Classification of COVID Related Tweets

Performance Comparison between different ML model and NLP techniques

Nisha Poudel
CSCE
University of Nebraska Lincoln
Nebraska, USA
npoudel2@huskers.unl.edu

Sujan Shrestha
CSCE
University of Nebraska Lincoln
Nebraska, USA
sshrestha11@huskers.unl.edu

Boyang Hu
CSCE
University of Nebraska Lincoln
Nebraska, USA
boyang.hu@huskers.unl.edu

***Abstract*—The 2020 COVID pandemic posed severe challenges in the public health system of all nations. Amidst the unprecedented situation the world is undergoing, we wanted to explore how people are reacting to COVID-19. We chose to focus on the tweets since Twitter has become a popular means of communication nowadays. Our research quickly discovered that the mass population was sharing valuable COVID related knowledge in terms of tweets, pictures, emojis, etc. In this research project, we attempt to use machine learning to mine COVID-related tweets from the internet. In this study, we collected 1678 tweets related to COVID-19 and 4567 general tweets from Lincoln, Nebraska. We solved a supervised classification problem {COVID vs. Non-COVID tweets} by using Naive Bias, Linear support vector model, and logistic regression machine learning models, and compared the results. After we ran the three models on the given dataset, we got 93% accuracy for Multinomial Naive Bayes, 92.4% accuracy for Linear SVM, and 93% accuracy for Logistic Regression model. So, we found all three models worked consistently the best for our COVID tweet classification problem.**

***Index Terms*—Linear Regression, Linear SVM, Multinomial Naive Bias, classification**

I. Introduction

'Chaos theory' suggests that even random-looking data can have hidden patterns or high valued information stored within it. Similarly, the COVID-19 related tweets may have some hidden valuable information stored within them. It could be the key to control and manage the COVID-19 crisis effectively. According to the latest CDC report, there are over 35.9 million cases of COVID worldwide to this date, and it is still growing exponentially [1]. Even with that many cases, we still have very little information on COVID, and information from hospitals about COVID is virtually nonexistent. Before we started our project, we believed that information on COVID-19 could not only come from databases generated by hospitals but also from tweets generated by people from many locations who might have experienced the pandemic firsthand. So, it is essential to predict any COVID-19 related tweets to extract the valuable information hidden within them.

II. Problem Definition

Applying machine learning to analyze COVID-19 data is a new hot topic; Currently, there aren't many good models that can distinctly identify COVID and non-COVID tweets. In addition to that, there is not enough data on COVID-19 that can support any model. Hospitals, health organizations have been working day and night to gather data on COVID, but it is just not enough. Moreover, many researchers are currently on hold due to the lack of useful COVID-19 data.

III. Research questions and Hypothesis

By classifying tweets into COVID and Non-COVID, we primarily hope to learn any new information on COVID. We hope to learn how different people are reacting to the COVID-19 crisis from their tweets. We also hope to learn information on people's sentiments, emotions, rage, or anger in the context of COVID. The findings from our study could be further be rationalized using sentiment analysis. We hope to discover any correlation among COVID-related tweets and features such as geographic location, which might help track how different regions are responding to the crisis. This information could later prove to be extremely beneficial in developing effective strategies to combat any pandemic. We begin our research by making a strong assumption that any COVID related tweet will have keywords such as 'COVID', 'CORONA', 'PANDEMIC' etc. [2]

During our research, we faced the following questions:

- How do we collect the tweets that satisfy our requirements?
- In the absence of good data, how can we correctly distinguish COVID tweets from Non-COVID Tweets?
- What machine learning models should we use to in order to distinctly classify the tweets?

IV. Data Summary

A. Dataset

In the absence of a proper dataset, we were compelled to create our own dataset that contained around 6000 COVID and non-COVID tweets. We used Twitter API to collect different tweets for our dataset. We began the data collection process by first downloading the COVID and non-COVID related data from Twitter for 7 days from Oct 28 to Nov 5. The spatial coverage of tweets was Lincoln, Nebraska(10 Km radius). We originally planned to extract tweets for the time period of May through August. However, we were stopped by Twitter's limitation for that date range as Twitter only allowed us to get tweets for the past 7 days and no more. We selected a 10km radius of Lincoln because we wanted to see how people in

Lincoln were reacting to COVID and reduce the number of tweets we were getting without it. In the end, 1628 COVID-related tweets were downloaded, and 4808 general tweets (that contained COVID and non-COVID tweets) were downloaded from Twitter. We then needed to filter COVID and non-COVID tweets in the general tweets to label the tweets correctly. The COVID related tweets were labeled as 1 and Non-COVID as 0. We made the distinction between COVID and no COVID tweet based on 36 keywords. [2] Then, the COVID and non-COVID tweets were combined in a data frame for further processing.

