Comparison of traditional machine learning models using the example of text classification of twitter data

Jan Burger - 765391

[1] Hochschule Darmstadt
[2] Information Science
[3] NLP-based Data Science
[4] Supervisor: Mina Schütz
[5] Submission date: 25.07.2021

Abstract. Traditional machine learning models have been around for many years. Since computing power has grown a lot over the years, the preferred method for machine learning nowadays are neural networks. Since the rise of neural networks, traditional machine learning models have moved into the background of research. That's why in this paper I want to focus on the traditional approach of machine learning and ask the question of which traditional machine learning model has the best performance on the task of text classification of twitter data. My research is based on the data set of Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Webe[4]. In the first step, I used typical preprocessing techniques like the removal of punctuation to prepare the data for the machine learning model. I then did some Exploratory Data Analysis to get insight to the data and extract new features. After that I tested different possibilities to train and test the machine learning models like using new features or testing whether the Z-normalization does make a difference in model performance. The traditional machine learning model with the best performance was the Logistic Regression with 94,3% accuracy.

1 Introduction

Traditional machine learning models are algorithms which already exist for a long time and are very well researched and documented. In contrast to that none traditional machine learning refers to newly developed algorithms which at the moment most research is done with. In this paper, I want to demonstrate the differences between traditional machine learning models and determine their performance. All of this will be done using the example of text classification of twitter data.

My work is based on the paper of Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Webe[4]. I will use their dataset of roughly 24 thousand extracted and classified tweets from Twitter to train and evaluate my machine learning models. In their original paper, they labeled the tweets in three different classes, 0 - hate speech 1 - offensive language 2 - neither. One big problem arises when doing research on hate speech. The definition of hate speech varies a lot in different papers. Fore example, one definition a research paper gave was the following: "HATE: This class contains tweets which highlight negative attributes or deficiencies of certain groups or individuals. This class includes hateful comments towards individuals based on race, political opinion, sexual orientation, gender, social status, health condition etc."[6].Another paper gave the following definition: 'Hateful: this class includes tweets which are offensive, and present hate, racist and segregative words and expressions."[7]. This shows that there is not a formal definition for hate speech. Different papers give different definitions on what hate speech is. The research paper which my work is based on itself[4] states that hate speech is directed towards a group of people and the goal is to harm them in some way.

Since the definition of hate speech is not clear yet, I decided to summarize the two categories hate speech and offensive language and call them offensive language+. This category consists of all the tweets which in the original paper[4] were labeled as hate speech or offensive language. My second category will simply be the tweets which were labeled as normal.

In traditional machine learning, different models have different assumptions about the data and therefore vary in performance. I want to evaluate which traditional machine learning model scores the best performance on text classification. Therefore, the following research question arises:

"Which traditional machine learning model has the best performance on text classification of twitter data?"

To evaluate the performance, the following four metrics will be collected for each model: accuracy, precision, recall and F1-Score. Since the problem statement is a classification task, I can only use supervised algorithms to solve this problem. These are the algorithms which I will collect the metrics for: Logistic Regression, K-Neighbours Classifier, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machine, Decision Tree, Random Forrest. I will use different parameters to test the models under different circumstances. For example, I will add new engineered features to the models and test them with and without those features. To create these features, I will use Exploratory Data Analysis (EDA) to get insight to the data and engineer new features. At the end, all the results will be summarized in a big result table which shows all the metrics for each model with all the parameters tested.

2 State of the art and related work

The paper of Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee* and Khairullah khan[3] describes the classification process of a piece of text in four major steps. Starting off with the preprocessing of the text by tokenizing, stemming and removing the stop words. After that follows the feature selection and the vector representation of the words using the TF-IDF approach. The last step is to train the algorithms with the selected features. In the article "A recent overview of the state-of-the-art elements of text classification" written by Marcin MichałMirończuk[5] the author compares four articles and analyses the elements that were used for text classification. All the mentioned articles used preprocessing as well as a vector space model as a basis for the machine learning algorithms. The articles mentioned used different techniques for dimensionality reduction. The four traditional machine learning models which were used by all four papers were decision tree, support vector machine, naive bayes and k-nearest neighbour.

