ML4Geo – Final Report

Twitter Sentiment Analysis Using Random Forest Classifier

Robert Wright

September, 2021

University of Heidelberg

Contents

1. Introduction

Twitter is a place for all sorts of people to gather. Politicians, celebrities and athletes discuss and interact with each other. As in any society-like state, human conflicts are inevitable. However, in recent years, Twitter was forced to penalize a record number of accounts for posting hate speech [1]. So far, the grand final of this alarming development was probably the permanent banning of Donald Trump's (the US president at that time) Twitter account [2].

Therefore, this project aims at training a relatively simple, yet robust, machine learning model to detect hate speech on Twitter: A random forest classifier is using the tweet text and location as features. In a later run, this project is going to focus solely on geographical information as feature space. When coding in python, the scikit-learn library [3] helps to solve such machine learning problems and is used throughout this project.

2. Data Preparation

The dataset used in this project was first gathered by Founta et al. [4] and is openly shared on GitHub[1]. The online version holds two columns: tweet ID and classification label. The

[1] `https://github.com/ENCASEH2020/hatespeech-twitter`

tweets were read by volunteers and annotated through crowdsourcing with one of the following labels: *abusive*, *hateful*, *normal* or *spam*. According to Twitter policy, researchers are not allowed to publish raw content (e.g., text) of tweets as users might want to delete personal tweets, which should not appear elsewhere then. Therefore, unique IDs are used to identify and download (online) tweets and their related meta-information.

For further information on how the dataset has been assembled, please refer to the corresponding publication.

2.1. Acquisition

In order to access the Twitter API and automate downloading of tweets, one needs credentials provided by Twitter. These can be obtained by applying for a developer Twitter account, which has to be verified. Then, `Tweepy`, a python library, searches Twitter for tweets that match the IDs provided in the aforementioned dataset. The tweet text and selected (geo-related) meta-information, such as coordinates or location name, are stored in a new data frame and combined with the classification labels.

The original dataset (from 2018) provides 99 996 tweet IDs, however, just 54 280 still happen to be online and are analysed in the following.

2.2. Preprocessing

In order to train a machine learning model on natural language, data preprocessing is of high importance.

As a first step, tweets without any geographical information are discarded. The Twitter documentation[2] lists several attributes (as part of the meta-information) that an object

[2] `https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet`

(i.e., tweet) can have. Most interestingly, `coordinates` represents the actual coordinates of a tweet as reported by a client application (e.g., location service on mobile phone) and `place` indicates the location the user selected (if any) while posting. The latter attribute has several child objects, such as `country_code`, `bounding_box` and `full_name`. A bounding box provides coordinates that enclose the geographical location of a tweet. Therefore, either by directly using the values of `coordinates` or by averaging the content of `place.bounding_box.coordinates`, one can compute the longitude and latitude of a tweet location.

Many tools were created to solve natural language processing (NLP) problems. Among other things, the python NLP library `spaCy` offers trained state-of-the-art models for various languages, which help to perform the following processing steps:

1. **Tokenization** is the important process of breaking a stream of textual data into characters, words or any other meaningful elements called tokens. This project makes use of word tokenization; the tweet text is split into individual strings holding single words.

2. **Lemmatization** reduces tokens to their base form or lemma, so they can be grouped together and analysed as a single item. For example, *walking*, *walks* and *walked* are all forms of the word *walk*, therefore *walk* is the lemma of all these words.

3. **Stop word removal**. These words do not contribute any meaning or sentiment to a sentence and can be safely discarded. `spaCy` includes a dictionary of common stop words, which are typically articles, prepositions or linking words (e.g., *is*, *are*, *and* in English language).

4. Apart from these general steps in NLP, Twitter content requires some custom alteration. In greater detail, the final preprocessing involves removing of hyperlinks, twitter handles, punctuation marks and numbers, again motivated by the fact that these do not add any important context.

2.3. Exploratory Analysis

Although the preprocessing should deliver adequate results, it is better to investigate the updated data frame by visualizing some key characteristics. Please note that in the following, only tweets originating from the US were considered. After data cleaning and preprocessing, their total number adds up to 1201.

Imbalanced Data

The current data frame is imbalanced, i.e., the different classes are not represented equally.

