SIADS 694/695 Team Project Report

Twitter Bot Detection

Team members:

Rania El-Shehety
Alexander Levin-Koopman
Jeffrey Olson

May 30, 2022

Summary and goals

We used a data set of Twitter account features and tweets to construct a classifier for determining whether accounts are human or automated (bot). Our classifier used supervised and unsupervised learning elements, and the principles of natural language processing and semantic analysis. Our finished classifier was able to identify accounts with accuracy exceeding 85 percent.

Twibot-20

Our team was able to get access to Twibot-20, a novel set of Twitter account data which is not publicly available. However we were granted access as long as we don't post it publicly, and it is used for academic purposes only. A description of Twibot-20 and its creation can be found in this 2021 preprint, by S. Feng et al. The dataset contains information from more than 200,000 Twitter accounts. The main features of interest for our project are:

1. A set of account features, drawn mainly from the Twitter API;
2. A capture of up to 200 of each account's most recent tweets;
3. On some accounts, around 11,000, a label of whether the account is human or "bot."

A bot account is one run via automation, and that needs minimal human intervention. Bot accounts serve a wide variety of purposes, including entertainment, announcement, advertising and promotion, manipulation of markets and elections, and spreading (mis)information. Most bot accounts do not self-identify as such; in fact, their success often relies on their ability to "pass" as human, which makes their detection challenging. Although the full account of the Twibot-20 research team's methodology is worth reading, in brief the labeling process was outsourced to human reviewers using a predetermined set of criteria, with a multiply redundant confirmation process. About 11,000 of the accounts in Twibot-20 are labeled; roughly half the labeled accounts are human. Much of our project focused on this part of the dataset.

Overall architecture of the learning model

Our approach to the creation of a learning model ended up being quite complex. Ultimately, we used the labeled data to train an ensemble of learning models, but the process included several isolated steps which employed their own unique approaches.

First, we engaged in early-stage processing of various kinds to create signals that would feed into the late-stage ensemble models. In one of these, NLP deep learning methods were used on the tweet capture of each account to train a model that would provide an initial prediction of whether the account was human. In the other, semantic methods were used (again on the tweet capture) to analyze each account's distribution of topical interests. Details on both are provided below.

Joining these signals with a large group of other more basic features that were extracted from the data set in various ways. These include things like age of the account, number of followers, sentiment analysis, and statistical measures of the tweet capture. Again, see below for more detail.

Early-stage processing

Supervised deep learning

For one of the early-stage signals, a neural network was trained using labeled data for an "initial prediction" of whether an account was human. The data used was derived only from the tweet capture associated with the account, and none of the account features.

In more detail, the tweet capture (a list of strings in the dataset) was joined into one long string for each account. A tf-idf vectorizer was trained on the entire training set, and then used to convert each capture into a weighted vector representation. A neural network with two hidden layers of 100 nodes each was constructed and trained with these vectors, using the human/bot labels. During the training a callback was in place, which allowed the neural net to simultaneously check loss and accuracy on a validation set; the training was halted as soon as overfitting was detected.

When fed new vectorized tweet captures, the trained neural network outputs a probability that the account is bot. This value is passed to the next stage of the learning model, discussed below.

The TensorFlow/Keras library was used to conduct this part of the project. In the initial approach, the individual tweets (instead of the combined total tweet capture) were used for training. Although the model could rapidly achieve accuracy upwards of 90 percent on the training set, validation sets rarely scored better than 65 percent -- the approximate accuracy on the training set after one epoch. In hindsight, it seems unrealistic to expect that a single tweet contains enough information to reveal whether it belongs to a human account.

Topic analysis

The goal of the topic analysis was to create a feature that captured the number of topics a user tweeted about according to the tweet capture. Our conjecture was that this spread of topics for a user would indicate a bot if the spread was narrow and a human if the spread was broad. To achieve this the first step was to preprocess the data. A few approaches were used for this, one was to only remove english stop words, another was to use regex to find words in the tweets using '\w\w+', this captured all word characters of two or more, then removing stopwords and stemming using nltk PorterStemmer. The third approach was to use the large spacy model then lematize the words if they were not stop words and only contained letters. This can be seen in section 1 of the extract_topic_distribution notebook. After this preprocessing the result was passed into a tfidf vectorizer using an n_gram range of 1 and 2, then fed to the model. The first approach created a massive and sparse matrix which created difficulties for model convergence, the second and third approaches created a slightly smaller but still sparse matrix. The stemming approach was significantly faster than the SpaCy approach but did not have the added benefits like name entity recognition. Section 1.2 extract_topic_distribution notebook.

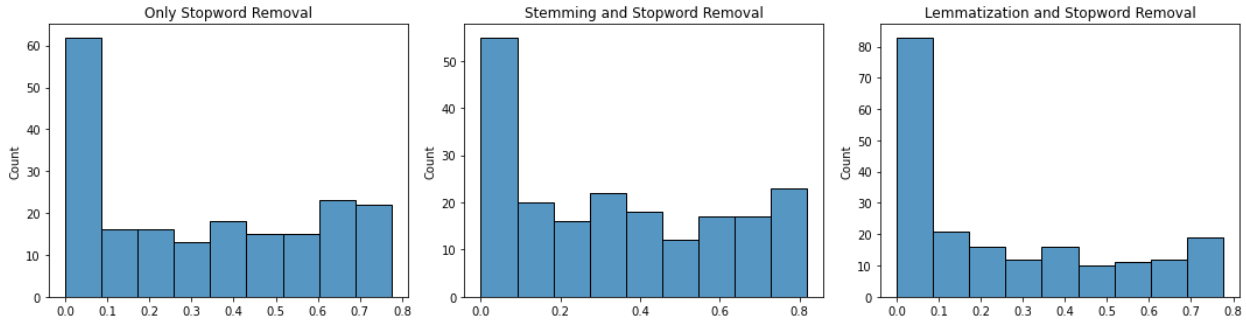
This initial approach had its difficulties; first, we tried to use Latent Dirichlet Allocation on the entire tweet dataset by combining each user's tweets into a single document or row in the matrix and trying to extract all of the topics in the dataset. The initial thought was to fit then transform the training data, extract the topics then transform the test set. One issue with this is the computation time. With matrices of this size the convergence time for LDA would take entirely too long, second if there were topics in the test set that were not seen in the training set they would be lumped into an "other" category by the model.

The second approach was to explode the tweet dataset creating a matrix with one tweet per row keeping the ids of the users as the index. This created a matrix of almost 2 million rows. This was then fed into the Spacy preprocessing function then the vectorizer and fed into a Non Negative Matrix Factorization model. This model performs better on small documents like tweets and is a bit faster than LDA. However this too ran into trouble converging because of the size and sparsity of the matrix and wouldn't converge.

