphs:

- ï¿½
- Ã¯Â¿Â½

So we just proceeded to substitute all of them with a single apostrophe.

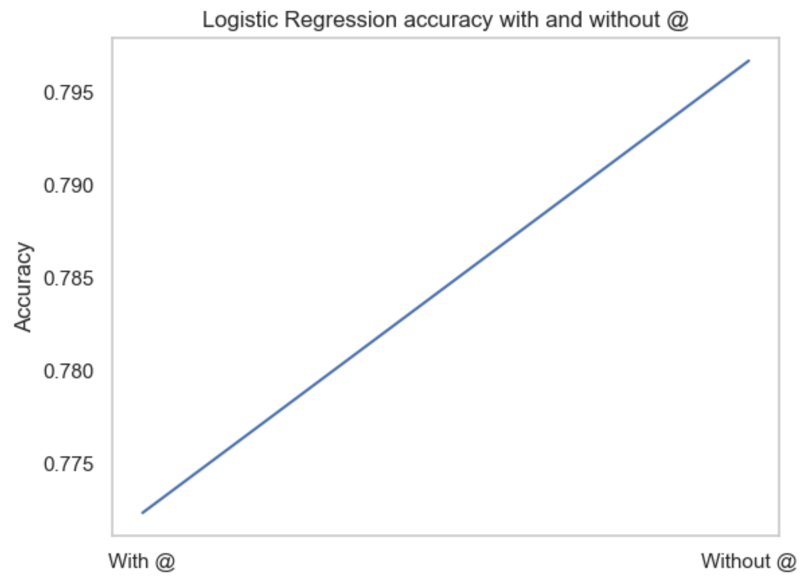
1.3 Data Trasformation

Before working on our dataset we still had some work to do, we needed to remove some information that in text classification are potentially non useful, which are tags, hastags, and urls.

To choose which of them mantain and in which form we performed some test to see if they presence or absence improved somehow the performance of the Logistic Regression Classifier. We choose to use him because it was the best trade-off of accuracy, speed and simplicity.

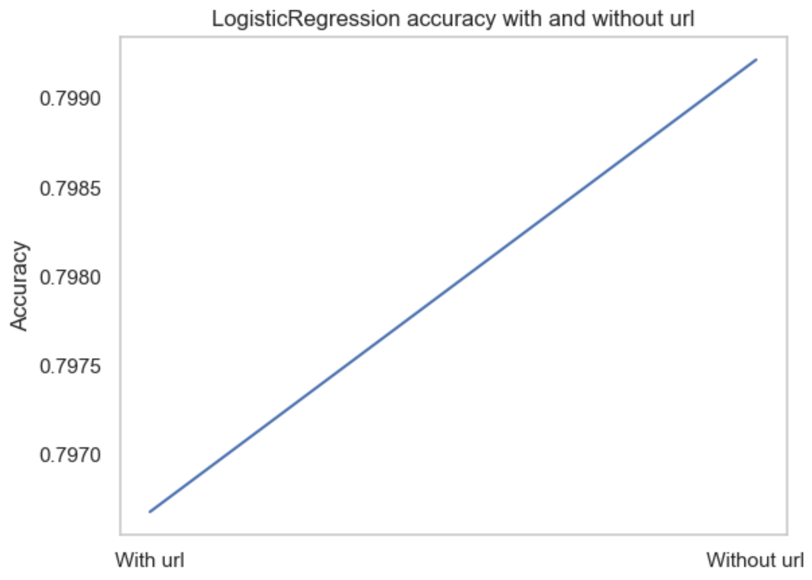
1.3.1 User Mentions (@username)

The debate here was whether to keep them or to substitute them with the token "user_mention", the results we obtained are this:



So since the accuracy is about 3% higher with them we could decide immediately to remove them without further analysis.

1.3.2 URLs

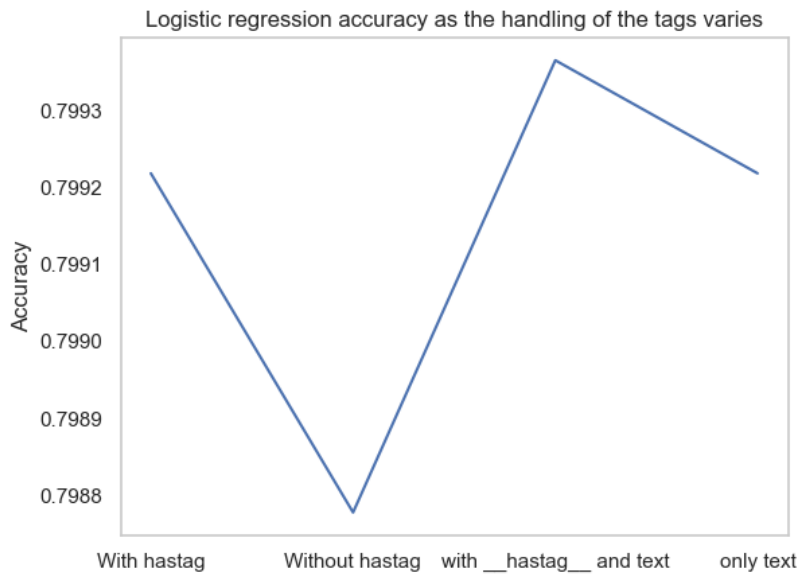


For the url instead the difference in accuracy of the Logistic Regression classifier was minimal, so we analyzed the Wilcoxon Test result: the two cases were actually statistically similar, leading us to decide to replace them with "__url__", which is the simpler solution because allow us to save less information

1.3.3 Hastags (word)

For the hastags the situation was a bit more complicated we had 3 different option depending on the amount of information we wanted to keep

- **"#twitter" → "twitter"**
  We replace it with just the "word". We keep the information on the content of the hastags but lose the information that they were contained in an hashtag
- **"#twitter" → "__hashtag__"**
  We replace it with just the token, as for the urls. We only keep the information that we had an hastag but lose the content completely
- **"#twitter" → "__hashtag__ twitter"**
  We replace it with the token and the "word", we keep the maximum amount of information possible.



The difference in accuracy amongst the option was very slight, so we performed statistical significance test, Wilcoxon, finding out that there are differences between using the dataset with or without hastag, but there there are no differences between using the dataset with original hastag and using it with the token __hastag__. We would use the original dataset, since, in the last phase of our work, it will implicate one less pre-processing step to apply to eventual new data we want to analyze

1.3.4 Text Elaboration

As we know to correctly extract information from text, we need to use the Bag of Words Method, which is a general platform to analyze case study for text classification. Furthermore we applied TF-IDF as weighting scheme for the representation.
Two aspect we focused on are:

- Filter the Stop Words
- Stemming

We have also to extract a vocabulary of relevant words and remove the ones which are too rare or too common: this procedure is analyzed in the chapter 2.