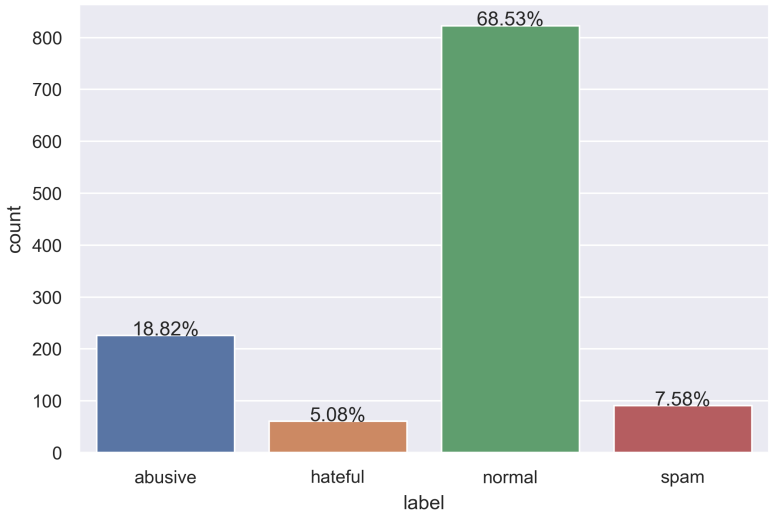
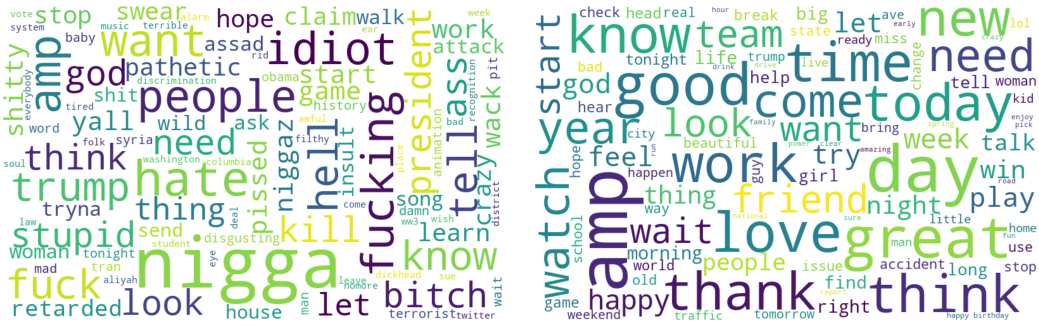


Figure 1: Count of occurrences of each class in the preprocessed dataset, for clarity the values are also shown as percentages on the respective data bar.

Figure 1 reveals that *normal* tweets are more common than any other label, in turn, *spam* and *hateful* tweets are rare. This is important to keep in mind as it determines how the performance of the machine learning model should be evaluated.

Word Cloud



(a) hateful (b) normal

Figure 2: Word cloud for tweet classes. The size of a single word is proportional to its number of occurrences within all tweets of the respective class. Sorry for vile language!

Another way of visualizing natural language are word clouds. These can be used to get a general idea of the most common words utilized in tweets. In figure 2, this was done for *normal* and *hateful* tweets in order to make the results as divergent as possible. Qualitatively speaking, the outcome seems reasonable, which indicates that the preprocessing steps have been successfully implemented.

Mapping Tweets

Last but not least, it is investigated whether the tweets show any form of geospatial bias. The map of figure 3 indicates that the classes of tweets are well distributed within the US and, as one would expect, rather follow population density.

