Spring 5-9-2022

A Machine Learning and Deep Learning Framework for Binary, Ternary, and Multiclass Emotion Classification of Covid-19 Vaccine-Related Tweets

Aditya Dubey
aditya.dubey@uconn.edu

A Machine Learning and Deep Learning Framework for Binary, Ternary, and Multiclass Emotion Classification of Covid-19 Vaccine-Related Tweets

Aditya Dubey
Computer Science & Eng.
Univ. of Connecticut, Storrs, CT 06269

I. Introduction & Motivation

Language and text are some of the most important instruments for conveying information and emotion. Identifying emotion and its subsets is a topic that has been pursued by psychologists, medical professionals, and behavioral scientists for decades. As social media has taken over the world over the last decade, interpreting the emotions behind online activities has become critical to understanding people's reactions and sentiments to events, theories, and policy. The onset of the coronavirus pandemic changed the way humans live and operate in society for over a year, with a vaccine being the main solution to stopping rising cases across the globe. Although vaccines have been developed over many years, the combined efforts of biotech giants such as Pfizer, Bio N' Tech, and Moderna, resulted in the fastest developed vaccine ever. In January of 2021, Biden announced getting 100M vaccinations within the first 100 days of his presidency to fasten pandemic recovery and ultimately curb infection rates. The resulting conversation across America was polarizing at best, revealing a large opinionated community of anti-vaxxers, many of whom continue to spread disinformation and fear about the vaccine. This was especially evident across platforms such as Facebook and Twitter, with the latter implementing misinformation warnings to filter false tweets. In this case, understanding the exact sentiment and emotion behind each tweet is key to understanding overall vaccine sentiment across media platforms. This paper mines public emotion towards the Covid-19 vaccine based on Twitter data collected over the past 6-12 months. Given this data and the goal of the paper, we aim to answer the following questions:

1. Can a framework be developed for machine learning and deep learning multiclass classification models to accurately infer one of four emotion groups represented by a vaccine-related tweet? The four emotion groups we are interested in are:
    a. Apprehension/Anticipation
    b. Sadness/Anger/Frustration
    c. Joy/Humor/Sarcasm
    d. Gratitude/Relief
2. Is there a significant binary distinction that can be made between tweets that express "negative" emotions (Apprehension, Anticipation, Sadness, Anger, and Frustration) and "positive" emotions (Joy, Humor, Sarcasm, Gratitude, and Relief)? Can a ternary distinction prove to be the better classification model?

Through these questions, the primary goal is not only to determine the overall acceptance and sentiment of the vaccines by the public but also to understand the steps public health officials can take to further educate hesitant and/or fearful citizens while also incentivizing it.

II. Literature Review

There is a plethora of research on the vaccine discussion in social media and the conversations surrounding it. Most current works that focus on emotion mining do a direct comparison between anti or pro-vax sentiment. Multi Classification of emotion is less prevalent for the specific use case of vaccines and the coronavirus pandemic. Works such as [2] by D'Andrea, focus primarily on unrelated vaccinations such as the flu, meningitis, and other childhood vaccines. Others such as [1] take a stance on anti vs pro covid vaccination directly. They focus on general takes related to fake news and opinions as well, both of which are sentiment-related but not nearly as closely related to our topic.

One work that aligned close to our paper was [3] by Chang, Ferreira, & Yang. The key distinction between our work and theirs is that they incorporate topic data such as news, government policy, and

COVID-19 events to determine if emotion falls within a specific category. Furthermore, their data is from middle-late 2020 (published in 2021), while our data is more recent and in tune with the current vaccine discussion. Therefore, we expect our analysis to offer further insight into emotion overlap in tweets, what types of tweets are apprehensive, sadness, joyful, or gratitude, and general methods that help models better discern between more nuanced types such as sarcasm and apprehension.

The paper that best aligns with our classification process however is [5] by Mondal, Gokhale. 6 emotions (joy, sadness, anger, fear, love, and surprise) are classified by using traditional machine learning algorithms such as SVMs, Neural Networks, and MLPs on TF-IDF & TextBlob features. We initially expect our analysis and results to reflect similar or slightly lower accuracies since we will be discerning between emotion groups such as Joy/Humor/Sarcasm and Apprehension/Anticipation. Additionally, our dataset is nearly three times smaller, so further rebalancing and feature engineering will be needed to approach similar performance.

III. Data Collection & Labeling

Data for this paper was collected over 1 week between March 4th, 2021, and March 12th, 2021. 3734 tweets were collected through the Twitter API using the key phrase "side effects". Each tweet was labeled based on 4 emotions:

- Apprehension/Anticipation **[N]**
    - *'"Is my COVID-19 vaccine working if I don't have side effects?"*
- Joy/Humor/Sarcasm **[J]**
    - *"Touring the town with Dad imagining all the places we'll go when it's safe. And FYI, woke up feeling great. Side effects from the vaccine were short-lived."*
- Gratitude/Relief **[G]**
    - *"I just got my ONE AND DONE Johnson &Johnson vaccine and 6 hrs later, I can report no side effects. Not even soreness at the injection site. Thank you President Biden!"*
- Sadness/Anger/Frustration **[S]**
    - *"Yeah forget what I said earlier about me not experiencing any side effects from the vaccine. I feel sluggish and have extreme joint pains. Send help"*

These 4 emotions are used for 2 primary reasons and were inspired by Plutchik's wheel of emotions. One, individuals who did not necessarily fear the vaccine but were still hesitant needed to be classified, thus an apprehension emotion was used. Two, joy versus gratitude were two different emotions that help us understand individual reactions post-vaccination, while most research thus far has only aimed to see pre-vaccination emotions. Therefore, based on these labels, the entire dataset is labeled and included in the final corpus.

IV. Preliminary Analysis and Preprocessing

In this section, we will discuss the three sections of the preliminary analysis performed: data preprocessing, word clouds, and the statistical analysis of tweets.

The first step in our data preprocessing was tokenization. We utilized the RegexpToekeizer from the nltk.tokenize library for this step and each tweet (originally a string) was broken up into a series of tokens (words) and each token was transformed to lowercase. Then the tokens were normalized, where stop words and any tokens containing Unicode characters were removed from the data. The stop words were imported from the nltk.corpus library. The stop words were adjusted after the creation and analysis of the word clouds, which are discussed further down below. After normalization, we performed stemming and lemmatization (using PortStemmer and WordNetLemmatizer from nltk.stem). Stemming is the process of reducing inflected words to their word stem or root form and lemmatization refers to the process of grouping together the inflected forms of a word so they can be analyzed as a single item or lemma. Through all these preprocessing steps, we are able to clean and reduce a tweet to a list of tokenized keywords that are significant for NLP. We ran these preprocessing steps on each of the four labeled sub-datasets. This concludes the data preprocessing steps we took. After this preprocessing, we have a dataset of 1792 tweets.

