Depression Detection in Reddit Posts Through Machine Learning and Sentiment Analysis

Vasco Vieira Costa
University of Aveiro
13 PUBLICATIONS **0** CITATIONS

Depression Detection in Reddit Posts Through Machine Learning and Sentiment Analysis

Vasco Vieira Costa
vascovc@ua.pt

January 17, 2025

Introduction

The advent of smartphone technology and the rise of social media have profoundly impacted daily life worldwide. During this time, mental health has declined, with increasing rates of depression and suicide [4]. Social Networking Sites (SNS) have come under significant scrutiny in the media, with various aspects of their use being analyzed. These include not only the amount of time spent on these platforms but also the intensity of engagement and problematic usage, each showing varying levels of association with mental health outcomes [2]. Research has also explored the causal relationship between social media use and psychiatric outcomes. Evidence suggests that reducing social media use is associated with a decrease in related symptoms [4].

Teenagers today are heavily dependent on Social Networking Sites (SNS), often finding it difficult to imagine life without them [5]. Between 2011 and 2018, rates of depression, self-harm, and suicide attempts increased significantly. Several studies have shown that heavy technology users are twice as likely as light users to report low well-being. Furthermore, even light users are affected due to the decline in personal relationships [9].

During the COVID-19 pandemic, teenagers' mental health was further impacted, with their social media engagement and emotional expression reflecting the pandemic's progression and their heightened need to stay connected, particularly among those at greater risk [11]. Although bullying and cyberbullying showed little to no increase during this period, anxiety and depression rose significantly compared to pre-pandemic levels [3].

Depression detection using Machine Learning (ML) methods applied to online content has been explored previously, utilizing data from platforms such as Facebook posts, Facebook comments, and YouTube comments, achieving varying levels of success [10]. Other approaches have focused on disruptive events, such as the COVID-19 pandemic, where the frequency of posts and comments on subreddits was closely aligned with the evolution of the pandemic [11].

This work aims to leverage the Kaggle dataset provided (*here*) to predict whether the author of a post might be experiencing depression [6].

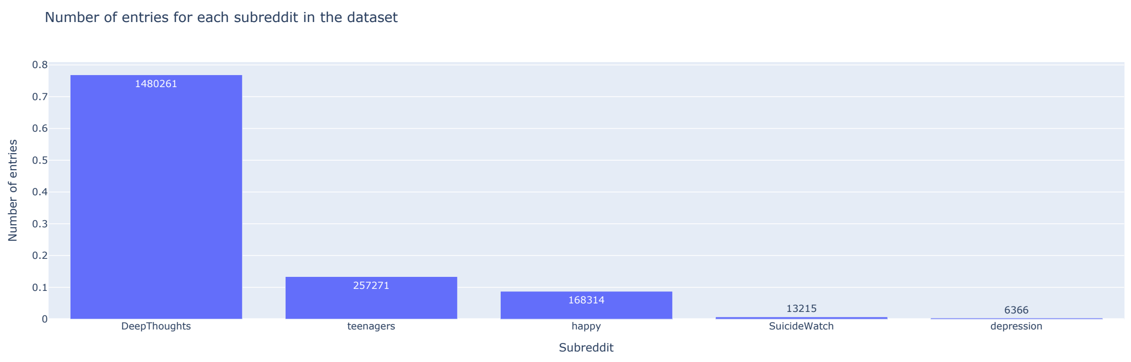
Data analysis

The dataset presented is an extraction from Reddit in particular from the subreddits of `r/teenagers`, `r/DeepThoughts`, `r/happy`, `r/SuicideWatch`, `r/depression` and a set of unknown sources which were not considered. These entries are mentioned in the Kaggle dataset page as being a subset of the data available in the `r/pushshift` subreddit page of the post (presented *here*) related to the end of the year 2023.

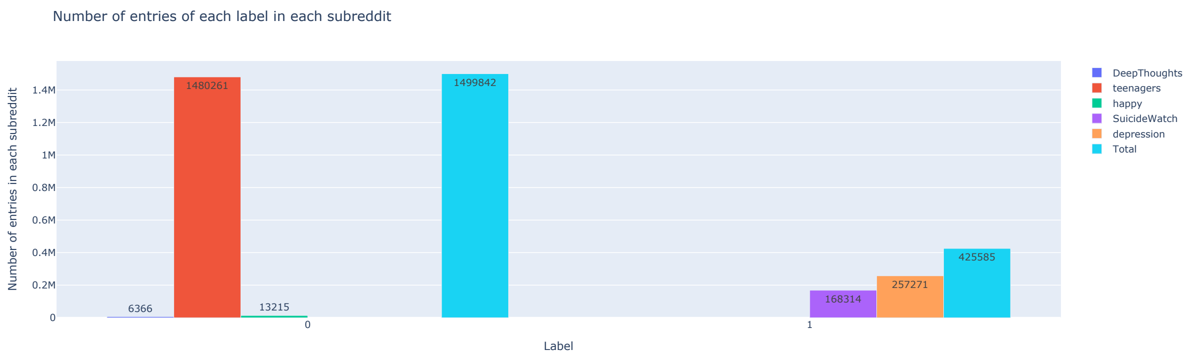
This tabular dataset is presented in a `CSV` format with 2.470.778 entries and 8 columns. Each entry is a Reddit post with its respective index, the subreddit from where it was extracted, post title, body, number of upvotes, time of post, number of comments, and the label, numbered with 1 in case of depression content and 0 otherwise. It is mentioned that the creation of this label is solely based on the indication from where it was extracted. In case the post is from `r/SuicideWatch` or `r/depression` then it is considered a 1. Therefore, the column indicative of the subreddit of origin will not be considered in the text analysis task.

Related to the subreddit where the post is from there is an unbalance as can be seen in Figure 1, in particular in Figure 1a where the predominance is of posts from

DeepThoughts. Figure 1b separates this information according to the label where it is clear that the number of entries of class 1 is much inferior to that of class 0.



