Real or Not: How Does the Use of Natural Language Processing Improve the Classification of Disaster Tweets?

Machine Learning, Group 97. Vrije Universiteit Amsterdam

Hongyu He Beata Haracewiat Matthias Debernardini Alexander Balgavy
Soufiane Jounaid

March 30, 2020

Abstract

Twitter is home to 321 million active users from all over the world. It is 36th on Alexa, and the verb "tweet" is now a mainstay in the English vernacular. Today, when news breaks, it breaks first on twitter. Often, when a disaster emerges, tweets begin to flood the Twittersphere about the situation. For emergency dispatchers, becoming aware of an accident as it unfolds and proactively deploying resources to the location can have a profound impact on the collateral damage. However, this necessitates the need for an intelligent system that can process and interpret tweets because no person or group of people can read and understand the constant stream of tweets. Because these tweets are mainly composed of natural language, the intelligent system needs to process the natural language so that it can output actionable suggestions.

1 Introduction

In this report, we describe the NLP (Natural Language Processing) pipeline to compete in the "Real or Not? NLP with Disaster Tweets" Kaggle competition. The goal of the competition is to decide, given a tweet, whether it pertains to a disaster or not. The NLP pipeline with the highest accuracy wins the competition. The prepared dataset was provided by Figure Eight Inc. and the pipeline consisted of Python libraries.

1.1 Hypothesis

Our research will focus on exploring the difference in the models' performance depending on the feature set. Similar research has shown the potential of NLP feature extraction techniques to improve the predictions [1]–[3]. Thus, our research will focus on investigating the hypothesis: *Linguistic features extracted using the natural language processing pipeline improve the classification of disaster tweets compared to classic derivable features.*

1.2 Approach

In order to investigate our hypothesis, we have decided on a structured approach. As the first step, we performed pre-processing on the provided training data, which included data analysis and data cleaning. This step consisted of investigating the class imbalance, dealing with missing values and seeking correlations between features and classes among others. Based on the latter we have built a set of features which could be potentially useful for the classification. These will be later used for the training pipeline. For model evaluation, cross-validation with 5 stratified folds was used. This ensured that every instance from the dataset has been used for the training at least once. Moreover, the folds contained approximately the same proportion of the target value as the original dataset. To form the final conclusions, we evaluated our models on the test data. The respective steps along with the reasoning are described in-depth in the following sections.

2 Method

2.1 Data preprocessing

The dataset required some initial preprocessing. Table 1 below shows the columns present in the training dataset, along with the amount of instances that did not have a value for the given column. The keyword is a word associated with (or representative of) a given instance, the location is optionally entered by the user when they are authoring a tweet, the text is the content of the tweet, and the target is the label of the tweet.

The first observation we made was that out of the total 7613 instances, none of them were missing a target label. This meant that all of the data was usable for supervised training and validation. Furthermore, all instances have an entry for the text column, so this column would be useful for feature extraction.

Column	Amount of instances with a null value for the column
keyword	61
location	2533
text	0
target	0

Table 1

We then saw that 61 instances were missing a value for the keyword column, which is approximately 0.8% of the whole dataset. Out of those 61 instances, 42 were labelled as positive and 19 as negative. Based on this, we decided to fill in the missing data by imputation: for the positive instances, we filled in the mode of the keyword for positive instances, and likewise for the negative instances. We reasoned that this was fine, as it would not really affect the dataset as a whole, since only a small percentage of the dataset had a missing value for that column. Furthermore, there was no significant difference between the amount of positive versus negative instances that had a missing value for 'keyword'.

In the case of the location, around one third of the instances had a null value for this column. Unfortunately, in this case it was not possible to do much: using the mode value for the location would not make sense, and we did not have enough other features which we could use to train a model to predict the location of an instance (predicting this from just the text would be difficult). Moreover, missing data for the location could also be expected in the real-world use case. Therefore, we decided to leave the empty values empty, and potentially create a model that could use the location values when present.

Next, we had to deal with duplicate instances in the dataset. There were no strict duplicates (i.e. duplicates where every column was duplicated); however, there were 69 instances that contained repeated text values. Out of those 69 rows, 37 rows contained the same text, yet were labelled differently. As this would be confusing for our model during training, we decided to simply drop rows with duplicated labels, because we did not have a good method to keep only the rows that were labelled correctly (doing this manually could lead to errors caused by subjectivity). Dropping duplicate rows did not have a major impact on the size of the dataset, as the operation only removed 110 instances, or around 1.44% of the data.

Some cleaning of the text values was also necessary, as they contained HTML entities and abbreviations that would not be understood by our natural language processing (NLP) pipeline, but could still carry meaning. In particular, we replaced HTML entities (e.g. `&`) with their English-word equivalent (e.g. "and"). We also expanded abbreviations to their full form, e.g. "ER" to "emergency room".

Finally, we noted the class imbalance of the dataset, which is shown in Table 2:

Class	Number of instances	As percentage of total number of instances
1 (is a disaster)	3198	42.62%
0 (is not a disaster)	4305	57.38%

Table 2

There were approximately 14.76% more negative instances than there were positive instances. The class imbalance problem arises when one class is represented by significantly more instances than the other class [4]. In our case, we decided that this distribution was tolerable and would not lead to a major problem, given the size of the data.

2.2 Extracting features from the text

Some of the columns, such as the keyword, would be immediately usable as a feature. However, the text still required further processing.

Some of the text values contained character sequences that had special meanings when used on Twitter. For example, alphanumeric strings beginning with an "@" symbol are "mentions", and are used to directly refer to another user's Twitter account in a message. Alphanumeric strings beginning with a "#" symbol are "hashtags", and are used to categorise a message (a user may add one or more categories which they deem fitting for their message). In our processing, we decided to also treat these two kinds of alphanumeric strings accordingly, as their presence could be a useful feature.

Overall, we noted the presence of several kinds of entities in the text: mentions of relief organisations, mentions of news organisations, other mentions, hashtags, organisations (companies, agencies, institutions, etc.), geopolitical entities (countries, cities, states), and facility entities (buildings, airports, highways, bridges, etc.). For each mention present in a message, we also noted whether the mentioned user was a known news organisation (e.g. CNN, BBC) , or a known relief organisation (e.g. MSF, Oxfam), as this could be a good indicator of whether the message referred to a disaster.

We checked whether the hashtags in a message and the message itself contained a synonym of the word "disaster" or "accident"; we obtained these synonyms from Roget's 21st Century Thesaurus. These synonyms were split up into 'levels': level 1 synonyms were synonyms deemed closer in meaning to the word 'disaster', and level 2 were less similar. We also extracted the subject, verbs, and objects for each sentence in each message, as these three grammatical features are generally part of the independent clause, and are useful in determining the topic of a sentence or message [5]. Finally, we checked whether the text contains words that could be used to describe 'damage', such as 'explode' or 'rupture'.

2.3 Feature selection

After processing the data, some analysis was necessary to determine which features would be useful in a model. To begin, we computed for each potential feature the amount of messages that had a value for that feature; you can see the results in Table 3 in the appendix.

Some of the features we extracted do not occur very frequently in the dataset. For example, only 0.03% of the tweets in the dataset have a hashtag that is a level 2 synonym of 'disaster', which amounts to two tweets. The same is true for level 1 synonyms of 'disaster', which occur in 0.12% of the hashtags of the tweets (9 instances). This means that they may not be useful features to consider, as they do not occur very often in general.