After the data preprocessing, we began the construction of the word clouds for our four labels: **[N]**, **[J]**, **[G]**, and **[S]**. Word clouds are visual representations of words in a data text that give greater prominence to words that appear more frequently in the text. We imported the word cloud library and designed a function that takes in a list of keywords obtained from the data preprocessing steps as input and outputs a word cloud for that respective collection. The input is converted into a large string and that test is then passed onto the imported WordCloud() class to then generate a word cloud. The word clouds were visualized using matplotlib. Each word cloud has a max_word limit set to 1000. The word clouds are shown in Figures 1,2,3, and 4 for the labels **[N]**, **[J]**, **[G]**, **[S]**, respectively. As explained before, the size of the word/token in the word cloud is proportional to the frequency of that word/token in the data. The word clouds shown in the below figures have been iterated upon. The original word clouds generated had the following words as very prominent: ['side', 'vaccine', 'effect', 'effects', 'second', 'covid', 'dose', 'iâ', 'amp', 'co', 'first']. These words were added to our stopwords since they were apparent in all four clouds and thus would not provide us with any useful information in our classification steps. This revision of the stopwords showcases the importance of the word cloud visualization since it helped us reduce redundant information among the four labeled datasets, allowing for the differences among the four to be more recognizable. Analyzing these word clouds also revealed more information about the labels. We see words such as hope, know, nervous, and hear to be prominent in the **[N]** word cloud. This makes sense since the **[N]** label represents apprehension & anticipation and those selected words often help project that tone. Similar to this observation, we see the terms sore, arm, Pfizer, and Moderna being prominent in the **[G]** label. The word clouds for **[J]** and **[S]** are similar in their prominent words.

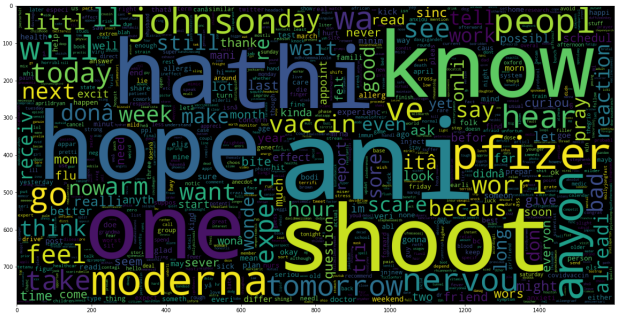


*Figure 1.1: **[N]** word cloud*

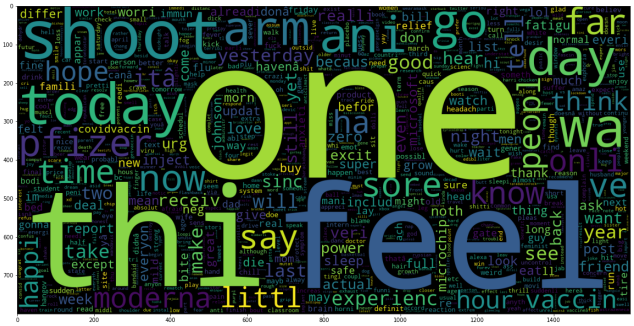


*Figure 1.2: **[J]** word cloud*

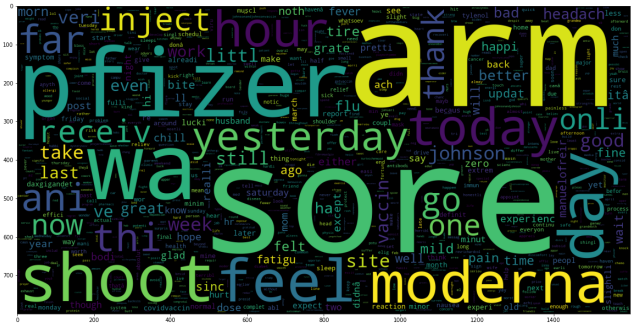


*Figure 1.3: **[G]** word cloud*

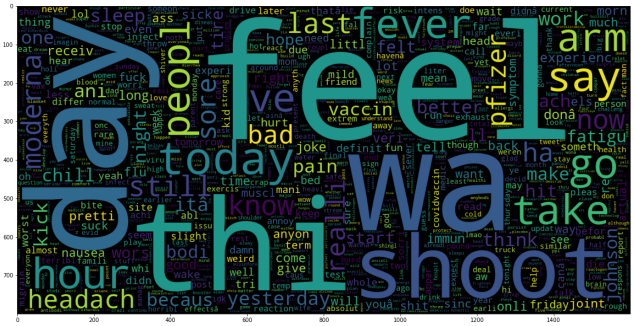


*Figure 1.4: **[S]** word cloud*

Parameter	N	J	G	S
Percentage Breakdown	19.5%	24.8%	20.0%	35.6%
Avg. Display Text Width	154	159	164	157
Avg. Follower Count	1571	3353	1813	2315

Avg. Friends Count	1375	1230	1590	1620
Avg. List Count	21	54	24	27
Avg. Status Count	28114	21862	27154	26163
Avg. Favorited Tweets Count	3	167	8	3
Avg. Retweet Count	0	16	0	0
Avg. Favorite Count	35838	31945	37733	32174

Table 1.5: Statistical Analysis of Words

IV. Social Features

Twitter serves as a platform that allows for both a great deal of data collection and analysis of said data in relation to individual users and their outreach. The impact of tweets, retweets, and virality of user comments are all important to understanding the overall function of social media in relation to emotion. The following section dives into the various parameter data:

- **Avg. Display Text Width:** It displays the number of characters in each tweet of the corpus. We can see that there is no significant difference across emotions. Therefore, tweets from one group are not necessarily longer than others.
- **Avg. Follower Count:** This represents the number of followers of the user who made the tweet. The tweets that are labeled as the joy emotion are linked to accounts that tend to have much higher follower counts than the other emotion groups.
- **Avg. Friends Count:** This shows the number of friends the account that made the tweet has. Across emotions, there seems to be no significant difference in friends count.
- **Avg. List Count:** This represents the average number of lists the account of the tweet has. The average number of lists people who made joy tweets are in is about two times as many as users who made tweets with every other emotion.
- **Avg. Status Count:** This value represents the cumulative number of tweets made by users for each emotion. The total count is approximately the same across all emotions but the joy emotion is lower than all others by approx. 4200 tweets minimum.
- **Avg. Favorited Tweets Count:** This parameter measures the number of tweets liked by the users making a tweet related to a specific emotion. There is an extremely high deviation of nearly 50x more favorited tweets for joyful tweets relative to Apprehensive and sad tweets.
- **Avg. Retweet Count:** This number represents the number of times a tweet is retweeted. All emotions had a value of 0 except for the joy emotion.
- **Avg. Favorite Count:** This is the parameter that checks how many users liked that specific tweet.