B. Exploratory Data Analysis(EDA)

Exploratory Data analysis is a crucial step in the process of building a machine learning model. This step provides detailed insights into our data. We may remove any redundant or highly correlated data before we fit them into our model. We can significantly reduce the dimensions of data through this process and thus reducing the model training time.
In addition to that, from EDA, we learned that the length of COVID tweets was smaller than that of Non-COVID with some outliers which was a very interesting find.

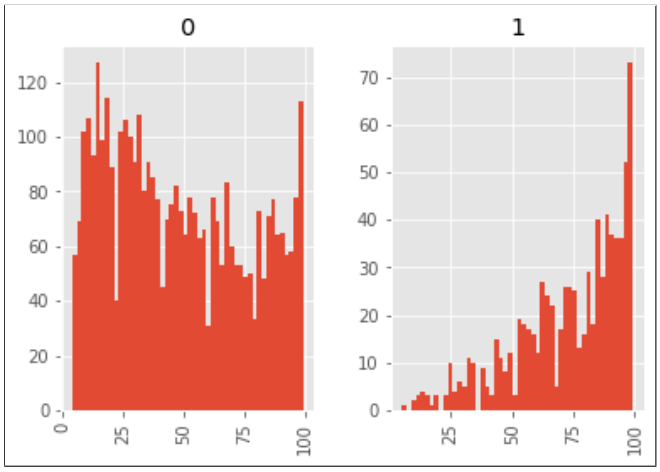

Fig. 1. Length of Tweets

C. Feature Extraction

A feature is anything that can correctly represent the given data. The process of extracting the important features from the given dataset is known as feature extraction [3]. In our research study, we used three steps for feature extraction.

- Text Normalization (Stemming or Lemmatization)
- Text Preprocessing (Tokenization, removing stop words, etc.)
- Vectorization of the features

1) Text Normalization : Languages we speak and write contain several words that are often derived from one another. Text Normalization reduces those words to their root form to speed up the learning process. Stemming and Lemmatization are Text Normalization (or sometimes called Word Normalization) techniques in Natural Language Processing that are used to prepare text, words, and documents for further processing. Generally, Lemmatization is more suitable for text classification [3]. During lemmatization, inflected words are grouped together so they can be analyzed as a single item, identified by the word's lemma or dictionary form. Thus, words like 'moved' and 'moving' will be reduced to 'move.' In this research, we will explore three text Normalization techniques.

- Lemmatization
- Stemming
- Lemmatization and Stemming

2) Text Preprocessing : Using Text features in a machine learning model can be computationally expensive. Therefore, we perform text preprocessing that tokenizes the texts, removes the stop words, and converts textual features to a matrix of token count form easily computed by a machine learning model. **Stop words** are words like "and", "the", "him", which are presumed to be uninformative in representing the content of a text. At the end of Text Preprocessing, we are left with a sparse representation of the word vector counts using scipy.sparse.csrmatrix [3]. In order to convert the features to a matrix of tokens, we use two types of feature/word embedding technique that are **Frequency based word embedding** and **Prediction based word embedding**

Frequency based word embedding The frequency-based methods use deterministic methods to compute word vectors. They do not consider the contextual nature of the words in the text while creating a numerical representation. Inside Frequency-based embedding, we explored two different techniques of computing the word vector.

- Count vectors using Bag of words
- TF-IDF vectors using TF-IDF transformer

Bag of Words is simply a collection of words to represent a sentence or text with word count and mostly disregarding the order in which they appear.Bag of words model is a collection of unigrams [3].There are some limitations of this model:

- It cannot capture phrases and multi-word expressions.
- It effectively disregards any word order dependence.
- It does not account for potential misspellings or word derivations.

A better and sophisticated model in bag of words for feature representation is the n-grams model. Instead of building a simple collection of unigrams (n=1), one might prefer a collection of bigrams (n=2), where occurrences of pairs of consecutive words are counted. We will be experimenting with different n-grams for correct representation of given data.