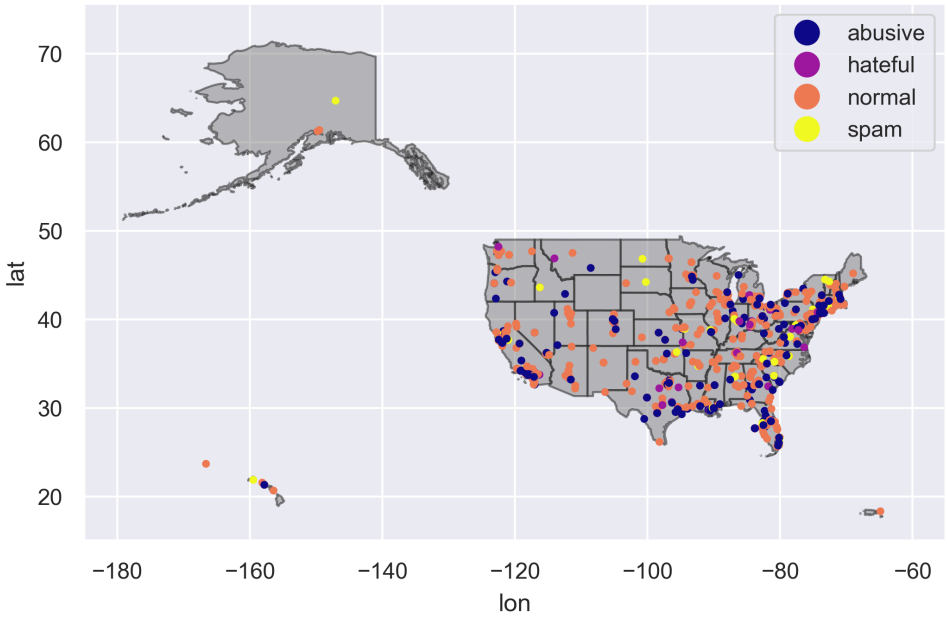


Figure 3: US map featuring tweet location and class label.

3. Random Forest Classifier

Decision trees are a popular method for various machine learning tasks. A single tree, however, tends to overfit its training set and learns highly irregular patterns, i.e., has low bias, but very high variance. Random forests operate by constructing a multitude (forest) of decision trees, thus averaging multiple deep decision trees, trained on different parts of the same training set. This method reduces the variance, which comes at the expense of a small increase in bias, but generally greatly boosts the performance in the final model.

Random forests generate reasonable predictions across a wide range of data while requiring little configuration. Further information on the scikit-learn implementation can be found in *Ensemble Methods* [5].

3.1. Bagging

Bootstrap aggregation (or bagging in short) is a simple and very powerful ensemble method to de-correlate trees of a random forest [6].

Bootstrapping refers to any test or metric that uses random sampling with replacement and is generally useful for estimating the distribution of a statistic (e.g., mean, variance) without using normal theory. Likewise, instead of training on all observations, each tree of the random forest is trained on a subset of the training data. The chosen subset is called a bag, and the remaining data points are referred to as out of bag samples. Multiple trees are trained on different bags, and later the results from all the trees are *aggregated*. Simply training many trees on a single training set would give strongly correlated trees, or even the same tree many times.

Random forests also include another type of bagging scheme, sometimes referred to as "feature bagging": At each node of a single tree, a random subset of features is selected. This prevents trees from becoming correlated, if only one or a few features are very strong predictors of a class, as these would be selected in many trees. It also makes the model more robust with regard to correlated features.

3.2. Hyperparameters

The following main parameters are adjusted in the random forest classifier of this particular project work.

- `n_estimators` is the total number of trees in the forest. Generally speaking, the larger, the better, but also the longer it will take to compute. In addition, results will stop getting significantly better beyond a critical number of trees.

- `max_features` is the size of the random subset of features to consider when splitting a node. The lower, the greater the reduction in variance, but also the greater

the increase in bias.

- `max_depth` controls the maximum depth of a tree.

- `min_samples_split` and `min_samples_leaf` determine the minimum number of samples required to split an internal node and to be at a leaf node, respectively.

4. Experiments

Scikit-learn provides an architecture to facilitate machine learning workflow: pipelines. A pipeline is simply a method of chaining multiple steps (such as feature engineering or model training) together, in which the output of the previous step is used as the input for the next step. In the following, the individual pipeline steps are explained.

4.1. Feature Engineering

The preprocessed dataset contains single word strings representing the tweet text and location coordinates stored as floating point numbers. While a random forest classifier can use the geographical information unaltered as numerical features, the raw text tokens cannot be fed directly to the algorithm as it expects a feature vector with a fixed size rather than a list of strings with variable length.