The statistical analysis of the words based on these social features is located in Table 1.

V. Data Preparation

This is the next step we take in our journey to emotion classification. We have preprocessed and analyzed the tweets, and now we have to prepare the data so that we can properly extract features from them and feed them into our classifiers. Data preparation first consists of remapping the class labels to numerical values and addressing the class imbalance.

A. Class Mappings

First, we created a new pandas data frame that combined all of the four preprocessed datasets for the labels **[N]**, **[J]**, **[G]**, and **[S]**. This data frame has a text column and a type column. The text column is our dependent variable (X or measured variable) and the type column is our independent variable (y or variable we are trying to predict). The text column contains all the preprocessed tweets and the type column contains the numeric label type for each of those corresponding tweets with the following mapping:

Mappings			
0 = N	1 = S	2 = J	3 = G

Figure 2.1 - Emotion Label Mappings

B. Data Splitting and Stratifying

We performed the following data splits:

- For multiclass models: 90% of our data training and 10% of our data for testing.
- For binary models: 85% of our data training and 15% of our data for testing.

These splits gave us the best performance for all our models. We performed these splits with stratified sampling. This way the percentage of all the classes in the training and testing data is the same. For some models, we toggled the stratified sampling as it helped with performance.

C. Class Balancing - Undersampling

Shown below is the emotion distribution of the data after we have preprocessed it:

```
N: 314, S: 323, J: 400, G: 574
```

Figure 3.1 - Multiclass Training Data Distribution

The first class balancing technique that we employed was random undersampling, without replacement, of our training data (for our multiclass models). There are a total of 1240 tweets shown here and as we can see, there are a large number of **[G]** labeled tweets, and the classes are not balanced due to this. To address this, we employed a random undersampling technique. Undersampling is a technique to balance uneven datasets by keeping all of the data in the minority class **[N]** and **[J]** and decreasing the size of the majority classes **[G]** and **[S]**. We resampled the data with different ratios and in proportion to the label's spread of the data. After this we obtained a uniform dataset with 350 tweets being sampled from each emotion type:

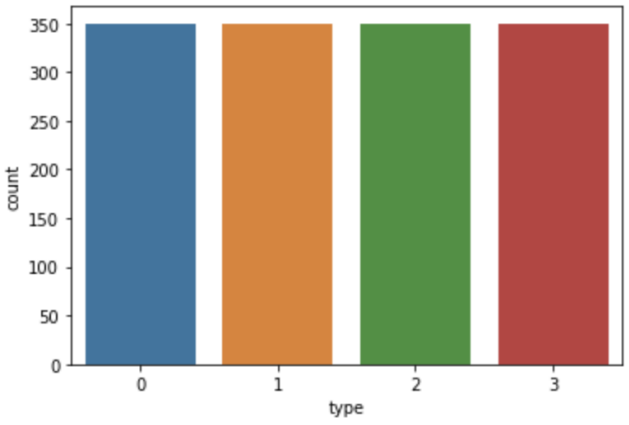


Figure 3.2 - Balanced Data Distribution - Undersampling

As stated before, this was only performed on the training data set. We applied this class balancing only for our multiclass models since it provided us with far better results than our other class balancing technique.

C. Class Balancing - Oversampling

Another class balancing technique that we employed was random oversampling, with replacement, of our training data (for our binary models). Random oversampling involves randomly selecting examples from the minority class, with replacements, and adding them to the training dataset. In our case, we performed random oversampling on our **[N]**, **[S]**, and **[J]** labels to increase their size to match the size of the **[G]** label of 945 samples. We obtained the following distribution for our training dataset:

```
N: 298, S: 305, J: 378, G: 542
```

Figure 3.3 - Binary Training Data Distribution

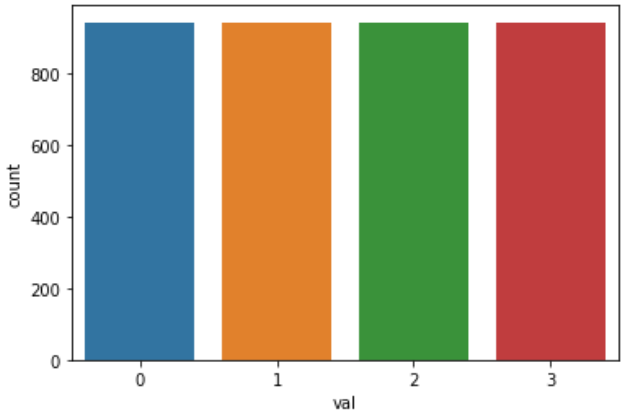


Figure 3.4 - Balanced Data Distribution - Oversampling

This random oversampling technique was not used in conjunction with the random undersampling technique, described before in section C. These class balancing techniques are to be used independently of each other. Random oversampling of the training data did not show any major benefits to the performance of models in multiclass classification, but it did boost performance for the binary classification and so we only used it for class balancing in our binary classification models.

VI. Feature Extraction

Before diving into the classification process, we have to begin with feature extraction. Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. The main goal of this is to reduce the number of features while still being able to capture the same information since having too many dimensions leads to more complex and less efficient models which may be overfitted to the training dataset.

A. TF-IDF

We utilized TF-IDF as one method for feature extraction. Term Frequency - Inverse Document Frequency or TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. We used the python machine learning library sklearn as the framework for our feature extraction and classifiers. This is how we performed our feature extraction with TF-IDF:

1. Using sklearn's train_test_split, we split our data into X_train, X_test, y_train, and y_test.
2. Next, we used the CountVectorizer from sklearn to vectorize our X variable (the text data). This converts a collection of text documents into a matrix of vectors, each of which stores the frequency (count) of each of the words that occur in the entire text. We then used the TfidfTransformer to transform the count matrix outputted by the count vectorizer into a normalized TF-IDF representation matrix.
3. Now our data split has become the following:
    a. X_train_tfidf, X_test_tfidf, y_train, y_test.

B. Unigram

Unigram feature extraction, in the context of this project, involved splitting our sentences into one-word sequences and using these converted sentence lists as our features. Through SkLearn's CountVectorizer, we were able to specify an n-gram size to extract a training and testing dataset of unigram features. We performed the following steps:

4. Using sklearn's train_test_split, we split our data into X_train, X_test, y_train, and y_test.
5. Using CountVectorizer, we transformed this vectorized data into a Unigram representation by setting the n-gram in the vectorizer to be 1. An ngram of 2 or 3 would represent bigram and trigram models.
6. Now our data split has become the following:
    a. X_train_tfidf, X_test_tfidf, y_train, y_test.