TF-IDF is a technique to quantify a word in documents and find out how relevant they are in the given context. For example: In a large text corpus, some words might show up very frequently but only carry less meaningful information. Moreover, these words might overshadow the contribution of less frequent words that may carry highly meaningful information because of the frequency.

	location	tweet	covid	length	tweet_lemma	tweet_stemm	tweet_lemma&stemm
6122	Lincoln, Nebraska	Even without horns.	0	19	even without horn .	even without horn .	even without horn .
6049	Lincoln, NE	Bout to leave work and cuddle with my pillows	0	45	bout to leave work and cuddle with my pillow	bout to leav work and cuddl with my pillow	bout to leav work and cuddl with my pillow
5525	Lincoln, NE	Tuned in	0	8	tuned in	tune in	tune in
4970	Lincoln	It's possible that came up with this idea on s...	0	100	it 's possible that came up with this idea on ...	it 's possibl that came up with thi idea on st...	it 's possibl that came up with thi idea on st...
374	Lincoln	You must book a session via the Lincoln SU Ten...	1	111	you must book a session via the lincoln su ten...	you must book a session via the lincoln su ten...	you must book a session via the lincoln su ten...

Fig. 2. Dataset after implementing text normalisation technique

Prediction based word embedding Prediction based embedding is a powerful way to compute the word vectors that actually considers the contextual nature of the words in the text while creating numerical representation. There are mainly two methods of creating word vectors in this category.

- word2vec
- doc2vec

A **Word2Vec** model is a simple ANN with a single hidden layer. It is helpful to use Word2Vec model if our objective is to have words with similar context occupy close spatial position or similar vector representation. It converts words into vectors by encoding their contextual information (relation of a word with other words in the Text corpus). So, it is very useful for semantic understanding of the words [3]. A Word2Vec model implementation involves 2 tasks that are pre-process Data, and Create word embeddings and training the Word2vec model.

The **Doc2vec** model is created by slightly augmenting the Word2vec model.It is a small extension to the Continuous Bag-of-Words (CBOW) Word2vec model. Instead of using just words to predict the next word, it also adds another feature vector, which is document-unique.

D. Visualization of Text Data

There are many techniques to visualize words.

- Word Cloud
- Dimensionality Reduction (2D)

Word Cloud is a technique to visualize words in Text Normalization Section. Word Cloud displays words in varying size based on their frequency in the Text Corpus. But, it doesn't show any contextual information where as dimensionality reduction technique is lowering the data to only important dimensions for proper visualization of the data. [4] There are 2 technique of dimensionality reduction: PCA and t-SNE

PCA stands for Principal component analysis. This technique of dimensionality reduction is very useful in case of **linear data**. **T-SNE** stands for T-Distributed Stochastic Neighbor Embedding. It is a Manifold Learning technique of dimensionality reduction and is very useful in case of **non-linear data**. The t-SNE tries to preserve the distances in a neighborhood. It differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance.T-SNE would be extremely slow because of the large size of vocabulary. That's why first we apply PCA to reduce the number of dimensions, then apply t-SNE [3].

V. Evaluation Metric

We evaluate our models based on 4 important categories. But, we decide the efficiency of a model based on the accuracy.

- Accuracy
- Precision
- Recall
- F1 score

For our machine learning models, we decided it was essential to predict COVID tweets as accurately as we can and we don't care if we missed one or two tweets from the given dataset which is why we try to maximize the recall for given model.

VI. Methods

There are many models that can be used for Text Classification and among them we will be using 3 machine learning models and comparing their results.

- Naive Bayes Classifier
- Linear Support Vector Machine
- Logistic Regression

A. Naive Bayes Classifier

Naive Bayes is a popular machine learning technique for text classification because it performs well despite its simplicity. It has been successfully used for many purposes, but it works particularly well with natural language processing (NLP) problems. Naive Bayes is a family of probabilistic algorithms that take advantage of probability theory and Bayes' Theorem to predict the tag of a text. Naive Bayes comes in different versions, depending on how text documents are represented [5]. We chose to use multinomial Naive Bayes classifier because it has performed fast, reliable, and accurate in a number of Natural Language Processing applications. Moreover, multinomial Naive Bayes is suitable for classification with discrete features.