The paper which my work will be based on[4] also works with the TF-IDF approach for vectorizing the words. They used several different machine learning models like logistic regression and support vector machine. As evaluation metrics they used accuracy, precision, recall and the F1-Score. This paper as well as several different other papers all show a similar approach which I will use in this paper. The difference is that most papers only choose a few machine learning models to train and don't extract further features from the text than the standard vectorization process. This is something which I will work on more in detail to test different models with different features.

3 Data

Figure one shows an impression of the original data from the paper my work is based on[4]. The Data set consists of three columns with the number of votes that each class received during the annotation of the tweets. The column class represents the final class of the tweet and the column tweets shows the raw unedited tweet.
As I already mentioned in the introduction, I am going to summarize hate speech and offensive language into one category called offensive language+. My raw data set will then consist of only two labels, which are normal and offensive language+. A preview of the data is shown in figure two.

4 Data Preprocessing

In this step, I am going to preprocess the raw tweets in order to make the text data more uniformly and therefore easier to process by machine learning models. I will apply the following preprocessing steps in the following order:

1. Making all words lowercase
2. Cleaning the tweets which includes removing mentions, hashtags, URL's and retweets
3. Converting Emoticons to text
4. Removing all newlines
5. Converting all chat words to full words
6. Removing all digits
7. Removing punctuation
8. Spelling correction
9. Removing stop words
10. Using wordnetlemmatizer to group words together to its basic form

I choose this order of preprocessing steps for several reasons. For step two and three, the tweet has to be in raw format in order to work properly. For example, URLs can only be removed if no characters of the URL are deleted. After step three, things like newlines and digits will be removed. I found a list of Emoticons[1] and chat words[2] with their respective representation as words on GitHub and Kaggle. I will use this list to convert chat words as well as emoticons to text. It is important that the chat word conversion will be done before the removal of digits and punctuation because chat words include digits and special characters, which will be removed after step six and seven. The next step after the removal of punctuation is the spelling correction. In this case, it makes sense to do the spelling correction before the stop word removal because some words might be corrected to a stop word in the process of spelling correction and afterwards removed by the stop word removal. After step eight and nine, the last step in the preprocessing chain will be the use of the wordnetlemmatizer. For this step, I used an already existing implementation from kaggle[2]. It is important to do this step at the end of the preprocessing chain because it might benefit from earlier steps like the spelling correction.

5 Exploratory Data Analysis

In this step, I am going to explore the data to get a better understanding of it. This step is the prerequisite for the feature extraction and feature selection.

5.1 Number of tweets per class

There are 20620 tweets in the data set labeled as offensive language+ but only 4163 labeled as normal. In order to have an equal distribution between normal and offensive language+ tweets, I use all the normal tweets and randomly select 4163 tweets from the offensive language+ tweets. This also helps to reduce RAM usage. Figure three shows the original distribution of normal and offensive language+ tweets.

5.2 Word count, character count and word length

Things like word count or word length can help to identify differences between each class and develop new features for the machine learning models. Figure four shows that the word count between the classes is nearly identical. Table one demonstrates that the character count and word length are also nearly the same.

5.3 Word clouds

Word clouds are a good way to visualize the most common words for each class. Figure five shows to most common words for normal tweets, and figure six shows the most common words for offensive language+ tweets. There is clearly a difference between the two categories. The offensive language+ tweets contain more negative and offensive words like b***h or h*e, while the normal tweets only have trash as the most negative word the gets often used.