(a) Number of entries in the dataset from each subreddit.



(b) Number of entries in the dataset from each subreddit and the respective label. (The 0 valued bars are not shown.)

Figure 1: Number of entries in the dataset related to the subreddit it originated.

Methodology

Due to the size of the original dataset and the hardware limitations this had to be shortened. The approach followed was to randomly sample 1000 entries from each of the subreddits leading to the composition depicted in Figure 2. This reduced dataset, like the original, is unbalanced, although on a different proportion, with 60% now being of label 0 and 40% of label 1 in a total of 5000 entries.

To conduct sentiment analysis of these entries the dataset was further simplified with the subreddit from where it originated being removed, the title and body combined into a single text entry, and the creation time also removed. The number of up-votes and comments despite not contributing to sentiment analysis were left as they could present relevant information for the application of ML methods. The programming language of Python was selected due to existing user familiarity and the vast number of existing libraries to conduct the analysis. In particular, the scikit-learn [7] and the NLTK [1] packages were used for ML and natural language operations.

The text entries were first transformed to lower case and then tokenized with the

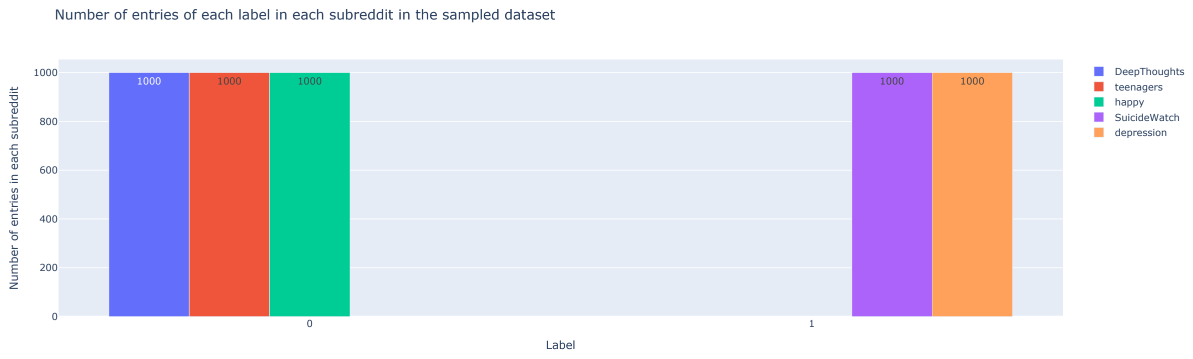


Figure 2: Number of entries in the sampled dataset from each subreddit.

stopwords, in English, and punctuation removed from each entry[1]. Two commonly used strategies in text analysis is to either perform stemming or lemmatization to normalize the text. Stemming is an algorithmic approach to reduce the words to their root forms, by removing suffixes and prefixes. In contrast, lemmatizing leverages lexical databases to produce lemmas, allowing for complete word transformation, changing *"better"* to *"good"*. With this in mind, different separate runs were conducted with the use of one or the other through the functions of `PorterStemmer` and `WordNetLemmatizer` available in `NLTK`.

The following step is then to convert the entries to matrix form to allow for the application of ML methods. Two separate approaches were followed to use a word counting methodology, `CountVectorizer`, and the term frequency-inverse document frequency (TF-IDF), `TfidfVectorizer`, methodology, both with the default options. While the first counts the frequency each word appears the second methodology adjusts the raw word frequency counts by considering how commonly a word appears across all documents in the corpus, reducing the weight of frequently occurring but less meaningful words. TF-IDF assigns a weight to each word that reflects its importance in a document relative to the entire corpus. This is calculated by multiplying the $\text{tf}(t,d)$ term frequency with the $\text{idf}(t)$ inverse document frequency, where t is the term and d is the document it relates to. The element of $\text{tf}(t,d)$ is the word count of each word in each document. The $\text{idf}(t)$ component is calculated differently with $\log \frac{n}{1+\text{df}(t)} + 1$ when using the default options. Where n is the total number of documents and $\text{dt}(t)$ is the number of documents that contain the term t. The resulting vectors are normalized by the Euclidean norm

$$v_{norm} = \frac{v}{||v||^2} = \frac{v}{\sqrt{\sum_{i=1}^{n} v_i^2}}.$$

Other ways of calculating the $\text{idf}(t)$ term are also possible, with the more traditional form of $\text{idf}(t) = \log \frac{1+n}{1+\text{df}(t)} + 1$, or even with $\text{idf}(t) = \log \frac{n}{\text{df}(t)} + 1$ [7].

With the words converted to their matrix form, it is possible to add the sentiment of that entry to serve as a label for the application of classifiers. Besides the word matrix and the label, it is also possible to add the number of upvotes and comments. To this final matrix, a scaling operation can be applied with the StandardScaler to normalize the dataset for the classifiers that are sensitive to this factor.

To the possible 4 combinations of lemmatization/stemming and word count/TF-IDF transformations to the reduced dataset a randomized hyperparameter search was con-

[1]The existing resources of NLTK were used for both these last two operations.

ducted in 5 fold cross validation in 80% of the dataset with the remainder 20% being left for testing on the final best model to ascertain the capabilities of the model to generalize to never seen instances. In this hyperparameter search, the metric considered was the f1-score in the macro average due to the binary labels' class imbalance scenario.

Furthermore, besides the traditional classifiers it is possible to take advantage of the algorithms in NLTK, namely the Sentiment Intensity Analyzer, (presented *here*), to analyze the sentiment of the sentences by the phrasing constructions and the words used. For each of the lower-cased textual entries, the polarity scores can be computed through the package functions to obtain a score of each function regarding the negative, positive, or neutral sentiment. With this output, it is then possible to relate the positive ones to say they are of positive label, 0, or negative, 1. This can be done through the compound value in case of positive or negative value, respectively. To the 0 values, the system can be either positively prone by considering the compound value of 0 to be of label 0 or negatively prone and associate it to 1. Both strategies were also studied. With this approach, it is then possible to perform the metric analysis as when applying ML methods.