B. Linear support Vector Classifier

Support vector machines is an algorithm that determines the best decision boundary between vectors that belong to a given group (or category) and vectors that do not belong to it. It can be applied to any vectors which encode any data. This means that in order to leverage the power of support

vector machine text classification, texts have to be transformed into vectors [6]. We chose to use the Linear Support Vector Classifier because we observed that our data was linear. We came to this conclusion after applying PCA to our data and visualizing it.

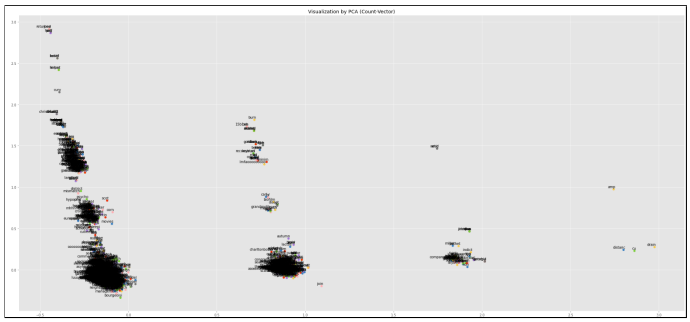

Fig. 3. Visualization by PCA (Count-Vector)

C. Logistic Regression Classifier

Logistic regression is a classification algorithm used to solve binary classification problems. The logistic regression classifier uses the weighted combination of the input features and passes them through a sigmoid function. Sigmoid function transforms any real number input to a number between 0 and 1. We chose to use Logistic Regression because it is a linear classifier with a decision boundary. Since we discovered our data is linear, this discriminative model is anticipated to provide diversity in our study. [7]

D. Hyperparameter tuning

When the number of features is higher than the number of data points, the logistic regression model tends to be underdetermined. To fix this problem, we need to introduce additional constraints that are known as hyperparameters. For Naive Bayes, we didn't have to tune in any hyper-parameters except the smoothing curve or alpha i.e. 0.01, to resolve the over-fitting issue. By using GridSearch, for Linear SVM, we were able to find the optimal 'C' i.e. 1 that controls the penalty margin. In the case of logistic regressions, using gridsearch, we found the optimal 'C' to be 10, 'solver' to be 'sag, and 'tol' to be 0.001.

VII. Results

For text normalization, stemming and lemmatization techniques performed better. However, precisely, the Stemming technique worked best for Multinomial Naive Bayes Classifier while lemmatization technique worked the best for Logistic Regression and Linear support vector classifier. Similarly, for text preprocessing bigram model from the bag of words technique was best performed for all machine learning models we used.

	Feature	Accuracy	Precision	Recall	F1 Score
0	uni_X_train_lemma	0.898469	0.849840	0.771014	0.808511
1	uni_X_train_stemm	0.721998	0.000000	0.000000	0.000000
2	uni_X_train_lemma_stemm	0.923449	0.925170	0.788406	0.851330
3	bi_X_train_lemma	0.930701	0.964158	0.779710	0.862179
4	bi_X_train_stemm	0.926672	0.956835	0.771014	0.853933
5	bi_X_train_lemma_stemm	0.923449	0.925170	0.788406	0.851330
6	tfidf_X_train_lemma	0.917002	0.951493	0.739130	0.831974
7	tfidf_X_train_stemm	0.917002	0.944853	0.744928	0.833063
8	tfidf_X_train_lemma_stemm	0.917808	0.941818	0.750725	0.835484
9	w2v_X_train_lemma	0.850121	0.800000	0.614493	0.695082
10	w2v_X_train_stemm	0.853344	0.822134	0.602899	0.695652
11	w2v_X_train_lemma_stemm	0.861402	0.850202	0.608696	0.709459
12	d2v_X_train_lemma	0.841257	0.774074	0.605797	0.679675
13	d2v_X_train_stemm	0.850927	0.807692	0.608696	0.694215
14	d2v_X_train_lemma_stemm	0.850927	0.803030	0.614493	0.696223