5.4 Sentiment Analysis

Sentiment analysis can be used to determine whether a tweet has a positive, negative or neutral sentiment. The polarity of the sentiment refers to whether a tweet is positive or negative, and the subjectivity refers to whether a tweet is objective or rather subjective. Figure seven shows that normal tweets on average have a higher polarity and therefore are more positive and have a lower subjectivity and therefore are more objective than tweets which are labeled as offensive language+.

5.5 Text standard

Python offers a module named "textstat". With this module, it is possible to determine the text standard for each tweet. The average text standard for normal tweets is 5.6 which is interpreted as 5th to 6th grade. The average text standard for offensive language+ tweets is 4.0. Therefore, the text standard is another metric that shows the difference between normal and offensive language+ tweets.

5.6 Dimensionality reduction

One problem of working with text data is that each word in the data set becomes a feature in the end. Without reduced dimensionality, it is impossible to get a visual impression of the data. That's why I use principal component analysis to reduce the data set to it's three principal components. Figure eight shows the data represented in 3D space.

Yellow point represent offensive language+ tweets and blue points represent normal tweets. Since the whole data set is visualized, there are more yellow points than blue points in this figure.
To interpret the three principal components correctly, I calculated the correlation of each feature from the original data set with each principal component. Figure nine, ten and eleven show the top five features with the highest absolute correlation for each principal component.
Principal component one correlates nearly 100% with the original feature text standard. That means the higher PC1 is, the higher the text standard of the tweet. PC2 and PC3 both have strong correlation with words that are very typical for offensive language. That means if PC2 or PC3 has either a very high value or a very low value, it usually contains lots of offensive words. It can clearly be seen that the normal tweets are located in the middle of the graph where both PC2 and PC3 are very low, meaning they don't contain any offensive words.

6 Feature extraction and feature selection

In the section of EDA I showed several statistics and gathered information about the data set. I will now use this analysis to extract and select new features. The statistics about word count, character count and word length did not show any significant differences between normal tweets and offensive language+ tweets. Therefore, no new features can be extracted here. The sentiment analysis showed a significant difference between normal and offensive language+ tweets. However, the polarity had a higher difference than the subjectivity. That's why I will only use the sentiment polarity as an additional feature. The text standard did also show a difference between the categories. Therefore, this feature will also be used for the machine learning models. To get a good overview, I will test the models once with and once without the additional features. This will ensure that in the end the best variant can be selected.

7 Model selection and Model training

I selected the following supervised machine learning models: Logistic Regression, K-Neighbours Classifier, Gaussian Naive Bayes, Multinomial Naive Bayes, Support Vector Machine, Decision Tree, Random Forrest. All the models will be tested once with the new features and once without them. I will also test whether the models perform better when the data is Z-normalized. The Multinomial Naive Bayes is only suitable for features with none negative values. Therefore, it can't be used with the new feature sentiment polarity because this feature involves negative values. Multinomial Naive Bayes can only be used with no new features and without standard scaler because this also produces negative values.Other than that, all models will be tested with the different parameters mentioned. To evaluate the model I will use the following four metrics: Precision, Recall, Accuracy and F1-Score. In addition to the parameters already discussed, I will use a grid search object in sklearn to test different hyperparameters for each model specifically.

8 Results

Figure 12 and 13 show the result table for all tested models with all parameters for each model. Figure 12 shows all results with Z-normalization, and figure 13 without Z-normalization. Each result table is broken down into four different levels. The first level is whether the Z-normalization was applied to the data. The second level is whether the newly created features were used. The third level is the vectorization method. For all models, Countvectorizer and TFIDF-Vectorizer were used. The last level are all the metrics like accuracy, precision, recall and F1-Score. The accuracy and the F1-Score are highlighted since they are the most important metrics for my research.