C. POS Tagging

Part of Speech (POS) tagging is a process to mark up the words in text format for a particular part of a speech based on its definition and context. This feature was used to introduce context by assigning word items such as JJ for an *adjective* or VB for a *verb*. The ideology behind this feature extraction method is that tweets with specific emotions follow specific sentence structures. A key use case of this is sarcastic tweets, which are theorized to have patterns such as idiosyncratic phrases of the type noun, adposition, and noun. To account for the sarcasm trait, this feature is expected to improve the overall classical accuracy of the Joy/Sarcasm/Humor label. For this feature extraction method, we exclusively chose to balance the classes as that was the best performance.

D. GloVe Word Embeddings

GloVe stands for Global Vectors for Word Representation and serves as an unsupervised learning algorithm that generates word embeddings based on the relationships between various words. These relationships are quantified by GloVe's predefined embedding rules file which assigns these numerical embeddings to each sentence according to their word relation contents. For the context of our project, we took the following steps:

1. Installed and unzipped the predefined GloVe embeddings dataset.
2. Created a GloVe loader function to parse each sentence in our preprocessed dataframe.
3. Apply the embeddings vectors associated with words found in a corresponding sentence.
4. Reshape the data to its original state.

5. Randomize the arrangement of embedding vectors and split them into a train and test dataset.

E. TextBlob

Our final feature extraction method implemented across the ML classifiers used for this research was TextBlob. This method returns the overall sentiment and polarity of a sentence based on its contents, allowing polarities greater than or less than 0.5 to be translated to strong positive or negative sentiment. We aim for this feature to help better discern between sarcastic and sad tweets. This polarity score was added to our data as metadata and was fed in with the text to our classifiers. This was also toggled and tested as we found out that incorporating this with other features did not provide us with any performance boost for our classifiers.

F. Auxiliary and Social Features

To provide our classification models with more information about our tweets, we included auxiliary and social features from the tweet metadata as input metrics for our models. We toggled these features to be included in and not as we tested our model performances so we could get the best performance per model. The social features that we used are the following (*all of these were defined in section IV. Preliminary Analysis and Preprocessing*):

- Favorite count
- Follower count
- Favorited tweets count

In addition to these features, we also added punctuation and text-based emoticon counts. Expressions of emotion use certain characters for emphasis and express more sentiment. Certain punctuation characters, specifically '!' and '?' carry an emotional value that can be used to help predict the sentiment of the tweet. We also chose to add n '@' and '#' to our punctuation count feature, as they also provide additional information as to if the tweet is part of a trend with the hashtag or if it is tagging someone. Finally, we incorporated the counts of text-based emoticons as input feature data. Unlike emojis, which are based on a Unicode value, text-based emoticons are 'smiley' or 'sad' faces drawn with simple characters and are easier to process into our models:

- A happy face is :) or ;) or =]
- A sad face is :(or ;(or =[

Shown below are the total number of punctuation and text-based emoticon counts for each label:

	@	?	!	#	:) ;) =]	:(;(=[
N	153	109	44	26	1	1
S	133	42	37	26	0	12
J	73	45	68	52	8	1
G	291	18	142	71	14	1

Figure 4.1 - Punctuation & Text-based Emoticon Counts

Other feature extraction methods that we explored were BERT Word embeddings and Word2Vec vectorization, but both yielded many lower-performing models on the basis of accuracy and the performance metrics listed in our results section.

VII. Machine Learning Classifiers

We implemented several different commonly used supervised models (known for performing well on multi-class classification tasks) through the sklearn package:

- **Logistic Regression:** Logistic regression is a binary classification model that uses a logistic function to model a binary dependent variable. In the case of our multiclass classification, we are using a multinomial logistic regression from the sklearn library.
- **SVC:** Support Vector Machines (SVMs) is a classification model type that estimates a hyper-plane boundary with margins to separate data into various classes. Using a linear kernel through sklearn, these models were trained across the 4 emotion classes.
- **KNN:** This classifier compares the distance between various data points (our vectorized data) to determine which ones associate the most and cluster them accordingly. This model arranged data into 4 clusters, for classification.

- **Multinomial Naive Bayes (Multi-NB):** This classifier is an extension of a Naive Bayes classifier and is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

A. Hyperparameter Optimization

This is the process of determining the right combination of hyperparameters that maximizes the model performance. We can see the model hyperparameters, with their finalized values after tuning, on table 4.1. To ensure optimal hyperparameters were used for all ML classifiers, *GridSearchCV* was implemented to determine these values. It is a SkLearn hyperparameter tuning algorithm that cycles through a range of hyperparameters fitted for each model, such as the kernel, alpha, gamma, and more depending on the classifier type. For our ML classifiers, we optimized between enabling prior fitting, and an alpha range from 1 to 0.00001. As will be discussed in our results, this resulted in ~10% increases in accuracy across the Multinational Naive Bayes and SVM models. Additional tuning methods used to get the ideal hyperparameters were keras-tuner with bayesian optimization and Scikit-learn optimize.

B. Performance Metrics

The following metrics were used to determine the feasibility of each model:

- *Accuracy:* This is the percentage of tweets in the validation that was labeled correctly, using this function:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

- *Precision and Recall (Binary definition)*

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

- *Precision-micro:* This is the sum of all true positives for all classes, divided by all the positive predictions:

$$\frac{(TP_1+TP_2+\ldots+TP_n)}{(TP_1+TP_2+\ldots+TP_n+FP_1+FP_2+\ldots+FP_n)}$$

- *Precision-weighted:* This is the weighted arithmetic mean of all the precision scores of different classes:

$$\frac{w_1*percision_1+w_2*percision_2+\ldots+w_n*percision_n}{n}$$

- *Recall-micro:* This is the sum of all true positives for all classes, divided by all the actual positives:

$$\frac{(TP_1+TP_2+\ldots+TP_n)}{(TP_1+TP_2+\ldots+TP_n+FN_1+FN_2+\ldots+FN_n)}$$

- *Recall-weighted:* This is the weighted arithmetic mean of all the recall scores of different classes:

$$\frac{w_1*recall_1+w_2*recall_2+\ldots+w_n*recall_n}{n}$$

- *Root Mean Square Error:* This is the standard deviation of the residuals. RMSE is a measure of the spread of the residuals and indicates how concentrated the data is around the line of best fit:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

- *F-Score:* This statistic gauges the weightage between precision and recall (can be defined as micro or weighted based on the precision and recall values):

$$F1 = 2x\frac{Precision * Recall}{Precision + Recall}$$

- *Mean Absolute Error:* This is the mean of the absolute values of the individual prediction errors over all instances in the test set:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{x}_i - \boldsymbol{x}|$$

D. Baseline & Binary Classification Test

The theoretical baseline of a 4-class model would be 25%. Due to the skew present in our data, the SkLearn dummy classifier was used to obtain a

baseline of 0.373 or 37.3%. When we toggled on class balancing this baseline became the theoretical 25%.

