CPTS 437 – Project Write-Up
Classification of Vaccine Attitudes via Tweet Content

Ethan Nelson
Aiden Walker
Reed Havens

12/3/23

Introduction

This paper will offer a high-level overview on the team's project to classify vaccine attitudes via Tweet content through the use of various machine learning (ML) models. This report will cover 5 primary sections:

1. Problem Statement and Significance
2. Data and Pre-Preparation Steps
3. Solution Approach
4. Results and Analysis
5. Conclusions and Future Work

This report is a part of a three-piece collection including the presentation video and code documentation and demo, all of which is available on the team's GitHub repository located below. Links to all additional deliverables are included here as well for convenience:

GitHub
- https://github.com/reed271/437_Final_Project/tree/main

Colab Demo & Documentation:
- https://colab.research.google.com/drive/12dLOr1zcvVOal9gBJ4AK_drWn2EGjKl5?usp=sharing

Demo Instructions
- Available on GitHub

Video Presentation:
- https://drive.google.com/file/d/1bFjmvdxD0aaPxaD9skVlQN57-hq2DY-1/view

Presentation Slides:
- https://docs.google.com/presentation/d/16xsHj9aoFwom8KvvDwoNtzcpZUbh_PFcndjy8gOOG0w/edit?usp=sharing

1. Problem Statement and Significance

The COVID-19 virus emerged in 2019 and quickly developed into an international pandemic, and it continues to affect the world today. The WHO estimates that since its emergence, COVID-19 is responsible for approximately 6.9 millions deaths, globally. Despite this danger, the pandemic saw an unprecedented rise in vaccine hesitancy, the delay or refusal of acceptance of vaccines despite ample supply and evidence. This international rise in vaccine hesitancy bolstered the spread of the virus and exacerbated the scale and length of the pandemic via the hampering of herd immunity. Furthermore, the trend has continued in years following and has resulted in a 3% vaccine exemption rate in kindergarteners, the highest rate ever reported by the CDC (Kekatos, 2023).

To combat this dangerous trend, researchers have sought to better understand the complaints raised by individuals so that they may better address and educate them. One major source of complaints from vaccine-hesitant individuals is X.com (formerly known as Twitter, but will be henceforth referred to as Twitter). These complaints range in variety, with some noting concern over unknown long-term side effects, some citing the effectiveness of natural remedies in comparison, and some claiming that the virus and vaccine were components of a government conspiracy, to name a few.

A major component of researching these adverse responses is effectively classifying their complaints and attitude towards vaccines. This allows for a better understanding of the distribution of attitudes, which can inform educational campaigns suited to address the largest, most accessible communities.

Our project objectives are twofold: the primary objective is to effectively classify individual Tweets into categories based on vaccine attitude using multi-class machine learning classifiers; the secondary objective is to evaluate the various ML models used and analyze their comparative performance. The primary objective is suited directly to the problem statement, while the secondary objective allows the team to flesh out known concepts and explore new ones in an effort to deepen their understanding of the machine learning process.

2. Data and Pre-Processing

The dataset used in this project is the Twitter Multilabel Classification Data set from Kaggle.

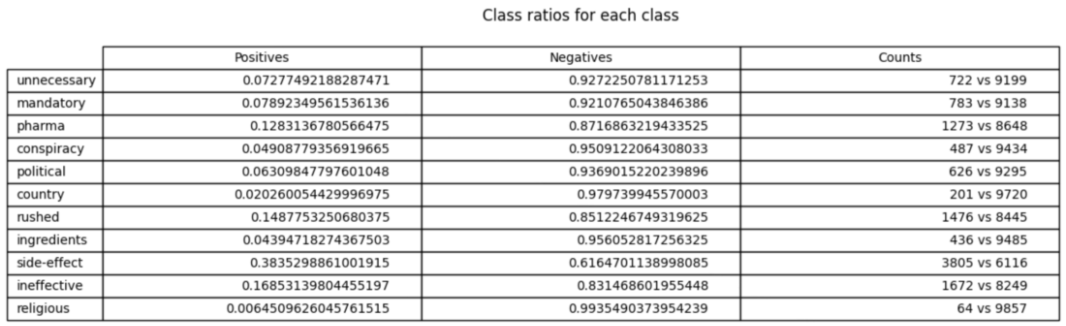
Source:
https://www.kaggle.com/datasets/prox37/twitter-multilabel-classification-dataset

This dataset contains 9,921 entries, where each entry contains a Tweet, entry ID, and an array of vaccine attitude labels expressed in the given Tweet. The following are the 12 candidate labels within the dataset into which we aim to classify Tweets:

1. **Unnecessary** – The tweet indicates vaccines are unnecessary, or that alternate cures are better.
2. **Mandatory** – The tweet suggests that vaccines should not be made mandatory.
3. **Pharma** – The tweet indicates that the Big Pharmaceutical companies are just trying to earn money
4. **Conspiracy** – The tweet suggests some deeper conspiracy, and not just that the Big Pharma want to make money
5. **Political** – The tweet expresses concerns that the governments / politicians are pushing their own agenda though the vaccines.
6. **Country** – The tweet is against some vaccine because of the country where it was developed / manufactured
7. **Rushed** – The tweet expresses concerns that the vaccines have not been tested properly or that the published data is not accurate.
8. **Ingredients** – The tweet expresses concerns about the ingredients present in the vaccines
9. **Side-Effects** – The tweet expresses concerns about the side effects of the vaccines, including deaths caused.
10. **Ineffective** – The tweet expresses concerns that the vaccines are not effective enough and are useless.
11. **Religious** – The tweet is against vaccines because of religious reasons

Tweets within this dataset may contain 0-11 of these labels. Therefore, the classification task will focus on categorizing the given Tweet into the most relevant category from those available, based on a measurable statistic.

This dataset required several pre-processing steps before it could be utilized in the Tweet classification problem solution. First, it was necessary to clean up the original dataset by removing most stop words (excluding select terms such as “no” and “not”) and filtering out superfluous symbols, such as “@” and “#”. Second, the Natural Language Toolkit (NLTK) library was utilized to help filter and lemmatize our Input Sentences, the cleaned-up and prepared Tweet content that can be utilized by our ML models. This ensures consistency across our input data and aids in normalization of both input and output. Finally, we convert our original Y data (the original set of output labels) to a binary output vector to facilitate the training of our models. Now our dataset is of a configuration that can be utilized by standard machine learning models.


Class ratios for each class

	Positives	Negatives	Counts
unnecessary	0.07277492188287471	0.9272250781171253	722 vs 9199
mandatory	0.07892349561536136	0.9210765043846386	783 vs 9138
pharma	0.1283136780566475	0.8716863219433525	1273 vs 8648
conspiracy	0.04908779356919665	0.9509122064308033	487 vs 9434
political	0.06309847797601048	0.9369015220239896	626 vs 9295
country	0.020260054429996975	0.979739945570003	201 vs 9720
rushed	0.1487753250680375	0.8512246749319625	1476 vs 8445
ingredients	0.04394718274367503	0.956052817256325	436 vs 9485
side-effect	0.3835298861001915	0.6164701138998085	3805 vs 6116
ineffective	0.16853139804455197	0.831468601955448	1672 vs 8249
religious	0.0064509626045761515	0.9935490373954239	64 vs 9857

3. Solution Approach

The team’s approach to the project was twofold: the primary objective was to effectively classify Tweets by their most relevant vaccine attitude label; the secondary objective was to explore the comparative effectiveness of various machine learning models in this task and evaluate their performance. These two goals in conjunction are intended to provide a meaningful application while ensuring the team was able to push the bounds of their existing knowledge and skills. The fundamental algorithm at play is text analysis, where the text of a Tweet receives a continuous classification probability score for each candidate label in Y. All tested models utilize the same training and testing data, such that the same preprocessing steps of lemmatization and filtering will be applied to all model input data. The team utilized the following seven multi-label classification models in our research:

1. Linear SVM
2. Multinomial Naive Bayes
3. Bernoulli Naive Bayes
4. Random Forest Classifier
5. Gradient Boosting Classifier

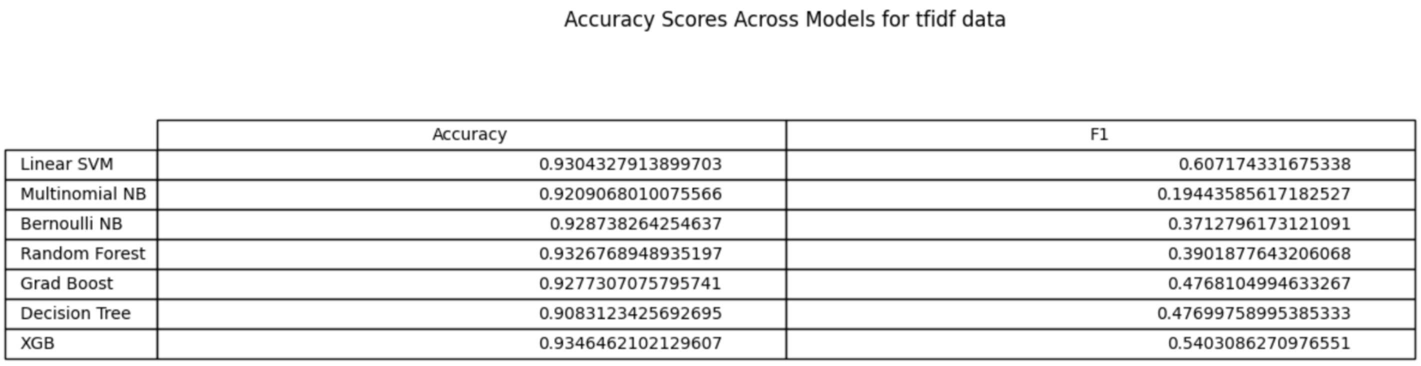
6. Decision Tree Classifier
7. XGB Classifier