Stop-word Filtering

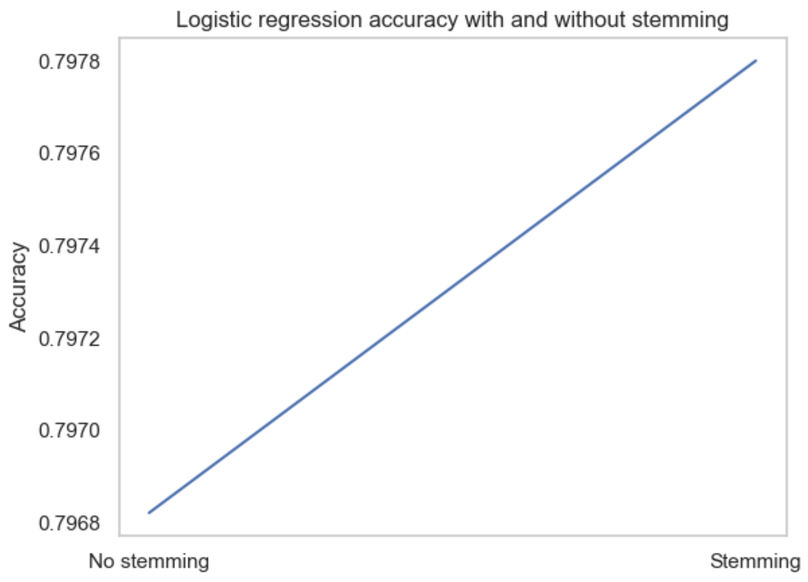
Stop Words are articles, conjunctions, prepositions, pronouns, words appearing very often in a particular language or in a context (respectively language-specific and domain-specific stop words).

For their grammar role they give little to none information and so can be considered as noise and have to be removed. So for all the analysis of thus project we removed them.

Stemming

When dealing with text we might want to make the system able to recognize words ignoring their declination, to do so we could use stemming, which is the process of reducing each word to its *root form* by removing eventual suffix.

In particular we had a whole debate about whether to keep stemming or not. To make the best choice we considered which version performed better on Logistic Regression obtaining these results.



Since as we can see from the image the difference in accuracy is rather low we performed again the Wilcoxon test concluding that there weren't significant differences in the distribution, so we chose the option which required less preprocessing action and decided to not apply stemming.

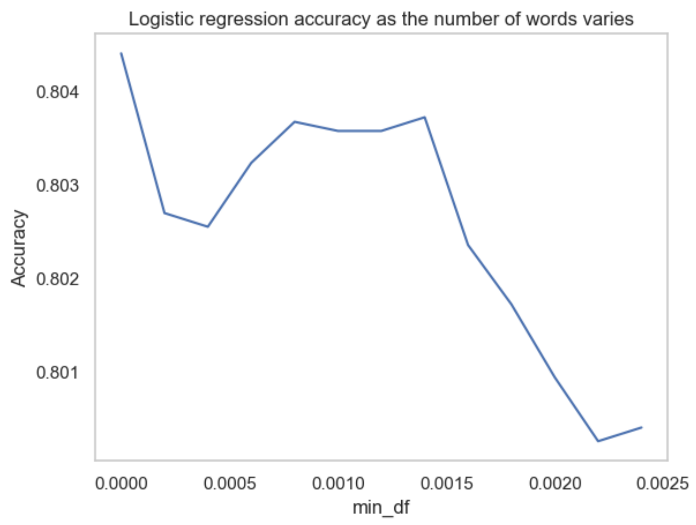
Chapter 2

Feature selection

Since we decide to represents text using the BOW technique, our features are the words (tokens). In our dataset, after the pre-processing phase, there are 27482 words, but how many if them are needed?
To answer this question we want to kept the words more frequent than a certain threshold *max_df* and under the threshold *min_df*. To choose the value of these two threshold we use a classification model (Logistic Regression) and we observe how the accuracy changes as the parameters vary.

min_df

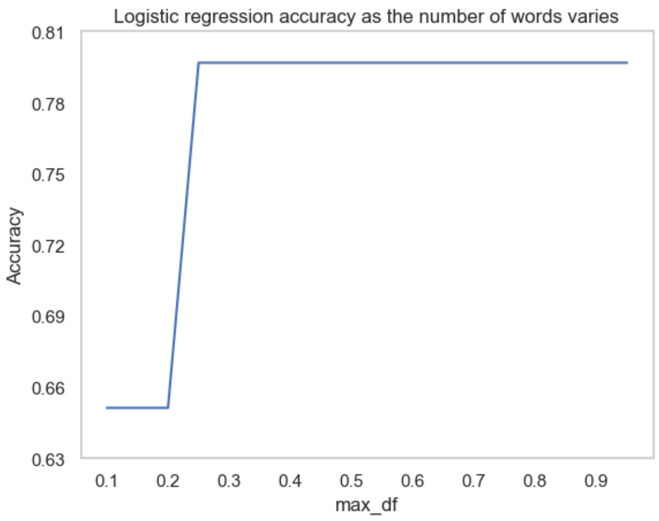
We use these threshold to ignore terms that have a document frequency strictly lower than the given threshold.



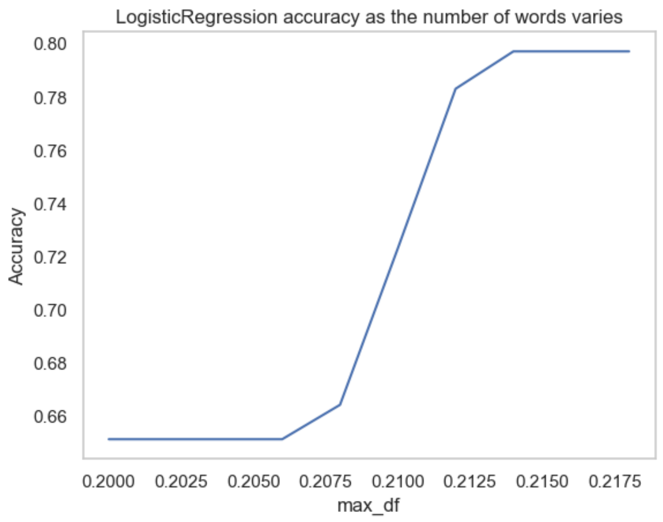
As we can see there are not big differences in term of accuracy as the parameter min_df varies: to choose which one is the best, we performed the Wilcoxon test with $\alpha = 0.05$. The Wilcoxon test indicates that there are no significantly differences in terms of accuracy, so we choose min_df = 0.012 in order to obtain the simplest model, since this parameters make us reduce the most the number of words.

max_df

We use these threshold to ignore terms that have a document frequency strictly higher than the given threshold.



Observing the results, we notice that we choose a too large step: in fact, we don't have feature with a frequency higher than 25%.
We reduce the step to observe the variation of results changing the number of features.



As we can see from the plot, the best value for the parameter max_df is 0.2126 (or 0.22 since there are no differences in terms of number of feature).

After set these two parameters, the number of feature is reduced to 1412: this is due to the high presence of meaningless terms in the original dataset.

Chapter 3

Result Comparison

We will now make a comparison between our result and the one of the original paper, the main differences in our two analysis regards not only the classification purpose but also the preprocessing steps since in the original analysis they replaced with tokens both urls, tags and hastags.

So their main focus was also to understand which fake tweets generator were better and more difficulty recognizable from detection algorithms.

3.1 Paper Results

They tested the difficulty in discriminating human-written tweets from machine-generated ones by evaluating 13 detectors: some of them exploiting text representations as inputs to machine-learning classifiers and others based on deep learning networks.

In the paper were used also Neural Network-based methods and word embedding technique, but we will exclude their performance for this comparison and just analyze the differences in results obtained with BoW methods with machine learning models.