Similarly, mentions of news organisations and relief organisations do not occur very often either: they are contained in 0.95% (71 instances) and 0.04% (3 instances), respectively. One reason for this may be that the list of Twitter accounts that we are using is too limited, and the tweets in our dataset simply refer to other news or relief organisation accounts. Unfortunately, there is no straightforward way of compiling a fully complete list of such accounts. Moreover, these features may still be useful, as if they are present, they may immediately be a strong indicator of the labelling of the message; to determine whether this is the case, further analysis was necessary, as discussed below.

When selecting features for inclusion in the model, we created plots and attempted to find potential correlations. Firstly, we looked at only the columns that were already provided in the dataset. For the keywords, we observed that generally, a keyword is not equally as often associated with positively labelled tweets as it is with negatively labelled tweets. This is shown in Figure 5 in the appendix, which is an excerpt from the full plot: the turquoise section (representing the amount of positive instances) and the red section (representing the amount of negative instances) are rarely the same size. The full plot is not included here for brevity; nevertheless, this excerpt should serve to show that the keyword can be a good indicator of the label. There are some discrepancies that are worth mentioning, namely that different forms of the same word (e.g. noun, adjective, adverb, plural, singular) may indicate different class labels. For example, the keyword 'deaths' generally indicates that an instance should be labelled as positive, whereas for keywords 'death' and 'dead', it is more likely that the instance is negative. This is probably because the English language contains many words whose meanings change depending on the context, or whether they are meant figuratively.

The location column was not useful, for two reasons. Firstly, it is entered by the user, so it either may be spelled in an unusual way (e.g. "M!$$!$$!PP!" meaning "Mississippi"), or even be a valid location. This is not trivial to clean or analyze, and due to time constraints, we did not investigate ways to do so. Secondly, there is no location present for one-third of the training dataset. Therefore, we decided not to consider this in our models.

Although, as shown above, less than one percent of the dataset contained a direct mention of a news organisation's account, it could still be used as a feature. As can be seen in Figure 6 in the appendix, if a message mentions a news organisation, it is more likely that it should be labelled as positive. Therefore, if present, we could potentially use this feature to predict a positive label for the instance.

Next, we analyzed the presence of level 1 synonyms of 'disaster'. As is shown in Figure 7 in the appendix, the difference is not that significant, but it is still somewhat more likely for the tweet to be labelled as positive when it contains a synonym than when it does not. Similar plots were created for level 2 synonyms and words relating to damage; these plots are omitted for brevity, but the results show that these two features would not be a good indicator of the instance being positive.

We also observed that tweets labelled as positive have a higher number of recognised entities (geopolitical entities, organisations, and facilities) in the text. This can be seen in Figure 8 in the appendix, where if a tweet contains five or more entities, it becomes more likely that it is a positive instance. However, there are only a handful of tweets with seven or eight entities, so it is possible that this observed 'relationship' may actually be a result of the small size of the training dataset.

Finally, we investigated the role of hashtags in the classification of tweets. As we said above, not many hashtags contained synonyms of the word 'disaster'. However, for those that did, it is clear from Figure 9 in the appendix that they were mostly labelled as positive.

We also considered the number of hashtags present in a tweet, and observed that as the number of hashtags increases, it is generally more likely that the tweet should be labelled as positive. However, this is not a clear relationship, as there are some outliers. Figure 10a and Figure 10b in the appendix show the same plot, but Figure 10b was zoomed in by plotting only tweets with four or more hashtags to make potential relationships more visible. These figures show that there is a higher proportion of positively labelled tweets that contain seven or more hashtags, but there are some outliers, such as at nine and eleven hashtags. Furthermore, similarly to the number of entities, it is important to consider that this apparent relationship may actually result from the small size of the training dataset. Nevertheless, the number of hashtags could still be a useful feature for label prediction.

Another reason for the hashtags in a tweet to potentially be a useful feature is that the most popular hashtags for the labels differ significantly. As can be seen in Figure 11a and Figure 11b in the appendix, out of the 30 most commonly occurring hashtags in each class, only four are the same for positive (turquoise) and negative (red) instances. In addition, those four hashtags are relatively generic: "best", "hot", "news", and "prebreak". This means that if a specific hashtag occurs in a given text, it is a likely indicator of the label that should be predicted for that instance.

3 Experiments

In an effort to carry out a proper experiment, we applied a variety of methods to our data. From Natural language processing to non-linear models, the following subsection describes the motivation behind the choices we made in this department and outlines our performance predictions.

We experimented with two techniques for our baseline models, excluding our linguistic features of course, because one of the primary questions we aim to answer in this paper is whether the added linguistic features we extracted using the natural language processing pipeline improve the classification of disaster tweets compared to classic derivable features.

3.1 First attempt: Logistic regression

For our first baseline model, we wanted a simple and statistical approach. Logistic regression serves our purpose nicely, taking multiple numerical features and outputting a binary response probability. In this specific case we employ this model with 2 feature extraction methods: Bag of words and Term frequency-inverse document frequency (TF-IDF). Bag of words is a simple text feature extraction method, it builds a vocabulary of unique words it encounters throughout the input and keeps a measure of their occurrences. TF-IDF takes the concept a step further by multiplying the measure (term frequency) by the term's inverse document frequency: in our setting this translates to multiplying the frequency of a word in a tweet by a measure of how rare or common it is in the corpus (set) of tweets.

Term frequency Inverse document frequency in our setting is defined as:

$$\text{TFIDF} = \frac{\text{number of occurrences of a term}}{\text{total number of words in the tweet}} \log\Big(\frac{\text{total number of tweets}}{\text{number of tweets containing the term}}\Big)$$

The logistic regression model then uses a logit function to estimate probabilities. This function is defined as:

$$F(x) = \frac{1}{1 - e^{-x}}$$

where

$$x = w_0 + \sum_{k=1} nw_k \cdot x_k$$

w_k are the regression coefficients of the model and x_k are the features. The results of this model can be depicted in a confusion matrix, as shown in Figure 12 in the appendix. These results are particularly bad, considering we have a large amount of false positives, meaning that a large number of non disaster related tweets were classified as disaster, if we consider cost imbalance, this is not as bad as having the opposite scenario. We can conclude on this basis that this regression model is indeed a bit brave (classifies most data points as positive).

3.2 Experimental setup: The Pipeline

To enable several experiments using different classification methods, some sort of pipeline needs to be established. To turn our data set in training data points, we assemble together a 5 component pipeline.

The Data preprocessing elaborated in previous sections is the first part of our pipeline,this section handles data cleaning and exploration, i.e generating more features. Consequently, the training data points become more and more complex throughout the pipeline, both in terms of data type and format. At the end of the process, the training data contains numerical, textual and categorical labeled data. Therefore, the second part of the pipeline is a column transformer from the Scikit-learn library which is responsible for serializing all kinds of extracted features from every column into scalar values. Then, the third part of the pipeline is a scalar which normalises these scalar values to the same range in order to prevent bias.

The experimental part is the fourth part, which is regarded as a slot where different classification methods are "plugged in" to produce predictions. Finally, the last part of the pipeline is the evaluation of the predictions, for this matter we used cross-validation with 5 stratified folds for all experiments, furthermore the loss measurement is based on the mean square error (MSE) of the classifiers. We rely on this measure as a primary means to compare classifiers and test our hypothesis In an effort to carry out a proper experiment, we applied a variety of methods to our data. From Natural language processing to non-linear models, the following subsection describes the motivation behind the choices we made in this department.and outlines our performance predictions.

We experimented with two techniques for our baseline models excluding our linguistic features to help us answer the research question.