Fig. 4. Summary of Multinomial Naive Bayes Classifier

	Feature	Accuracy	Precision	Recall	F1 Score
0	uni_X_train_lemma	0.914585	0.885417	0.777439	0.827922
1	uni_X_train_stemm	0.914585	0.870000	0.795732	0.831210
2	uni_X_train_lemma_stemm	0.916197	0.878378	0.792683	0.833333
3	bi_X_train_lemma	0.923449	0.936330	0.762195	0.840336
4	bi_X_train_stemm	0.925866	0.931159	0.783537	0.850993
5	bi_X_train_lemma_stemm	0.916197	0.878378	0.792683	0.833333
6	tfidf_X_train_lemma	0.913779	0.951020	0.908537	0.582600
7	tfidf_X_train_stemm	0.908944	0.928287	0.704268	0.799308
8	tfidf_X_train_lemma_stemm	0.906527	0.924000	0.911585	0.813264
9	w2v_X_train_lemma	0.843674	0.712025	0.685976	0.698758
10	w2v_X_train_stemm	0.844480	0.699681	0.667683	0.683307
11	w2v_X_train_lemma_stemm	0.595488	0.829268	0.518293	0.637899
12	d2v_X_train_lemma	0.796938	0.743590	0.353659	0.479339
13	d2v_X_train_stemm	0.769541	0.564417	0.560976	0.562691
14	d2v_X_train_lemma_stemm	0.732474	0.496241	0.804878	0.613953

Fig. 5. Summary of Linear Support Vector Classifier

A. Multinomial Naive Bias, Linear SVM, Logistic Regression : A comparison

The details of the result is depicted as below:

As can be seen from Fig. 4, 5, 6 that the performance of stemmized or lemmatized bigram data is somewhat consistent in all the machine learning model. This implies that choosing the best text normalization and prepossessing technique is a crucial part of a text classification problem. The optimal results for all three models are as follows.

VIII. Conclusion and Future Work

Since we used the optimal hyperparameter, we got a really good result for all three machine learning models. As mentioned in the evaluation metrics, we are also concerned with the Recall score, and Multinomial Naive Bayes gave us a slightly better recall score than the others. Thus, we

	Feature	Accuracy	Precision	Recall	F1 Score
0	uni_X_train_lemma	0.898469	0.849840	0.771014	0.808511
1	uni_X_train_stemm	0.721998	0.000000	0.000000	0.000000
2	uni_X_train_lemma_stemm	0.923449	0.925170	0.788406	0.851330
3	bi_X_train_lemma	0.930701	0.964158	0.779710	0.862179
4	bi_X_train_stemm	0.926672	0.956835	0.771014	0.853933
5	bi_X_train_lemma_stemm	0.923449	0.925170	0.788406	0.851330
6	tfidf_X_train_lemma	0.917002	0.951493	0.739130	0.831974
7	tfidf_X_train_stemm	0.917002	0.944853	0.744928	0.833063
8	tfidf_X_train_lemma_stemm	0.917808	0.941818	0.750725	0.835484
9	w2v_X_train_lemma	0.850121	0.800000	0.614493	0.695082
10	w2v_X_train_stemm	0.853344	0.822134	0.602899	0.695652
11	w2v_X_train_lemma_stemm	0.861402	0.850202	0.608696	0.709459
12	d2v_X_train_lemma	0.841257	0.774074	0.605797	0.679675
13	d2v_X_train_stemm	0.850927	0.807692	0.608696	0.694215
14	d2v_X_train_lemma_stemm	0.850927	0.803030	0.614493	0.696223

Fig. 6. Summary of logistic Regression

TABLE I
OPTIMAL RESULTS

	Accuracy	**Precision**	**Recall**	**F1-score**
Multinomial Naive Bayes	0.93	0.964	0.779	0.863
Linear Support Vector classifier	0.924	0.931	0.784	0.83
Logistic Regression	0.93	0.964	0.779	0.864

concluded that Multinomial Naive Bayes is a better model for COVID tweet classification.

A. Future Work

We have limited our research to the tweets of Lincoln, Nebraska. So in the future, the spatial coverage of tweets can be extended so that our model can be more generalized. Moreover, we can increase the temporal coverage and collect more tweets specific to COVID outburst. We can further rationalize our findings using sentiment analysis or pattern analysis.