In the end they obtained better results using neural networks and these results suggested that some technique used to generate deepfake text are very difficult to unmask also for expert human annotators.

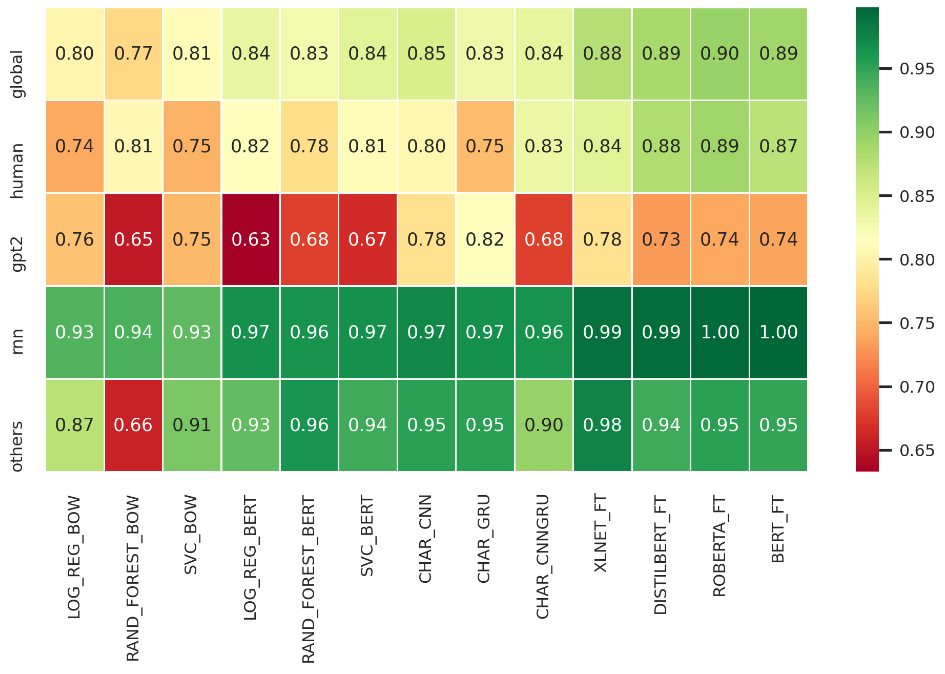
3.2 Comparison

3.2.1 Paper

These were the ending results

Method	HUMAN			BOT			GLOBALLY
	Precision	Recall	F1	Precision	Recall	F1	Accuracy
LOG_REG_BOW	0.841	0.749	0.792	0.774	0.859	0.814	0.804
RAND_FOREST_BOW	0.759	0.798	0.778	0.787	0.747	0.767	0.772
SVC_BOW	0.851	0.754	0.800	0.779	0.869	0.822	0.811
LOG_REG_BERT	0.846	0.820	0.833	0.826	0.851	0.838	0.835

This is the confusion matrix. For the purpose of our project we have to consider as a unique value all the different types of bot classes that we have here



	LOG_REG_BOW	RAND_FOREST_BOW	SVC_BOW	LOG_REG_BERT	RAND_FOREST_BERT	SVC_BERT	CHAR_CNN	CHAR_GRU	CHAR_CNNGRU	XLNET_FT	DISTILBERT_FT	ROBERTA_FT	BERT_FT
global	0.80	0.77	0.81	0.84	0.83	0.84	0.85	0.83	0.84	0.88	0.89	0.90	0.89
human	0.74	0.81	0.75	0.82	0.78	0.81	0.80	0.75	0.83	0.84	0.88	0.89	0.87
gpt2	0.76	0.65	0.75	0.63	0.68	0.67	0.78	0.82	0.68	0.78	0.73	0.74	0.74
rnn	0.93	0.94	0.93	0.97	0.96	0.97	0.97	0.97	0.96	0.99	0.99	1.00	1.00
others	0.87	0.66	0.91	0.93	0.96	0.94	0.95	0.95	0.90	0.98	0.94	0.95	0.95

3.2.2 Our results

The detailed final results are explained in the notebook attached, here we show again the confusion matrix on the test set for the 3 classifier used to give you a visual comparison with the one of the paper.

In the table below we have some statistical score.

	BOT			HUMAN			
	F1	Precision	Recall	F1	Precision	Recall	ACCURACY
Logistic Regression	0.78	0.85	0.71	0.81	0.75	0.87	0.79
SVC	0.78	0.86	0.71	0.81	0.75	0.88	0.80
Random Forest	0.77	0.91	0.67	0.82	0.74	0.93	0.80