3.3 Baseline method: Linear SVC with derivable features (non-NLP)

The method that we used for our baseline is Support Vector Classifier (SVC), a statistical model built using support vector machines. The goal behind SVM is to find the decision surface that maximizes the margin between the data points of the two classes (disaster related or not in this case). This decision surface is best illustrated through the following figure from [6]:

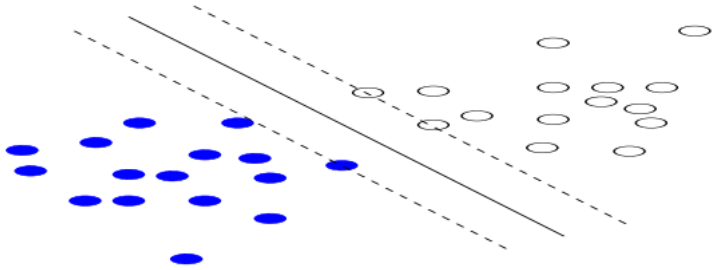


Figure 1: A decision line (solid) and a margin (distance between dashed lines), both represent the decision surface.

Mathematically, this decision surface (for linearly separable data) can be written as a hyperplane [6] where the vector x is a data point we want to classify whereas w and b are learned from the training set :

$$vec(w_i) \cdot vec(x_i) - b = 0$$

Let y_i be the classification of a data point $vec(x_i)$. If it evaluates to +1 for positive and -1 for negative, the SVM optimization problem can be simply written as the following two inequalities:

$$\begin{cases} vec(w_i) \cdot vec(x_i) - b \geq +1, & \text{for } y_i = +1 \\ vec(w_i) \cdot vec(x_i) - b \leq -1, & \text{for } y_i = -1 \end{cases}$$

One way to motivate this choice for our task would be to showcase comparative studies involving SVM and text classification, and indeed we do provide a few studies showing that Support Vector Machines can provide state-of-the-art performance in text classification tasks [6], [7]. We will show later that our Linear variant of SVC however loses horsepower against its non linear counterparts, this is partly due to our added NLP features breaking the linear separability property of our dataset. More on this in the next section.

3.3.1 Feature Pre-elimination

Before deploying the linear SVC method, we were eager to evaluate the usefulness of some features. To this end, we set up a quick comparison (not using our full pipeline) comparing the results of using the id feature or not. As expected, The metrics shown in table Figure 13 demonstrate that the id feature has a negative impact on the prediction, we decided to omit it for later stages.

3.3.2 Exploiting Feature Space with Kernel Tricks

Before we proceed with adding more and more features, we need to uncover some properties of the training data. In this subsection we employ a variety of kernels for our SVC model (using kernel trick), furthermore based on the results we will be able to make assumptions about the shape and condition of the data.

First, an assumption is made that the feature space before extracting more features should be linearly separable. To prove this, several experiments using SVC with 3 kernel types, rbf [8], polynomial and sigmoid, are conducted. It turns out, based on the results shown in table Figure 14 and Figure 15, this is indeed the case that the linear kernel outperforms other 2 non-linear kernels using only basic features serialized from the original dataset. Our assumption holds so far. Furthermore, based on the training set accuracy, the SVC with `poly` kernel seems to have overfitted and the overall standard deviation of the predictions is quite high.

3.4 Non-baseline methods: (NLP) Entity-related Features

Without using any NLP-related features, the performance of the prediction is about 50 - 60% accuracy if taking the variations into account. The current scores are by no means close to yielding a useful model. However, since the feature space is still rather naïve, exploring more features with natural language processing seems promising.

To build our NLP component, we used one of the best performing NLP libraries at the moment, SpaCy [9], [10]. Since concrete entities are reflected through the tweets, we will use these as central features and derive the other features from them. Following this protocol, we end up with 5 features : the entities themselves, the labels/types of the entities, the words dependent on the entities (Part of speech), the positions of the entities in a sentence (Part of speech) and the derived words of the entities.

3.4.1 SVC - The Second Try

Having the previous performances on hand, more experiments were undertaken using SVC. Similar to the previous method, an experiment using the linear kernel was first conducted. Then, experiments using non-linear kernels are followed. The results of experiments in question are summarised in Figure 16, Figure 17, Figure 18, and Figure 19 in the appendix.

As a corollary, the performances of most of the non-linear kernels have surpassed that of the linear kernel. We, therefore, conclude that, after adding the 5 entity-related features, the feature space is no longer linearly separable. Most importantly, the scores of the predictions have been improved a lot which gives away the possibility of having further improvement by adding more relevant features. In addition, the prediction produced by the polynomial kernel is exceptionally bad, therefore we decided to omit it from the next experiment.

3.4.2 Classification ensemble with Adaboost

Although the performance got improved by adding those entity features, the bias of the results remains huge as the accuracy is still below 65%. Thus, ensemble methods with boosting [11] in turn become the next attempt with the purpose of reducing bias and improving accuracy. An adaboost classifier is then plugged into the pipeline. The results of this experiment are illustrated in Figure 20 in the appendix.

Surprisingly, the result is not good at all as it barely reaches the same accuracy as that of the SVC with the polynomial kernel. Therefore, this method was discarded at this phase.

3.4.3 Bagging with Random Trees

Decision trees are by nature one of the most intuitive machine learning algorithms humans ever designed, most people have at some point employed this line of reasoning while trying to make a decision, hence its grand popularity in classification. Let's deviate slightly from the statistical math heavy illustrations and turn to a simple example that shows how this decision trees function in Figure 2:

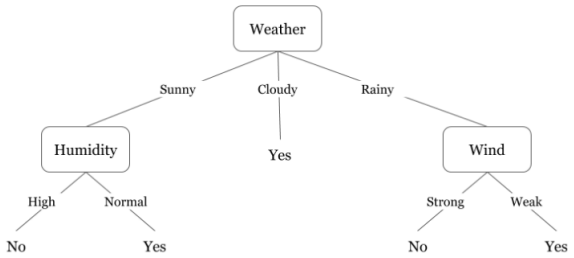


Figure 2: Graphical representation of a decision tree on the problem: Should we play badminton or not based on weather conditions

As you can see, a tree is a collection of if-then conditional rules to reach a binary decision "yes-no" (we only illustrated binary classification for our purpose, but it can also classify labels) [12].

For our purpose, we are using decision trees in an ensemble, more accurately, a random forest, one of the most famous bagging methods, particularly famous for feature selection, a deciding factor in our decision to conduct experiments using this method. This model works by spawning multiple decision trees each with a random subset of features and training data, it then pools the predictions together and takes the average, take this as a multitude of humans making predictions about a specific matter, if the matter is hardly predictable, pooling and averaging predictions can help reach a more precise result [13]. The prediction results using the Random Forest classifier with our current limited set of 9 features are demonstrated in tables Figure 21, Figure 22, and Figure 23 in the appendix.

As the results show in the tables, using the random forest method improves the accuracy to about 68% which is the most promising result we have so far. However, when looking at the variance and the training set accuracy, it shows us that the model is prone to overfitting, especially when the number of bagged trees reaches about 500, and the accuracy starts dropping. For this reason, we decided in the following experiments, that a maximum depth for each considered decision tree should be set.

3.5 Feature Post-processing: selecting the right features

At this stage, lots of features have been extracted and utilised. As elaborated in the data preprocessing section, both quality and quantity of the features play a vital role in the classification process. Therefore, we decided to conduct feature post-processing before adding more features.

3.5.1 Imputation and Feature Importance

In the conducted experiments, missing values are filled by dummy values or labels like "missing" or "unknown". However, as discussed in the previous section, missing keywords should be replaced by the mode of each class, the real disaster class and the fake disaster ones. Thus, a simple imputation by replacing those dummy fillers with corresponding mode values is implemented. However, surprisingly, the prediction accuracy compared to before shows no noticeable improvement. This is shown in Figure 24 in the appendix.