REFERENCES

[1] E. Koeze and N. Popper. Virus change the way we internet). [Online]. Available: https://www.nytimes.com/interactive/2020/04/07/technology/coronavirus-internet-use.html

[2] R. Lamsal. Coronavirus (covid-19) tweets dataset). [Online]. Available: https://ieee-dataport.org/open-access/coronavirus-covid-19-tweets-dataset

[3] M. R. Hasan. Feature extraction for text analytics. [Online]. Available: https://github.com/rhasanbd/Text-Analytics-Beginners-Toolbox/blob/master/TextAnalytics-I-FeatureExtraction.ipynb/

[4] ——. Visualization of text data. [Online]. Available: https://github.com/rhasanbd/Text-Analytics-Beginners-Toolbox/blob/master/TextAnalytics-II-Visualization.ipynb/

[5] K.-M. Schneider. On word frequency information and negative evidence in naive bayes text classification). [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-30228-5_42

[6] B. P. Colas F., "Comparison of svm and some older classification algorithms in text classification tasks. in: Bramer m. (eds) artificial intelligence in theory and practice," IFIP International. Federation for Information Processing, vol 217. Springer, Boston, MA., 2006. [Online]. Available: https://doi.org/10.1007/978-0-387-34747-9_18

[7] V. M. Tomas PRANCKEVIČIUS, "Comparison of naïve bayes, random forest,decision tree, support vector machines, and logistic regression classifiers for text reviews classification." Baltic J. Modern Computing, (2017), N,, 2017.

IX. APPENDIX

We included the detailed results at the end of the report.