The first thing that stands out is that the models trained with the vectorization method TFIDF all have a very low overall performance. The accuracy is always close to 50%. The model which performed best was the Logistic Regression without the Z-normalization and only with text based features. The model had an accuracy of 94,4% and nearly the same F1-Score with 94,2%. In general, the model which performed best with the Z-normalization was the Logistic Regression. The model which performed best with the Z-normalization was the Random Forrest. With 94,2% accuracy it was very close to the best performance of the Logistic Regression. The model performance of the Decision Tree and the Random Forrest improved when the new features were added. One other thing that stands out is the recall score of the K-Neighbours Classifier with Z-normalization and with only text based features. The model has a accuracy of only 50% but a recall score of nearly 100%.

9 Discussion

Overall, my research question can be answered with these results. The best tradition machine learning model for text classification of twitter data is the Logistic Regression without the Z-normalization and without any new added features, using the Countvectorizer as a vectorization method. The model had an accuracy of 94,4%.

The low performance of the models with the TFIDF-Vectorizer can be explained with the high frequency of which offensive words exist in tweets. The more tweets a term appears in, the less it is weighted. But it would be better to give these insulting words a higher value because they are deciding when it comes to offensive language. The Countvectorizer does the opposite thing and values words higher which appear more often by simply counting the occurrences of the words. Another interesting result is that the K-Neighbours Classifier has a very low accuracy but a very high recall of nearly 100% when using the TFIDF-Vectorizer. This means that the model is basically marking every tweet as offensive. This results in a 50% accuracy, since the number of offensive language+ tweets and the number of normal tweets are equal.

10 Limitations

One big limitation which doing this research was that there was not enough computing power available to test certain things. For example, I wanted to include spellchecking in the preprocessing part, but the spellchecker would have taken over ten hours to finish. That's why this preprocessing step did not take part in my research. Due to the same reason, I was also not capable of using the dimensionality reduction technique TSNE. Although the PCA already gave good insight into the data, it would have been interesting to see the results of another algorithm.

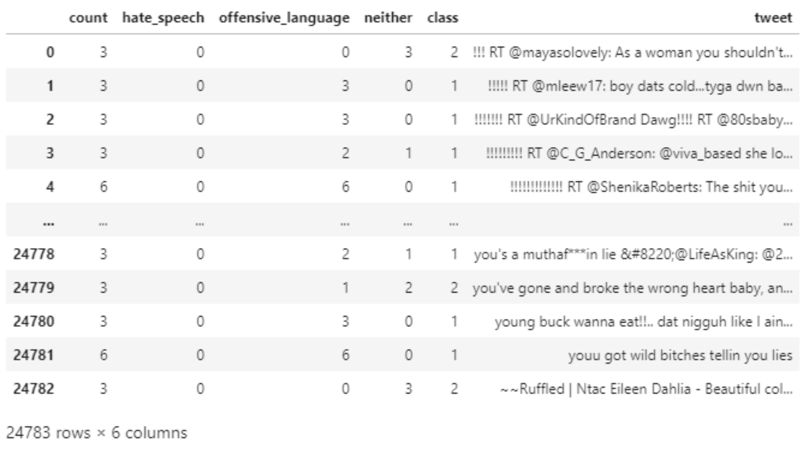
11 Conclusion and Future work

To sum it all up, my research showed that the traditional machine learning algorithm Logistic Regression has the best overall performance using no Z-normalization and only text based features. I used several preprocessing steps to prepare the data for the machine learning models. After that I tested the models with different parameters like Z-normalized data or with new developed features. For the future it would be interesting to see whether different new features like the sentiment subjectivity which I did not test in this research can enhance the performance of the models. As my research showed, there is no need to test the models with the TFIDF-Vectorizer since the performance will always be lower than with the Countvectorizer.