Additionally, we examine the feature importance by looking into the top-10 features that are selected by the random forest classifier. In the appendix, Figure 25 shows the top-10 features without text data and Figure 26 illustrates the important features in the text data. After analysis, each elected feature category plays an important role in the classification process.

3.5.2 Polynomial Feature Extension

Without using more NLP features, the simplest way to achieve massive features is to create new features based on existing features. One of the approaches is to generate polynomial and interaction features the way resembles that of the polynomial kernel trick. To this end, a polynomial feature generator from Scikit-learn library is inserted into the middle of the pipeline.

Originally, 38278 features including all text data have been used in previous experiments. After inserting a polynomial order-two feature generator, the number of features of each data point is extended to 767869266 (about 770 million). However, when feeding this amount of features into the random forest (our current best model) the training process could not be accomplished after 5 hours of training in the supercomputer DAS5 [14], we thus decided to go with the SVC model with the rbf kernel. The result is shown in table Figure 27 in the appendix.

To our surprise, the performance did not improve as expected. Therefore, we conclude that more features do help in terms of the classification only if the added features are relevant and carefully selected.

3.6 Final Model Selection: Using full NLP feature set

Finally, after several experiments, the ideal computational models that fit into our scenario are narrowed down to Random Forest. At this final stage, we will make use of all NLP features described in Table 3 together with the 5 entity-related features to evaluate the models.

3.6.1 The Black-Box - Neural Net

Before our final run of the Random Forest model, it would be really interesting to see how all the features perform in a black-box neural network. We thus chose the most straightforward NN method, Multi-layer Perceptron (MLP) classifier [15] to see whether there would be any promising results. Again we provide a usability case study [16] on why this model would be a promising choice. The results of this experiment are shown in Figure 28 and Figure 29 in the appendix.

As illustrated in the resulting tables, there is indeed small performance gain by enlarging the neural network. However, considering the exponential increase of the training and evaluation time, we have to give up this method and stick with the Random Forest model.

3.6.2 Tuning Hyperparameters of the final model - Random Forest

Since examining the best number of trees and the maximum number considered for each tree are known to be the most time-consuming part, we, therefore, make tuning of these two parameters a stand-alone process by drawing the out-of-bag (OOB) error curve [17] in order to see the convergence region so that we can to narrow down the range of the parameters and in turn shrink the random search grid.

By setting up 3 pipelines with different `max_features` parameters, we continually tune their `n_estimators` parameters which resulted in the OOB error curve showing in Figure 3 in the appendix.

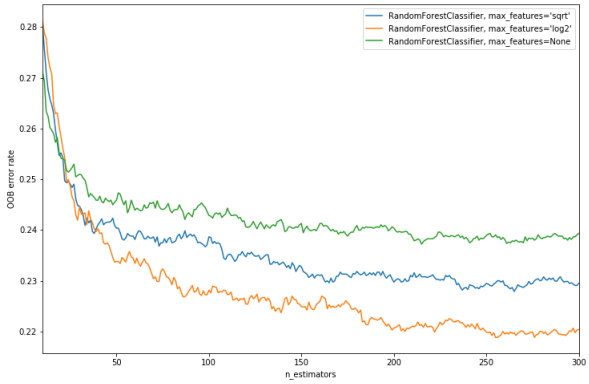


Figure 3

Based on the OOB error curve that we have, the best parameter for `max_features` would be `sqrt`. For the `n_estimation` parameter, there is a clear convergence around 250. Thus, we finally shrink down the parameter distribution into the following,

```
random_distro ={
     'bootstrap': [True, False],
     'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
     'max_features': ['sqrt'],
     'min_samples_leaf': [1, 2, 4, 6, 8],
     'min_samples_split': [2, 5, 10],
     'n_estimators': [220, 250, 270]
}
```

Further, a randomized search on hyperparameters is conducted using the Scikit-Learn library. The result of the random search is shown in Figure 30 in the appendix.

```
{'n_estimators': 250,
 'min_samples_split': 6,
 'min_samples_leaf': 4,
 'max_features': 'sqrt',
 'max_depth': 80,
bootstrap': True}
```

By comparing the base model of the random forest whose performance is shown in table Figure 31 in the appendix, an improvement on the cross-validation accuracy of about 3% can be concluded and, perhaps most importantly, according to the training set accuracy, the tuned model less prone to be overfitting.

4 Results and evaluation

Table Figure 4 is a tabulated summary of the results from the models (as described in the experimental section) with the following; Accuracy of the model, Cross-Validation MSE and its standard deviation, the sum of the evaluation and training time and the section where this is discussed.

After evaluating all these models, we decided to explore the random forest further. Despite the NLP producing satisfactory results, the time to finish is prohibitive.

5 Conclusions

Out of all the models, Random Forests produce the best results (especially with hyperparameter tuning). This is due to the high accuracy and the short training times. The five-part pipeline serves as an excellent infrastructure for future experiments. The linear SVCs performed well, especially when the feature space is not linearly separable (baseline and non-baseline sections). Ensemble methods were beneficial when the results have high variance and bias. A higher number of features do help in terms of the classification only if the added features are relevant and carefully selected. While black-box methods hold a large amount of potential, the costs are very high compared to others.

Method	Training Accuracy (%)	CV MSE (%)	CV MSE StDev (%)	Total Time (h:m:s)
MLP (100 hidden layers)	99.8	64.9	9.0	00:52:31
MLP (500 hidden layers)	99.8	66.5	5.9	13:42:40
Random Forest (20 trees)	98.7	68.2	6.0	00:00:54
Random Forest (200 trees)	99.5	69.2	6.6	00:06:28
Random Forest (500 trees)	99.5	68.9	6.2	00:14:30
Random Forest w/ HP tuning	84.4	71.3	6.0	N/A
Linear Kernel	97.5	61.2	8.7	00:12:23
RBF Kernel	90.2	65.7	7.0	00:11:10
Polynomial Kernel	87.3	65.5	3.5	00:12:52
Sigmoid Kernel	76.8	59.0	3.9	00:10:16

Figure 4

6 Discussion

In this report, we demonstrate the results of various machine learning models in the task of natural language classification. The end goal was to explore potential solutions to the problem of classifying tweets if they pertain to disasters or not. With some success, we demonstrated that this is possible but not with high levels of confidence. This lack of success stems from the problem statement being overly broad, in that it is not precisely clear what a disaster is and what is not. There is not an authoritative source of truth that defines what is and is not a disaster because the meaning of this word is different according to different people. Rightly so as different people have different perspectives and sets of values. For example, to Donald Trump, the North American Free Trade Agreement is a disaster. However, citizens of Mexico would say that it is an overwhelming success that has brought wealth and prosperity to their nation. So is NAFTA a disaster or not? To Americans that worked in Automotive part suppliers (brakes, engines, etc.) NAFTA was a disaster since it decreased the number of available jobs in the country, but the opposite is true for Mexico and other trading partners of the US. This begs the question, to whom would such an intelligent system serve? Americans? It could, but it would be unfair. Therefore the scope of this problem needs to be narrowed down to something more specific that is unquestionably objective. Without the problem adequately modeled, there is no set of equations that can consistently deliver accurate results for all cases.

One way to improve this system is to target, for example, natural disasters or automotive accidents. An additional improvement would be to include a knowledge base that allows the system to understand that, for example, Kansas is situated in tornado valley, and so if a tweet becomes geotagged from Kansas, then it is more likely about a disaster. This knowledge-base could also have a memory, that if there is a known disaster occurring in a geographical area, then tweets from that area will be much more likely to be about a disaster.