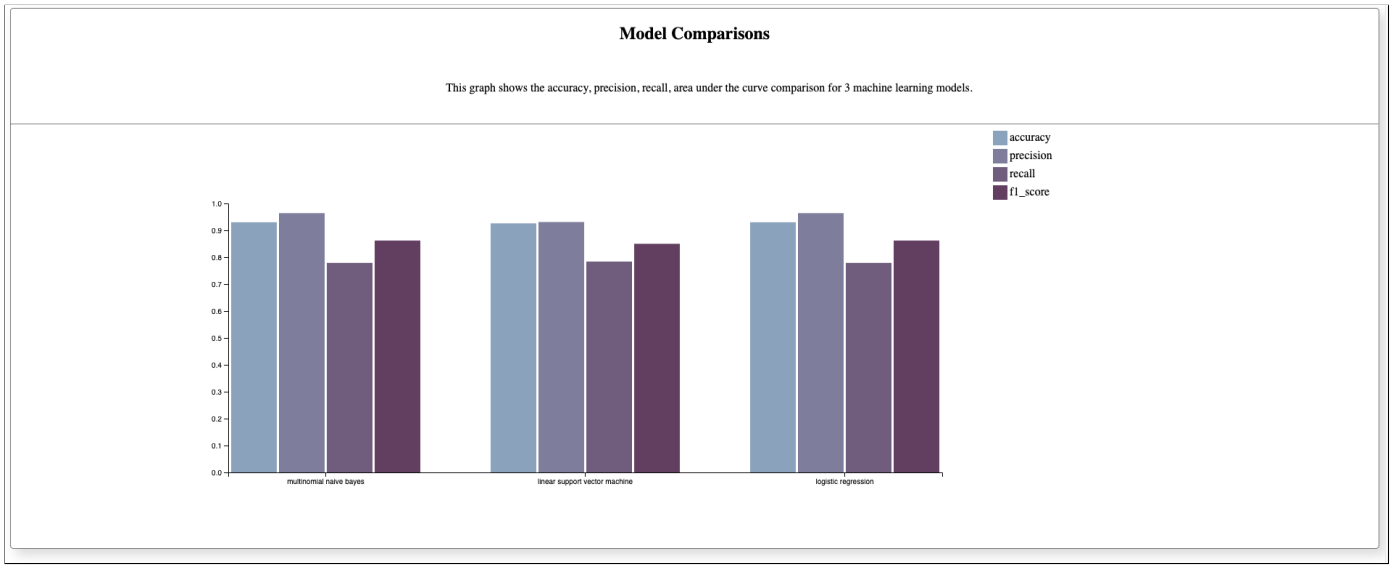

Fig. 7. Model Results Comparison

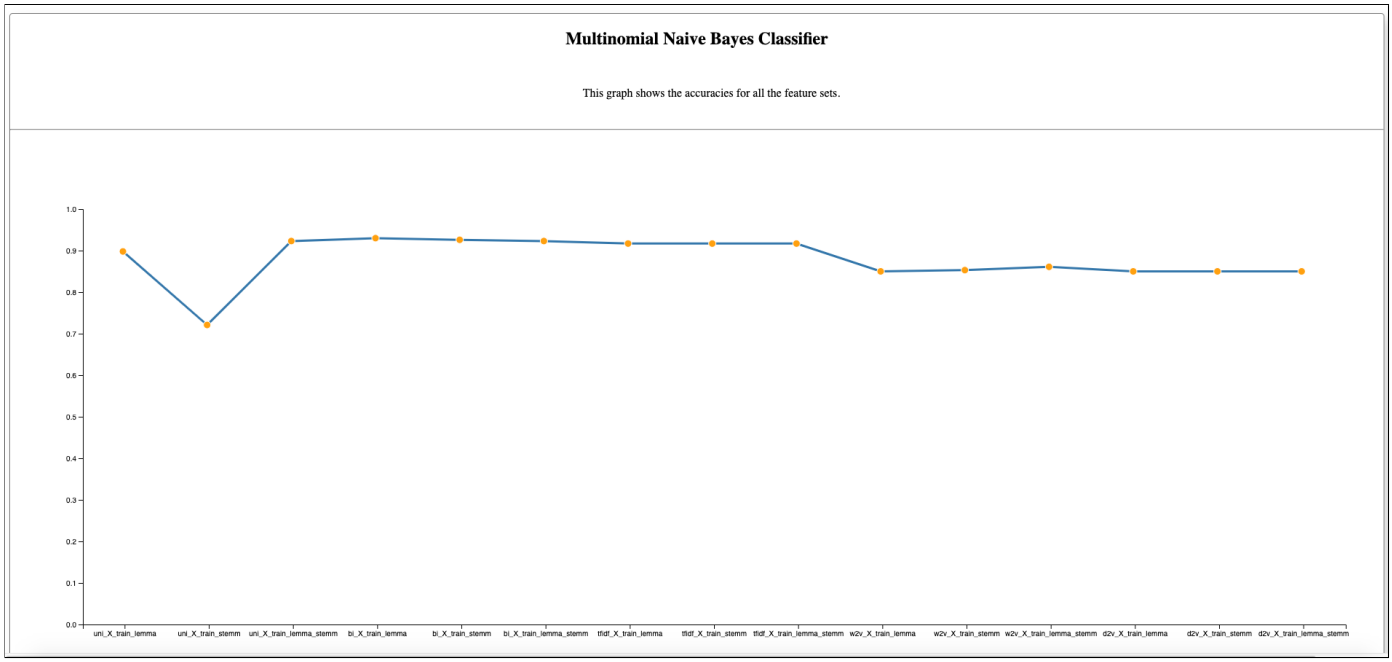

Fig. 8. Naive Bayers Classifier Results