In order to address this, each individual token occurrence frequency is treated as a numerical feature. A corpus of tweets can thus be represented by a matrix with one column per tweet and one row per token. Consequently, the length of the feature vector of a single tweet is equal to the total number of different tokens. As most tweets typically use a very small subset of the words used in the corpus, the resulting vector will have many feature values that are zero. Remaining values are computed using the term frequency–inverse

document frequency (tf–idf) transform. It multiplies the tf, the number of times a term occurs in a given tweet, with an idf component, which is computed as

$$\text{idf}\,(t) = \log \frac{1+n}{1+\text{df}\,(t)} + 1, \tag{1}$$

where n is the total number of tweets and $\text{df}\,(t)$ is the number of tweets in the corpus that contain the term t. Tf–idf takes into account that some words are very present in a corpus, hence carrying very little meaningful information about the actual content and shadowing the frequencies of rarer yet more interesting terms. The maximum number of features `max_features` (i.e., length of feature vector) returned by the tf–idf method can also be adjusted.

4.2. Cross-Validated Grid Search

The tf–idf feature extractor is pipelined with the random forest classifier. It is a supervised machine learning experiment, which means that the parameters introduced in section 3.2 must be manually set. However, there is still a risk of overfitting *on the test set* because the parameters can be tweaked until the estimator performs optimally. In other words, knowledge of the test set "leaks" into the model and evaluation methods overestimate the generalization performance. Therefore, yet another part of the data must be held out as a so-called validation set. Training proceeds on the training set, after which the model is evaluated using the validation set, and when the experiment seems to be successful, final evaluation is done on the test set.

By dividing the data into three distinctive sets, the number of samples available for model training drastically decreases, and thus the final result can depend on a particular choice of partitioning the data. A separate validation set is no longer needed when using *cross-validation* (CV), this procedure is illustrated in greater detail in figure 4.

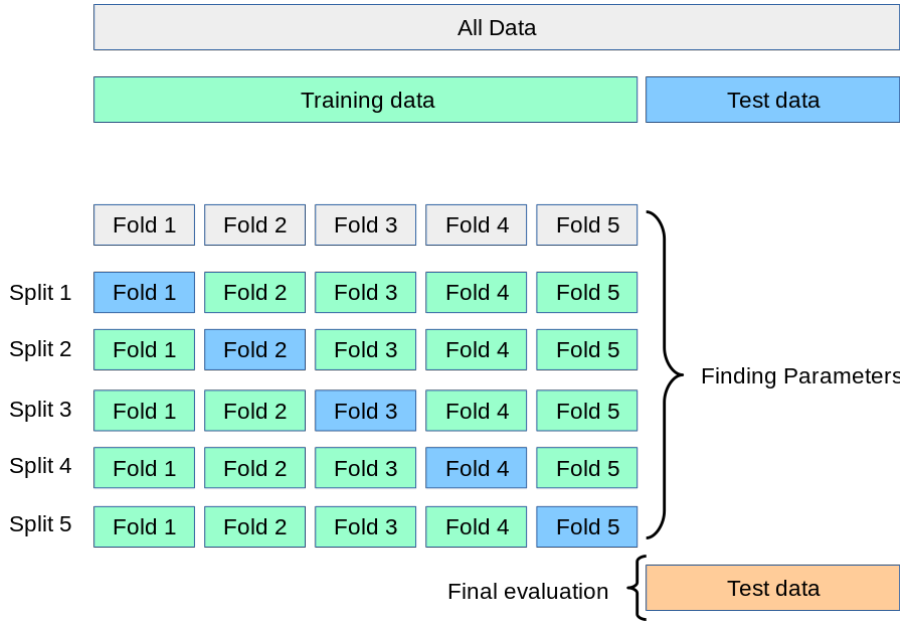


Figure 4: In this basic approach of CV, the training set is split into $k = 5$ smaller sets, $k - 1$ of the folds are used as training data (green) and the resulting model is validated on the remaining part of the data (blue). This procedure is followed for each of the k folds. The overall model performance is assessed by averaging the single runs. A test set is held out for final evaluation. Diagram taken from scikit-learn documentation[3].

The hyperparameter space is searched for the best cross validation score, *grid search* exhaustively evaluates all possible combinations of parameter values and retains the best result.

The random forest classifier is trained on a reduced dataset, focussing on tweets originating from within the US.