The third approach was to take the matrix of a user's tweets one user at a time and fit a NMF model on their tweets to extract the topics per user. This created another problem as we had to input the number of topics we wanted the NMF model to extract. With these topic extraction models if we input 2 topics to extract it will find 2 and if we tell it to extract 10 it will find 10. This final approach abandoned the topic spread as being a number out of total topics and instead focused on the shape of the distributions output by the model. The input matrix was a matrix of a user's tweets lemmatized and vectorized with one tweet per row. This was fed into the NMF model to extract the number of topics equal to the total number of tweets. The output of the NMF model was the matrix of topics and words H and the tweets and topics W. We then took the W matrix and took the argmax of each row. Section 2.1 extract_topic_distribution notebook. The resulting list became the topic captured by each tweet. This distribution of topics was then normalized by the number of tweets and the final generated features were descriptive statistics of the distribution. These include the skewness, kurtosis, standard deviation and the gini

correlation. This last one measures the dispersion of the distribution where the other three measure the peakedness and the spread. Below is an example of one of the outputs of the nmf model and the topic distribution of the users tweets.

	only stopword removal	with stemming	with lemmatization
skewness	0.217854	0.270507	0.664794
kurtosis	-1.415758	-1.306865	-0.975609
standard deviation	0.261645	0.26982	0.258526
gini coefficient	0.460829	0.456085	0.571356



This can be seen in Section 2.2 extract_topic_distribution notebook.

This final approach was able to converge faster even though it had to be calculated for each user, and was used with the Spacy lemmatization preprocessing as this gave a distribution that was able to extract acute and separable topic spreads.

Feature extraction

In addition to the advanced analysis techniques just described, we also employed more basic methods to extract features which we thought might be useful in assisting the learning model distinguish human and bot accounts.

The following lists the features which were developed and added to the training set at this stage:

1. Standard statistical measures of the tweet capture: minimum, maximum, and average tweet length, as well as standard deviation in tweet length.
2. Aspects of user and screen names: their length, and the number of distinct characters appearing in the user name. (An automated process for account creation may use randomized character assignments, leading to more distinct characters.)
3. Whether the accounts were “verified” by Twitter. According to Twitter, this is used to indicate that “an account of public interest is authentic.”
4. Whether the account has a URL associated with it. Twitter accounts may feature a URL as part of their profile, but many choose not to.
5. Measures of engagement: how many followers, how many friends, how many “favourites”, number of statuses for each user and the listed count.
6. The age of the account (in days before May 1, 2022).
7. The length of the description of the user’s profile.
8. The average sentiment per user. We did this by using the nltk package in python where we used the sentiment module that takes a sentence as an input and outputs its sentiment as a number that’s negative if the sentiment is negative and a positive number if sentiment is positive. We then average the values across the user’s tweets to get the average sentiment per user.
9. Tweets similarity per user.
10. The number of links in a user's tweets is normalized by their total number of tweets. This becomes the average number of links per tweet.
11. The number of mentions normalized by a user's total tweets. This becomes their average number of mentions per tweet.
12. The number of retweets normalized by the user's total number of tweets. This becomes the ratio of retweets to tweets by a user.
13. The total number of tweets per user.

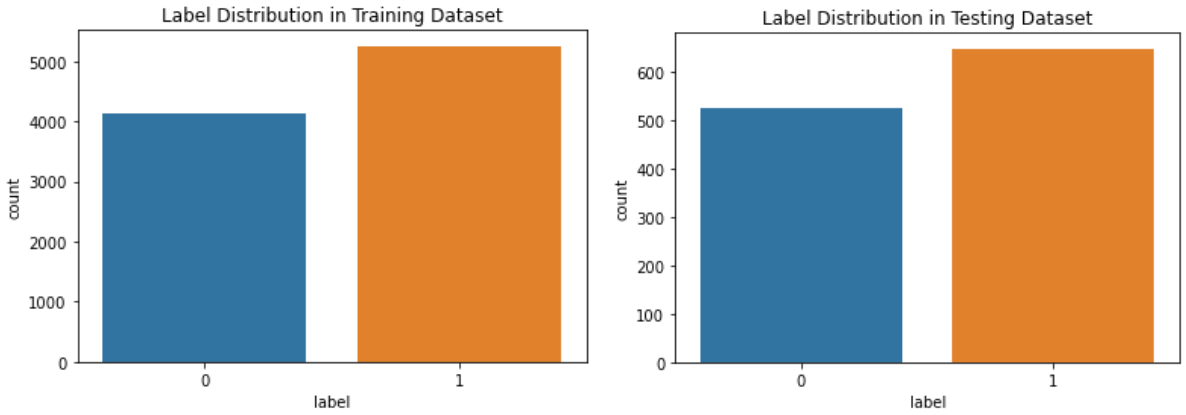
Our hypothesis is that a user that writes tweets that are similar to each other is a bot; however if the user writes tweets that aren't similar to each other, then it's a human. We will test our hypothesis after generating the similarity feature when doing the Exploratory Data Analysis on the final dataset.

In our journey to calculate the similarity between the tweets we first planned to train our word2vec model, since we have a large dataset of approximately three million tweets. However, training such a model needed high computational power (for example, GPU) and would take a very long time. That's why we decided to use a pretrained model since there are many available online. We planned to use the BERT model in the sentiment analyzer package in python, however it had the drawback of outputting an error if the word wasn't included in the trained model vocabulary. We could have solved this problem by adding a try except clause in our code, that fills the vector of the word that isn't found in the vocabulary with zeros. However, we decided to use a pretrained model in the spacy library that doesn't have this problem, instead, it produces a vector of zeros by itself for the words not in the vocabulary of the trained model. In addition to that, the spacy model has the advantage of taking a sentence and generating the word2vec for it, instead of taking each word and then averaging their vectors to get the vector representation of the whole sentence. After producing the vector representation of each sentence for a certain user, we append the vector for each sentence in a matrix until we have all the vectors for all the sentences of each user. After that we calculate the cosine similarity for each matrix and therefore have an average similarity score for each user. Finally, we append the similarity scores for each user in the users dataset that we will use to feed the supervised machine learning model.

