Comparing the Effectiveness of Support Vector Classifier and Stochastic Gradient Descent in Hate-Speech Detection

Dania Noman Ali *

October 17, 2023

Abstract

The increased use of Social Media with easy access to most people in the world has given rise to a multitude of problems; with cyberbullying and online hate-speech standing out as significant issues. With the choice of a user to maintain there anonymity and post most things that would be considered uncivil in a one-to-one real life conversation, has led to a widespread dissemination of online hate-speech, posing significant societal challenges and determinantal effects to an individual's mental health. In this paper, we explored two simple Classifiers, Support Vector Classifier (SVC) and Stochastic Gradient Descent (SGD) which are compared and analysed through there accuracy score to determine there effectiveness in detecting hate-speech within the context of Twitter data. To train the models, a publicly available dataset by Analytics Vidhya which can be found on Kaggle.com is used which contains 32k tweets labelled with a '1' if it is sexist/racist or '0' if it's not. The goal of this paper is identifying the differences in performances in hate-speech detection by the two classifiers

In Latex there are three different types of headings: sections, subsections, and subsubsections. Below you can see examples of how to make sections, subsections, and subsubsections.

1 Introduction

Cyberbullying, predominately in the form of hate-speech is a widespread phenomenon especially in the context of Twitter tweets. Sharing an individual's opinion with billions of people all around the globe, with the option to stay completely unknown has led Social Media to be a safe haven for the propagation of hate-speech using remarks that might be sexist/racist, derogatory against certain ethnicities, targeting religious minorities and/or defaming another individual based off there certain characteristics [1].

*Advised by: Maria Konte of Georgia Institute of Technology

In light of its latest rebranding to 'X', the social media platform has accrued a substantial user base, boasting approximately 450 million active participants. Official reports from Twitter indicate that these users collectively contribute to an average daily volume of approximately 500 million tweets. Each user, on average, invests approximately 30.9 minutes of their daily activity on the platform.

Notably, the scale of content generation on this platform is significant, with users capable of generating up to 2400 posts per day. It is worth emphasizing that this disproportionately large volume of user-generated content is disseminated with minimal to no pre-posting filtration or content moderation measures in place, rendering the platform susceptible to the proliferation of hate speech and other forms of harmful content. Social media companies invest millions of dollars in dealing with such issues, which mostly includes manual moderation and deleting posts/tweets that are deemed as 'offensive', hateful references, incitement, slurs and tropes, dehumanization and hateful imagery [2]. A study by the European Sociological Review, investigates the impact of perceived social acceptability on online hate speech and suggests that interventions based on descriptive norms, such as moderate censorship, can significantly reduce hate speech and guide future interventions in online communities to prevent the spread of hate [3]. Therefore, the goal of this paper is to investigate and compare the effectiveness of hate-speech detection by the two classifiers, namely, Support Vector Classifier and Stochastic Gradient Descent.

2 Text Classification

Text Classification is an important task under Natural Language Processing (NLP) whose primary objective being the automated allocation of text into predetermined categories. Examples of tasks that could be achieved by text classification are:

- Sentiment Analysis
- Classifying Emails as Spam or Non-spam
- Automation of answering queries of customers
- Categorizing News articles according to there topics

Text classification falls under the category of supervised learning, which means it relies on a dataset where each document is labelled with its respective category. This labelled data is used to train a classifier, which can then categorize new text documents accordingly. This process forms the basis for many text-related tasks and applications. For this research paper, we would be delving into two types of Classifiers, Support Vector Classifier (SVM) and Stochastic Gradient Descent (SGD). The aim of this paper is to find out which classifiers performs better in hate-speech detection. Stochastic Gradient Descent

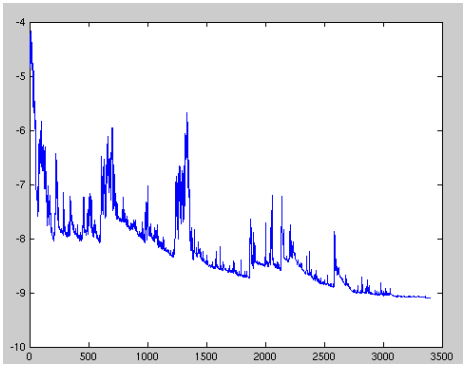


Figure 1: SGD frequently updates with significant variations, resulting in substantial fluctuations in the objective function, as depicted in Image 1:

2.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a variation of the Gradient Descent algorithm employed in machine learning optimization. It specifically targets the inefficiencies that arise when dealing with extensive datasets in machine learning projects. In the case of SGD, rather than using the entire dataset in each iteration, it selects a single random training example or a small batch to calculate the gradient and update the model parameters. This random selection introduces an element of randomness into the optimization process, hence the term "stochastic" in its name. The primary advantage of using SGD lies in its computational efficiency, especially when working with large datasets. It substantially reduces the computational cost per iteration compared to traditional Gradient Descent methods that require processing the entire dataset. Moreover, SGD excels in online learning scenarios, where data streams continuously, enabling real-time model adaptation with incremental updates. Here are the key steps involved in the SGD process: 1. Initialization: The model's parameters are randomly initialized. 2. Setting Parameters: You determine the number of iterations and the learning rate (alpha) for parameter updates. 3. Stochastic Gradient Descent Loop: The following steps are repeated until the model converges or reaches the maximum number of iterations: a. Shuffle the training dataset to introduce randomness. b. Iterate over each training example (or a small batch) in the shuffled order. c. Compute the gradient of the cost function concerning the model parameters using the current training example (or batch). d. Update the model parameters by taking a step in the direction opposite to the gradient, scaled by the learning rate. e. Assess convergence criteria, such as differences in the cost function between gradient iterations. 4. Return Optimized Parameters: Once the convergence criteria are met or the maximum iterations are reached, the optimized model parameters are returned.

SGD enhances efficiency by updating parameters one at a time, making it considerably faster and suitable for online learning applications. [4,5,6,7].

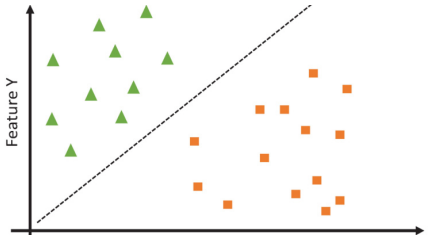


Figure 2: How an SVC Works - Source: ©SHUTTERSTOCK.COM/SIDHARTHA CARVALHO

tweet	labels
would you please ask these shameless @user @user give	1
goodnight my friends... stay blessed and highly favored!!! thursday fitfam	0
black women demonic porn	1
vandals turned a jewish family's menorah into a swastika" antisemitism hate	1
i'm pretty sure that warm weather and sun is my meditation sunshine meditate quietÃ	0

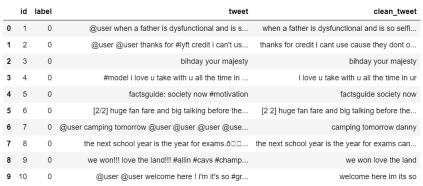
Table 1: Examples of tweets from the dataset and there corresponding labels, where it has a value of '1' if the tweet contains hate-speech (which is defined as sexist/racist remarks for the simplicity of this dataset), and a label of '0' if it doesn't.

2.2 Support Vector Classifier

2.2 Support Vector Classifier is a special case of the much broader, Support Vector Machine, that is primarily focused on classification tasks. SVC is a type of Supervised Learning, whose main goal is to find a hyperplane that best separates two classes or more classes (Binary or multiclass classification). SVCs work by maximizing the margin between the decision boundary (hyperplane) and the nearest data points from each class. These nearest data points are called support vectors. SVCs can handle both linear and nonlinear classification problems, depending on the choice of kernel function

3 Preparation of Dataset

The dataset used for this paper is from Kaggle.com, provided by Analytics Vidhya, named Twitter Sentiment Analysis. The dataset contains around 32,000 tweets, where label '1' denotes the tweet is racist/sexist and label '0' denotes the tweet is not racist/sexist. The usernames in this dataset are replaced by @user for the sake of copyright issues. This dataset was chosen due to large number of tweets, to better test the ability of the two classifiers. [8]