To preview a contrast between multinomial classification and "binary" classification, we created a new dataset where all tweets labeled as **[J]** or **[G]** (the Joy & Gratitude emotion groups) were relabelled as "0" for positive sentiment, and all tweets labeled as **[N]** and **[S]** (the Apprehension & Sadness emotion groups), were relabelled as "1" for negative sentiment. For binary classification, our baseline was 50%.

VIII. Deep Learning Models

In addition to the several different machine learning classifiers trained, we implemented several deep learning models to determine if this approach resulted in higher predictive capability. The following models were developed with Keras & TensorFlow 2.0:

- **Multi-Layer Perceptron (MLP):** This is a feed-forward Artificial Neural Network with an input, hidden, and several output layers for the classification. An ANN of this type takes in multiple inputs with corresponding weights, processes this data based on the strength of each input, and returns a singular output that corresponds to all possibilities of inputs. We used the ReLu activation function to minimize data loss associated with the gradient loss function.
    - **MIMO -** A MLP that Involves Multiple Input Layers and/or Multiple Output Layers. With this, we can incorporate other metadata features from the tweet data into our X variable for our models to predict emotions.

- **Convolutional Neural Network(CNN):**
  This is a type of neural network that is designed to process generative or descriptive tasks such as visual processing and natural language.

- **Bi-directional LSTM:**
  A sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backward direction. This allows a model to understand the relationships between words **before and after** each other, thus improving theoretical accuracy.

Our first model is the MLP MIMO model, which is structured in figure 4.1. As can be seen in the model architecture, one input layer is used to feed in pre-engineered GLoVe word embeddings based on our tweet data. The second input layer feeds in our numerical data, like our social (favorite counts, tweet length) and our TextBlob polarity score. While an LSTM layer operates on our language data, a simple dense neural network operates on numerical data to combine and generate one final output of weights. The output of these layers is concatenated and passed through several more dense layers for the final output. Our second model is the Bidirectional LSTM model which is structured as shown in figure 4.2.

The next deep learning model is our CNN Text model which is composed of two embedding layers for GloVe word embedding features. This is followed by a series of Convolutional and Dropout layers to decrease overfitting while maintaining higher accuracies. This model type was chosen since CNNs can better correlate weights of various vectorized sentences over MLPs which can overfit quickly.

The final deep learning model is our basic MLP neural network. This neural network (NN) was configured with SkLearn similar to our other ML classifiers and thus we tested it on the same feature extraction methods.

Each of these deep neural networks went through its own tuning process. The Bidirectional LSTM, MIMO, and CNN were run multiple times with various combinations of layer counts, neurons per layer, and validation splits. The basic MLP neural network was able to utilize GridSearchCV, similar to the ML classifiers, to obtain its optimal hidden layer sizes and shapes.

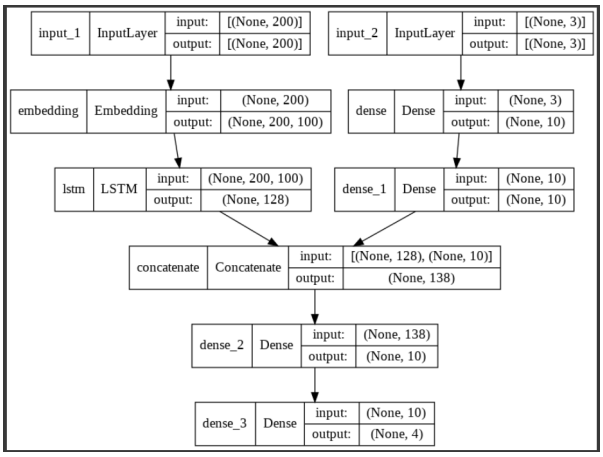
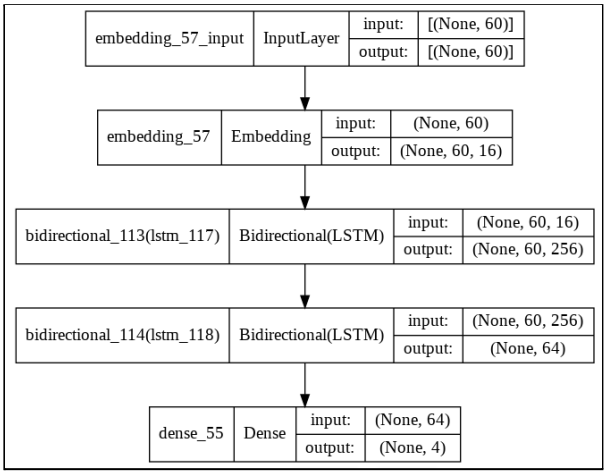
Model	Hyperparameters
Sklearn Models (used the default sklearn parameters with some changes)	
Logistic Regression	C = 1.0
KNN	Default sklearn parameters with n_neighbors = 15
SVM	Default sklearn parameters with C=10 and gamma=0.0001
Multinomial NB	Default sklearn with alpha=1, class_prior=None, fit_prior=False
NN (MLP)	solver='lbfgs', alpha=1e-5, max_iter=500, hidden_layer_sizes=(512,512,512)
Tensorflow Models	
Bidirectional LSTM	
MIMO (MLP)	