Random Forest

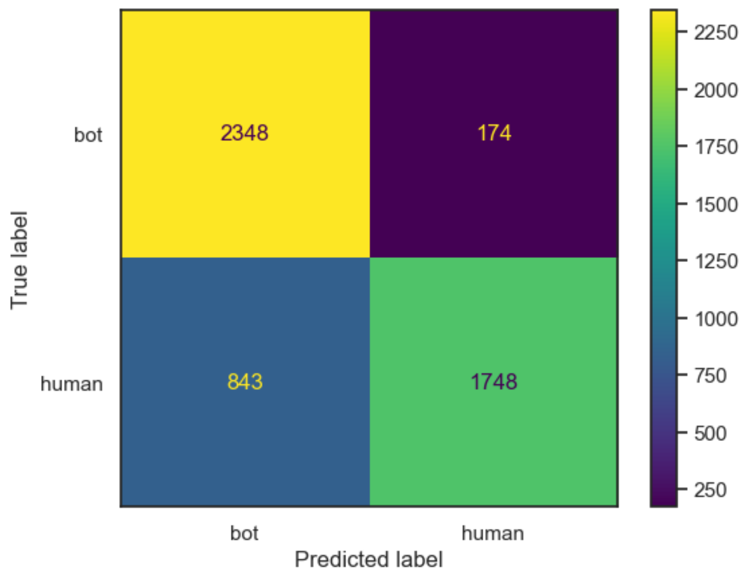


Figure 3.1: Confusion Matrix for the Test set

Logistic Regression

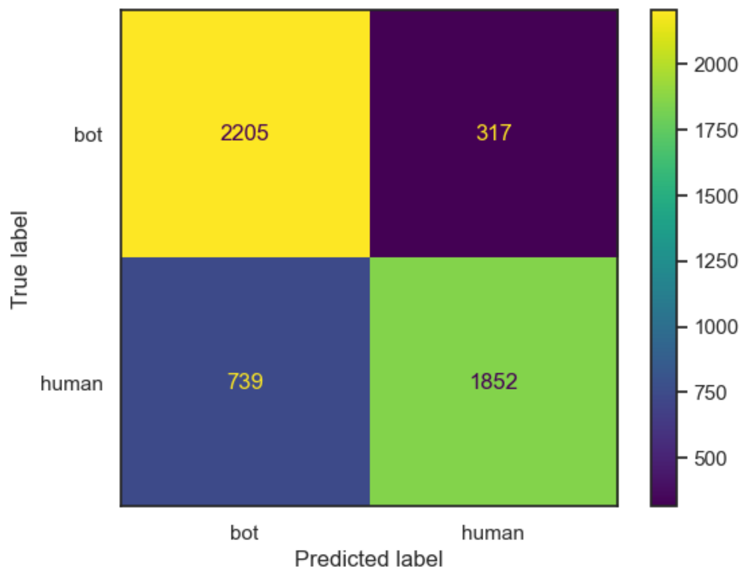


Figure 3.2: Confusion Matrix for the Test set

SVC

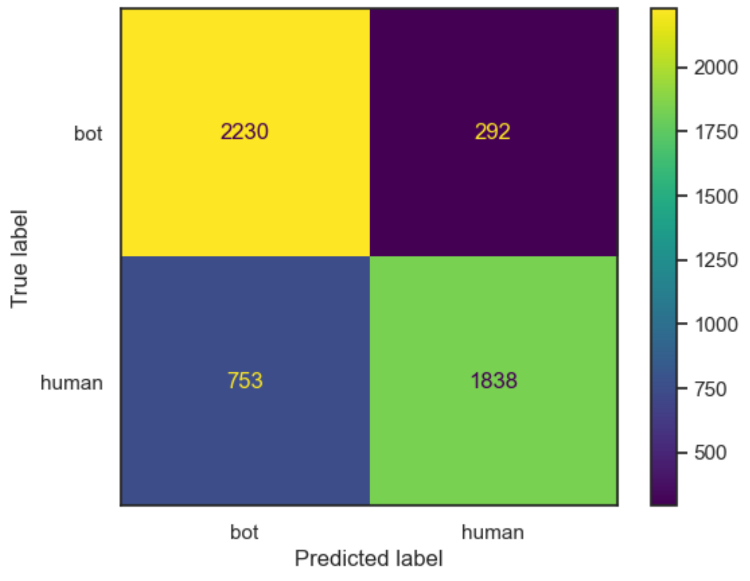


Figure 3.3: Confusion Matrix for the Test set

3.3 Comment

As you can observe from the above sections our results are very very close to the ones obtained in the original paper, but our classifiers are faster and lighter since we reduced the number of features of BOW to 1412, while in the paper such they use 25000 words.

Chapter 4

Model choice

4.1 Analysis on parameters chosen

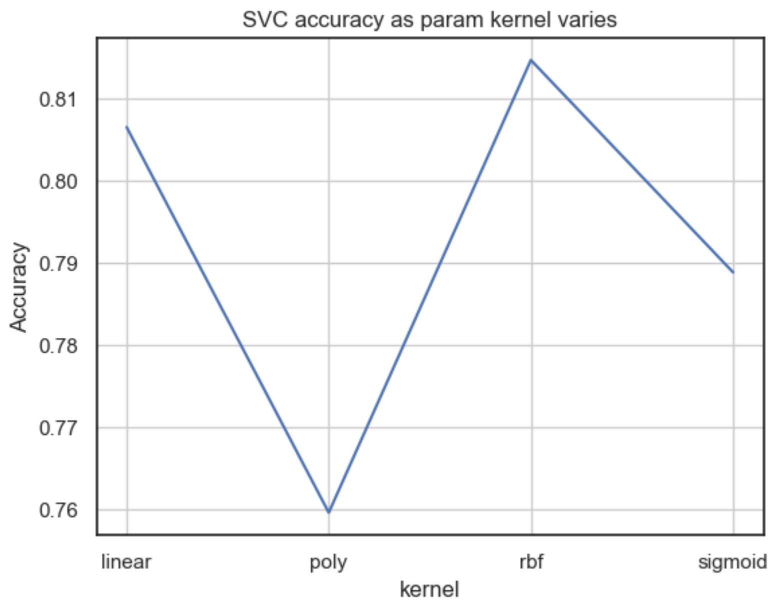
After obtaining about the same results in terms of accuracy of the paper, we ask ourself if the parameter chosen are the best available.

Logistic regression

In this case, the unique parameter is C, and C=1 is the better one due to the accuracy results and Wilcoxon test.

SVC

In this case we can choose between four type of kernel functions: Linear, Polynomial, rbf and Sigmoid.



Except for the case of polynomial function, there are no big differences in terms of accuracy, and Wilcoxon Test confirm that.
So the kernel function = linear is the better one.

Random Forest

In this case we have several parameter to test:

- min_samples_leaf, the minimum number of samples required to be at a leaf node.
- max_depth, the maximum depth of the tree.
- min_samples_split, the minimum number of samples required to split an internal node.
- n_estimators, the number of estimators.

For each of them we try different parameters, but every time there are no differences in terms of accuracy. Performing the Wilcoxon test with $\alpha = 0.05$ we always choose the simplest model.
To conclude, we compare our Random Forest with TweepFake one:

- min_samples_leaf = 4 (TweepFake: 2)
- max_depth = 30 (TweepFake: 30)

- min_samples_split = 50 (TweepFake: 15)
- n_estimators = 100 (TweepFake: 500)

The results we obtain are the following:

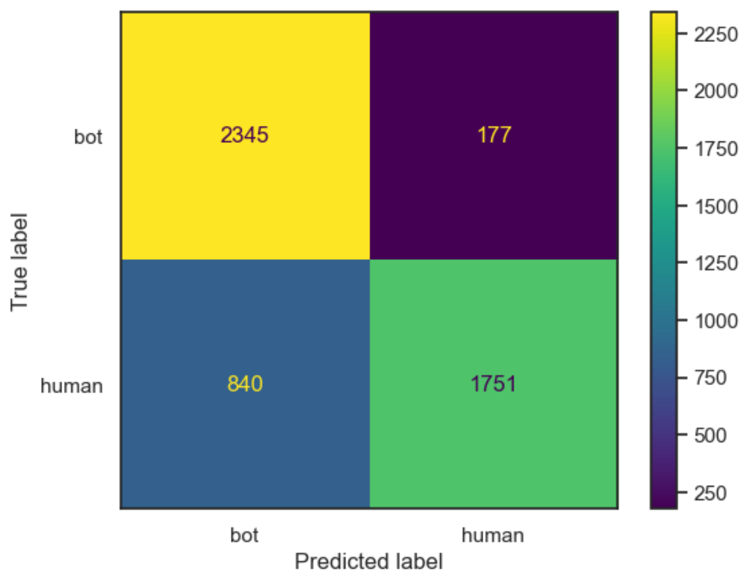
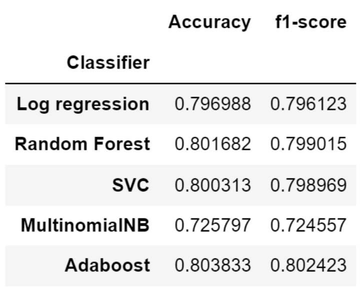


Figure 4.1: Confusion Matrix for the Test set

with an accuracy of 0.80.
Since the accuracy is the same, we think that these parameters are better than the ones choice by TweepFake, because in this way the model is lighter.

4.2 Introduction of new models

We try other models too: we try a Multinomial Native Bayesian Classifier and Adaboost with 100 estimators.
In the following table, the accuracy and f1 scores are reported.