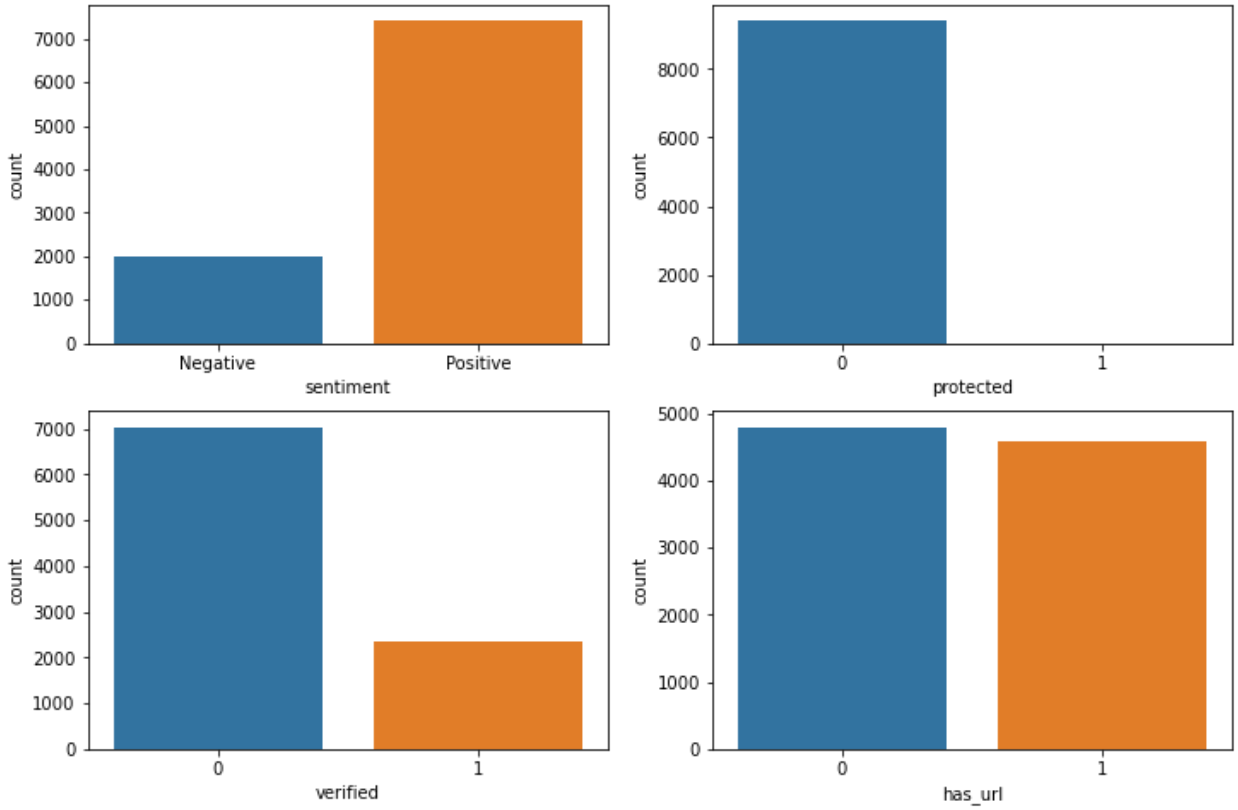
Exploratory Data Analysis

After extracting the features from the original dataset, we divided the produced dataset into training and testing datasets. Our training dataset contains information about 9395 users and 31 columns where we only use 26 of them in our supervised learning section. Since the features are extracted from the dataset, we don't face the issue of having missing values in our features. On the other hand, the testing dataset contains information about 1176 users and the same number of features. We analyzed first the distribution of each feature and then we analyzed the correlation between each feature and the label feature. We also divided the categorical and numerical features.

When analyzing the label feature of whether the user is a human or bot, we observe that the dataset is almost balanced in both the training and testing datasets (as seen in the below figures), with a slight increase in the bot percentage (label 1) than the human percentage (label 0).

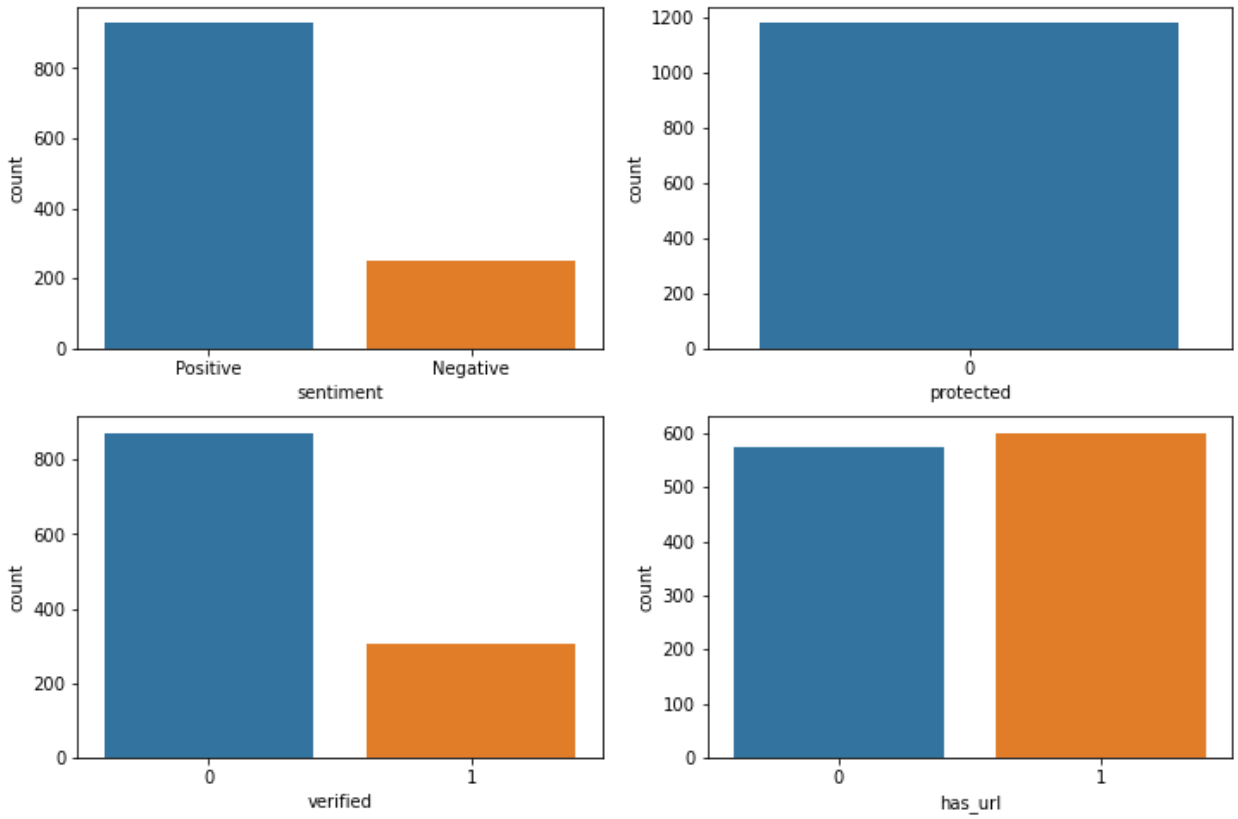


Regarding the categorical features in the training dataset, we can observe from the below figure that the majority of the users tweets have a positive sentiment. Also,the majority of the users accounts aren't verified while all of them aren't protected. Finally, we can see that the number of