Results

Since the sentiment analysis is the same for all the combinations of transformations to the dataset this only needs to be shown once. This can be applied directly to all of the dataset entries because there is no training procedure and the metrics obtained are shown in Table 1. In this, both the 0 favored metric, *Sentiment_0*, and the 1 favored metric are demonstrated, *Sentiment_1*. From this sentiment analysis, it can be seen that the 0-prone approach performs better than the other. It also shows that there are a considerable number of cases where the sentiment is considered neutral, something that the labeling does not allow to be a reality.

Table 1: Sentiment metric reporting through the polarity score obtained from the use of the NLTK Sentiment Intensity Analyser.

		0.0	1.0	macro avg	weighted avg	accuracy
	f1-score	0.741	0.653	**0.697**	0.706	
Sentiment_0	precision	0.778	0.614	**0.696**	0.713	**0.704**
	recall	0.707	0.698	**0.703**	0.704	
	f1-score	0.703	0.631	0.667	0.674	
Sentiment_1	precision	0.766	0.572	0.669	0.689	0.671
	recall	0.649	0.703	0.676	0.671	
support		3000	2000	5000	5000	-

Lemmatization and Word Count

For the dataset transformed through Lemmatization the dataset created presented 19.234 features. In Table 2 the metric results from the best models from the hyperparameter search in the 80% portion of the dataset and tested in the remainder of the dataset are presented with the macro average, *m. avg*, and the weighted average, *w. avg*, and the accuracy, *acc*.

The testing of the models in the training set itself is also presented in Table 3. From here it can be seen that some classifiers perform better with the AdaBoost and GradientBoosting outperforming the remainders with the k-NN suffering from a high level of overfitting, high performance on the training set and low in the testing.

Table 2: Metric results of the best model from the hyperparameter search trained in the 80% portion of the dataset in the remainder for testing for the dataset transformed with lemmatization and word count.

Classifier	metric	0.0	1.0	m. avg	w. avg	acc
	f1-score	0.886	0.809	0.847	0.855	
AdaBoost	precision	0.851	0.868	**0.860**	0.858	0.857
	recall	0.923	0.758	0.840	0.857	
	f1-score	0.769	0.638	0.704	0.717	
DecisionTree	precision	0.756	0.655	0.706	0.716	0.718
	recall	0.782	0.622	0.702	0.718	
	f1-score	0.586	0.557	0.572	0.574	
GaussianNB	precision	0.698	0.475	0.587	0.609	0.572
	recall	0.505	0.672	0.589	0.572	
	f1-score	0.881	0.815	**0.848**	0.854	
GradientBoosting	precision	0.869	0.833	0.851	0.854	0.855
	recall	0.893	0.798	**0.845**	0.855	
	f1-score	0.749	0.521	0.635	0.658	
k-NN	precision	0.690	0.624	0.657	0.664	0.671
	recall	0.820	0.448	0.634	0.671	
	f1-score	0.870	0.808	0.839	0.845	
LogisticRegression	precision	0.875	0.801	0.838	0.846	0.845
	recall	0.865	0.815	0.840	0.845	
	f1-score	0.827	0.644	0.736	0.754	
RandomForest	precision	0.746	0.827	0.787	0.779	0.767
	recall	0.927	0.528	0.727	0.767	
	f1-score	0.839	0.770	0.805	0.812	
SVM	precision	0.856	0.749	0.803	0.813	0.811
	recall	0.823	0.792	0.808	0.811	
support		600	400	1000	1000	-

Table 3: Metric results of the best model from the hyperparameter search trained in the training portion and tested in it. For the dataset transformed with lemmatization and word count.

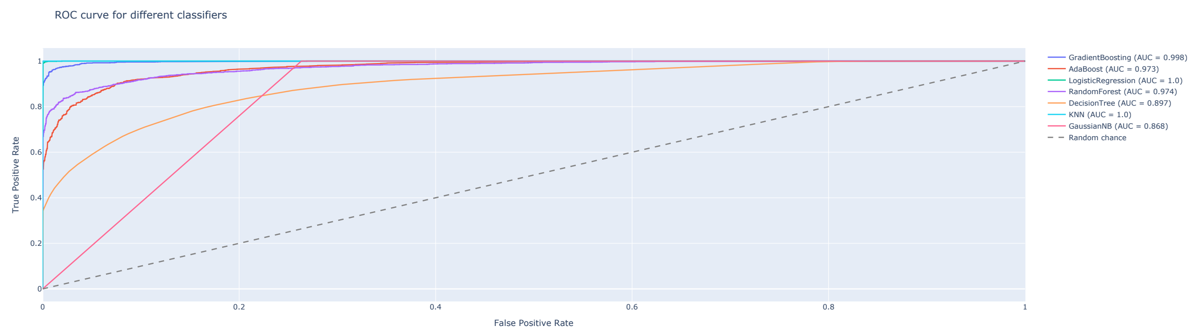
Classifier	metric	0.0	1.0	m.avg	w.avg	acc
	f1-score	0.936	0.895	0.915	0.919	
AdaBoost	precision	0.906	0.945	0.926	0.922	0.920
	recall	0.967	0.850	0.909	0.920	
	f1-score	0.862	0.771	0.817	0.826	
DecisionTree	precision	0.831	0.822	0.826	0.827	0.828
	recall	0.895	0.727	0.811	0.828	
	f1-score	0.868	0.851	0.860	0.861	
GaussianNB	precision	1.000	0.741	0.871	0.896	0.860
	recall	0.767	1.000	0.884	0.860	
	f1-score	0.981	0.971	0.976	0.977	
GradientBoosting	precision	0.971	0.988	0.979	0.978	0.978
	recall	0.992	0.956	0.974	0.978	
	f1-score	1.000	1.000	**1.000**	1.000	
k-NN	precision	1.000	1.000	**1.000**	1.000	1.000
	recall	1.000	1.000	**1.000**	1.000	
	f1-score	0.997	0.996	0.996	0.996	
LogisticRegression	precision	0.994	1.000	0.997	0.997	0.996
	recall	1.000	0.991	0.996	0.996	
	f1-score	0.905	0.812	0.858	0.868	
RandomForest	precision	0.826	1.000	0.913	0.896	0.874
	recall	1.000	0.684	0.842	0.874	
	f1-score	0.998	0.997	0.998	0.998	
SVM	precision	0.996	1.000	0.998	0.998	0.998
	recall	1.000	0.994	0.997	0.998	
support		2400	1600	4000	4000	-