	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so selfi...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for credit i cant use cause they dont o...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	i love u take with u all the time in ur
4	5	0	factsguide: society now #motivation	factsguide society now
5	6	0	[2/2] huge fan fare and big talking before the...	[2 2] huge fan fare and big talking before the...
6	7	0	@user camping tomorrow @user @user @user @use...	camping tomorrow danny
7	8	0	the next school year is the year for exams.ð□□...	the next school year is the year for exams can...
8	9	0	we won!!! love the land!!! #allin #cavs #champ...	we won love the land
9	10	0	@user @user welcome here ! i'm it's so #gr...	welcome here im its so

Figure 3: Shows the cleaned tweets after preprocessing. This is essential for preparing Twitter text data for subsequent analysis or modeling by eliminating noise and unwanted characters from the tweets.

3.1 Support Vector Classifier

3.1.1 Preprocessing the Data

For the SVC Model, the data was cleaned by the 'tweet-preprocessor' library. It begins by installing the tweet-preprocessor library and imports essential libraries, including re for regular expressions and preprocessor (aliased as p) for tweet preprocessing. The regular expressions REPLACE NO SPACE and REPLACE WITH SPACE are defined to facilitate text cleaning. The custom function clean tweets is the core of the preprocessing pipeline. Given a DataFrame as input, it iterates through each tweet, utilizing tweet-preprocessor to remove Twitter-specific elements such as URLs and mentions.

3.1.2 Splitting The Data

The final step is to split the data into a training set and an evaluation set by the use of 'train test split' function. The training set denoted by 'x train' contains the cleaned tweet values and the corresponding target labels are denoted by the 'y train'. The testing set, x test, contains tweet values for evaluation, and y test has the respective target labels. The stratify=y parameter ensures a similar class distribution between the original dataset and the splits, useful for classification tasks. A random seed, random state=1, ensures reproducibility, and test size=0.3 allocates 30

3.1.3 Text Vectorization

Text Vectorization is a process through which numerical values are assigned to text. Although, there are multiple techniques to employ this method, we would be using the CountVectorizer() from the scikit-learn library which operates on a list of text documents stored in the document's variable. The CountVectorizer() records the number of repeats or the frequency of the word that occurs and prepares a matrix. For example, if data was: ["Twitter is fun", "I like posting tweets on twitter", "Some people on twitter post tweets that are mean!"] The matrix would be made as followed:

	are	fun	is	like	mean	on	people	post	posting	some	that	tweets	twitter
0	0	1	1	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0	0	1	0	0	1	1
2	1	0	0	0	1	1	1	1	0	1	1	1	1

Figure 4: Example of how CountVectorizer() from sklearn works

3.2 Stochastic Gradient Descent

3.2.1 Preprocessing the Data

A Python function is designed to perform the preprocessing for the SGD Classifier. The function, aptly named 'clean-text', accepts two parameters: a DataFrame (df) and the name of the text field within the DataFrame (text-field).

1. Text Lowercasing: Initially, the code employs the str.lower() method to convert all text within the specified 'text field' to lowercase. This step ensures uniformity in letter casing, mitigating potential discrepancies in the text data.

2. Text Cleaning and Tokenization: The subsequent operation is performed using a lambda function and the re.sub function from the re library. This operation serves the following purposes:

   - Removal of Twitter usernames or mentions (e.g., "@username").
   - Elimination of non-alphanumeric characters and special symbols from the text.
   - Exclusion of hyperlinks (e.g., "https://www.example.com") from the text.
   - Disregard for the "RT" (retweet) tag at the beginning of a tweet.
   - Removal of any remaining URLs.

3. Returning the Processed DataFrame: Finally, the function returns the DataFrame df with the specified 'text-field' transformed after the cleaning and tokenization operations.