These seven models utilize different underlying methods for multi-label classification, and as such, produce varying results. In the evaluation of each model, a variety of training and testing data is utilized in search of optimal performance. A custom 'generate_data()' function is utilized to generate different sets of train and test data; it creates a set of count vectorized features as well as a set of TF-IDF vectorized features and adds them to our master Data dictionary. It also creates variations within these two sets, modifying the min_df and ngram ranges. Modifying the min_df value allows us to filter greater or fewer infrequent words at a chosen rate of infrequency; modifying the ngram value allows us to change the number of words in our analyzed strings. Modifying these two parameters in our test and train data allows us to further optimize the results of our classification task, regardless of the model ultimately chosen.

The following section will discuss the empirical results of the project, as well as analysis of those results and of the performance of each classifier.
Analysis using LDA (Latent Dirichlet Allocation) along with Textstat's text feature extract did not yield comparatively good results - and so count and TF-IDF methods were preferred.
In addition, we attempted feature reduction via sklearns PCA analysis and Select K Best, but these were ineffective at helping reduce the feature sizes, due to various issues like the multiple classes and the huge feature/item count sizes.

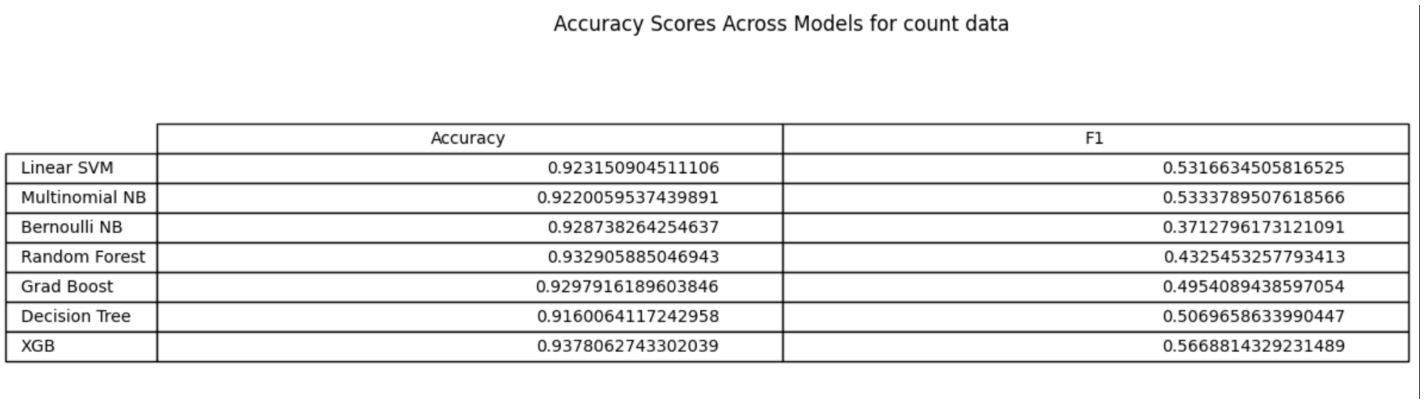
4. Results and Analysis

The team measured each model's performance via F1 Score, a comprehensive measure of accuracy that combines a model's precision and recall into a single statistic. These two metrics are often in tension, and by measuring the F1 Score, we are able to ascertain a more complete picture of the model's overall performance. In addition, the accuracy of each model was measured, to give an overall idea of how it is performing. The scores are calculated for each class, and are averaged to show the overall performance.