This analysis can be further extended with the aid of the graphic plots presented in Figure 3 where it is possible to analyze different plots for the application of the best models obtained through the hyperparameter search, trained on the whole dataset and tested in it. In particular, in Figure 3a the Receiver Operating Characteristic (ROC) curve is presented where the false positive rate is compared against the sensitivity. Figure 3b shows the Precision-Recall curve and Figure 3c the Detection Error Trade-off curve.

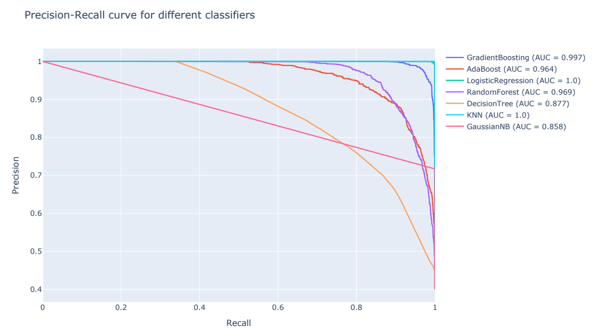
These graphs show a great performance of the models, however, such high values in conjunction with the results from Table 2 demonstrate that such models are overfitting. Meaning that their application in real scenarios would be lacking.

Lemmatization and TF-IDF

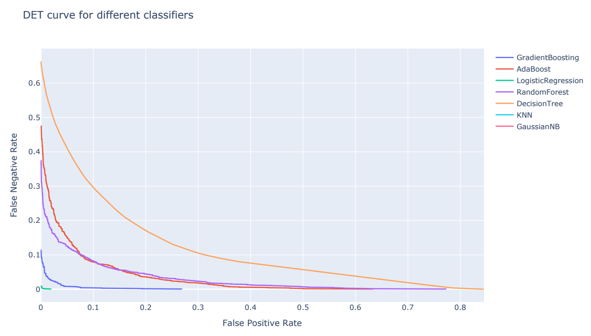
This analysis takes the change to use a TF-IDF approach instead of the word count procedure presented before. The results obtained are similar, as can be seen in the results of Table 4 where the best models obtained from the hyperparameter search in 80% of the dataset and trained in it are shown on the testing of the remaining data. Table 5 shows how the trained models on the entirety of the dataset and tested in it behave. From these, the k-NN continues to suffer from overfitting, and into the top performers the LogisticRegression trades places with the GradientBoosting.



(a) Receiver Operator Characteristic curve.



(b) Precision-Recall curve.



(c) Detection Error Trade-off curve.

Figure 3: Curve characteristics for the best model in the hyperparameter search trained with the entire dataset and tested in it. For the dataset transformed with lemmatization and word count.

In a visual representation, Figure 4, this overfitting situation is confirmed by showing great performance of the models in the trained data with them lacking in the testing of unseen instances, in Table 4. Moreover, the RandomForest's bad performance is also replicated where the prediction is always the majority class.

Table 4: Metric results of the best model from the hyperparameter search trained in the 80% portion of the dataset in the remainder for testing for the dataset transformed with lemmatization and TF-IDF.

Classifier	metric	0.0	1.0	m.avg	w.avg	acc
	f1-score	0.882	0.808	**0.845**	0.853	
AdaBoost	precision	0.856	0.851	**0.853**	0.854	0.854
	recall	0.910	0.770	0.840	0.854	
	f1-score	0.771	0.627	0.699	0.713	
DecisionTree	precision	0.748	0.660	0.704	0.713	0.716
	recall	0.795	0.598	0.696	0.716	
	f1-score	0.608	0.559	0.584	0.588	
GaussianNB	precision	0.702	0.486	0.594	0.615	0.585
	recall	0.537	0.658	0.597	0.585	
	f1-score	0.878	0.796	0.837	0.845	
GradientBoosting	precision	0.843	0.854	0.849	0.848	0.847
	recall	0.915	0.745	0.830	0.847	
	f1-score	0.743	0.435	0.589	0.620	
k-NN	precision	0.659	0.604	0.632	0.637	0.647
	recall	0.852	0.340	0.596	0.647	
	f1-score	0.864	0.814	0.839	0.844	
LogisticRegression	precision	0.899	0.773	0.836	0.849	0.843
	recall	0.832	0.860	**0.846**	0.843	
	f1-score	0.750	0.000	0.375	0.450	
RandomForest	precision	0.600	0.000	0.300	0.360	0.600
	recall	1.000	0.000	0.500	0.600	
	f1-score	0.855	0.776	0.816	0.823	
SVM	precision	0.845	0.790	0.818	0.823	0.824
	recall	0.865	0.762	0.814	0.824	
support		600	400	1000	1000	-

Table 5: Metric results of the best model from the hyperparameter search trained in the training portion and tested in it. For the dataset transformed with lemmatization and TF-IDF.