This preprocessing function is a critical for the application of the SGD model, as it ensures that the text data is appropriately formatted and devoid of noise, enabling the subsequent model to effectively detect hate speech in a consistent and reliable manner.

3.2.2 Mitigating Class Imbalance through Un-sampling

The dataset that was taken had a majority of label '0' tweets and a class imbalance was encountered. To address this class imbalance, unsampling techniques were employed which begins by first segregating the training dataset into majority (which was label '0' tweets - non-hate-speech tweets) in this case, and

Figure 5: shows the code that employs the unsampling to mitigate class imbalance

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$
$$IDF(t) = log\frac{N}{1+df}$$
$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

Figure 6: How a TF-IDF works

minority (label '1' - hate-speech tweets). Resampling was done to the minority class to ensure that the minority and majority has an equal class size. The unsampled minority class is then combined with the original minority class, created a balanced training set. This rebalancing process prevents model bias towards the majority class and enhances the model's ability to detect hate-speech.

3.2.3 Text Classification

The pipeline consists of three fundamental stages. First, it employs the CountVectorizer to convert raw text data into numerical features, capturing the term frequencies of words within the documents. Following this, the TfidfTransformer is applied to transform these features into TF-IDF representations, factoring in the significance of terms across documents. Finally, the pipeline incorporates the SGDClassifier, a Stochastic Gradient Descent-based linear classifier, to make accurate predictions. This comprehensive pipeline seamlessly integrates text data preprocessing and classification, harnessing the TF-IDF methodology for informative feature extraction and the SGD classifier's efficiency, making it well-suited for large-scale text classification tasks, particularly in the context of hate-speech detection

3.2.4 Splitting the Data

Utilizing the train-test-split function from the sklearn.model-selection module, the dataset is divided into two distinct subsets: the training set and the testing set. X-train and y-train capture the feature matrix and corresponding labels of the training data, essential for training the SGD classifier. Alternatively, X-test

Figure 7: Text Classification Pipeline for Hate-Speech Detection

Classifier	Accuracy Score	F1 Score
Support Vector Classifier	94.8378	56.5408
Stochastic Gradient Descent	96.9179	96.9471

Table 2: Metric Evaluation of the Classifiers

and y-test house the feature matrix and labels of the testing data, dedicated to assessing model performance. This segregation ensures that the classifier is rigorously evaluated on unseen data. Furthermore, by specifying random-state=0, we establish reproducibility, enabling consistent and replicable results across multiple runs.

3.2.5 Model Training and Prediction

Firstly, the pipeline-sgd, which encapsulates the text classification pipeline including vectorization and classifier components, is fitted to the training data (X-train and y-train). This operation trains the SGD classifier on the prepared training dataset, enabling it to learn patterns and associations within the text data. Following the model training, predictions are generated for the testing data (X-test) using the trained model. The predict method is applied, producing y-predict, which comprises the model's predictions for hate-speech labels on the testing dataset. This step represents the critical evaluation phase, where the model's effectiveness in hate-speech detection is quantified by comparing its predictions (y-predict) to the actual labels (y-test) from the testing dataset.

4 Results

Scores of the Classifiers

The results indicate that the Stochastic Gradient Descent (SGD) Classifier outperformed the Support Vector Classifier in terms of both Accuracy and F1 Scores. Specifically, the SGD achieved an accuracy score of 96.9179, which is notably higher than the Support Vector Classifier's accuracy score of 94.8378. Similarly, in terms of the F1 score, the SGD Classifier achieved a significantly higher score of 96.9471, while the Support Vector Classifier scored 56.5408. This improvement in performance may be attributed to several factors.

One crucial difference lies in the pre-processing of the text data. The SGD Classifier implemented a custom text cleaning function, 'clean-text,' which performed operations such as converting text to lowercase and removing special characters, mentions, URLs, and other non-alphanumeric characters. In contrast, the Support Vector Classifier relied on the 'tweet-preprocessor' library for pre-processing, which may not have been as extensive. Notably, the SGD's pre-processing included removing mentions and URLs, contributing to cleaner text data.