References

1. https://github.com/neelshah18/emot.
2. https://www.kaggle.com/pocooo/nlp-data-preprocessing-all-you-need/notebook.
3. Lam Hong Lee* Khairullah khan Aurangzeb Khan, Baharum Baharudin. A review of machine learning algorithms for text-documents classification.
4. Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language, 2017.
5. Marcin Michał Mirończuk and Jarosław Protasiewicz. A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, 106:36–54, 2018.
6. Sayar Ghosh Roy, Ujwal Narayan, Tathagata Raha, Zubair Abid, and Vasudeva Varma. Leveraging multilingual transformers for hate speech detection. *CoRR*, abs/2101.03207, 2021.
7. Hajime Watanabe, Mondher Bouazizi, and Tomoaki Ohtsuki. Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access*, 6:13825–13835, 2018.

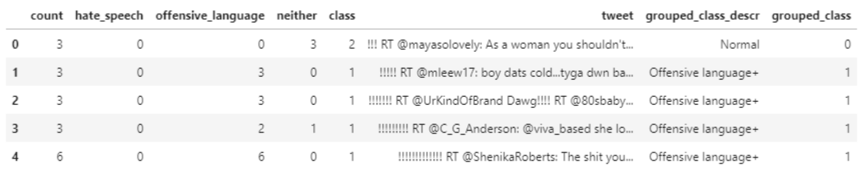
Appendices



	count	hate_speech	offensive_language	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
...	...	...	...	...	...	...
24778	3	0	2	1	1	you's a muthaf***in lie &#8220;@LifeAsKing: @2...
24779	3	0	1	2	2	you've gone and broke the wrong heart baby, an...
24780	3	0	3	0	1	young buck wanna eat!!.. dat nigguh like I ain...
24781	6	0	6	0	1	youu got wild bitches tellin you lies
24782	3	0	0	3	2	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...

24783 rows × 6 columns

Fig. 1. Original Data set[4]



	count	hate_speech	offensive_language	neither	class	tweet	grouped_class_descr	grouped_class
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...	Normal	0
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	Offensive language+	1
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	Offensive language+	1
3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	Offensive language+	1
4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	Offensive language+	1

Fig. 2. Modified Data set with only two labels

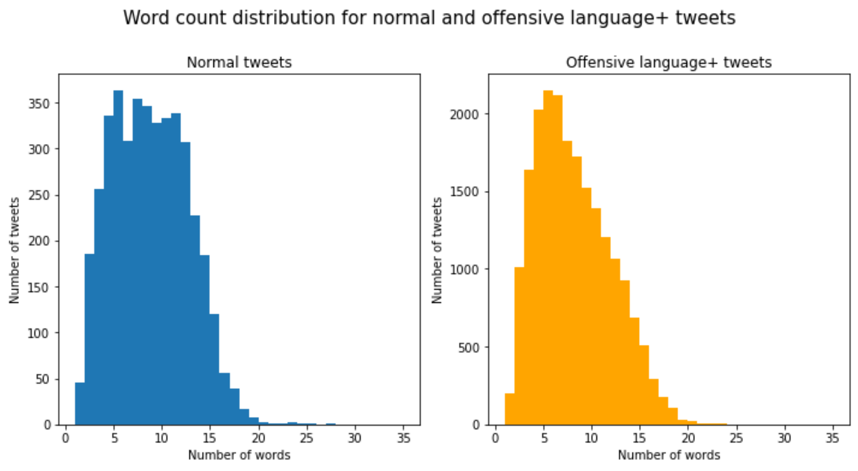


Fig. 3. Word count distribution for normal and offensive language+ tweets

Table 1. Overview of basic text statistics

Type of statistic	Normal tweets	Offensive language+ tweets
Avg. word count	8.3	7.7
Avg. character count	50.3	42.8
Avg. word length	6.0	5.5