Classifier	Accuracy	f1-score
Log regression	0.796988	0.796123
Random Forest	0.801682	0.799015
SVC	0.800313	0.798969
MultinomialNB	0.725797	0.724557
Adaboost	0.803833	0.802423

Figure 4.2: Comparison between different models

We can notice from the table above that the different models seems to have no differences in terms of accuracy and f1-score (except for the Multinomial Native Bayesian were the scores are the worst).
Since accuracy level are very similar, we try to improve it using a Voting Classifier.
We try two different combination for voting classifier:

- Random forest, SVC, Adaboost
- Random forest, SVC, Logistic Regression

In both case, we obtain worse results since the accuracy obtained is about 0.75, so we excluded the voting classifier models.
To choose among the other models. since the Wilcoxon test between Random Forest, SVC, Adaboost and Logistic regression shows no differences between them, we choose the simplest model: Logistic Regression.

Chapter 5

Data Stream

In the dataset we retrieve from the author of the paper ”TweepFake: About detecting deepfake tweets”, aren’t present the creation Timestamp. Before analyzing the accuracy of our model with recent tweets, we need to know the range of dates in which original dataset Tweets have been generated.
We need to rewrite the code of TweetedAt[1] for Python. TweetAt is a web service and library to extracts date and time from the tweet ID by reverse-engineering Twitter Snowflake[2].
In particular extracts the timestamps for post-Snowflake IDs directly from it and estimate timestamps for pre-Snowflake IDs.[2]

Dataset age

After retrying the Timestamp of every tweet, we find that we have 10 years of tweets, but the majority of them are from mid 2018 to mid 2020. For this reason we ask ourself if the chosen classifier works with the same results also with tweets from 2020-07-10 to 2023-01-10.
To identify eventual concept drifts, we classify tweets accumulated month by month.

5.1 How did we retrieve the tweets?

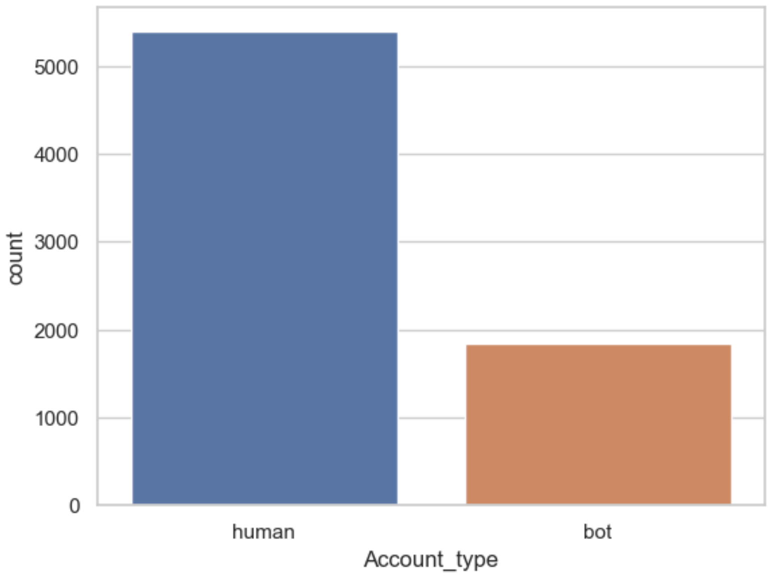
We use Twint, ”an advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles without using Twitter’s API”.

[1]https://github.com/oduwsdl/tweetedat

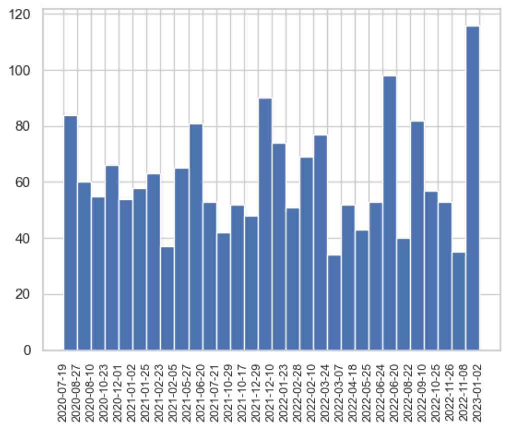
[2]More details on the inplementation of such estimation can be found here: https://ws-dl.blogspot.com/2019/08/2019-08-03-tweetedat-finding-tweet.html.

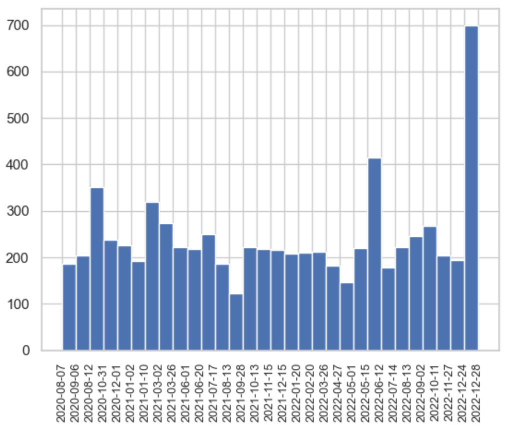
We retrieve tweets from all the users (both human and bot) we have in the original dataset: we obtain many features, but we are interested only in Tweet text, account type and creation timestamp.

Observing the totality of the collected Tweet from 2020-07-10, we can notice how they are unbalanced.



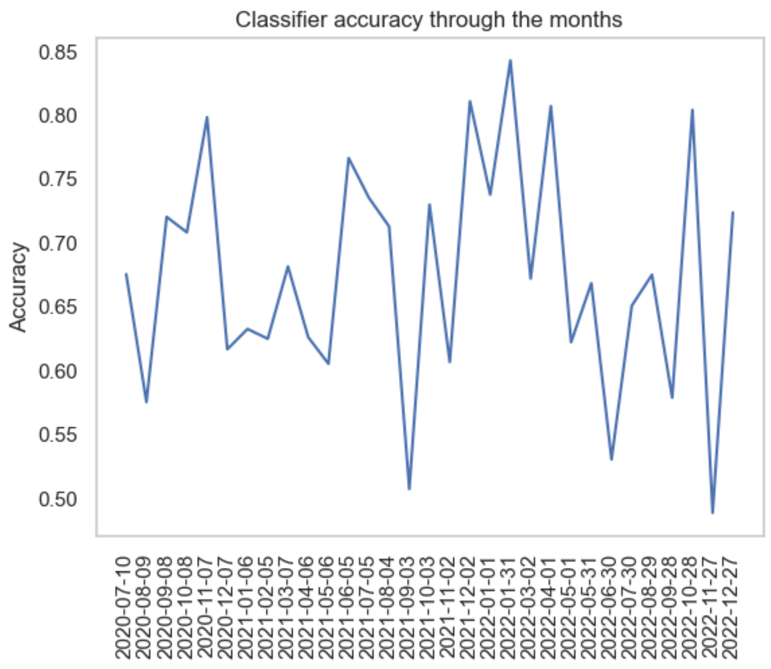
But, after all the distribution of human and bot tweet trhough the month are quite similar:





5.2 Logistic Regression Classifier

We try to use the Logistic Regression Classifier to classify the new tweets, even if we don't expect to obtain good results.