[3] `https://scikit-learn.org/stable/_images/grid_search_cross_validation.png`

5. Results

The optimized hyperparameters of the machine learning experiment are listed in table 1. The total number of input features (including coordinates and tokens) is 4093 per tweet, although in reality the tf–idf method can limit this quantity to `max_features`.

Estimator	Parameter	Value
Random forest	`n_estimators`	100
	`max_features`	None (all features)
	`max_depth`	None (no limitation)
	`min_samples_split`	4
	`min_samples_leaf`	1
Tf–idf	`max_features`	500

Table 1: Hyperparameter values of random forest classifier and tf–idf as returned by grid search.

Grid search performed 3 folds for each of 1296 candidates, totalling 3888 fits. The evaluation metric is the weighted F1 score (average of precision and recall), as the data is imbalanced with regard to the class label. The optimized hyperparameters return an F1 score of 0.819 on the training set, while generalization performance can be assessed by taking into account the classification report in table 2 and the confusion matrix of figure 5.

	Precision	Recall	F1 score	Support
abusive	0.77	0.84	0.80	44
hateful	0.40	0.18	0.25	11
normal	0.87	0.92	0.89	170
spam	0.14	0.07	0.09	15
weighted avg	0.78	0.82	0.80	240

Table 2: Classification report based on predicting test dataset. **Precision** or positive predictive value is defined as $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$, measuring the proportion of correct positive predictions. **Recall** is defined as $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$, measuring the proportion of correctly identified positive labels. A perfect model has a value of 1 in both metrics. **Support** is the number of samples the individual scoring parameters are based on.

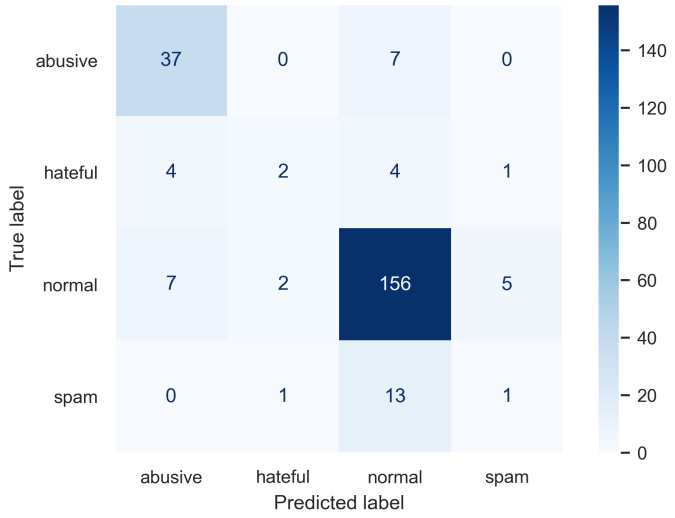


Figure 5: Confusion matrix of random forest classifier on US tweet data. By definition, entry i, j is the number of observations actually in group i, but predicted to be in group j.

Put together, this indicates that the random forest classifier can predict *normal* and *abusive* tweets reasonably well, but fails to detect *hateful* and *spam* tweets. The former is randomly predicted as another class, while the latter is almost exclusively recognized as *normal.*

Random forests can be used to assess the importance of variables, as illustrated in figure 6. The rank (i.e., depth) of a feature indicates its relative importance as, generally speaking, features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples. Scikit-learn combines this fraction of samples with the decrease in (Gini) impurity from splitting them to compute a normalized estimate of the predictive power of that feature.

The geotagged tweets allow to investigate the geospatial distribution of predictions as done in figure 7, however, none is identifiable (i.e., false predictions show no geographical pattern).