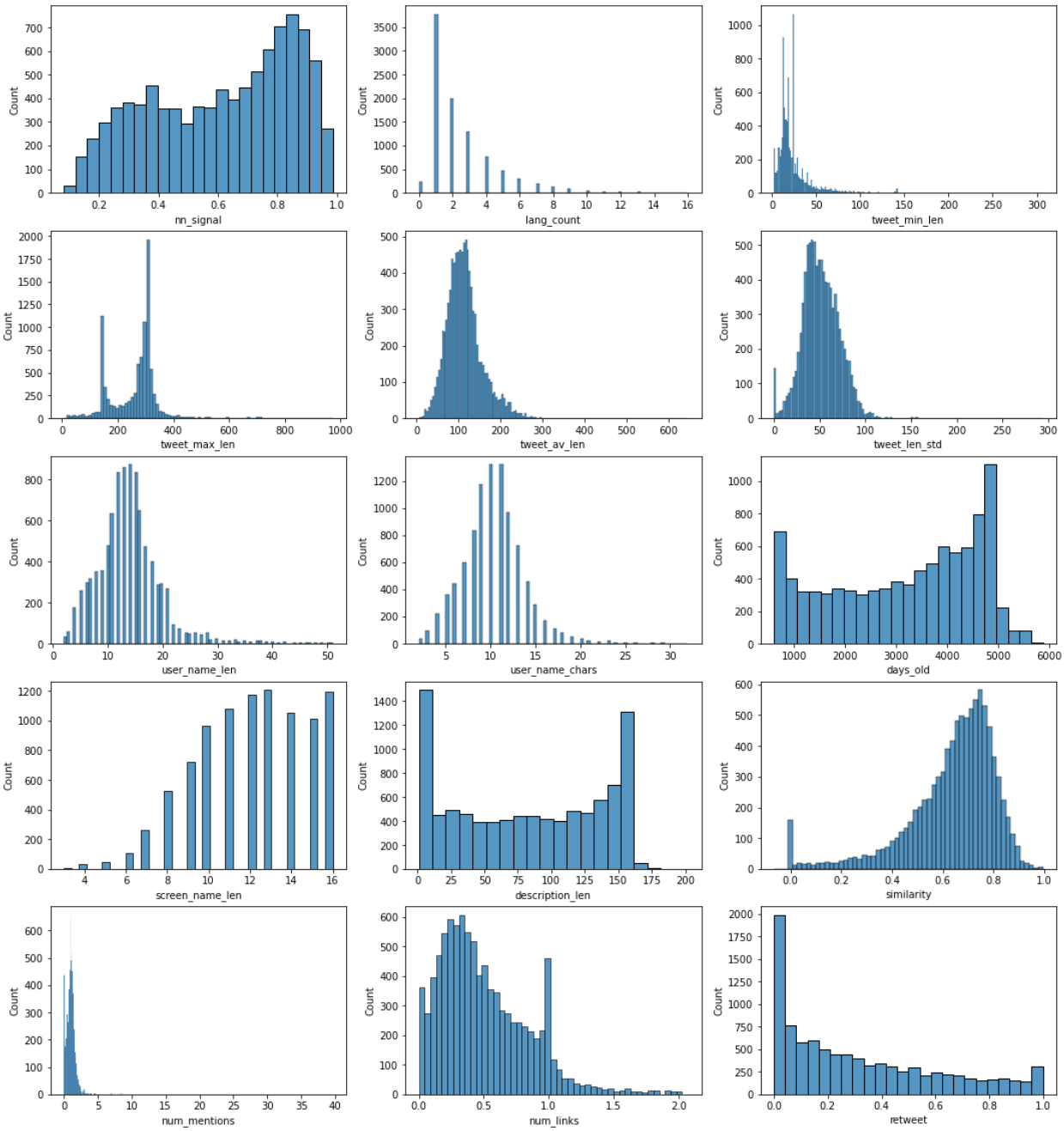


users tweets that contain URLs is equal to the number that doesn't.

We can also observe the same patterns of distributions in the testing dataset as seen in the below figure.

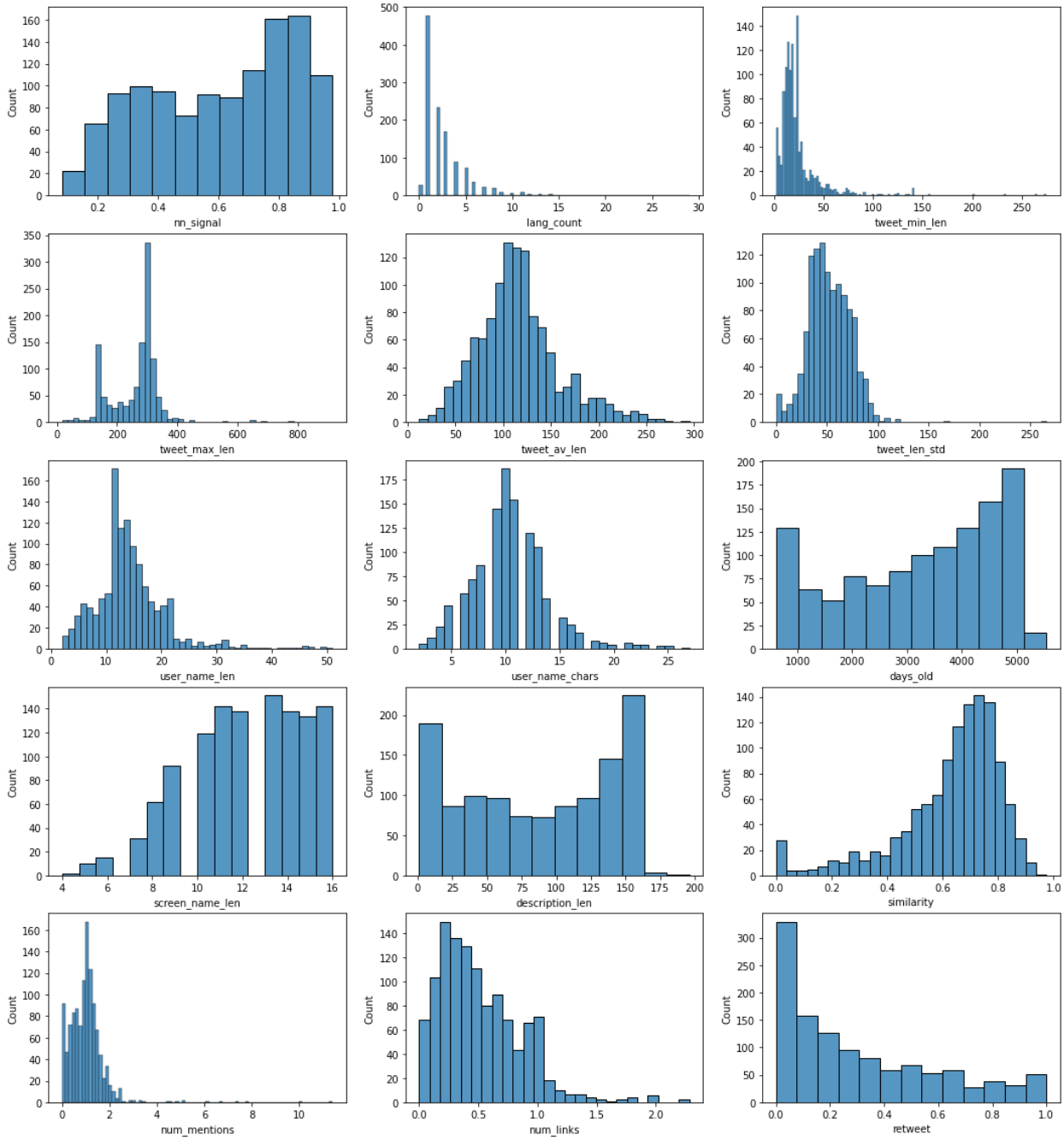


Regarding the numerical features distribution, we can observe from the figure below that the majority of the numerical features are normally distributed with some skewness with a very few of them uniformly distributed. For the “nn_signal” feature which represents the probability of a user to be a bot, we can see that it has an approximately skewed distribution to the left with the majority of the values to be around 0.9, meaning that the majority of the users in the training data have a high probability of being a bot. We will verify this when looking at this distribution in comparison to the label distribution. For the “lang_count” feature showing the languages used by each user, we can see that it also has a skewed distribution to the right in this case where the majority of users use one language in their tweets. For the “tweet_min_len” feature, we can see that it also has a right skewed distribution, with the majority of users' tweets having a minimum length of around 20. However, the length of the shortest tweet across all users has a length of two. Regarding the “tweet_max_len” feature, we can see that the majority of users’ tweets have a maximum length of around 300. However, the length of the longest tweet across all users has a length of 973. For the “tweet_av_len” and “tweet_len_std” features we can see that they have similar distributions normally distributed with a very slight skewness to the right, where the majority of users’ tweets have an average length of around 120 and standard deviation of around 45. For the “user_name_len” and “user_name_chars” features which represent the length of the username and the number of distinct characters used in the username respectively, we can observe them having a rightly skewed distribution, with the majority of users having a username of length around 15 and around 10 distinct characters used. For the “days_old” feature representing the age of the user’s account, we can see a