If this system were to be designed and implemented, then a thorough discussion about the acceptable level of risk associated with false positives and negatives. As in how costly would a mistake be from this system trying to classify tweets? It is conceivable that if the risks are low, then it does not warrant a machine learning solution and all the complexity that comes from it. Along this vein of discussion, perhaps no classifier is the best classifier. Twitter could conceivably reserve a keyword in their service to flag to some trivial system that a particular tweet is, in fact, about a disaster. If a user wants to let others know that they are in a disaster, then they can use it. This system could be abused or misused, but the same holds for the machine learning models.

Regardless of the implementation, this system would be incredibly useful to organizations that dispatch critical resources to regions affected by disasters. Such a system would allow for a more automated and proactive process that could make the difference in having a memorable disaster or a mundane event.

References

[1] D. T. Nguyen, K. Al-Mannai, S. R. Joty, H. Sajjad, M. Imran, and P. Mitra, "Rapid classification of crisis-related data on social networks using convolutional neural networks," *CoRR*, vol. abs/1608.03902, 2016. arXiv: 1608.03902. [Online]. Available: http://arxiv.org/abs/1608.03902.

[2] K. Stowe, M. J. Paul, M. Palmer, L. Palen, and K. Anderson, "Identifying and categorizing disaster-related tweets," in *SocialNLP@EMNLP*, 2016.

[3] K. D. Rosa, R. Shah, B. Lin, A. Gershman, and R. E. Frederking, "Topical clustering of tweets," 2011.

[4] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," vol. 6, pp. 429–449, 2002, 5, ISSN: 1571-4128. DOI: 10.3233/IDA-2002-6504.

[5] R. Nordquist, *Subjects, verbs, and objects*, 2020.

[6] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999, pp. 42–49.

[7] J. Dumais, J. Piatt, J. Heckermann, and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *7th International Conference on Information and Knowledge Management*, 1998, pp. 148–155.

[8] K. O Sullivan, "Comparing the effectiveness of support vector machines and convolutional neural networks for determining user intent in conversational agents," 2018.

[9] M. Neumann, D. King, I. Beltagy, and W. Ammar, "Scispacy: Fast and robust models for biomedical natural language processing," *CoRR*, vol. abs/1902.07669, 2019. arXiv: 1902.07669. [Online]. Available: http://arxiv.org/abs/1902.07669.

[10] *Spacy 101: Everything you need to know*, Accessed: 2020-03-29.

[11] R. Maclin and D. W. Opitz, "Popular ensemble methods: An empirical study," *CoRR*, vol. abs/1106.0257, 2011. arXiv: 1106.0257. [Online]. Available: http://arxiv.org/abs/1106.0257.

[12] P. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Pearson, 2006.

[13] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *International Journal of Computer Science Issues(IJCSI)*, vol. 9, Sep. 2012.

[14] H. Bal, D. Epema, C. Laat, R. Van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff, "A medium-scale distributed system for computer science research: Infrastructure for the long term," *Computer*, vol. 49, pp. 54–63, May 2016. DOI: 10.1109/MC.2016.127.

[15] H. Ramchoun, M. Amine, M. A. Janati Idrissi, Y. Ghanou, and M. Ettaouil, "Multilayer perceptron: Architecture optimization and training," *International Journal of Interactive Multimedia and Artificial Inteligence*, vol. 4, pp. 26–30, Jan. 2016. DOI: 10.9781/ijimai.2016.415.

[16] H. T. Ng, W. B. Goh, and K. L. Low, "Feature selection, perceptron learning, and a usability case study for text categorization," in *SIGIR '97*, 1997.

[17] M. Skocik, J. Collins, C. Callahan-Flintoft, H. Bowman, and B. Wyble, "I tried a bunch of things: The dangers of unexpected overfitting in classification," *bioRxiv*, 2016. DOI: 10.1101/078816. eprint: https://www.biorxiv.org/content/early/2016/10/03/078816.full.pdf. [Online]. Available: https://www.biorxiv.org/content/early/2016/10/03/078816.