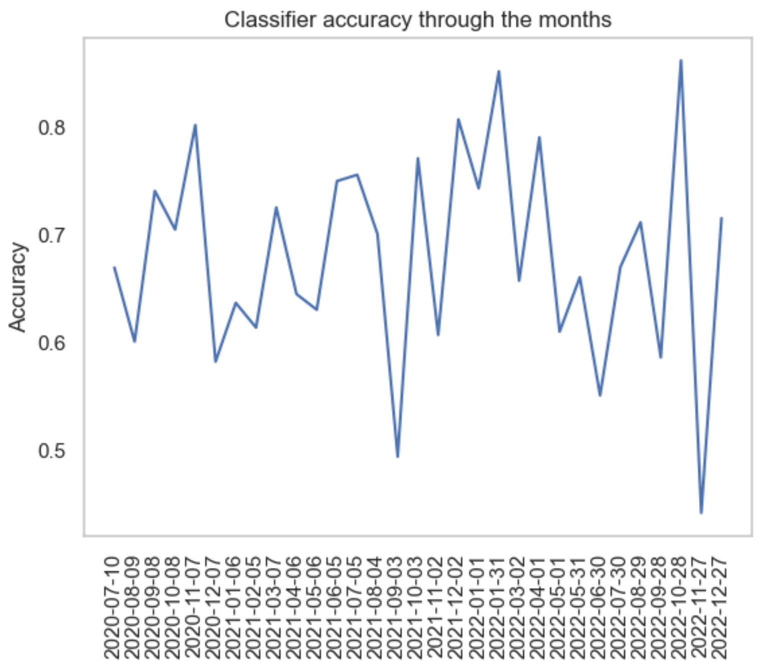
We can notice the presence of a kind of recurring concept drifting.

5.3 Ensamble method

After the previous poor results, we try to use one of the previous ensamble method we test even if they obtain worse results with the original dataset. Our ensamble method is a voting classifier with the following classifiers (each of them with the same voting weight):

- LogisticRegression
- Support Vector Classifier
- RandomForestClassifier

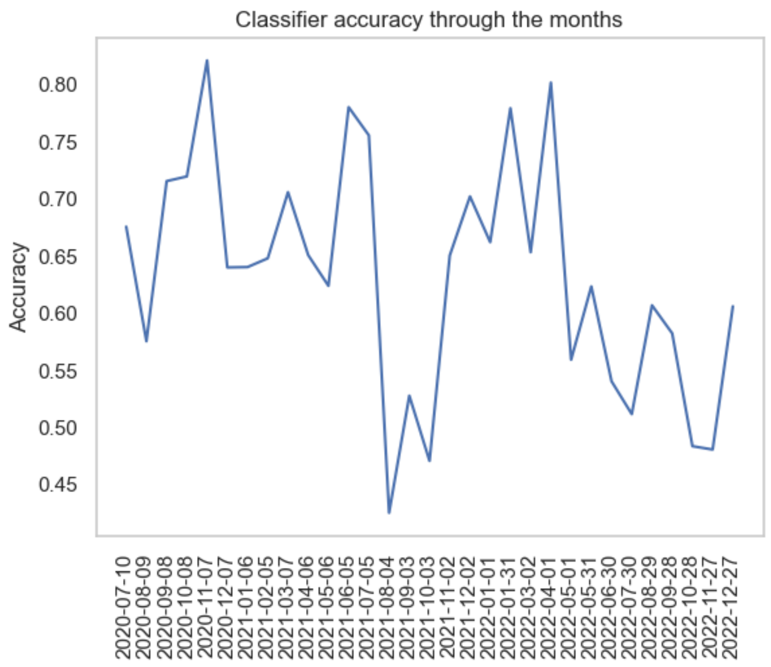
The accuracy we obtain:



Apparently we don't improve our results.

5.4 Retrain

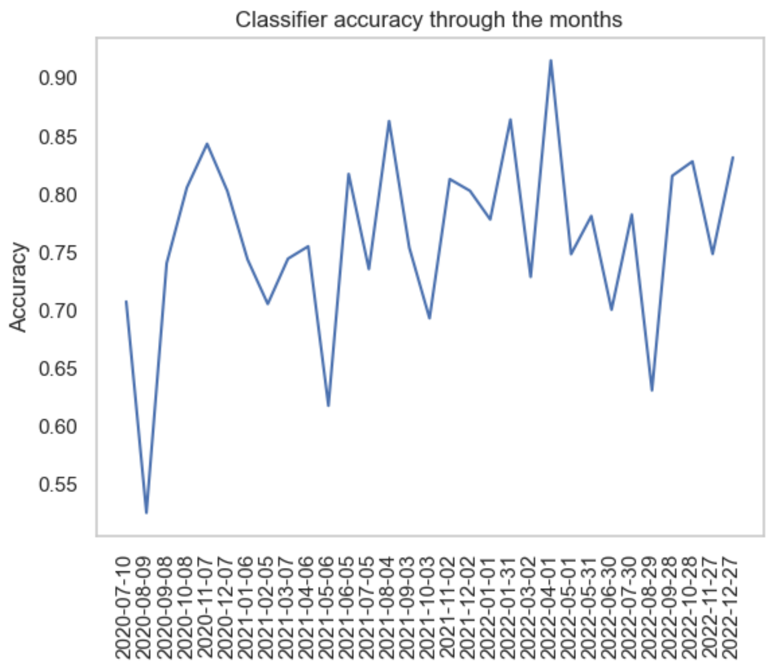
In this approach, the training set is extended with each new labelled chunk of tweets accumulated for each month and the whole classification pipeline is retrained from scratch.
Since we know that the newest tweets are unbalanced, we need to add SMOTE to the classification pipeline.



This approach don't give us better results and is very slow, since in the real environment, it need to be retrained once per month taking several seconds. If during the retraining phase we receive new Tweets need to be classified, the classification take more time.

5.5 Incremental

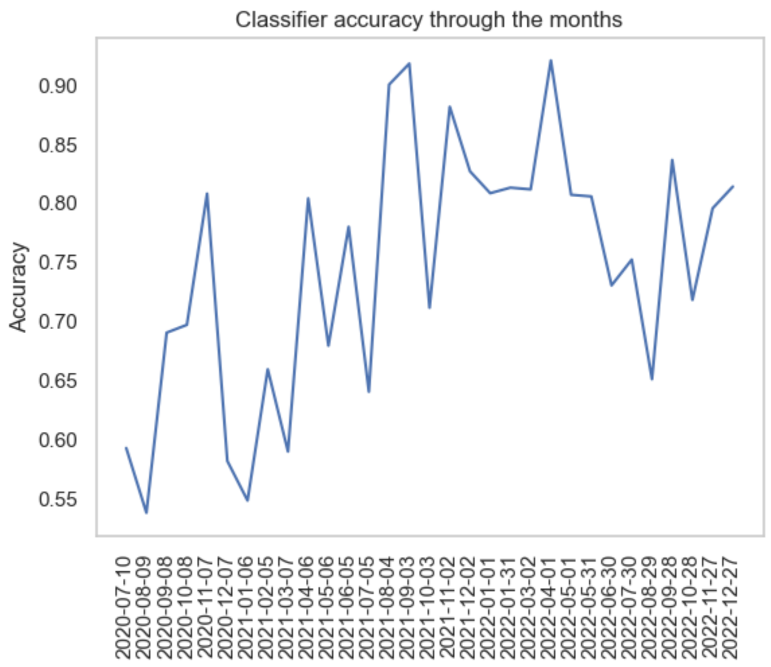
In this approach the classification model is updated based on the new labeled chunk of tweets with a partial fitting.
Only the classifier is updated, whereas the vocabulary generated based on the initial TF-IDF vectorization, is left unchanged throughout the online monitoring.
Since we need partial fitting, we are forced to change our classifier to use one that is suitable for partial fitting: given these constraints, we choose to use Native Bayesian classifier.