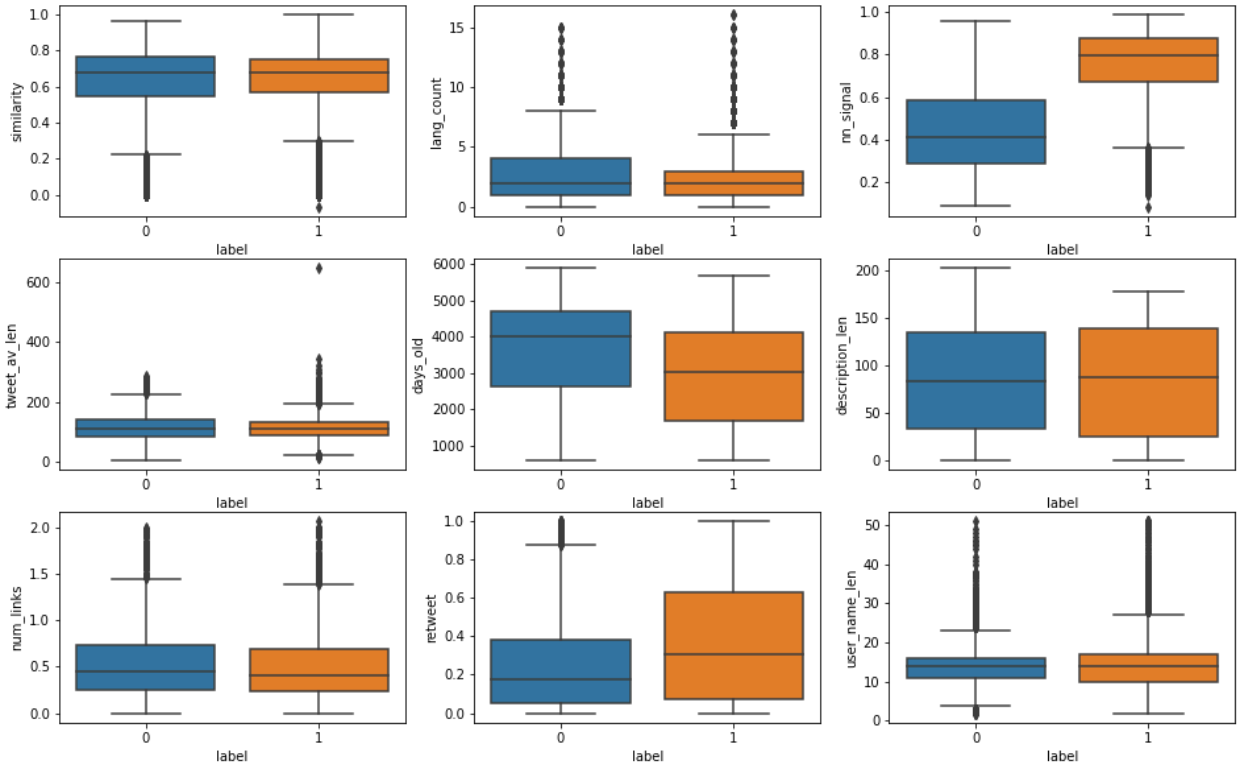
uniform distribution between 1000 to 5000 days (~3 to ~14 years). For the “screen_name_len” , we can observe that the majority of users screen names have a length of around 12-16. The “description_len” also has a uniform distribution between around 0 and 175. For the “similarity” feature representing the similarity across tweets of each user, we can observe a left skewed distribution with the majority of users having high similarity around 0.7 between their tweets. Based on our hypothesis, this means that the majority of users might be bots, since the similarity between the tweets are high. We will confirm the hypothesis when we compare this distribution with the label’s distribution. For the “num_mentions”, “num_links” and “retweet” features which represent the number of mentions, number of links and whether tweets are retweeted or not for each user, are normalized by the total number of tweets for each user.

Regarding the “num_mentions” feature we notice a right skewed distribution with a very low number of mentions across the users’ tweets. We can see a similar pattern in the “num_links” feature where for the majority of users, less than 50% of their tweets contain links. Finally, for the “retweet” feature it’s kind of uniformly distributed with the majority of users’ tweets having lower retweet percentages.
We can observe that the distributions of the numerical features in the testing dataset are similar to those in the training dataset as seen in the below figure, with the length of the shortest tweet