7 Appendix

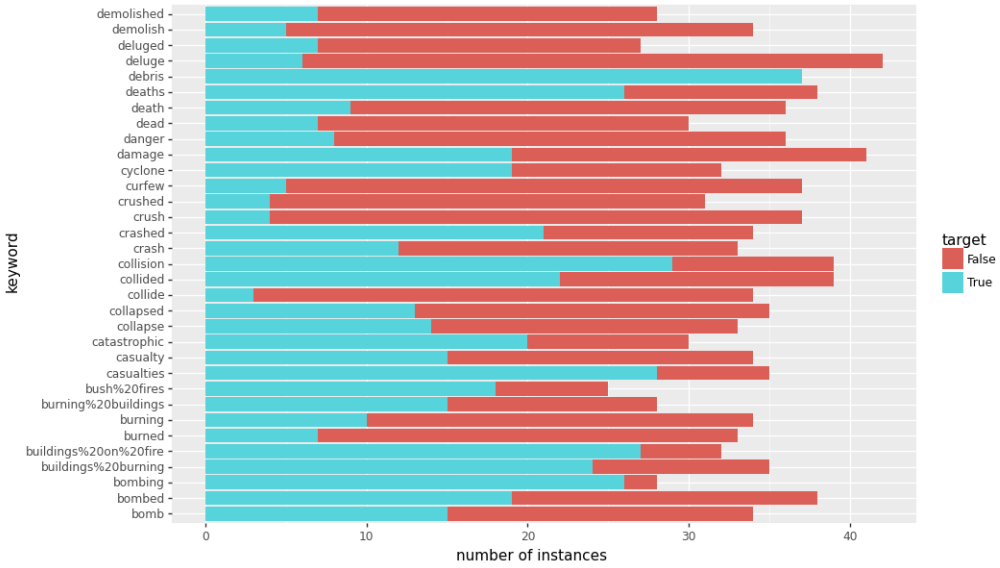


Figure 5

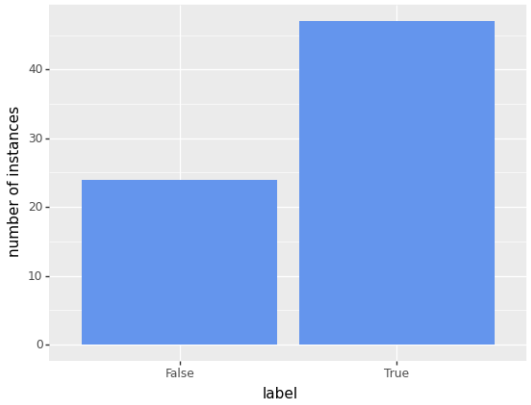


Figure 6

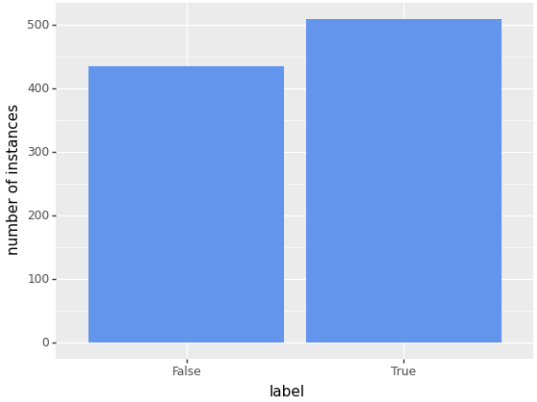


Figure 7

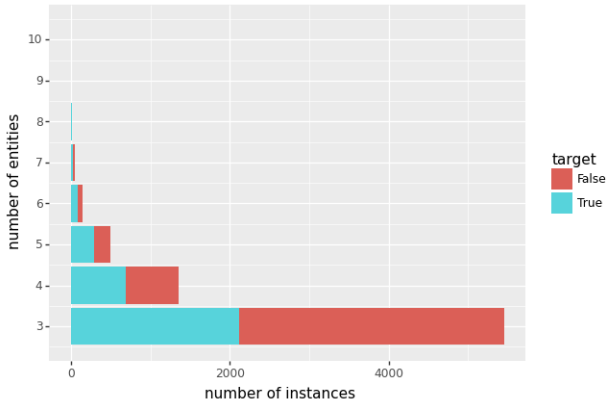


Figure 8

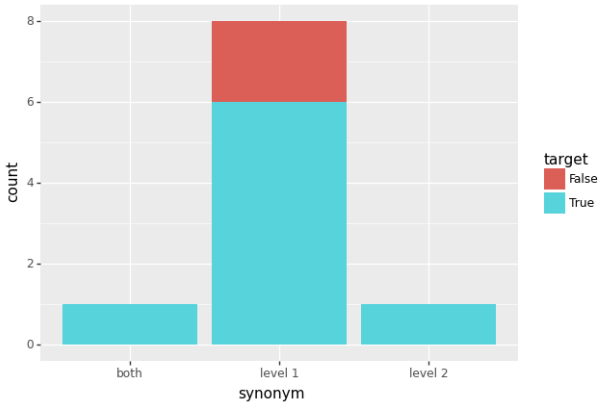
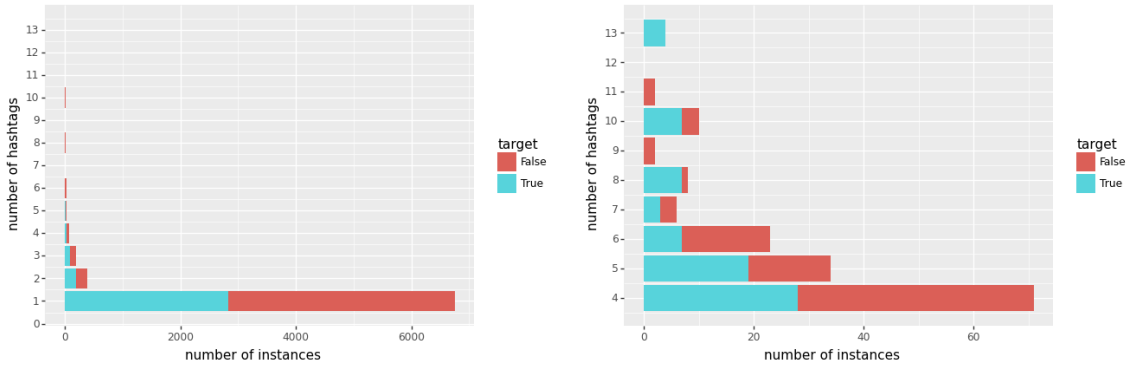
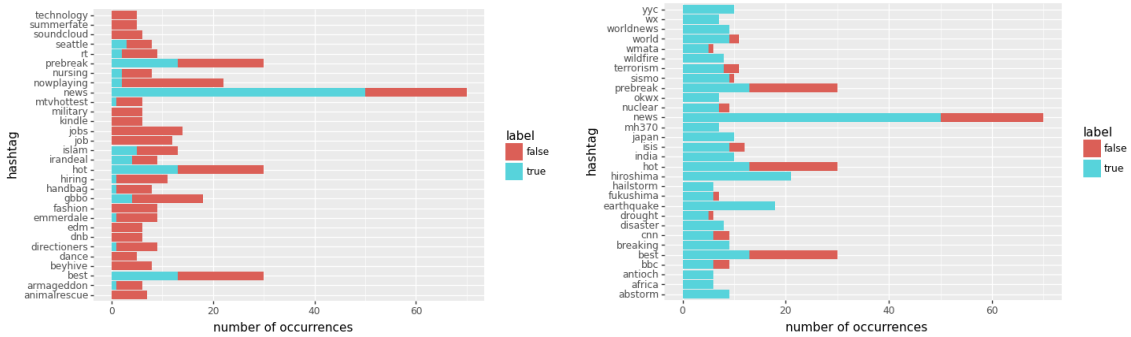


Figure 9



(a) (b)

Figure 10



(a) (b)

Figure 11

Feature	**Percentage of total instances with a value for the feature**
Contains keyword	100%
Contains location	66.92%
Contains hashtag	22.44%
Contains hashtag with level 1 synonym of "disaster"	0.12%
Contains hashtag with level 2 synonym of "disaster"	0.03%
Contains subject	52.9%
Contains verb	79.23%
Contains object	79.39
Text contains level 1 synonym of "disaster"	12.58%
Text contains level 2 synonym of "disaster"	5.1%
Text contains words relating to damage	8.0%
News organisation mentioned	0.95%
Relief organisation mentioned	0.04%
Contains mentions	25.4%
Contains one or more organisations	59.72%
Contains one or more geopolitical entities	15.19%
Contains one or more facilities	1.73%

Table 3

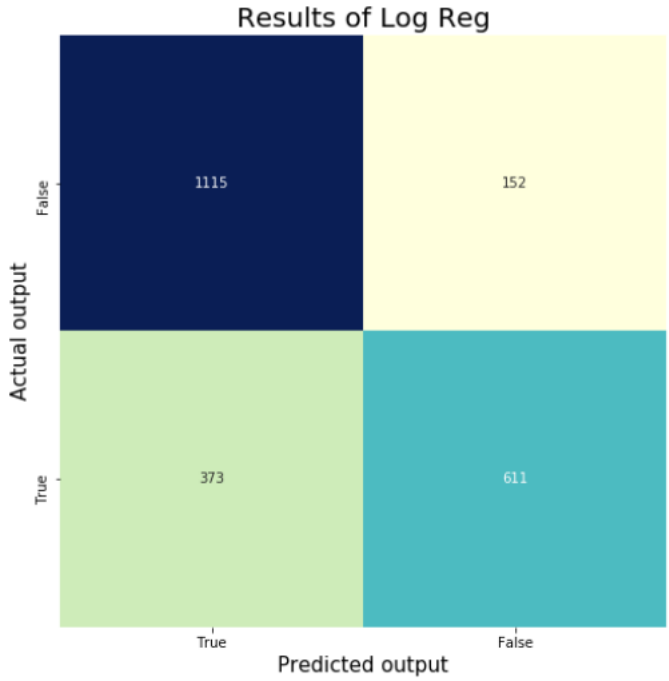


Figure 12: Confusion matrix of 30% of the tweets

1.1 linear kernel with id:

```
                precision    recall  f1-score   support

           0       0.64      0.77      0.70      4342
           1       0.58      0.42      0.48      3271

    accuracy                           0.62      7613
   macro avg       0.61      0.59      0.59      7613
weighted avg       0.61      0.62      0.61      7613

Cross-validation MSE: 0.498 ± 0.276
Training Set Accuracy: 0.903
```

1.2 linear kernel without id:

```
                precision    recall  f1-score   support

           0       0.67      0.69      0.68      4342
           1       0.57      0.54      0.55      3271

    accuracy                           0.63      7613
   macro avg       0.62      0.62      0.62      7613
weighted avg       0.62      0.63      0.63      7613

Cross-validation MSE: 0.626 ± 0.103
Training Set Accuracy: 0.942

Evaluation Time Taken: 00:06:15
Training Time Taken:   00:00:49
```