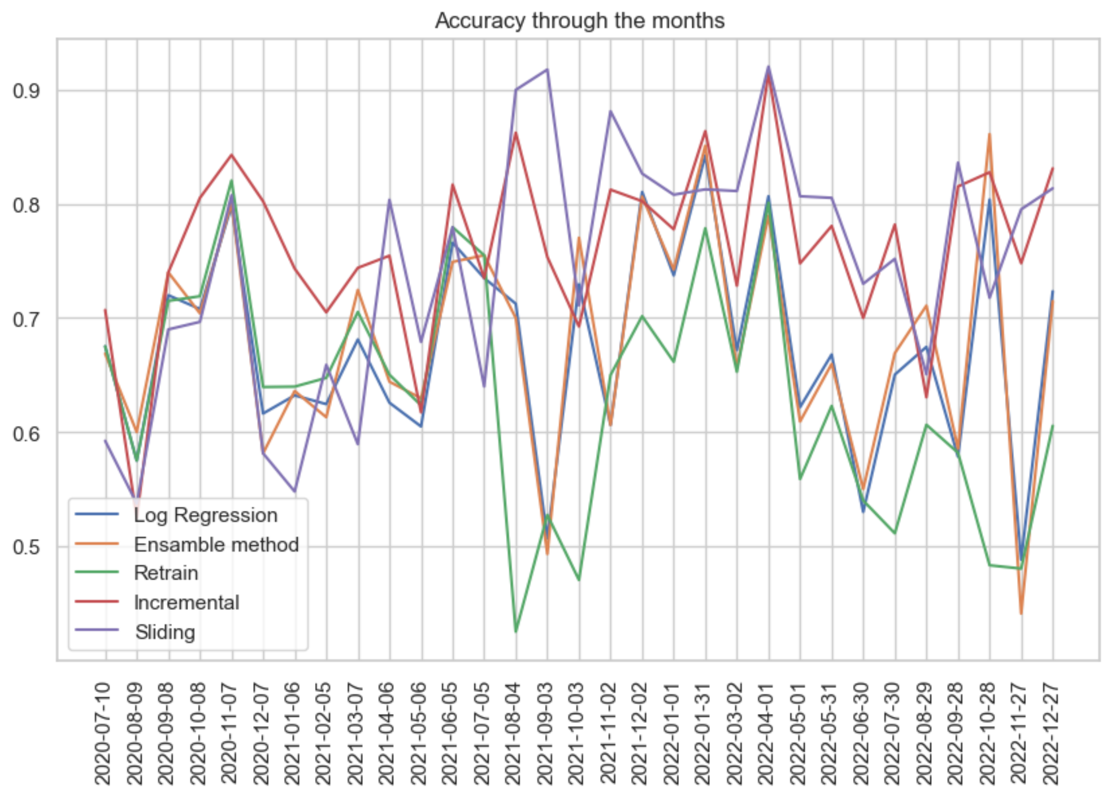
Concept drift still influence the accuracy, but resuts seem much better than the ones obtain before.

5.6 Sliding

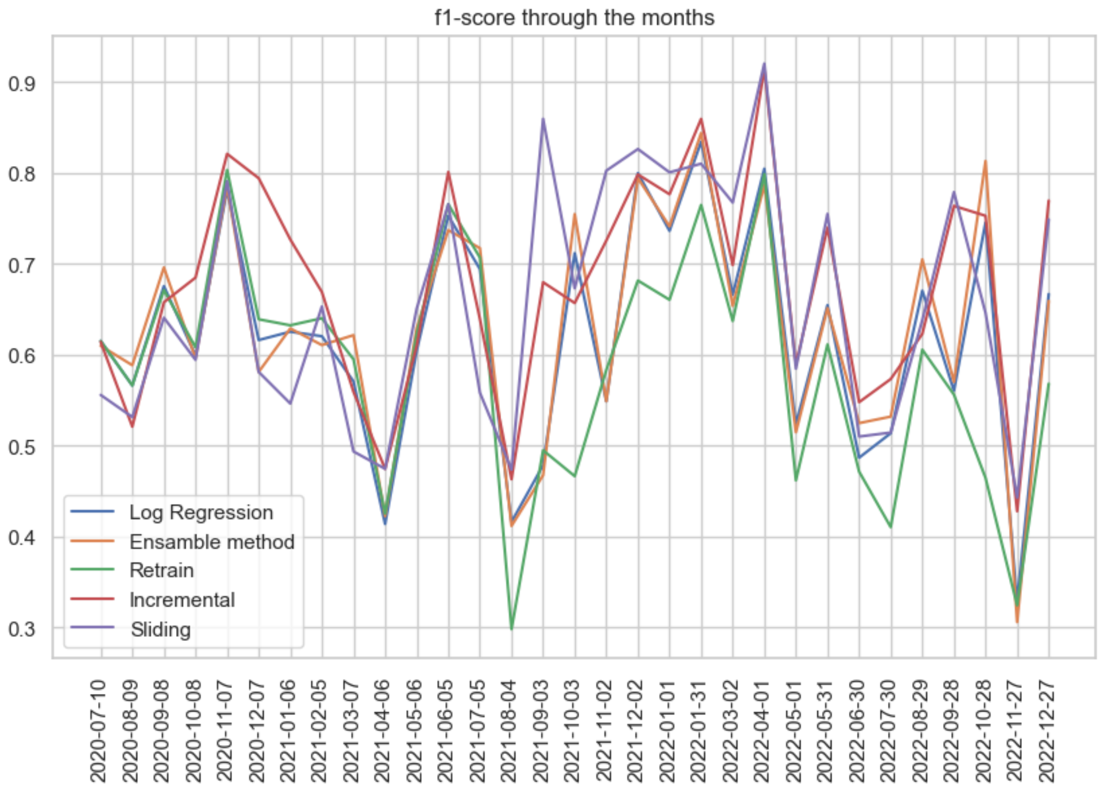
This approach is analogous to retrain, but oldest tweets are removed from the training set. Since we have Tweet from 2010, we choose to keep only tweets from '2019-01-06' as starting point, then add tweets of the recentest two months and remove the oldest two months.
Since we have unbalanced data, this solution could let us to some problems since newest tweet are much more less than the oldest ones. We try to deal with the problem by using SMOTE.


Classifier accuracy through the months
Accuracy
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0.90
2020-07-10
2020-08-09
2020-09-08
2020-10-08
2020-11-07
2020-12-07
2021-01-06
2021-02-05
2021-03-07
2021-04-06
2021-05-06
2021-06-05
2021-07-05
2021-08-04
2021-09-03
2021-10-03
2021-11-02
2021-12-02
2022-01-01
2022-01-31
2022-03-02
2022-04-01
2022-05-01
2022-05-31
2022-06-30
2022-07-30
2022-08-29
2022-09-28
2022-10-28
2022-11-27
2022-12-27

5.7 Comparison



Since we have unbalanced dataset, we want to compare also f1-score results.