Accuracy Scores Across Models for tfidf data

	Accuracy	F1
Linear SVM	0.9304327913899703	0.607174331675338
Multinomial NB	0.9209068010075566	0.19443585617182527
Bernoulli NB	0.928738264254637	0.3712796173121091
Random Forest	0.9326768948935197	0.3901877643206068
Grad Boost	0.9277307075795741	0.4768104994633267
Decision Tree	0.9083123425692695	0.47699758995385333
XGB	0.9346462102129607	0.5403086270976551

Figure 1 – Test Accuracy & F1 Scores Across Models for TF-IDF Data


Accuracy Scores Across Models for count data

	Accuracy	F1
Linear SVM	0.923150904511106	0.5316634505816525
Multinomial NB	0.9220059537439891	0.5333789507618566
Bernoulli NB	0.928738264254637	0.3712796173121091
Random Forest	0.932905885046943	0.4325453257793413
Grad Boost	0.9297916189603846	0.4954089438597054
Decision Tree	0.9160064117242958	0.5069658633990447
XGB	0.9378062743302039	0.5668814329231489

Figure 2 – Test Accuracy & F1 Scores Across Models for Count Data

Figure 1 showcases the results of the models when trained on TF-IDF vectorized data, while Figure 2 showcases the results of the models when trained on count vectorized data. Globally, we observe the best performance in the Linear SVM (TF-IDF). In the count vectorized group, the XG classifier offers the best performance, while the Linear SVM is relatively close behind. We must note that in both primary data groups, the Linear SVM and XG Classifier were the leading two models in F1 Score. Now, we will break down the results by classifier and explain each model's performance.

Linear SVM

The Linear SVM is our highest-performing model overall, reporting an F1 Score of 60.7% on the TF-IDF set. In the count set, the model still performed very well and came in second place, reporting a score of 53.2%. This results in an average score of 57%. The team has debated whether average score indicates anything meaningful and has tended towards the negative – certain models perform better in certain contexts, and optimal performance is fairly divorced from average performance across all configurations. Given that we're focused on optimal performance, we've chosen to focus less on average score across both data sets.

The Linear SVM operates by attempting to find a hyperplane between decision boundaries (in this case, the boundary between vaccine attitude labels) in n-dimensional space. It then classifies new entries based on their spatial proximity to the existing hyperplanes, and modifies the hyperplanes if necessary. The performance of the SVM in our experiments demonstrated that the dataset is generally linearly separable, and as such, it is no surprise that the Linear SVM performs well – given an existing decision boundary, it should learn that boundary in a finite amount of time.

Multinomial Naive Bayes

The Multinomial Naive Bayes (MNNB) model reported mediocre performance in comparison to other models, showcasing 53.3% accuracy on the count set, and 19.4% accuracy on the TF-IDF set. The Multinomial NB classifier is often used to classify text documents due to its ability to assign probabilities to output labels and utilize reduced complexity in comparison to other models, leading to reduced training time and computational load. Ultimately, these benefits of the MNNB model are difficult to observe in our experiment, where our dataset is relatively small and we are not directly analyzing time and computation costs of training a model. It is likely that the MNNB would score relatively higher were our dataset larger or if our score incorporated some degree of training time.

Bernoulli Naive Bayes

The BNB classifier reported an F1 score of 37.12% on the count set, and a score of 37.12% on the TF-IDF set. The BNB model did not perform well relative to the other models, but this was fairly expected. The BNB is best suited to handling binary or boolean features, which is to say that continuous textual features such as those in our data set are not well suited to the BNB classifier. This model helps to serve as a control within our experiment, demonstrating that the performance of the other classifiers is indeed due to their nature and suitability to the given problem, rather than a statistical anomaly. This is supported by the lack of delta between performance scores for the BNB classifier.

Random Forest

The Random Forest Classifier (RFC) achieved an F1 Score of 42.4% on the count set, and a score of 40.06% on the TF-IDF set. The RFC is a robust model that can be used for both classification and regression, and as such, supports both atomic and continuous features. The RFC's relatively low performance could be indicative of a few things; first, it may demonstrate an imbalance in our data set, such that there is a fairly uneven distribution of our 12 class labels. It may indicate that the RFC is sufficiently distinct from the other models such that it would require specialized tuning to optimize; here, we would likely experiment with the maximum depth, number of trees, and number of samples per node, and we could likely find a superior combination. However, for more out-of-the-box problems, the RFC seems relatively unfit. It requires further tuning to optimize the problem, or it may be that our dataset is simply imbalanced and thus not well suited to the RFC.