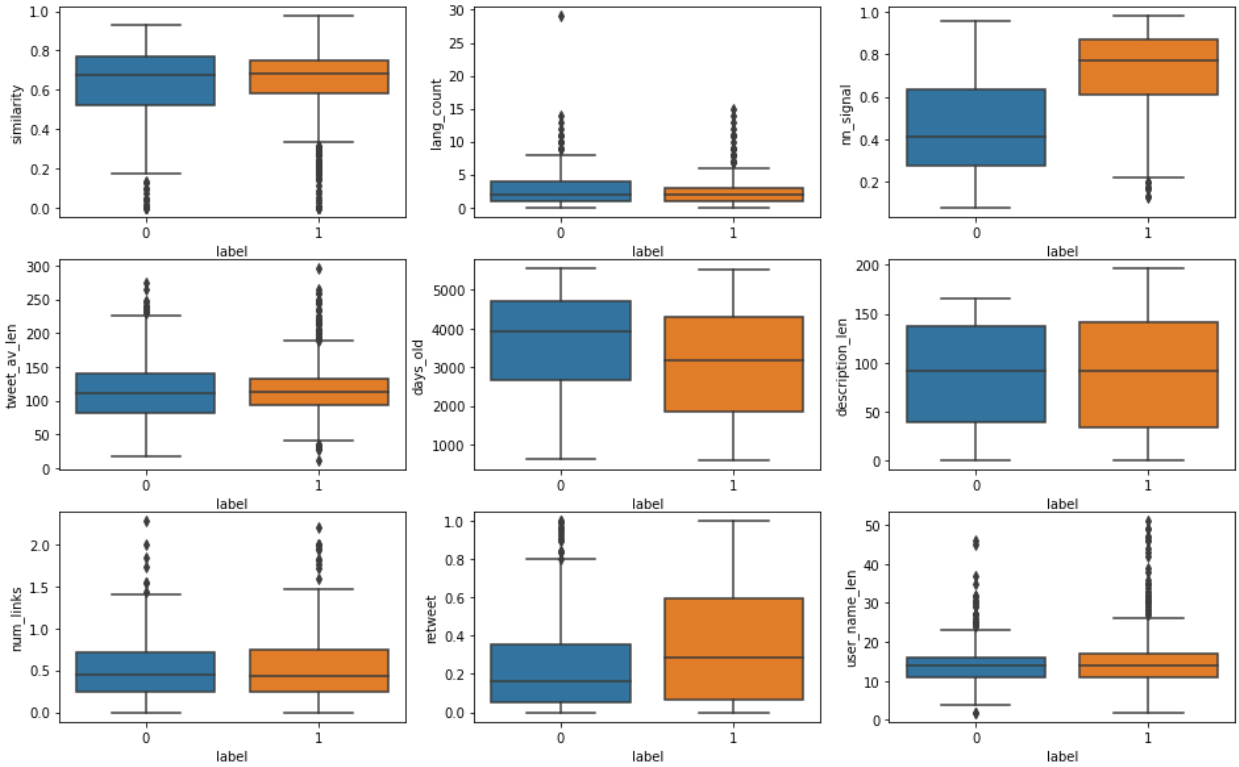


in the testing dataset to be two and the length of the longest tweet in the testing dataset to be 916.

Now we will be looking at the comparison between some features' distributions and the label's distribution to assess their relationship. The objective of these comparisons is to check whether the distribution of certain features will differ between the labels human and bot. Regarding the similarity feature, we already had the hypothesis that users that have higher similarity scores across their tweets are more likely to be bots than humans. However, as noticed in the below figure, the similarity scores for both types of account have similar distributions, which technically refutes our hypothesis. The second feature that we will check its effect on the different user types is the number of languages used by users. We can notice that the average number of languages used by users of both types (human and bot) is the same, however, for the human users we can see a larger dispersion in the language count meaning that human users tend to use more languages than bot users. The third feature to study is the "nn_signal" which represents the probability of a user to be a bot, we can see that of course it has higher values for the bot user type confirming the output of the neural network fed on the tweets. Looking at the tweet average length feature, we can notice no difference across user types, meaning both have a similar average tweet length. For the age of the users accounts we can observe that on average human users have older accounts than bot users accounts. Regarding the description length, number of links and username lengths features, we can see no difference on average across the two types of users. Finally, regarding the retweet feature we can see that the bot users have a retweet average that's higher than that of the human users.



We can also observe similar relationships between the previous features and the label feature in the testing dataset too as seen in the below figure.



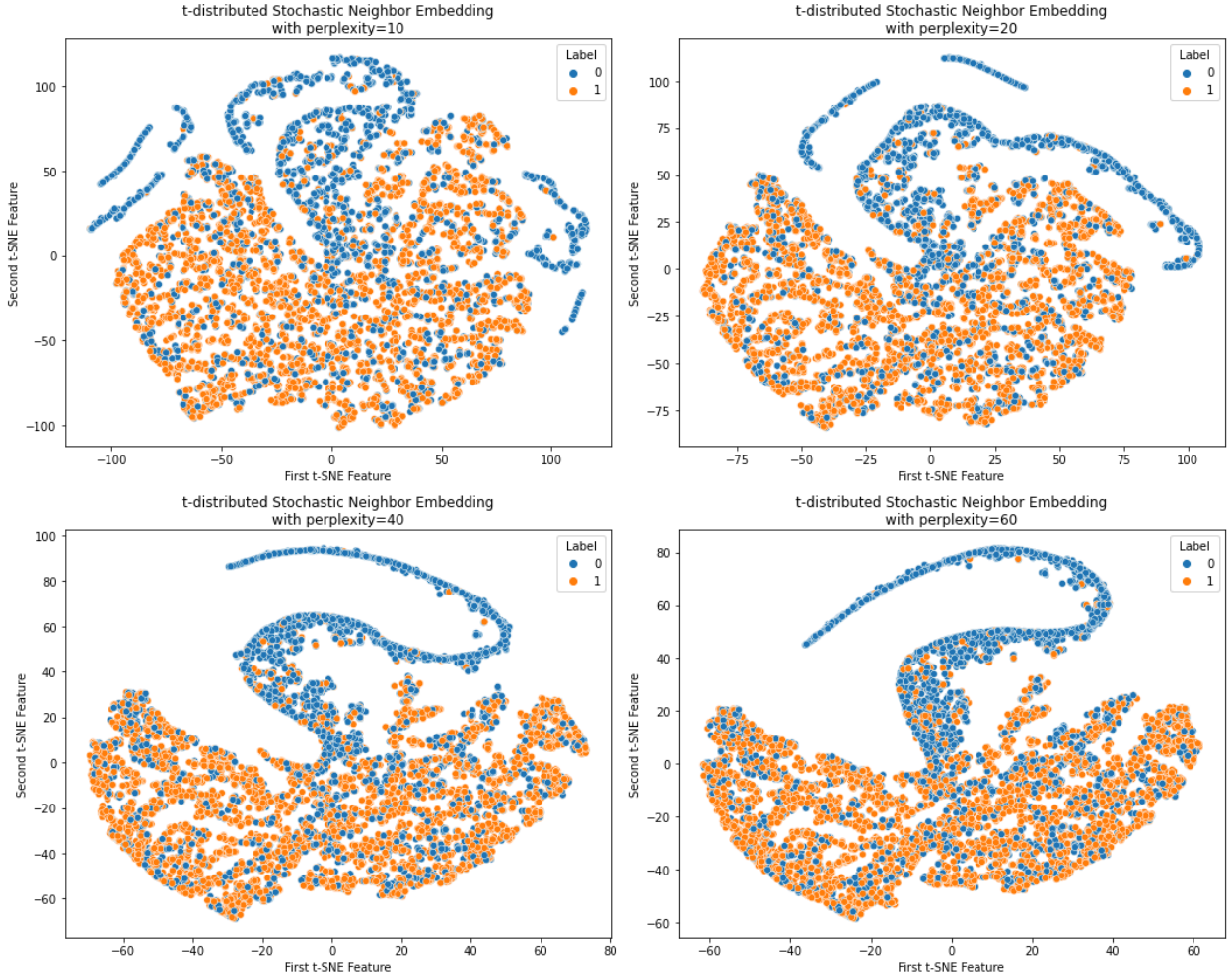
In general, the similarity of the distributions and relationships between features across the training and testing datasets is a good sign that can point to having datasets from the same population which means the model should perform well on both datasets, which will be proved in the supervised learning section. However, one drawback from this similarity is that overfitting might happen across both datasets, which might need further validation of the model (in the future) on other datasets to ensure its generalizability.