Additionally, the SGD Classifier addressed the imbalance in the dataset by resampling, which involved oversampling the minority class. This step is

crucial for dealing with imbalanced data sets and can significantly impact model performance. In contrast, the Support Vector Classifier did not incorporate such measures, which might have influenced the differences in model performance.

Another distinguishing factor is the choice of text vectorization. The Support Vector Classifier employed CountVectorizer() with binary representation, while the Stochastic Gradient Descent used TfidfVectorizer(). The choice of vectorizer can influence the representation of text features, with TF-IDF potentially capturing term importance more effectively than CountVectorizer.

Furthermore, both models employed different evaluation metrics. The Support Vector Classifier relied on the Accuracy score for model evaluation, while the Stochastic Gradient Descent used the F1 score. The choice of evaluation metric is essential, and the F1 score, utilized by the SGD Classifier, is particularly suited for imbalanced datasets, which could have contributed to its higher overall score.

These differences in preprocessing, class imbalance handling, text vectorization, and evaluation metrics collectively explain the superior performance of the Stochastic Gradient Descent Classifier in this study.

5 Conclusion

In this research paper, we conducted an in-depth analysis of two classifiers, the Support Vector Classifier (SVC) and the Stochastic Gradient Descent (SGD), to assess their effectiveness and suitability in the context of hate-speech detection. In the present era, where the proliferation of hate-speech on social media platforms poses a pressing concern, the need for robust classifiers is paramount. Our study revealed that both classifiers demonstrated promising results in identifying hate-speech; however, the Stochastic Gradient Descent (SGD) classifier exhibited superior performance with an impressive F1 score of 96.96, as opposed to the Support Vector Classifier's accuracy score of 94.84.

Several critical factors contributed to this discrepancy in performance. Firstly, the preprocessing techniques employed by each classifier played a pivotal role. The SGD classifier utilized an extensive custom text cleaning function, 'clean-text,' which not only converted text to lowercase but also effectively removed mentions, URLs, and various non-alphanumeric characters. Moreover, the SGD classifier proactively addressed class imbalance through resampling. In stark contrast, the Support Vector Classifier relied on a pre-made function for preprocessing and failed to account for class imbalance.

The choice of text vectorization further distinguished the classifiers. The Support Vector Classifier opted for CountVectorizer() with binary representation, while the SGD classifier made use of TfidfVectorizer(), a decision that enhanced its capacity to capture term importance more effectively.

our study showcased the potential of both classifiers in the area of hate-speech detection. Nonetheless, it is evident that the Stochastic Gradient Descent (SGD) classifier, with its comprehensive preprocessing, class imbalance handling, and advanced vectorization technique, emerged as the more powerful

tool for this critical task. Further research and experimentation are needed in order to refine our understanding of the most effective approach and to continue addressing the ever-evolving challenge of hate-speech detection in the digital age.

References

[523] Ml: Stochastic gradient descent (sgd). 2023.

[623] Difference between batch gradient descent and stochastic gradient descent. 2023.

[Alv17] Winter F Alvarez, A. Normative change and culture of hate: An experiment in online environments. . *European Sociological Review*, 2017.

[Ban22] S. Bansal. A comprehensive guide to understand and implement text classification in python. *Analytics Vidhya*, 2022.

[Bot18] Curtis F. E. Nocedal J Bottou, L. Optimization methods for large-scale machine learning. *arXiv.org*, 2018.

[Dea12] Corrado G. S. Monga R. Chen K. Devin M. Le Q. V. Mao M. Z. Ranzato M. A. Senior A. Tucker P. Yang K. Ng A. Y Dean, J. Large scale distributed deep networks - neurips. large scale distributed deep networks. *NeurIPS (Conference on Neural Information Processing Systems)*, 2012.

[Hui19] P Huilgol. Accuracy vs. f1-score. *medium.com*, 2019.

[Too] A. (n.d.) Toosi. Twitter sentiment analysis.

[Twi] Twitter. X's policy on hateful conduct x help. twitter. *rules-and-policies/hateful-conduct-policy*.

[Zha18] Z Zhang. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *arXiv*, 2018.