Figure 4.1 - Model Hyperparameters

IX. ML and DL Classifier Performance

		MULTICLASS							BINARY					
ML Model	**Metric**	**TF-IDF**	**TF-IDF (Social+ Aux)**	**TF-IDF (Social+ Aux+text blob)**	**Unigram**	**Unigram (Social+ Aux)**	**Unigram (Social+Aux +textblob)**	**POS Tagging**	**TF-IDF**	**TF-IDF (Social+ Aux)**	**TF-IDF (Social+ Aux+text blob)**	**Unigram**	**Unigram (Social+ Aux)**	**Unigram (Social+Aux +textblob)**
Log Reg.	**Accuracy**	**0.52**	**0.61**	**0.62**	**0.51**	**0.58**	**0.58**	**0.61**	**0.69**	**0.75**	**0.77**	**0.71**	**0.78**	**0.79**
	RMSE	1.25	1.07	1.05	1.13	1.02	1.04	1.04	0.58	0.54	0.5	0.6	0.5	0.48
	F-Score micro	0.52	0.61	0.63	0.51	0.58	0.57	0.61	0.69	0.75	0.77	0.71	0.78	0.77
	F-Score weighted	0.5	0.61	0.62	0.44	0.47	0.46	0.61	0.69	0.74	0.76	0.71	0.76	0.76
	MAE	0.74	0.6	0.59	0.69	0.58	0.58	0.59	0.39	0.27	0.23	0.37	0.24	0.23
KNN	**Accuracy**	**0.49**	**0.54**	**0.54**	**0.49**	**0.53**	**0.54**	**0.49**	**0.65**	**0.7**	**0.73**	**0.65**	**0.72**	**0.72**
	RMSE	1.35	1.22	1.22	1.39	1.33	1.3	1.25	0.9	0.61	0.55	0.92	0.6	0.6
	F-Score micro	0.48	0.54	0.54	0.48	0.52	0.52	0.49	0.65	0.7	0.73	0.65	0.72	0.72
	F-Score weighted	0.43	0.51	0.51	0.47	0.49	0.5	0.49	0.64	0.7	0.73	0.65	0.69	0.69
	MAE	0.86	0.75	0.74	0.9	0.83	0.83	0.82	0.6	0.32	0.29	0.55	0.36	0.36
SVM	**Accuracy**	**0.54**	**0.61**	**0.61**	**0.51**	**0.55**	**0.58**	**0.56**	**0.62**	**0.69**	**0.71**	**0.7**	**0.74**	**0.76**
	RMSE	1.26	1.2	1.2	1.24	1.18	1.16	1.12	1.0	0.58	0.54	0.6	0.54	0.52
	F-Score micro	0.54	0.61	0.61	0.5	0.55	0.58	0.55	0.62	0.69	0.71	0.7	0.74	0.76
	F-Score weighted	0.53	0.60	0.60	0.49	0.55	0.57	0.55	0.62	0.68	0.71	0.69	0.73	0.72
	MAE	0.76	0.69	0.67	0.8	0.71	0.69	0.69	0.7	0.4	0.28	0.32	0.3	0.27
Multi NB	**Accuracy**	**0.54**	**0.65**	**0.67**	**0.51**	**0.62**	**0.62**	**0.61**	**0.7**	**0.81**	**0.81**	**0.7**	**0.75**	**0.75**
	RMSE	1.25	1.15	1.11	1.25	0.65	0.65	0.72	0.6	0.42	0.43	0.6	0.52	0.52
	F-Score micro	0.52	0.65	0.67	0.5	0.6	0.61	0.58	0.7	0.81	0.81	0.7	0.75	0.75
	F-Score weighted	0.51	0.65	0.63	0.5	0.6	0.6	0.56	0.7	0.8	0.80	0.7	0.74	0.74
	MAE	0.8	0.55	0.53	0.8	0.65	0.65	0.67	0.4	0.19	0.19	0.4	0.28	0.27
Deep Learning Classifiers														
NN (MLP)	**Accuracy**	**0.55**	**0.59**	**0.63**	**0.53**	**0.6**	**0.61**	**0.55**	**0.72**	**0.74**	**0.74**	**0.71**	**0.75**	**0.75**
	RMSE	1.23	1.1	1.14	1.3	1.1	1.07	1.12	0.54	0.52	0.52	0.61	0.5	0.5
	F-Score micro	0.55	0.59	0.63	0.5	0.59	0.6	0.55	0.72	0.74	0.74	0.71	0.75	0.75
	F-Score weighted	0.54	0.57	0.62	0.44	0.58	0.59	0.53	0.72	0.73	0.73	0.71	0.72	0.72
	MAE	0.86	0.67	0.57	0.99	0.7	0.61	0.69	0.28	0.27	0.27	0.35	0.25	0.25

Figure 5.1 - Classifier Performance Metrics Tbl.

DL Model		GloVe (Social+Aux)	GloVe (Social+Aux)
bidirectional LSTM	**Accuracy**	**0.64**	**0.71**
	RMSE	1.12	0.58
	F-Score micro	0.63	0.7
	F-Score weighted	0.64	0.71
	MAE	0.68	0.33
MIMO (MLP)	**Accuracy**	**0.62**	**0.67**
	RMSE	1.18	0.9
	F-Score micro	0.62	0.67
	F-Score weighted	0.6	0.67
	MAE	0.65	0.61

Figure 5.2 - DL Model Accuracy Table

XI. Classifier Confusion Matrices

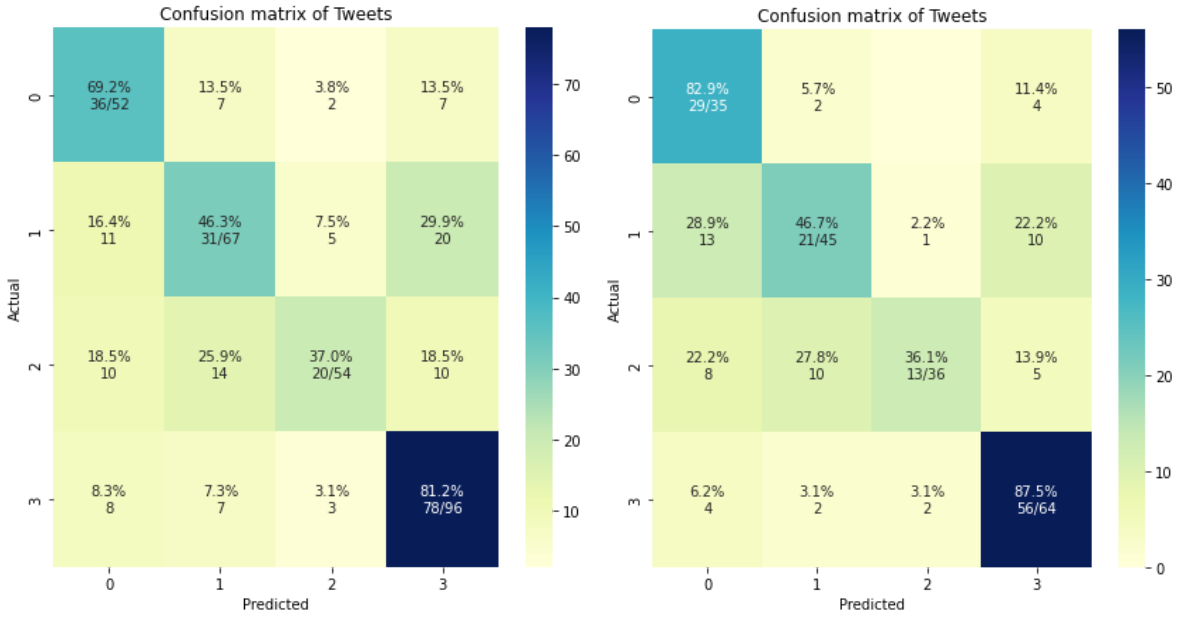


Figure 6.1 and 6.2 - Multi NB Tfidf Confusion Matrix (Before and After undersampling)