Classifier	metric	0.0	1.0	m.avg	w.avg	acc
	f1-score	0.950	0.920	0.935	0.938	
AdaBoost	precision	0.927	0.958	0.942	0.939	0.938
	recall	0.974	0.884	0.929	0.938	
	f1-score	0.921	0.872	0.897	0.902	
DecisionTree	precision	0.893	0.920	0.906	0.904	0.902
	recall	0.952	0.829	0.890	0.902	
	f1-score	0.899	0.879	0.889	0.891	
GaussianNB	precision	1.000	0.784	0.892	0.914	0.890
	recall	0.816	1.000	0.908	0.890	
	f1-score	0.988	0.982	0.985	0.986	
GradientBoosting	precision	0.978	0.997	0.988	0.986	0.986
	recall	0.998	0.967	0.983	0.986	
	f1-score	1.000	1.000	**1.000**	1.000	
k-NN	precision	1.000	1.000	**1.000**	1.000	1.000
	recall	1.000	1.000	**1.000**	1.000	
	f1-score	0.985	0.977	0.981	0.982	
LogisticRegression	precision	0.989	0.970	0.980	0.982	0.982
	recall	0.980	0.984	0.982	0.982	
	f1-score	0.750	0.000	0.375	0.450	
RandomForest	precision	0.600	0.000	0.300	0.360	0.600
	recall	1.000	0.000	0.500	0.600	
	f1-score	0.999	0.998	0.998	0.998	
SVM	precision	0.998	1.000	0.999	0.999	0.998
	recall	1.000	0.996	0.998	0.998	
support		2400	1600	4000	4000	-

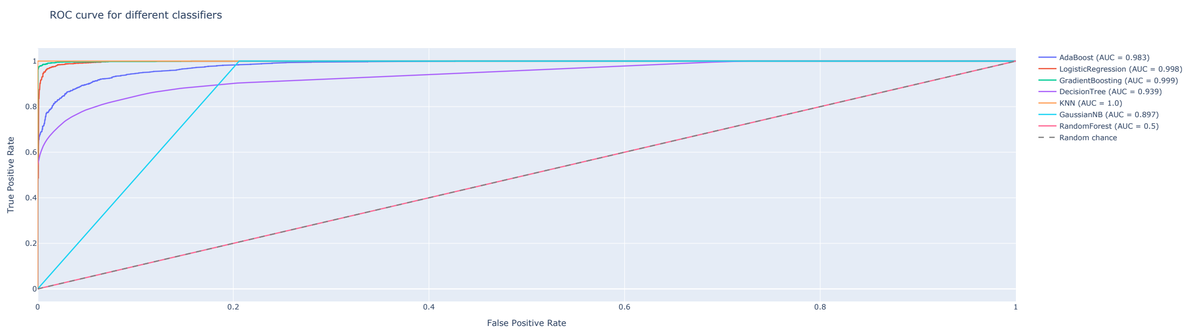
Stemming and Word Count

The transformed dataset through word Stemming presented fewer features than the Lemmatization process creating 14.688 features. In Table 6 the best models from the hyperparameter search and training in 80% of the dataset and tested in the remainder are shown. Table 7, on the other hand, reflects these models retrained on all of the dataset and tested in this set of data. An overall analysis of these values show a better performance of the process of Stemming opposed to Lemmatization, namely in the performance of the RandomForest. However, the situation of overfitting is still present as is the case of the *k*-NN classifier.

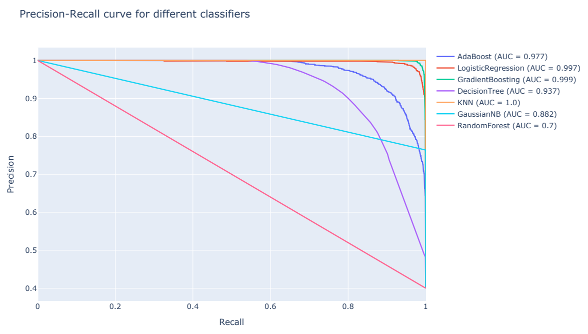
Visually, Figure 5, corroborates this information with some classifiers presenting very high values that do not correspond to the performance the models would have in unsen instances.

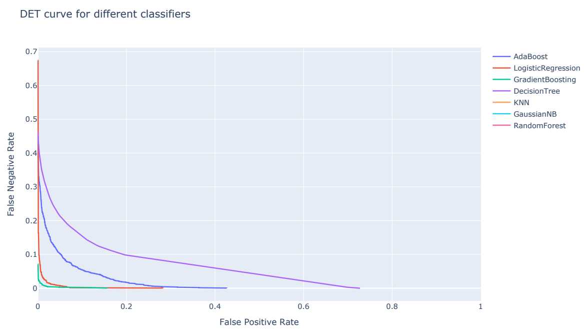
Stemming and TF-IDF

With the use of the TF-IDF approach instead of the Word Count, the results for the best models with the hyperparameter search in the 80% of the dataset, trained in



(a) Receiver Operator Characteristic curve.

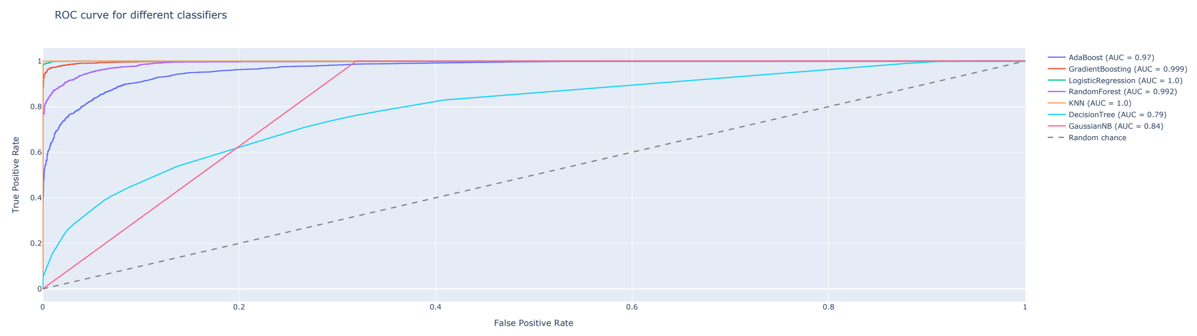




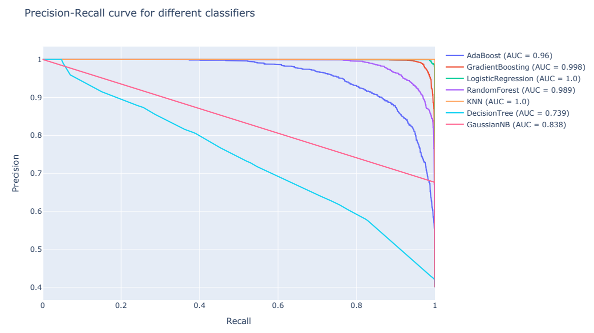
(b) Precision-Recall curve.