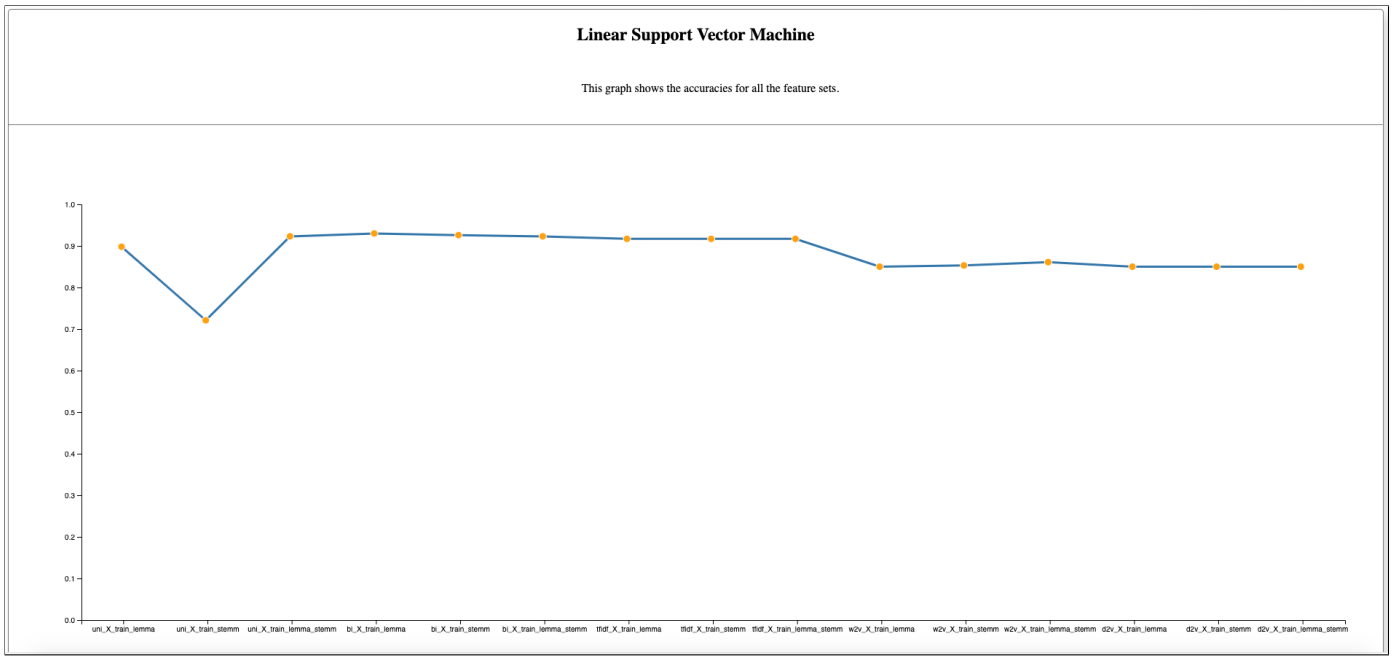

Fig. 9. Linear Support Vector Machine Results

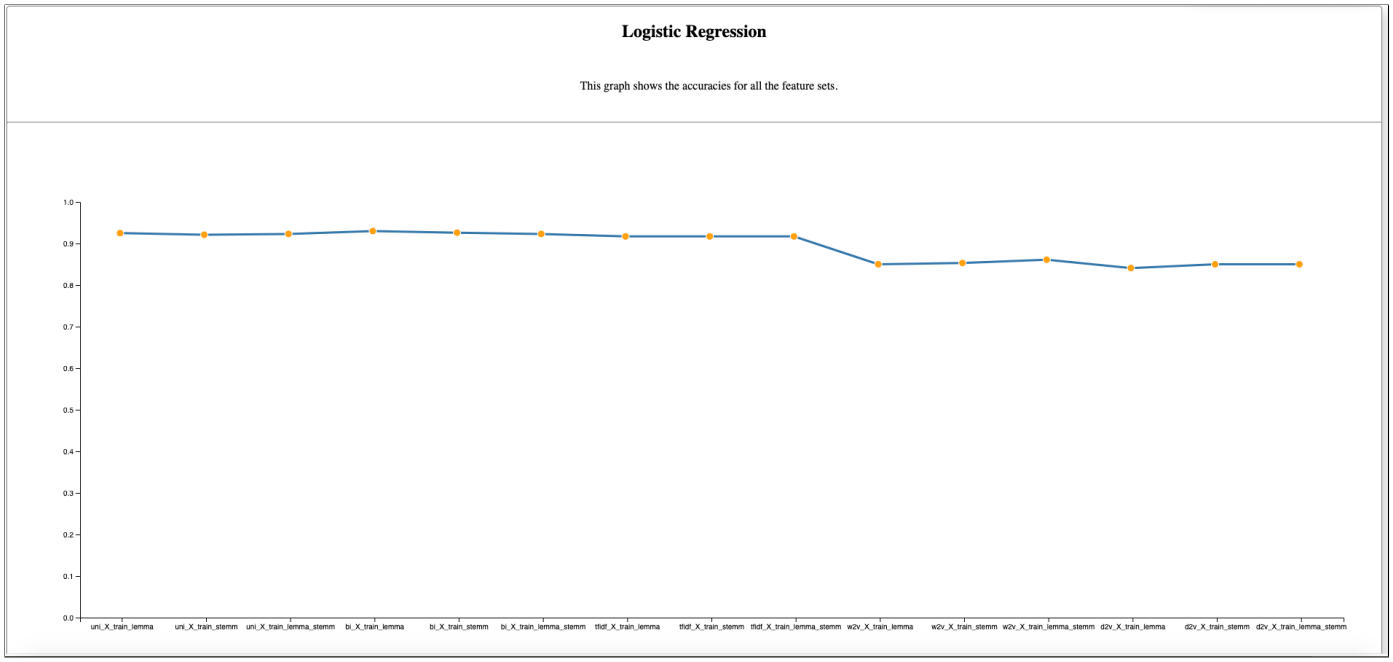

Fig. 10. Logistic Regression Results