Unsupervised Learning

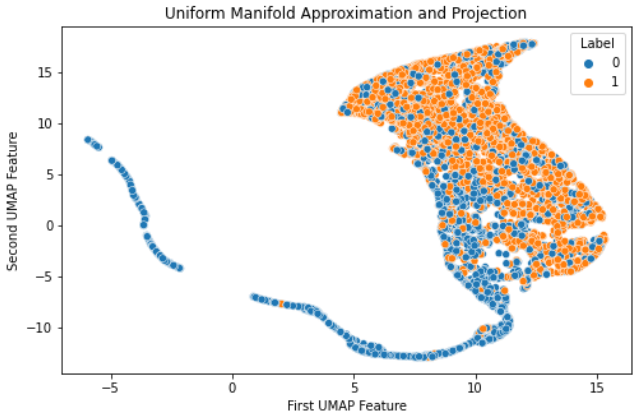
Using unsupervised learning in this project came in multiple forms first it was used to extract topics using a Non-Negative Matrix factorization as described above in the topic extraction section, second we used pretrained language detection and sentiment analysis models as well as a spacy model for vectorizing the tweets in the similarity calculation. Finally we want to use unsupervised learning to visualize a two dimensional projection of our final dataset to detect any possible separation in the classes.

There are multiple ways to visualize this projection using dimensionality. Here we tried three kinds: Multidimensional Scaling (MDS), t-Distributed Stochastic Neighbor embedding (t-SNE), and Uniform Manifold Approximation and Projection (UMAP). All of which can be seen in the unsupervised_vis.ipynb notebook. With a sample of 2000 data points (as it was quite slow) MDS did not achieve a viable separation between the classes as it is good at keeping the global structure of the data but not as good at detecting local neighborhood structure. Section 2.1 unsupervised_vis notebook. Thus the plot resulted with a few points separated from the rest and

not much could be gleaned from the result. On the other hand t-SNE is very good at preserving local neighborhood structure but not good with global structure, this resulted in a clear tail of points belonging to the human users extending out from a mixed cluster of bots and humans. The t-SNE algorithm needs to be tuned with a perplexity argument and as can be seen below this argument creates different visualizations. We can see that the best separation is created with a perplexity value of 40 or 60. Section 2.2 unsupervised_vis notebook.



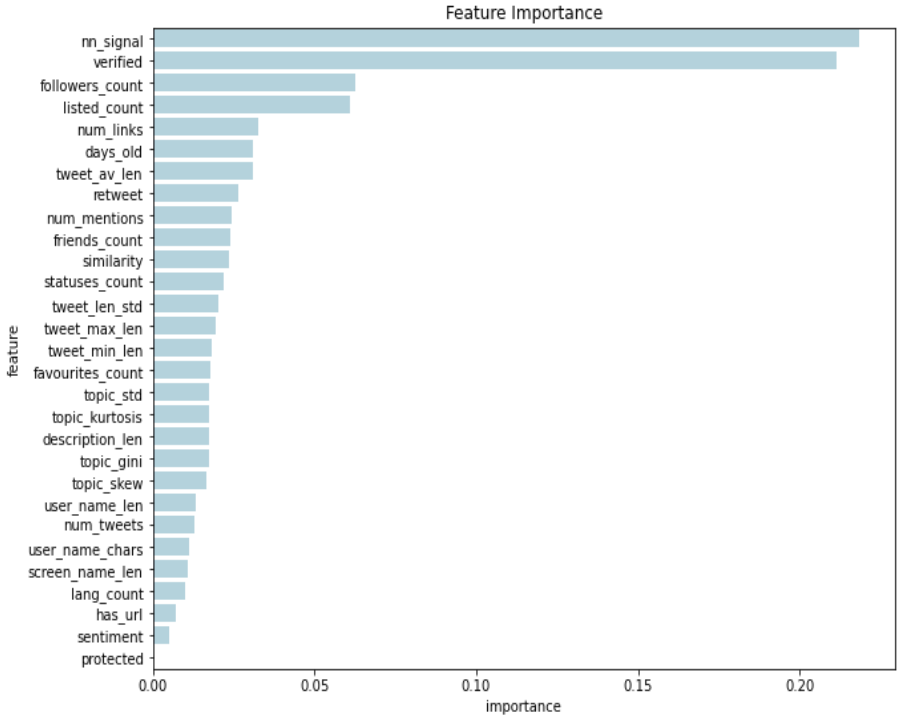
Although we can see the local neighborhood structure using t-SNE, it does not preserve any global structure. The last dimensionality reduction is UMAP; it preserves the local structure while also preserving global structurel. This gave us the best visual representation of the data. Section 2.3 unsupervised_vis notebook. We can clearly see the tail of human users and a large cluster of bots mixed with human users. Within the cluster it seems that the bots can be separated generally in the upper right sections of the plot.



Supervised Learning Models

For the final Supervised Learning portion, we decided to create an ensemble method of multiple models to create two classification models: the first is a voting classifier and the second is a stacking classifier. The theory behind this model combination is that each model on its own has built in biases and different types of difficulties with certain types of data. By combining models they can counteract each other's flaws and perform better and more consistently on unseen data.

We wanted to combine different types of models and ended up with two linear models LogisticRegression and SGDClassifier using modified huber loss, three tree based models lightgbm (gradient boosted trees) RandomForestClassifier, and AdaboostClassifier, and one support vector machine SVC. In section 1 of the final_supervised_model notebook. The helper functions are defined. One of these functions score_model was used in conjunction with the hyperopt tuning library to tune the individual models. In section 2 we loaded the data, dropped unneeded columns, converted string columns to integers and showed the stats of each dataset. Finally the data was scaled using the StandardScaler. We wanted to include a step to show how important each feature was to the predictions. The RandomForestClassifier has an attribute from the fitted model that outputs a list of feature importances using an “impurity-based” importance measure. From this we can see the most important feature even above verified was the output signal from the deep learning model. This plot was created in section 2.5 of the final_supervised_model notebook.



To create a baseline a untuned LogisticRegression model was passed to the score function which runs cross validation using roc_auc as the scoring method and it scored high at .94. Then each of the models was tuned using hyperopt in section 3.2. During tuning they only achieved marginally better results than the base LogisticRegression model.

The ensembling portion in Section 3.3 again only showed marginally better performance than the base model. For both methods they only outperformed the base model by .002 using the roc auc cross validation function.

In section 4 the base model and the voting and stacking classifiers were compared on the test dataset, and again the base model performed almost as well with the roc_auc and accuracy metrics.