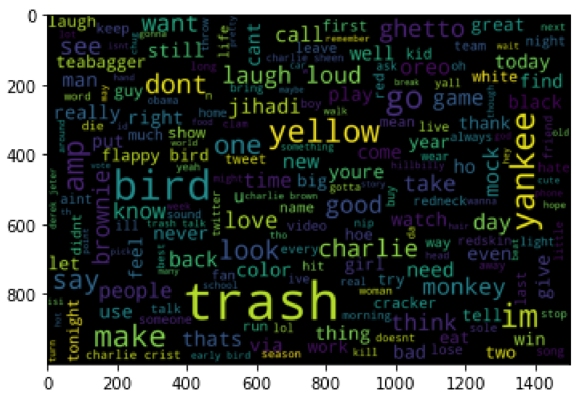


Fig. 4. Word cloud - Normal tweets

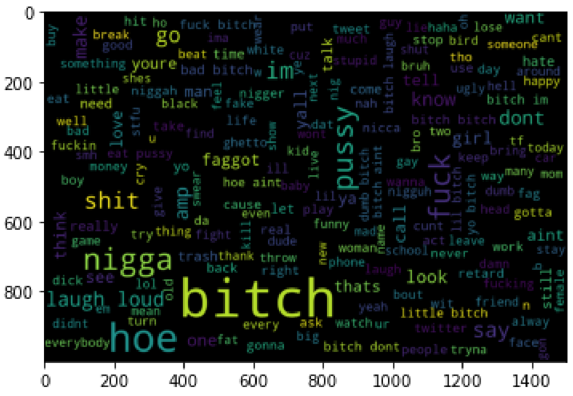


Fig. 5. Word cloud - Offensive language+ tweets

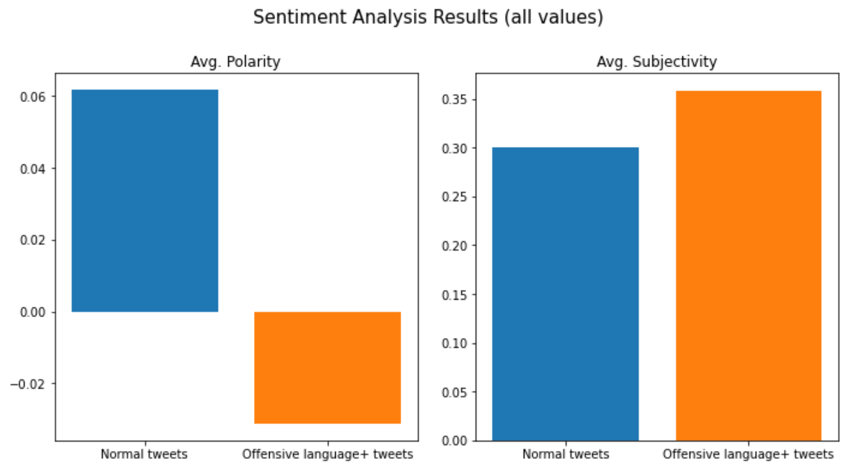


Fig. 6. Sentiment Analysis

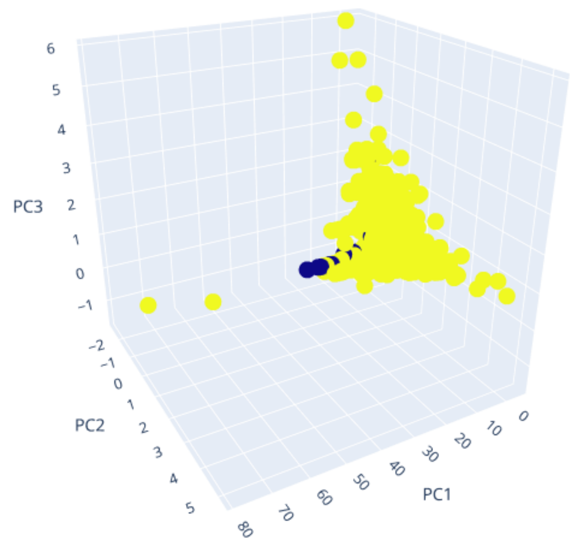
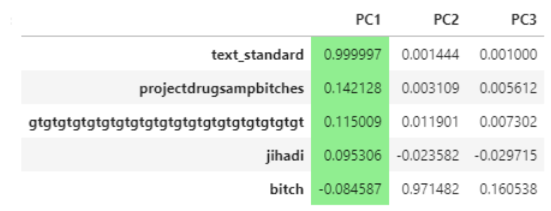
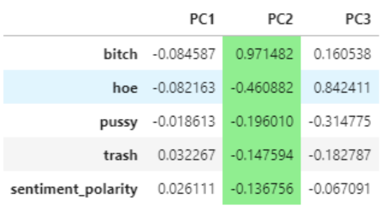