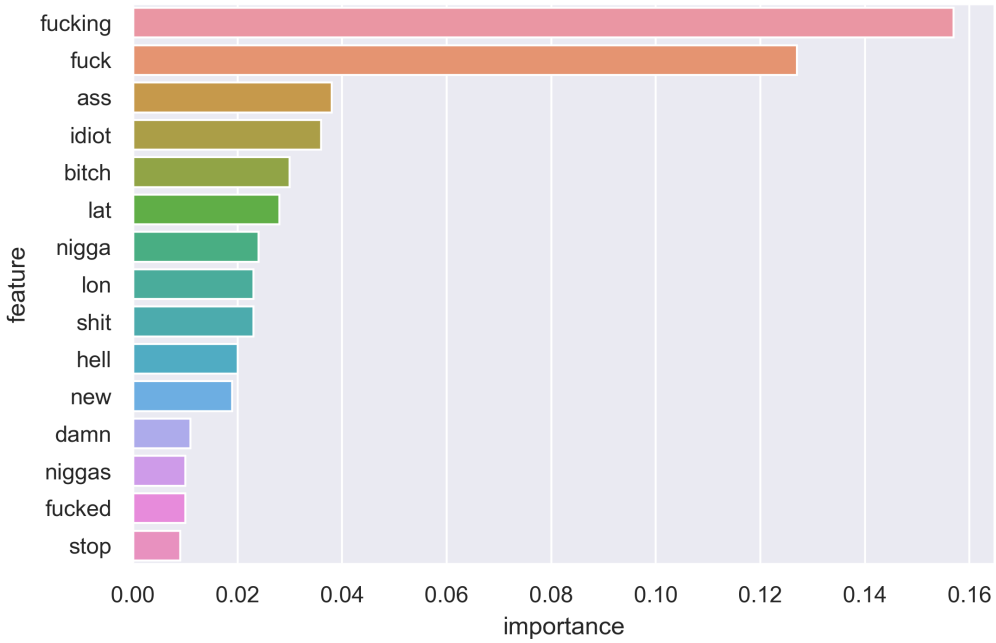


Figure 6: Importance of single features in random forest classifier (impurity-based). The values on the y-axis sum up to 1; the higher the values, the more important is the contribution of the matching feature to the prediction function.[4]

[4] `spaCy`'s trained English language pipeline automatically applies lemmatization to tweets. Apparently, this does not work flawlessly, as *fucking* and *fucked* appear in the processed text samples. This behaviour is further investigated in `lemmatization-test.ipynb`; access in appendix B.

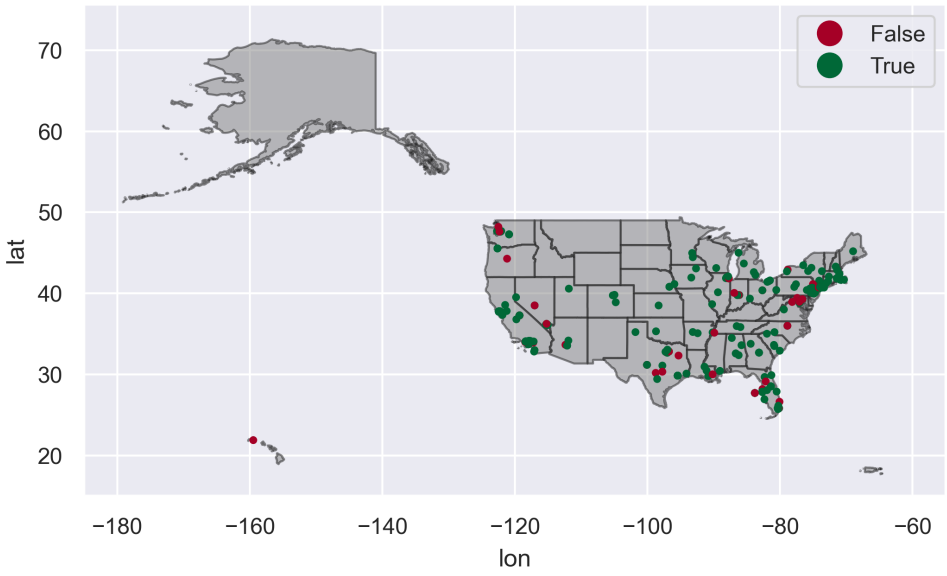


Figure 7: US tweet map illustrating where class labels of test set are predicted correctly (true) and incorrectly (false) by random forest model.

6. Discussion

This project trained a random forest classifier on tweet data featuring four distinctive labels. The generalization performance of the model highly varies with regard to the class of a tweet. More specifically, the feature importance analysis indicates that the model learned how to classify *abusive* tweets, while predicting every non-*abusive* tweet as *normal*. This behaviour is confirmed by the confusion matrix: High numerical entries on the diagonal are only present for *abusive* and *normal* class. However, please note that the total amount of data has been very limited after data preparation, *hateful* and *spam* tweets being affected severely, as these classes are the most infrequent ones of the data set. The classifying behaviour of the machine learning model is explained by the nature of the data, since the probability of a non-*abusive* tweet being actually *normal* is high.