Untuned LogisticRegression Model performance on the test dataset:

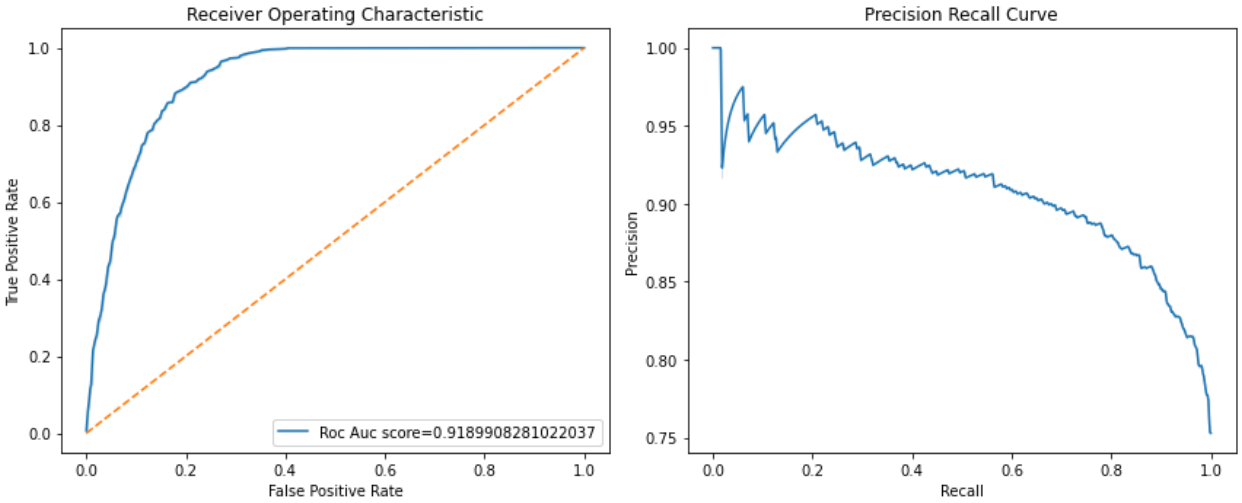
Metrics:

roc_auc	0.918991
accuracy	0.856293
precision	0.843840
recall	0.907550
f1	0.853188

Confusion Matrix:

	Predicted Positive	Predicted Negative
Actual Positive	418	109
Actual Negative	60	589

ROC/AUC and Precision/Recall curves



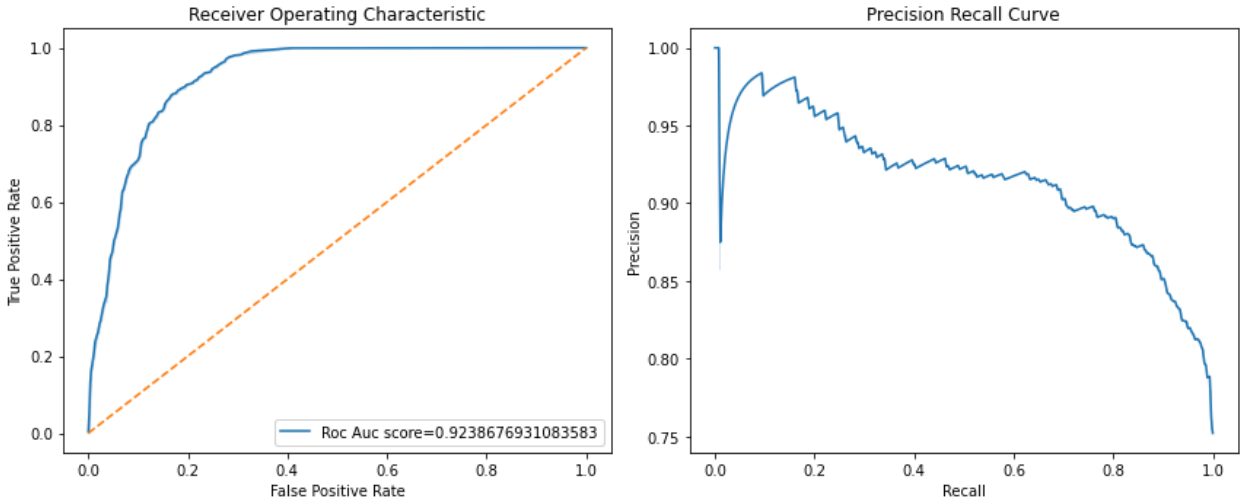
The stacking ensemble classifier with six tuned models on the test dataset:

Metrics:

roc_auc	0.923868
accuracy	0.857143
precision	0.846043
recall	0.906009
f1	0.854167

Confusion Matrix:

	Predicted Positive	Predicted Negative
Actual Positive	420	107
Actual Negative	61	588



In general ensemble methods will perform better than base untuned models, however in my opinion in this case it came down to the data. The data was linearly separable by class and thus the decision boundary was able to be found by all models linear or not.

Error analysis

Our classifier has multiple stages, some of them with little heuristic content, and also draws on a wide and diverse range of features. As a result, it is very challenging to say anything definitive about why specific accounts are misclassified. However, we can try to draw some information from how the classifier behaves overall.

Let's look first at bots that were misclassified as human. We compare the average and median feature values of these accounts with the bot and human accounts, to see if some features might correlate. The "listed_count" feature stands out: in bots classified as human, the average value was 154.2, and median 1. For all bots, these values were even higher: 485.2 and 5 respectively. And for humans, much higher: 3713.1 and 205. It seems strange that such a determinative feature in the test set was not emphasized sufficiently by the classifier. These numbers are roughly in line with what appears in the training set, so sampling variation is not the explanation. There are similar discrepancies in other features related to engagement, where the misclassified human accounts had low values in line with the bot overall average instead of the much higher human average. It is hard to square this with the importance of these metrics according to the ensemble's RandomForestClassifier (above).

The case of humans misclassified as bots is even harder to understand. We can find certain features -- "followers_count" for example -- where this set's average is closer to the human average than the bot one, but not nearly to the extent that we saw in the earlier example. We wondered if, as we postulate below when discussing ethical issues, this type of mislabeling might occur more often when the account tweets frequently in non-English languages. We did not develop a "primary language" feature, so we performed some visual inspection of the data

set itself. If anything, non-English languages seemed to occur more in the set of accounts correctly identified as human than in the misclassified ones.

One thing that we do notice in both types of misclassifications is that the average value of "nn_signal" is often closer to that of the opposite class than to its own. Given the importance of this feature to the RandomForestClassifier, and its general accuracy, this might be the best explanation for the occurrences of mislabeling.

Ethical Considerations

A functioning system for bot detection among Twitter accounts could possibly serve many purposes. In almost all the ones we can imagine, the system would facilitate efforts to eliminate, or at the very least suppress, the presence of bots on the platform. It is important to ask who might be harmed by this.

First, not all bot accounts are malicious or without value. Some provide autonomous creative or parodic expression; others provide updates on real-world events. Obviously many Twitter users value the contributions of bot accounts, so we anticipate that any project to remove bots would need further layers of learning to distinguish malicious bots from the rest.

Of more relevance to our project is the effect that misclassification might have. Human accounts classified as bots may be suppressed by Twitter and its clients, and some populations of Twitter users may be disproportionately affected by such misclassification. One concern we have is that Twitter accounts which tweet mainly or exclusively in a non-English language may suffer from mislabeling more often. For example, at some steps we used English stopwords to improve the performance of e.g. tokenization; but we did not do anything similar for other languages. Although we didn't see evidence that this was a problem for our classifier, it merits further study. It also should be mentioned that the training is only as good as the labeling; although the methods employed by the Twibot-20 research team seem reasonable and rigorous to us, we support their interrogation by people better qualified than us.

Statement of Work

Rania El-Shehety

- Preparing list of features to be extracted
- Feature extraction for some user level and tweet level features including Similarity calculation and sentiment analysis for user tweets
- Exploratory data analysis on final dataset for both training and testing datasets

Alexander Levin-Koopman

Dataset acquisition
Data preparation
feature extraction
Topic modeling

Data combination notebook
Final Supervised learning notebook

Jeffrey Olson
Data preparation
Early-stage neural network
Feature extraction, including language count feature
Report preparation and editing