Fig. 7. Principal Component Analysis



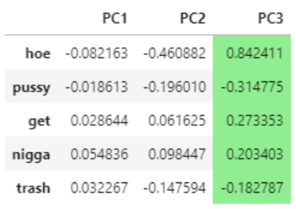
	PC1	PC2	PC3
text_standard	0.999997	0.001444	0.001000
projectdrugsampbitches	0.142128	0.003109	0.005612
gtgtgtgtgtgtgtgtgtgtgtgtgtgtgtgtgtgt	0.115009	0.011901	0.007302
jihadi	0.095306	-0.023582	-0.029715
bitch	-0.084587	0.971482	0.160538

Fig. 8. Correlation between raw features and PC1



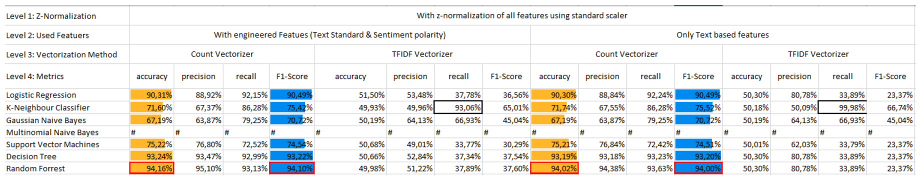
	PC1	PC2	PC3
bitch	-0.084587	0.971482	0.160538
hoe	-0.082163	-0.460882	0.842411
pussy	-0.018613	-0.196010	-0.314775
trash	0.032267	-0.147594	-0.182787
sentiment_polarity	0.026111	-0.136756	-0.067091

Fig. 9. Correlation between raw features and PC2



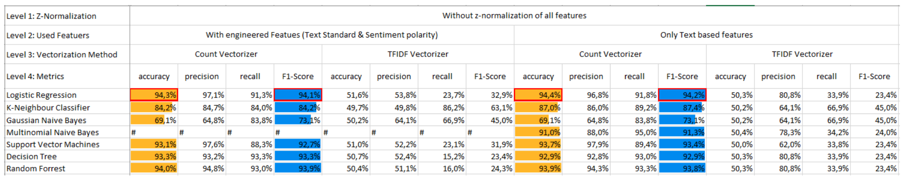
	PC1	PC2	PC3
hoe	-0.082163	-0.460882	0.842411
pussy	-0.018613	-0.196010	-0.314775
get	0.028644	0.061625	0.273353
nigga	0.054836	0.098447	0.203403
trash	0.032267	-0.147594	-0.182787