Repeating the experiment with a larger (geotagged) dataset might lead to an improved model, which can also actively predict the other tweet classes. Alternatively, in scikit-learn the `class_weight`-parameter of a random forest classifier determines the weight each class has when computing the impurity of an internal node, and can be used to penalize misclassifying the minority classes. Moreover, the bagging method of such ensemble learning models allows to downsample the majority class, thus growing trees on more balanced bootstrap samples.

The latitude and longitude of a tweet are important features according to figure 6. However, when training a random forest classifier on geographical tweet information only, the model performance is bad. This result corresponds to the US map in figure 3, which did not show any clear geospatial distribution of the tweet class labels. Further information and figures on this random forest classifier approach can be found in appendix A. To summarize, solely using tweet coordinates is not sufficient to train a random forest classifier on a data frame with limited number of samples.

A. Further Modelling: Tweet Coordinates Only

In this experimental run, a new feature space is introduced. All text information is discarded, and the random forest classifier is solely trained on the tweet coordinates. Apart from that, the machine learning workflow is unaltered. Table 3 and figure 8 clearly prove that such a model is almost exclusively predicting every tweet as *normal*, since the class label is the most prominent one of the imbalanced dataset.

	Precision	Recall	F1 score	Support
abusive	0.56	0.12	0.19	43
hateful	0.00	0.00	0.00	17
normal	0.67	0.95	0.79	160
spam	0.00	0.00	0.00	20
weighted avg	0.55	0.65	0.56	240

Table 3: Classification report

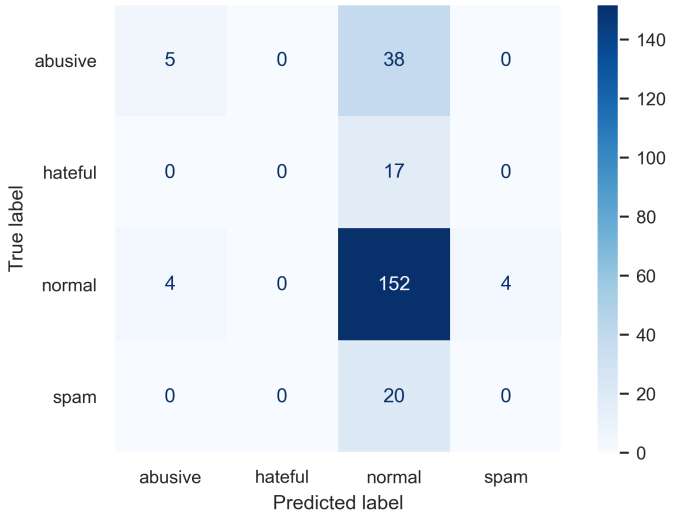


Figure 8: Confusion matrix

B. Code & Data Availability

The programming part of this machine learning project (and all necessary files to run it) can be found on GitLab[5]. The code is written in python and stored as jupyter notebooks.

[5] https://gitlab.com/vegan_schnitzel/twitter-sentiment-analysis

References

[1] Kurt Wagner. *Twitter Penalizes Record Number of Accounts for Hate Speech*. TIME. URL: https://time.com/6080324/twitter-hate-speech-penalties/ (visited on 07/28/2021).

[2] Ben Collins and Brandy Zadrozny. *Twitter permanently suspends President Donald Trump*. NBC News. URL: https://www.nbcnews.com/tech/tech-news/twitter-permanently-bans-president-donald-trump-n1253588 (visited on 07/28/2021).

[3] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Antigoni-Maria Founta et al. "Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior." In: *11th International Conference on Web and Social Media, ICWSM 2018*. AAAI Press, 2018.

[5] *Ensemble Methods*. scikit-learn. URL: https://scikit-learn/stable/modules/ensemble.html (visited on 10/02/2021).

[6] *Random Forest/Bagging*. In: *Wikipedia*. Page Version ID: 1044042624. Sept. 13, 2021. URL: https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=1044042624#Bagging (visited on 10/02/2021).