Figure 13

2. RBF kernel

```
                precision    recall  f1-score   support

           0       0.64      0.76      0.69      4342
           1       0.57      0.42      0.48      3271

    accuracy                           0.61      7613
   macro avg       0.60      0.59      0.59      7613
weighted avg       0.61      0.61      0.60      7613

Cross-validation MSE: 0.614 ± 0.105
Training Set Accuracy: 0.928

Evaluation Time Taken: 00:07:44
Training Time Taken:   00:01:00
```

Figure 14

3. polynomial kernel

	precision	recall	f1-score	support
0	0.55	0.85	0.67	4342
1	0.25	0.07	0.10	3271
accuracy			0.51	7613
macro avg	0.40	0.46	0.39	7613
weighted avg	0.42	0.51	0.42	7613

```
Cross-validation MSE: 0.513 ± 0.082
Training Set Accuracy: 0.966

Evaluation Time Taken:  00:08:17
Training Time Taken:   00:01:15
```

Figure 15

linear kernel

	precision	recall	f1-score	support
0	0.65	0.68	0.67	4342
1	0.55	0.52	0.54	3271
accuracy			0.61	7613
macro avg	0.60	0.60	0.60	7613
weighted avg	0.61	0.61	0.61	7613

```
Cross-validation MSE: 0.612 ± 0.087
Training Set Accuracy: 0.975

Evaluation Time Taken: 00:10:52
Training Time Taken:   00:01:41
```

Figure 16

rbf kernel

	precision	recall	f1-score	support
0	0.65	0.85	0.74	4342
1	0.67	0.40	0.50	3271
accuracy			0.66	7613
macro avg	0.66	0.63	0.62	7613
weighted avg	0.66	0.66	0.64	7613

```
Cross-validation MSE: 0.657 ± 0.070
Training Set Accuracy: 0.902

Evaluation Time Taken: 00:09:49
Training Time Taken:   00:01:21
```

Figure 17

polynomial kernel

	precision	recall	f1-score	support
0	0.64	0.92	0.75	4342
1	0.74	0.30	0.43	3271
accuracy			0.65	7613
macro avg	0.69	0.61	0.59	7613
weighted avg	0.68	0.65	0.61	7613

```
Cross-validation MSE: 0.655 ± 0.035
Training Set Accuracy: 0.873

Evaluation Time Taken: 00:11:37
Training Time Taken:   00:01:15
```

Figure 18

sigmoid kernel

	precision	recall	f1-score	support
0	0.63	0.69	0.66	4342
1	0.53	0.45	0.49	3271
accuracy			0.59	7613
macro avg	0.58	0.57	0.57	7613
weighted avg	0.58	0.59	0.58	7613

```
Cross-validation MSE: 0.590 ± 0.039
Training Set Accuracy: 0.768

Evaluation Time Taken:  00:09:07
Training Time Taken:   00:01:09
```

Figure 19

Adaboost with entity features (100 ensemble)

	precision	recall	f1-score	support
0.0	0.66	0.79	0.72	4305
1.0	0.61	0.44	0.51	3198
accuracy			0.64	7503
macro avg	0.63	0.62	0.61	7503
weighted avg	0.64	0.64	0.63	7503

```
Cross-validation MSE: 0.642 ± 0.083
Training Set Accuracy: 0.801

Evaluation Time Taken:  00:00:33
Training Time Taken:   00:00:03
```

Figure 20

20 trees entity features:

	precision	recall	f1-score	support
0	0.66	0.90	0.76	4342
1	0.75	0.39	0.51	3271
accuracy			0.68	7613
macro avg	0.71	0.65	0.64	7613
weighted avg	0.70	0.68	0.66	7613

Cross-validation MSE: 0.682 ± 0.060
Training Set Accuracy: 0.987

Evaluation Time Taken: 00:00:49
Training Time Taken: 00:00:05

Figure 21

200 trees entity features:

	precision	recall	f1-score	support
0	0.67	0.91	0.77	4342
1	0.77	0.41	0.53	3271
accuracy			0.69	7613
macro avg	0.72	0.66	0.65	7613
weighted avg	0.71	0.69	0.67	7613

Cross-validation MSE: 0.692 ± 0.066
Training Set Accuracy: 0.995

Evaluation Time Taken: 00:05:43
Training Time Taken: 00:00:45

Figure 22: caption

500 trees entity features:

	precision	recall	f1-score	support
0	0.67	0.91	0.77	4342
1	0.76	0.40	0.52	3271
accuracy			0.69	7613
macro avg	0.71	0.65	0.65	7613
weighted avg	0.71	0.69	0.66	7613

Cross-validation MSE: 0.689 ± 0.062
Training Set Accuracy: 0.995

Evaluation Time Taken: 00:12:53
Training Time Taken: 00:01:37

Figure 23

replace keyword by mode using SVC rbf kernel

	precision	recall	f1-score	support
0	0.66	0.79	0.72	4342
1	0.62	0.46	0.53	3271
accuracy			0.65	7613
macro avg	0.64	0.62	0.62	7613
weighted avg	0.64	0.65	0.64	7613