Fig. 10. Correlation between raw features and PC3



Level 1: Z-Normalization	With z-normalization of all features using standard scaler															
Level 2: Used Featuers	With engineered Featues (Text Standard & Sentiment polarity)								Only Text based features							
Level 3: Vectorization Method	Count Vectorizer				TFIDF Vectorizer				Count Vectorizer				TFIDF Vectorizer			
Level 4: Metrics	accuracy	precision	recall	F1-Score	accuracy	precision	recall	F1-Score	accuracy	precision	recall	F1-Score	accuracy	precision	recall	F1-Score
Logistic Regression	90,31%	88,92%	92,15%	90,49%	51,50%	53,48%	37,78%	36,56%	90,30%	88,84%	92,24%	90,49%	50,30%	80,78%	33,89%	23,37%
K-Neighbour Classifier	71,60%	67,37%	86,28%	75,42%	49,93%	49,96%	93,06%	65,01%	71,74%	67,55%	86,28%	75,52%	50,18%	50,09%	99,98%	66,74%
Gaussian Naive Bayes	67,19%	63,87%	79,25%	70,72%	50,19%	64,13%	66,93%	45,04%	67,19%	63,87%	79,25%	70,72%	50,19%	64,13%	66,93%	45,04%
Multinomial Naive Bayes	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
Support Vector Machines	75,22%	76,80%	72,52%	74,54%	50,68%	49,01%	33,77%	30,29%	75,21%	76,84%	72,42%	74,51%	50,01%	62,03%	33,79%	23,37%
Decision Tree	93,24%	93,47%	92,99%	93,22%	50,66%	52,84%	37,34%	37,54%	93,19%	93,18%	93,23%	93,20%	50,30%	80,78%	33,89%	23,37%
Random Forrest	94,16%	95,10%	93,13%	94,10%	49,98%	51,22%	37,89%	37,60%	94,02%	94,38%	93,63%	94,00%	50,30%	80,78%	33,89%	23,37%

Fig. 11. Result table 1 - With Z-normalization



Level 1: Z-Normalization	Without z-normalization of all features															
Level 2: Used Featuers	With engineered Featues (Text Standard & Sentiment polarity)								Only Text based features							
Level 3: Vectorization Method	Count Vectorizer				TFIDF Vectorizer				Count Vectorizer				TFIDF Vectorizer			
Level 4: Metrics	accuracy	precision	recall	F1-Score	accuracy	precision	recall	F1-Score	accuracy	precision	recall	F1-Score	accuracy	precision	recall	F1-Score
Logistic Regression	94,3%	97,1%	91,3%	94,1%	51,6%	53,8%	23,7%	32,9%	94,4%	96,8%	91,8%	94,2%	50,3%	80,8%	33,9%	23,4%
K-Neighbour Classifier	84,2%	84,7%	84,0%	84,2%	49,7%	49,8%	86,2%	63,1%	87,0%	86,0%	89,2%	87,4%	50,2%	64,1%	66,9%	45,0%
Gaussian Naive Bayes	69,1%	64,8%	83,8%	73,1%	50,2%	64,1%	66,9%	45,0%	69,1%	64,8%	83,8%	73,1%	50,2%	64,1%	66,9%	45,0%
Multinomial Naive Bayes	#	#	#	#	#	#	#	#	91,0%	88,0%	95,0%	91,3%	50,4%	78,3%	34,2%	24,0%
Support Vector Machines	93,1%	97,6%	88,3%	92,7%	51,0%	52,2%	23,1%	31,9%	93,7%	97,9%	89,4%	93,4%	50,0%	62,0%	33,8%	23,4%
Decision Tree	93,3%	93,2%	93,3%	93,3%	50,7%	52,4%	15,2%	23,4%	92,9%	92,8%	93,0%	92,9%	50,3%	80,8%	33,9%	23,4%
Random Forrest	94,0%	94,8%	93,0%	93,9%	50,4%	51,1%	16,0%	24,3%	93,9%	94,3%	93,3%	93,8%	50,3%	80,8%	33,9%	23,4%

Fig. 12. Result table 2 - Without Z-normalization

I hereby certify that I have written this thesis on my own and have not used any sources other than those indicated in the list of sources. All passages taken verbatim or in spirit from published or unpublished sources are identified as such. This also applies to sources which I myself have prepared for other purposes. The drawings or illustrations in this work have been prepared by myself or have been provided with an appropriate source reference. This work has not been submitted in the same or similar form to any other examination or examination authority. I am aware that if the above declaration proves to be incorrect, this constitutes an attempted de-registration which may result in expulsion.

Darmstadt, July 25, 2021

[illegible]