(c) Detection Error Trade-off curve.

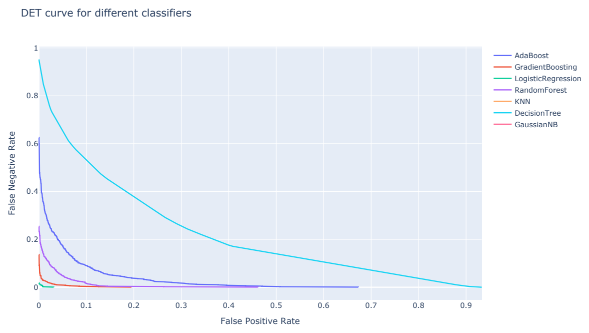
Figure 4: Curve characteristics for the best model in the hyperparameter search trained with the entire dataset and tested in it. For the dataset transformed with lemmatization and TF-IDF.



(a) Receiver Operator Characteristic curve.



(b) Precision-Recall curve.



(c) Detection Error Trade-off curve.

Figure 5: Curve characteristics for the best model in the hyperparameter search trained with the entire dataset and tested in it. For the dataset transformed with stemming and word count.

Table 6: Metric results of the best model from the hyperparameter search trained in the 80% portion of the dataset in the remainder for testing for the dataset transformed with stemming and word count.

Classifier	metric	0.0	1.0	m.avg	w.avg	acc
	f1-score	0.889	0.816	**0.853**	0.860	
AdaBoost	precision	0.856	0.872	**0.864**	0.863	0.862
	recall	0.925	0.768	0.846	0.862	
	f1-score	0.732	0.520	0.626	0.647	
DecisionTree	precision	0.687	0.589	0.638	0.648	0.656
	recall	0.783	0.465	0.624	0.656	
	f1-score	0.545	0.563	0.554	0.552	
GaussianNB	precision	0.703	0.463	0.583	0.607	0.554
	recall	0.445	0.718	0.581	0.554	
	f1-score	0.884	0.820	0.852	0.859	
GradientBoosting	precision	0.873	0.836	0.855	0.858	0.859
	recall	0.895	0.805	0.850	0.859	
	f1-score	0.747	0.524	0.636	0.658	
k-NN	precision	0.691	0.619	0.655	0.662	0.670
	recall	0.813	0.455	0.634	0.670	
	f1-score	0.878	0.822	0.850	0.855	
LogisticRegression	precision	0.889	0.807	0.848	0.856	0.855
	recall	0.867	0.838	**0.852**	0.855	
	f1-score	0.859	0.788	0.824	0.831	
RandomForest	precision	0.859	0.789	0.824	0.831	0.831
	recall	0.860	0.788	0.824	0.831	
	f1-score	0.853	0.765	0.809	0.818	
SVM	precision	0.833	0.795	0.814	0.818	0.819
	recall	0.873	0.738	0.805	0.819	
support		600	400	1000	1000	-

Table 7: Metric results of the best model from the hyperparameter search trained in the training portion and tested in it. For the dataset transformed with stemming and word count.

Classifier	metric	0.0	1.0	m.avg	w.avg	acc
	f1-score	0.927	0.881	0.904	0.909	
AdaBoost	precision	0.898	0.931	0.914	0.911	0.910
	recall	0.959	0.836	0.898	0.910	
	f1-score	0.755	0.553	0.654	0.674	
DecisionTree	precision	0.705	0.636	0.670	0.677	0.684
	recall	0.813	0.489	0.651	0.684	
	f1-score	0.836	0.826	0.831	0.832	
GaussianNB	precision	1.000	0.703	0.852	0.881	0.831
	recall	0.719	1.000	0.859	0.831	
	f1-score	0.989	0.984	0.986	0.987	
GradientBoosting	precision	0.980	0.999	0.989	0.987	0.987
	recall	0.999	0.969	0.984	0.987	
	f1-score	1.000	1.000	**1.000**	1.000	
k-NN	precision	1.000	1.000	**1.000**	1.000	1.000
	recall	1.000	1.000	**1.000**	1.000	
	f1-score	0.996	0.994	0.995	0.995	
LogisticRegression	precision	0.993	0.999	0.996	0.996	0.996
	recall	1.000	0.989	0.994	0.996	
	f1-score	0.978	0.967	0.973	0.974	
RandomForest	precision	0.969	0.982	0.975	0.974	0.974
	recall	0.988	0.952	0.970	0.974	
	f1-score	0.980	0.968	0.974	0.975	
SVM	precision	0.961	0.999	0.980	0.976	0.975
	recall	1.000	0.939	0.969	0.975	
support		2400	1600	4000	4000	-

this portion and tested in the remaining are presented in Table 8. With the training and testing of this model in the entire dataset in Table 9. From here the Adaboost and the LogisticRegression remain some of the best performers with the k-NN overfitting the data.