Cross-validation MSE: 0.647 ± 0.086
Training Set Accuracy: 0.909

Evaluation Time Taken: 00:10:46
Training Time Taken: 00:01:19

Figure 24

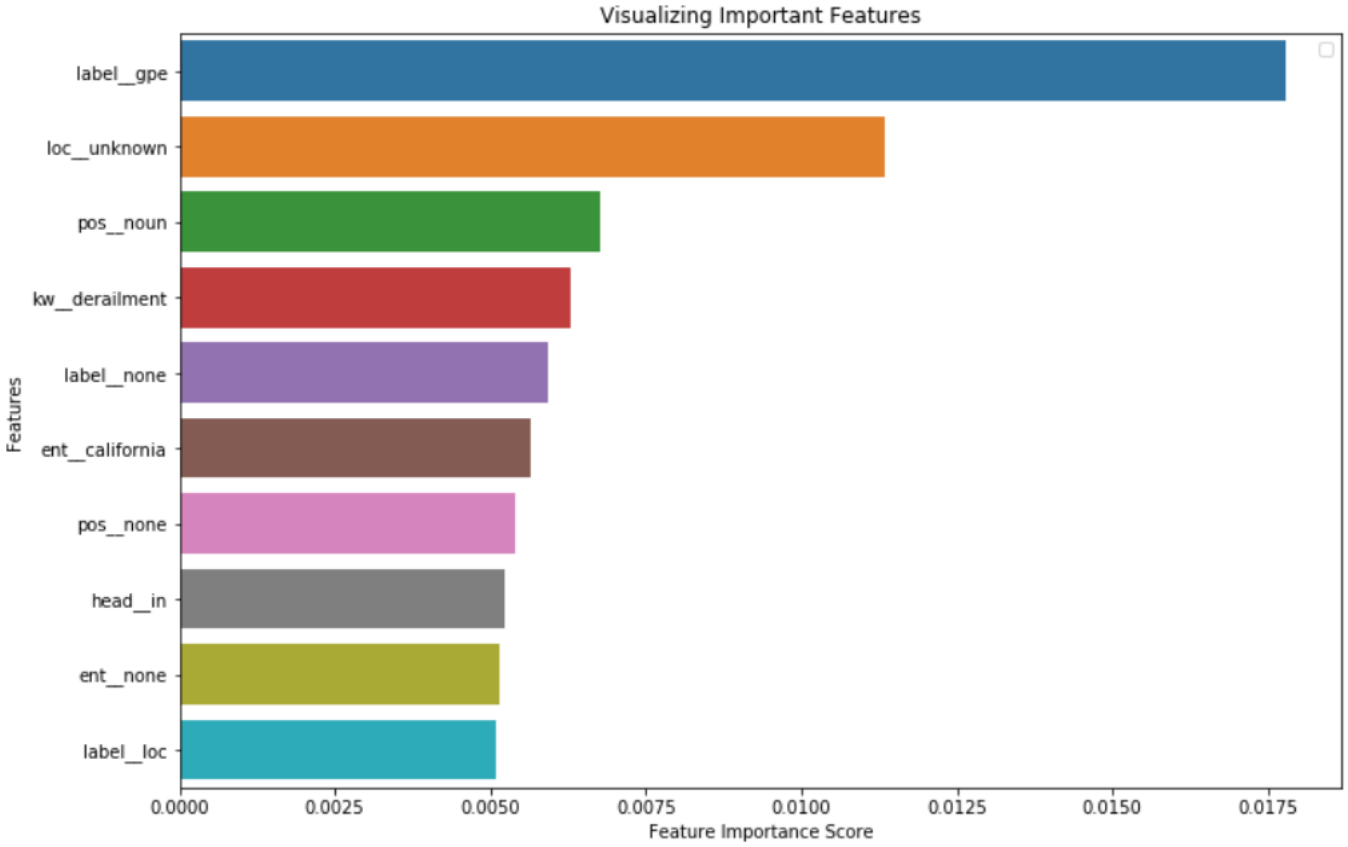


Figure 25

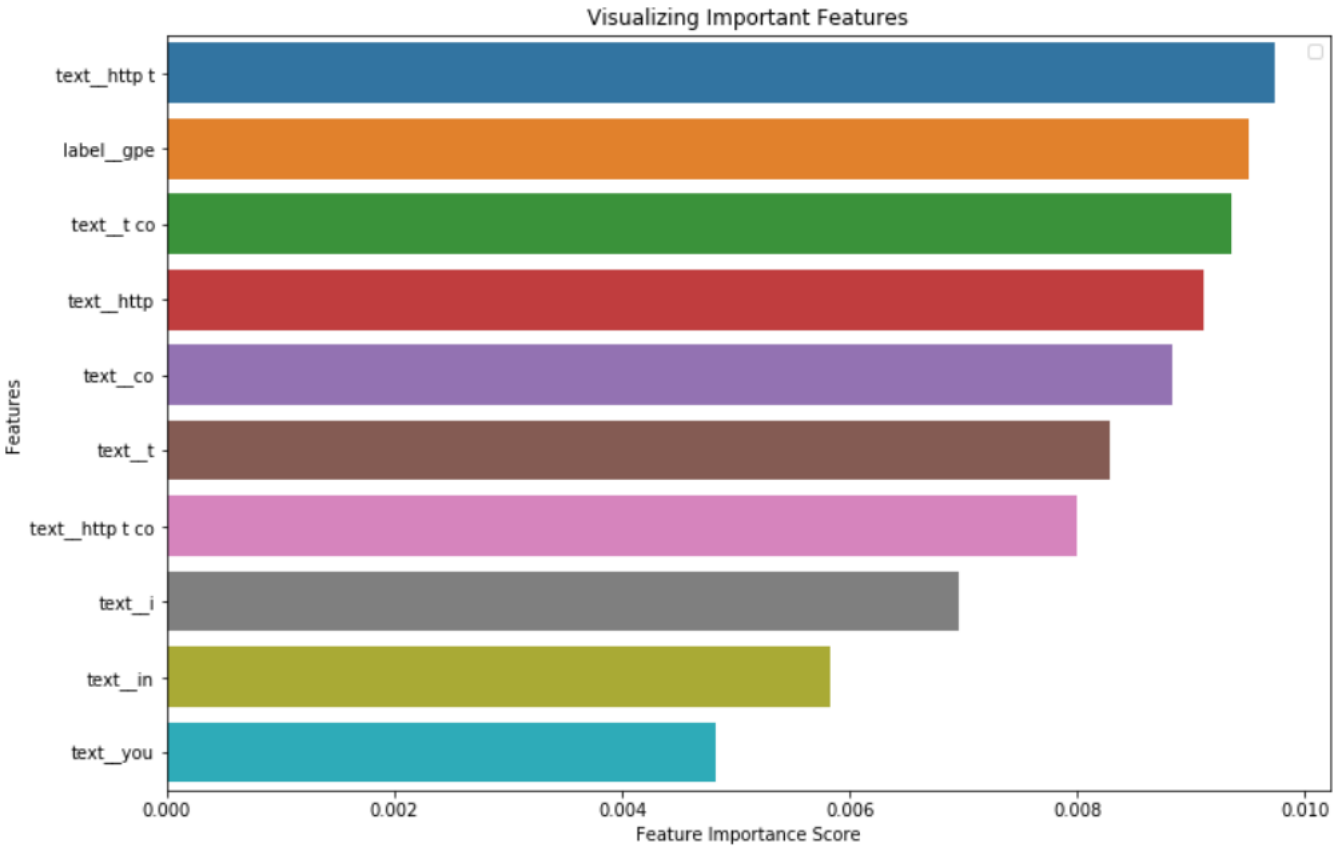


Figure 26

SVC with rbf kernel (770M features)

	precision	recall	f1-score	support
0	0.65	0.87	0.74	4342
1	0.69	0.38	0.49	3271
accuracy			0.66	7613
macro avg	0.67	0.62	0.62	7613
weighted avg	0.67	0.66	0.63	7613

Figure 27

MLP with all features

`solver='lbfgs', alpha=1e-5, activation='tanh'`

Params:

`hidden_layer_sizes=(100, 100, 100), max_iter=1000`

```
                precision    recall  f1-score   support

           0.0       0.72      0.64      0.68      4305
           1.0       0.58      0.66      0.62      3198

      accuracy                           0.65      7503
     macro avg       0.65      0.65      0.65      7503
  weighted avg       0.66      0.65      0.65      7503

  Cross-validation MSE: 0.649 ± 0.090
  Training Set Accuracy: 0.998

  Evaluation Time Taken:  00:47:30
  Training Time Taken:   00:05:01
```

Figure 28

MLP with all features

`solver='lbfgs', alpha=1e-5, activation='tanh'`

Params:

`hidden_layer_sizes=(500, 500, 500), max_iter=2000`

```
                precision    recall  f1-score   support

           0.0       0.72      0.69      0.70      4305
           1.0       0.60      0.64      0.62      3198

      accuracy                           0.67      7503
     macro avg       0.66      0.66      0.66      7503
  weighted avg       0.67      0.67      0.67      7503

  Cross-validation MSE: 0.665 ± 0.059
  Training Set Accuracy: 0.998

  Evaluation Time Taken:  09:23:30
  Training Time Taken:   04:19:10
```

Figure 29

Base Random Forest Classifier

	precision	recall	f1-score	support
0	0.68	0.90	0.77	4342
1	0.77	0.43	0.55	3271
accuracy			0.69	7613
macro avg	0.72	0.67	0.66	7613
weighted avg	0.72	0.70	0.68	7613

Cross-validation MSE: 0.680 ± 0.060
Training Set Accuracy: 0.844

Figure 30

Random Forest Classifier After Tuning Hyperparameters

	precision	recall	f1-score	support
0	0.68	0.90	0.77	4342
1	0.77	0.43	0.55	3271
accuracy			0.72	7613
macro avg	0.72	0.67	0.66	7613
weighted avg	0.72	0.70	0.68	7613

Cross-validation MSE: 0.713 ± 0.060
Training Set Accuracy: 0.844

Figure 31