To make these comparisons more readable we choose to rank the classifiers, based on the value of their accuracy month by month. In case of a tie finish, the rank follow these order:

1. Incremental
2. Sliding
3. Log Regression
4. Ensamble
5. Retrain

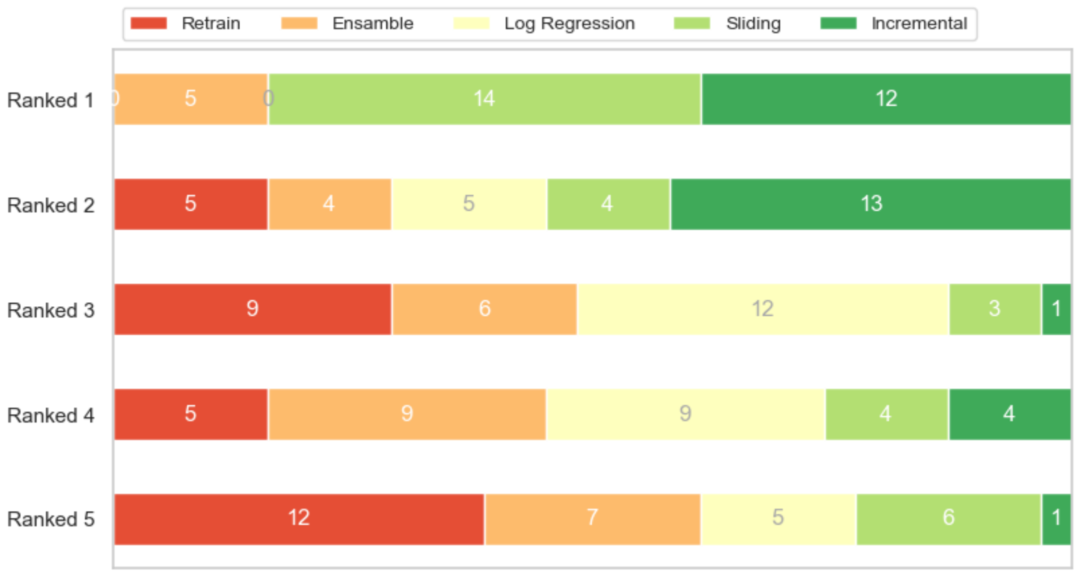


Figure 5.1: Classifiers accuracy rank

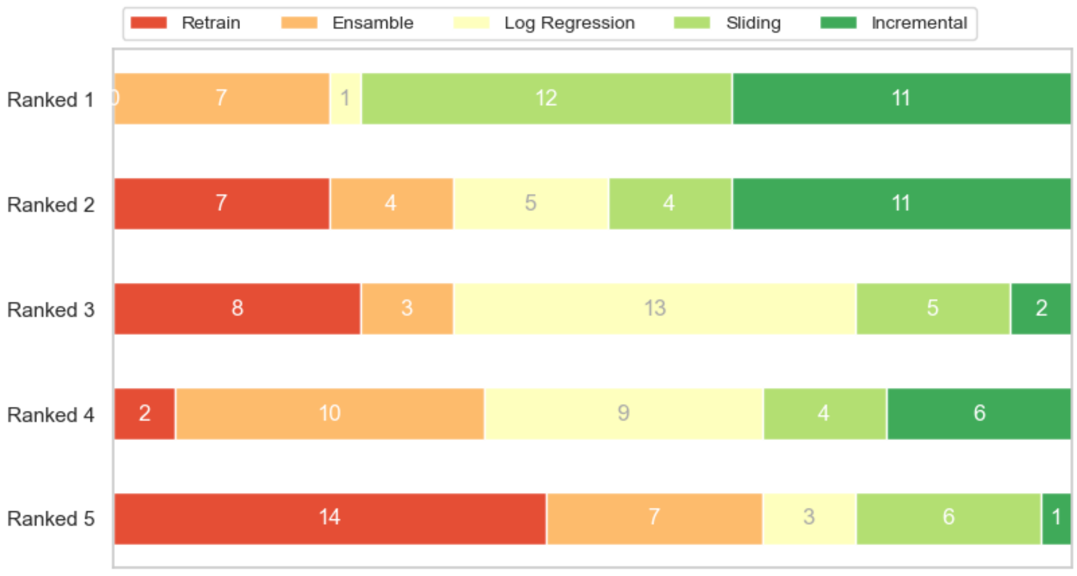


Figure 5.2: f1-score accuracy rank

Based on the two ranks, sliding and incremental appear as the better techniques, with sliding as the best.
These result could be due to an over-training problem, since several account are no longer active and we have a lower diversity of accounts.

Chapter 6

Test on tweets of unseen people

What if we want to classify tweets of new people?
Since we don't find any new known bot user imitating a real human profile, we first collect real Tweets, then we generate new Deepfake tweets using https://tweethunter.io/generate-tweets.
We obtain a total of 192 Tweets equally divded between human and bot generated.
We obtain the following results:

Technique	Accuracy	f1-score
Log Regression	0.77	0.77
Incremental	0.61	0.57
Sliding	0.55	0.49
Ensamble	0.81	0.81

We actually expected such results: because we don't fit again the ensamble model and the single classifier on the new scraped data, so probably the other two model are in a overtrain situation.
Hence it's normal for the sliding or increment refitting to lead to overtraining; this is explained also by the fact that the models are trained on actual people, so it's normal for them have poor accuracy and f1-score if the topic of discussion change drastically or, as in this case, the people who write the tweets changed.
For instance if our purpose was to monitor always the same users in a prolonged period of time then incremental and sliding method would be suitable.
Considereing instead our actual needs which are to have a generic classifier then ensamble and log regressionr results are still quite good; still we can't actually choose the "best" classifier since their performances on unseen people are evaluated only on Tweets from 2022-12-27 until the end of January

2023. We can't perform ulterior scraping because we already retrieved all the available tweets but to make an accurate comparison we would need to evaluate the accuracy of the model on tweets of the same account on more months to analyze the behaviour trough time also for the "new accounts" since the correct only one months could be luck (due for example to a coincidence of topics).

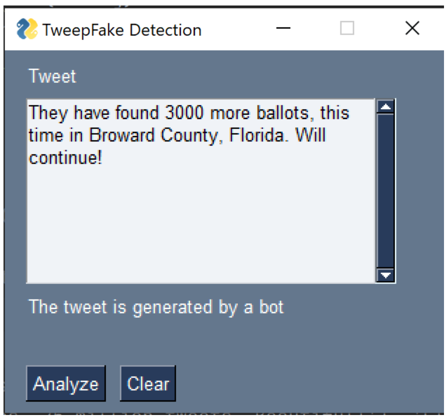
Chapter 7

Application

We implement an application with a GUI that, given a Tweet as input, it gives as output the prediction of the classifier.
Since we would have the most general possible classifier, we decide to use the Voting classifier with Random Forest, SVC, Logistic Regression.

7.1 Explanation

The application takes the classification model as a pickle file from the disk. Then it takes the Tweet as input and predict the results.
When the application is closed, all the Tweets with their prediction are saved in a csv file.



They have found 3000 more ballots, this time in Broward County, Florida. Will continue!

The tweet is generated by a bot

Bibliography

[1] Tiziano Fagni, Fabrizio Falchi, Margherita Gambini, Antonio Martella, and Maurizio Tesconi. Tweepfake: About detecting deepfake tweets. *Plos one*, 16(5):e0251415, 2021.

[2] Twitter Inc. Announcing snowflake, 2010.