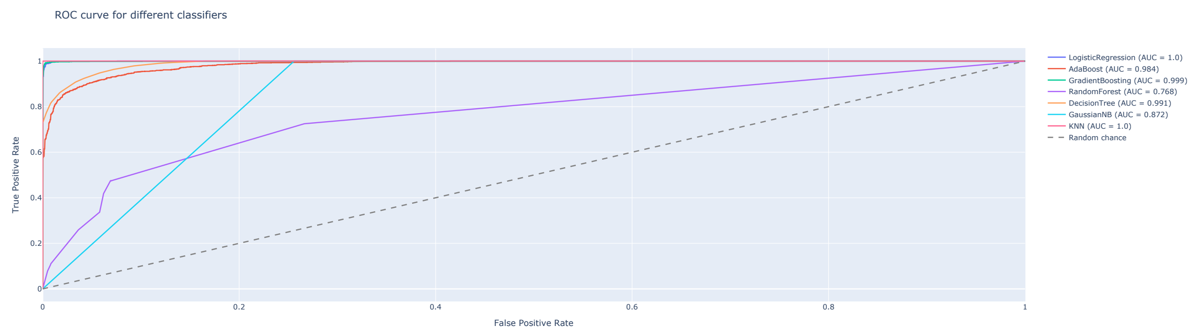
Figure 6 shows the diverse curve characteristics of the models trained on the entire dataset and tested in this. This serves as a confirmation of the models overfitting the data.

Summary

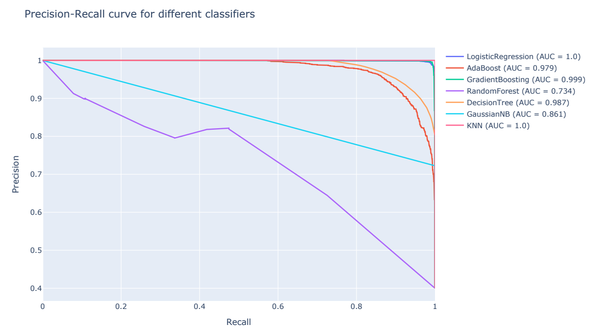
Despite the different approaches taken the results obtained are very similar in performance, with the overall metric results being close. The Adaboost, GrandientBoosting, and LogisticRegression are the best performers while not completely overfitting the data. Something that occurs in some models, particularly in the k-NN in all the cases. Something which is further compounded by the high dimensionality of the dataset which presents much more features than entries.

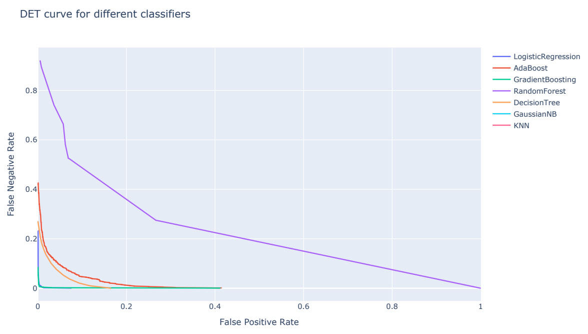
Such high dimensionality needs to be addressed and the selection of the TF-IDF parameters, TfidfVectorizer, instead of the default ones can present better results and this can also be considered as a hyperparameter to optimize. This philosophy can also be applied to the CountVectorizer. This presents even greater computational challenges and was out of the scope of this work due to the limited resources available.

Moreover, other metrics, besides the f1-score, can be analyzed in different ways. Here the approach was to consider a real-world application of the model where a set for testing



(a) Receiver Operator Characteristic curve.





(b) Precision-Recall curve.

(c) Detection Error Trade-off curve.

Figure 6: Curve characteristics for the best model in the hyperparameter search trained with the entire dataset and tested in it. For the dataset transformed with stemming and TF-IDF.

Table 8: Metric results of the best model from the hyperparameter search trained in the 80% portion of the dataset in the remainder for testing for the dataset transformed with stemming and TF-IDF.

Classifier	metric	0.0	1.0	m.avg	w.avg	acc
	f1-score	0.887	0.819	0.853	0.860	
AdaBoost	precision	0.865	0.854	**0.859**	0.861	0.861
	recall	0.910	0.788	0.849	0.861	
	f1-score	0.736	0.624	0.680	0.691	
DecisionTree	precision	0.752	0.606	0.679	0.693	0.690
	recall	0.722	0.642	0.682	0.690	
	f1-score	0.571	0.567	0.569	0.569	
GaussianNB	precision	0.709	0.474	0.591	0.615	0.569
	recall	0.478	0.705	0.592	0.569	
	f1-score	0.866	0.782	0.824	0.832	
GradientBoosting	precision	0.839	0.825	0.832	0.833	0.834
	recall	0.895	0.742	0.819	0.834	
	f1-score	0.766	0.370	0.568	0.608	
k-NN	precision	0.651	0.709	0.680	0.674	0.659
	recall	0.932	0.250	0.591	0.659	
	f1-score	0.881	0.826	**0.854**	0.859	
LogisticRegression	precision	0.890	0.815	0.852	0.860	0.859
	recall	0.873	0.838	**0.855**	0.859	
	f1-score	0.810	0.590	0.700	0.722	
RandomForest	precision	0.722	0.799	0.761	0.753	0.740
	recall	0.922	0.468	0.695	0.740	
	f1-score	0.864	0.796	0.830	0.837	
SVM	precision	0.864	0.797	0.830	0.837	0.837
	recall	0.865	0.795	0.830	0.837	
support		600	400	1000	1000	-

Table 9: Metric results of the best model from the hyperparameter search trained in the training portion and tested in it. For the dataset transformed with stemming and TF-IDF.