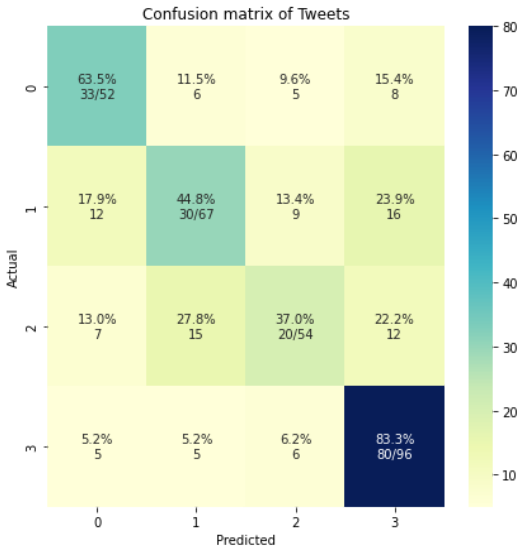


Figure 6.3 - NN (MLP) Confusion Matrix

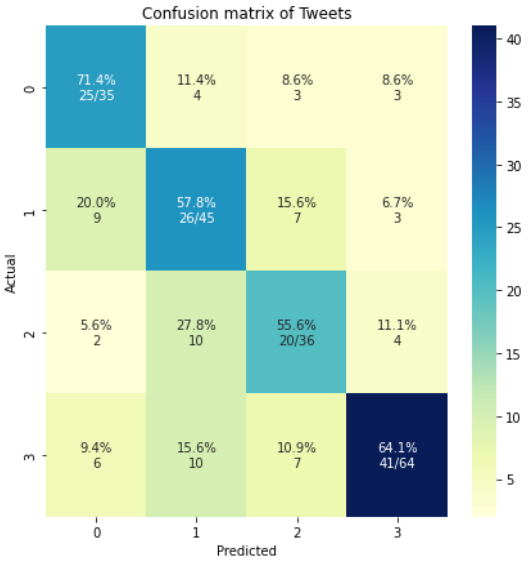


Figure 6.4 - NN (MLP).Confusion Matrix (After undersampling)

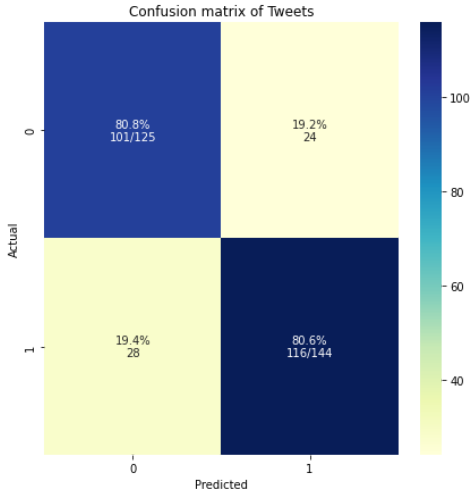


Figure 6.5 - Multi NB. Confusion Matrix (oversampling)

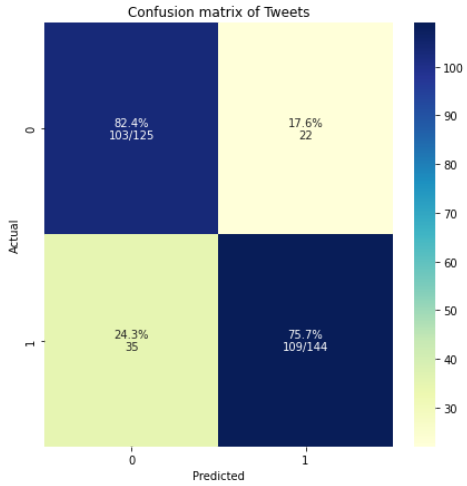
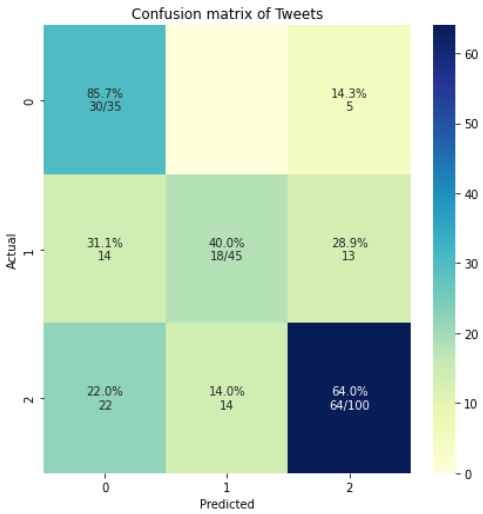


Figure 6.6 - Log. Reg, Confusion Matrix (oversampling)



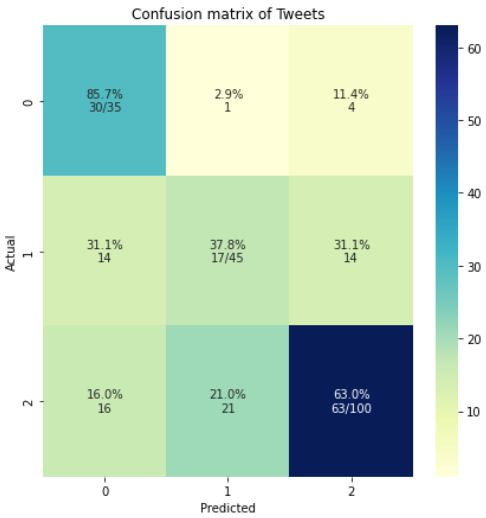


Figure 6.7 and 6.8 - Ternary Classification matrix for MultiNB and NLP (NN)

XII. Discussion

Based on our ML and DL results for multiclass classification, the best performing machine learning classifier was the multinomial naive Bayes classifier on TF-IDF features with an accuracy of ~66%. The best performing deep learning model was the NN (MLP) Text Model with TF-IDF feature extraction at an accuracy of ~63% (the Bidirectional LSTM with GloVe feature extraction also closely matched this performance).

Looking at the Multinomial-NB model, its MAE score of 0.53 is within a 10% tolerance of what can be defined as a good classifier based on accepted ranges for MAE values. We can also see that this model performs the best overall across most of the feature extraction methods used including unigram and POS tagging. The logistic regression model is a close runner-up with the Multi-NB. The poorest performing model appeared to be the KNN, with accuracies ranging from 54% on TF-IDF to 49% on POS tagging.

When discussing the impact of our auxiliary and social features, for our multiclass classification models, the additional social features, punctuation counts, and text-based emoticon counts did provide a slight boost in performance improving accuracy by ~2-3% across the board. The impact of these features, however, is more pronounced when we look at the binary classification problem with these models.

When compared to the multiclass baseline of 25%, discussed earlier, every model across all feature extraction methods achieves a success rate greater than 30-40% above this value. However, to determine whether model performance continued to be impacted by a lack of difference among positive and negative type emotions, binary classification was tested as well, which we will discuss now.