Gradient Boosting Classifier

The Gradient Boosting Classifier (GBC) scored 49.2% on the count set, and 48.12% on the TF-IDF vectorized set. The GBC is an ensemble classifier that continuously trains on the error of the previous iteration, similar to the gradient descent algorithm. This model saw moderate performance across the board, but still fell behind the Linear SVM by a large margin. This dip in performance can be explained as follows: we have not sufficiently tuned the hyperparameters of the GBC which may hamper performance; additionally, the GBC is more sensitive to the scale of input features than the SVM. Furthermore, it is possible our base learners employed are too weak to improve on pace. The GBC has fallen behind the SVM, but it is believed that performance could be improved through hyperparameter tuning.

Decision Tree Classifier

The Decision Tree Classifier (DTC) scored 49.7% on the count set, and 47.1% on the TF-IDF set. It must be noted that the ensemble classifiers seem to output relatively similar performances, all approximately between 40-50% on their F1 scores. The common denominators across all models are the input data sets – this suggests that this commonality may be caused by the data. Ensemble methods can be susceptible to imbalanced data, and these consistent results would suggest that the input data set is indeed imbalanced in class distribution (i.e.: by output label). This could also suggest that the ensemble methods require more fine-tuning and optimization than other algorithms such as the Linear SVM.

Extreme Gradient Boost Classifier

The Extreme Gradient Boost (XGB) Classifier scored 56.6% on the count set, and 54.03% on the TF-IDF set. This ranks it as the second-best performing classifier behind the Linear SVM. XGB outperforms our other ensemble learning methods because it automatically addresses many of our previous issues. XGB includes a regularization term to combat overfitting, absent in Random Forests; it also provides a built-in feature importance score to help discriminate among features; XGB is effective at capturing non-linear relationships, and it comes with an optimized splitting-criteria algorithm. These built-in additions help to address our previous shortcomings in our ensemble learners, and the effect is observed in XGB’s comparatively higher F1 score than the other ensemble learners.

4. Conclusions and Future Work

The team’s primary objective of classifying Tweets by vaccine attitude was successful. The team created a model, a Linear SVM classifier, that has achieved an F1 score of 60.71% when classifying Tweets into one of twelve categories, and an accuracy score of 93.04%. This is deemed successful based on the simple probability of classifiers. A perfect binary classifier, such as a coin, will produce an average score of 50% through random guessing, via: (100 / (number of classes)) = expected score for random guesses. Given that we have 12 classes, we may say our expected accuracy for random guessing would be 100/11 = 9.09%, which our models have far exceeded. However, our class distributions are uneven. Because 89% of our outputs are in fact ‘0’ in the Y vector, we should expect a default accuracy of 89%, which our models have all exceeded in their base accuracy score. Therefore, the team believes that we have fulfilled our primary objective.

Our secondary objective was to explore the comparative performance of machine learning models in the execution of the primary objective to deepen team understanding and learning. This objective was also successful. When training models on various sets of count vectorized and TF-IDF vectorized data, we see that the Linear SVM classifier reports the highest performance overall, while the XGB Classifier reports the second highest performance overall. These results were somewhat expected, yet still offered crucial insights to the team.

The Linear SVM was the best performing model for a number of reasons. SVMs allow for linear separability in higher dimensions, and the TF-IDF vectorization of words results in a high-dimension space. Linear patterns present in this space allow the SVM to make effective classifications. Additionally, SVMs are well-suited to handling sparse feature spaces, a form that text documents most often embody. As a model, SVMs have been well-documented in their ability to effectively classify text, particularly because of their ability to generalize beyond the training set. The XGB Classifier is quite different, instead utilizing an ensemble of weak learners and gradient descent operations to improve upon the model's error rate continuously. The XGBC also comes with many built-in tools and features to address the shortcomings of other ensemble learners, such as an included regularization term, feature importance analysis, parallelization and scalability, and an ability to handle imbalanced datasets. Additionally, weak learner ensembles are generally quite effective at capturing underlying linguistic patterns within a text. Ultimately, it is clear why the Linear SVM and XGBC outperformed all other models tested, and it primarily comes down to the robustness of the model, the fit for the given task and dataset, and the lack of required hyperparameter tuning.

If research were to continue along this tract, the team would pit the Linear SVM and the XGBC against each other to determine which can achieve the highest possible F1 score. This optimization process would primarily involve hyperparameter tuning, in which the team would test the models with various configurations of hyperparameters to determine the optimal setup for the given task. For the XGBC, this would be variables such as number of trees, max depth, minimum child rate, and L1 and L2 regularization terms, to name a few. For the SVM, we may be tuning variables such as class weights, loss function, penalty for error, and the regularization parameter, to name a few. By modifying the hyperparameters of these two models, the team believes that they could achieve higher F1 scores than those found in this report.