Classifier	metric	0.0	1.0	m.avg	w.avg	acc
	f1-score	0.952	0.924	0.938	0.941	
AdaBoost	precision	0.930	0.960	0.945	0.942	0.941
	recall	0.975	0.890	0.933	0.941	
	f1-score	0.954	0.932	0.943	0.946	
DecisionTree	precision	0.957	0.928	0.943	0.946	0.946
	recall	0.952	0.936	0.944	0.946	
	f1-score	0.871	0.854	0.863	0.864	
GaussianNB	precision	1.000	0.745	0.873	0.898	0.863
	recall	0.772	1.000	0.886	0.863	
	f1-score	0.993	0.989	0.991	0.991	
GradientBoosting	precision	0.987	0.999	0.993	0.992	0.992
	recall	0.999	0.980	0.990	0.992	
	f1-score	1.000	1.000	**1.000**	1.000	
k-NN	precision	1.000	1.000	**1.000**	1.000	1.000
	recall	1.000	1.000	**1.000**	1.000	
	f1-score	0.995	0.993	0.994	0.994	
LogisticRegression	precision	0.995	0.993	0.994	0.994	0.994
	recall	0.995	0.992	0.994	0.994	
	f1-score	0.818	0.604	0.711	0.732	
RandomForest	precision	0.728	0.827	0.777	0.767	0.750
	recall	0.934	0.476	0.705	0.750	
	f1-score	0.999	0.998	0.998	0.998	
SVM	precision	0.998	0.999	0.999	0.999	0.998
	recall	1.000	0.997	0.998	0.998	
support		2400	1600	4000	4000	-

was initially left out. Making the hyperparameter search in a subset of the data as would be the case for the real application. This allowed for a study of the models on how they could behave in deployment, Tables 2, 4, 6 and 8. However, due to the high dimensionality of the datasets constructed a more in-depth analysis was conducted to the possibility of overfitting, and the same was confirmed.

Furthermore, the preprocessing procedure for text analysis is of major importance especially when dealing with online content that presents multi-cultured content. Meaning that different languages and alphabets are present, posing additional challenges for the implementation. Which would either require a much larger dataset to be able to expand to all the possibilities or a way to interpret them all together, an alternative that poses other challenges due to different habits and customs. This situation, in combination with the labeling procedure, justifies a poor performance from the NLTK sentiment analysis.

Final Remarks

Mental health is a critical component of overall well-being, and it is essential to address it proactively to prevent extreme outcomes, such as suicide. This is particularly important for teenagers, who are especially vulnerable to mental health challenges, including stress, anxiety, and depression. The rapid physical, emotional, and social changes that occur during adolescence often intensify feelings of isolation or inadequacy, potentially leading to serious mental health issues if left unaddressed.

In this project, classifiers were constructed to determine whether a subject who authored a post is likely suffering from depression. This approach would enable the early identification of potential mental health concerns and facilitate timely intervention. The results demonstrate that the classifiers achieve significantly better-than-random predictions, with F1-scores, precision, and recall exceeding 0.85 on a separate testing set.

Despite the promising results, challenges remain, particularly in terms of dataset size and computational feasibility. The smallest dataset, derived from the transformation using stemming and word count, was 1.57 GB in size, while the largest, which used lemmatization and TF-IDF, reached 2.07 GB. These dataset sizes led to computational times that were not always practical.

Another limitation is the reliability of the true labels in the dataset, which is influenced by the data collection process. According to the analysis by Zhang et al. [11], based on the subreddits from which this dataset originates, there are inherent ambiguities. Specifically, distinguishing whether a user is experiencing depression or merely posting in related subreddits is more complex than validating the source of a post. These factors, combined, contribute to the moderate performance of sentiment analysis methods, such as those provided by the NLTK package. Moreover, the dataset's linguistic diversity poses additional challenges. Although the focus of this analysis was on English, not all dataset entries were guaranteed to be in English, limiting the effectiveness of the tools and methods.

Natural language processing presents inherent complexities, one of which is the high dimensionality of data. As observed in this project, this challenge can be addressed using novel techniques, such as those proposed by Singh et al. [8], which aim to reduce the computational burden. Further work in this area is crucial, as machine learning methods can enhance the early detection of mental health concerns, enabling faster responses and mitigating risks in vulnerable populations.

References

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[2] Simone Cunningham, Chloe C. Hudson, and Kate Harkness. "Social Media and Depression Symptoms: a Meta-Analysis". In: *Research on Child and Adolescent Psychopathology* 49.2 (Feb. 2021), pp. 241–253. ISSN: 2730-7174. DOI: 10.1007/s10802-020-00715-7.

[3] Elizabeth Englander. "Bullying, Cyberbullying, Anxiety, and Depression in a Sample of Youth during the Coronavirus Pandemic". In: *Pediatric Reports* 13.3 (2021), pp. 546–551. ISSN: 2036-7503. DOI: 10.3390/pediatric13030064.

[4] S. N. Ghaemi. "Digital depression: a new disease of the millennium?" In: *Acta Psychiatrica Scandinavica* 141.4 (2020), pp. 356–361. DOI: https://doi.org/10.1111/acps.13151.

[5] Chirag Gupta, Dr Sangita Jogdand, and Mayank Kumar. "Reviewing the impact of social media on the mental health of adolescents and young adults". en. In: *Cureus* 14.10 (Oct. 2022).

[6] Rishabh Kausish. *Reddit Depression Dataset.* 2024. DOI: 10.34740/KAGGLE/DSV/9361420.

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

[8] Ksh. Nareshkumar Singh et al. "A novel approach for dimension reduction using word embedding: An enhanced text classification approach". In: *International Journal of Information Management Data Insights* 2.1 (2022), p. 100061. ISSN: 2667-0968. DOI: https://doi.org/10.1016/j.jjimei.2022.100061.

[9] Jean M Twenge. "Why increases in adolescent depression may be linked to the technological environment". In: *Current Opinion in Psychology* 32 (2020). Socio-Ecological Psychology, pp. 89–94. ISSN: 2352-250X. DOI: https://doi.org/10.1016/j.copsyc.2019.06.036.

[10] Zannatun Nayem Vasha et al. "Depression detection in social media comments data using machine learning algorithms". In: *Bull. Electr. Eng. Inform.* 12.2 (Apr. 2023), pp. 987–996.

[11] Saijun Zhang et al. "Teens' Social Media Engagement during the COVID-19 Pandemic: A Time Series Examination of Posting and Emotion on Reddit". In: *International Journal of Environmental Research and Public Health* 18.19 (2021). ISSN: 1660-4601. DOI: 10.3390/ijerph181910079.