As can be seen in our data, all classifiers performed better with binary classification and were able to discern that joy and gratitude emotions were in contrast with apprehension and sadness tweets. This “binary” classification is a separation of positive and negative emotions, but due to the spread of the emotions that are contained within the groups, there is an overlay over the groups and the error is shown in the tables.

Despite this error, we were able to successfully develop an acceptable binary classification model, which was our Multinomial-NB coupled with Tfidf. This model had the highest prediction accuracy of 81% with an MAE score of 0.19 and an RMSE score of 0.43. This was also after incorporating the punctuation counts and text-based emoticon counts. Other ML models, like the logistic regression with unigram, also performed well with a 79% accuracy and the best performance from our deep learning models came from our NN (MLP).

The impact of our auxiliary and social features on our binary classification problem was far more beneficial. By including the social features, punctuation counts, and the text-based emoticon counts, we saw an increase in performance accuracy of around 4-6% over most of our models, with the Multi-NB model getting a 10% accuracy increase. This shows that it is important to consider these features for emotion classification as they convey value towards the sentiment.

Shown in figure 6.7 and 6.8 is the ternary classification metrics for the highest-performing machine learning and highest-performing deep learning classifiers. This ternary classification was achieved by combining the middle two groups (S [Sadness / Anger / Frustration] and J [Joy / Humor / Sarcasm]). These two classes were combined since they share some similarities with the emotions they have. After performing this three-way split, we can see that there is still a significant missclassification that still occurs with the middle combined emotion group. However, one important discovery from doing this is that the other two emotion group accuracies increased from performing the 4-way classification task. From this we can draw that the emotion group labels themselves may be causing the inaccuacis. For the N and G group, they both contain emotions that are similar and unique. When analyzing the plutchik's wheel of emotions (), they exist in theri own regions and are distinct. In the other S and J combined group, the emotions seem to spread all over the distribution of emotions and do not seem to be able to be put in their own unique and separate corner.

Moving on, when we rebalanced our training data (with undersampling) to exactly 350 tweets per emotion class, we yielded fairly similar accuracies

across all models. However, looking at the balanced data confusion matrices in figures 6.2 and 6.4, we can see a shift in emotion classification. Once the class balancing was performed on the training data, we can see that there were more true positives in the confusion matrices. This occurrence is the case for both our ML and DL models and we can see both the Multinomial-NB and the NN (MLP) classifier had more correct predictions fall on the diagonals of their confusion matrices. The main indication from the data rebalancing process is that although direct accuracy did not change significantly for both the ML and DL models, a rebalance allowed each of our ML and DL classifiers to better determine which emotion a tweet falls under. From the NN (MLP) classifier matrix, we can also see that a DL model can obtain a better distinction among the emotions and could outperform the ML models with more data and tuning. Based on how the data rebalancing affected our confusion matrices, we can safely conclude that if the dataset was much larger and thus more balanced across all 4 classes, we would see much better performance and accuracy across all of our models.

Another result that complements this point is that when we performed random oversampling of our training dataset on our binary classification problem, we also observed an increase in performance accuracy, which is where we obtained our best performing Multinomial-NB model, and it also boosted other ML model performances by ~ 5%. This shows that having more training data helps the models develop more robust reasoning and prediction capabilities.

Diving deeper into the results, we can see that the overall performance of our models is up to par with the general performance for multiclass emotion classification problems in related research, such as [3] by Chang, Ferreira, & Yang with their overall accuracies for J and N at approximately 65% and 62% respectively. We can also see in [5] by Mondal & Gokhale that more complex emotions such as surprise and love have accuracies of 62% and 52% respectively based on an SVM model. We can see (from our 4x4 confusion matrices) that our multiclass models did a pretty good job distinguishing the individual emotion groups that were labeled as N and G, and they had little more trouble with isolating the S and J emotion groups. The J label of Joy/Humor/Sarcasm is what we think causes this ambiguity since the emotion of sarcasm is more complex and often incorporates both positive and negative sentiments. Accounting for the fact that our data is labeled by emotion groups, not individual emotions adds additional complexity that justifies our models as having the shown performance given the relatively smaller dataset trained on. If we had a larger dataset to train our models on, we are certain our metrics would be far better. All in all, we conclude that this performance is a success for us

XIV. Conclusions & Future Research

As mentioned earlier, this paper analyzes vaccine emotion on Twitter through a dataset of collected and labeled tweets within one week in March, around the time of vaccines being made available for select members of the public. This final analysis indicates that reorganizing the way our training data is balanced so that it has an equal emotion distribution and incorporating additional social features, such as punctuation counts and text emoticon counts (on top of feature extraction methods like TF-IDF), improved our per emotion classification drastically. For multiclass classification, the Multinomial-NB model gave 66% accuracy and the NN (MLP) model gave approximately 63% accuracy. For binary classification, our Multinomial-NB (coupled with TF-IDF) had the highest prediction accuracy of 81% followed by the logistic regression with 79%. Going back to the initial questions we aimed to assess and answer with this research, we are now able to answer both of them with confidence. We have shown that a framework can be developed for ML and DL multiclass classification models to accurately infer one of four emotion groups represented by a vaccine-related tweet and we have also shown, with our binary classification results, that a significant binary distinction can be made between tweets that express “negative” emotions (Apprehension, Anticipation, Sadness, Anger, and Frustration) and “positive” emotions (Joy, Humor, Sarcasm, Gratitude, and Relief). Given all this, we are satisfied with these results.

Our future research will involve gathering more data to provide all models more information and

minimize data rebalancing that undersamples a higher frequency emotion or that samples with replacement. Redesigning the emotion labels so eahc class is conationg its own distinct subset of emotions is also a promising step that we should take, as that should greatly reduce inaccruais in the models overall. Additionally, we would like to explore alternate sarcasm detection feature extraction methods other than POS tagging to further reduce tweet misclassification.

REFERENCES

[1] Eleonora D'Andrea, Pietro Ducange, Alessio Bechini, Alessandro Renda, Francesco Marcelloni, *Monitoring the public opinion about the vaccination topic from tweets analysis*, *Expert Systems with Applications,* Volume 116, 2019, Pages 209-226,

[2] On J, Park H, Song T, *Sentiment Analysis of Social Media on Childhood Vaccination: Development of an Ontology* J Med Internet Res 2019;

[3]Monselise M, Chang C, Ferreira G, Yang R, Yang C, *Topics and Sentiments of Public Concerns Regarding COVID-19 Vaccines: Social Media Trend Analysis*, J Med Internet Res 2021;

[4]Karami A, Zhu M, Goldschmidt B, Boyajieff HR, Najafabadi MM. COVID-19 Vaccine and Social Media in the U.S.: Exploring Emotions and Discussions on Twitter. *Vaccines*. 2021; 9(10):1059. https://doi.org/10.3390/vaccines9101059

[5]Mondal, Gokhale. Multi-Label Classification of Parrott’s Emotions, 2020