Modelling Citizen Perception on Climate Change based on Twitter Data

Sarang Pramode (sp872), Rajesh Mudidana (rm775) and Krishna Datta (kd352)

1. Abstract:

Our objective is to inform the general public if information relayed on twitter is Factual/News or Opinion, with further classifications under Opinion to signify sentiment. We aim to tackle misinformation surrounding a major crisis we are facing - Climate Change. In this study a model which will classify the sentiment of a tweet as one of four types - Factual Change, Belief of Change, Neutral, and Non-Believer. To classify tweets we implemented several multi-class classification algorithms which included Multinomial Naive Bayes, Random Forests , LSTM and Logistic Regression trained on thirty thousand tweets and algorithm performance has also been compared with LSTM achieving optimum performance with a training accuracy of 70.1% and a testing accuracy of 65.4%.

2. Introduction

Climate Change and its effects are leading us to irreversible damage and we need to take action today. While policymakers and lobbyists take care of the direction the world should move in, the people and their opinions are largely influenced by social media and misinformation. Twitter is arguably the most important platform for social engagement, and it is no surprise that the climate conversation has all sorts of opinions, fuelled by the actions and tweets of famous personalities.
One way to control misinformation and help people educate themselves is to signify certain tweets as misinformation or news reporting.

Many companies are built around lessening one's environmental impact or carbon footprint. They offer products and services that are environmentally friendly and sustainable, in line with their values and ideals. They would like to determine how people perceive climate change and whether or not they believe it is a real threat. This would add to their market research efforts in gauging how their product/service may be received.

With this context, we are being challenged with the task of creating a Machine Learning model that is able to classify whether or not a person believes in climate change, based on their novel tweet data.

3. Background

3.1 Data Description

The dataset we used aggregates tweets pertaining to climate change collected between Apr 27, 2015 and Feb 21, 2018. In total, 43943 tweets were collected. Each tweet is labelled independently by 3 reviewers. This dataset only contains tweets that all 3 reviewers agreed on (the rest were discarded).

Dataset:
https://drive.google.com/file/d/1Kec5vWu_q_WS_v6hxWhDJ5rkpk_KisEQ/view?usp=sharing

3.2 Data Structure

TweetId	Unique Id of each tweet
Message	Actual text body of the tweet
Sentiment	Sentiment assigned to each tweet as defined below

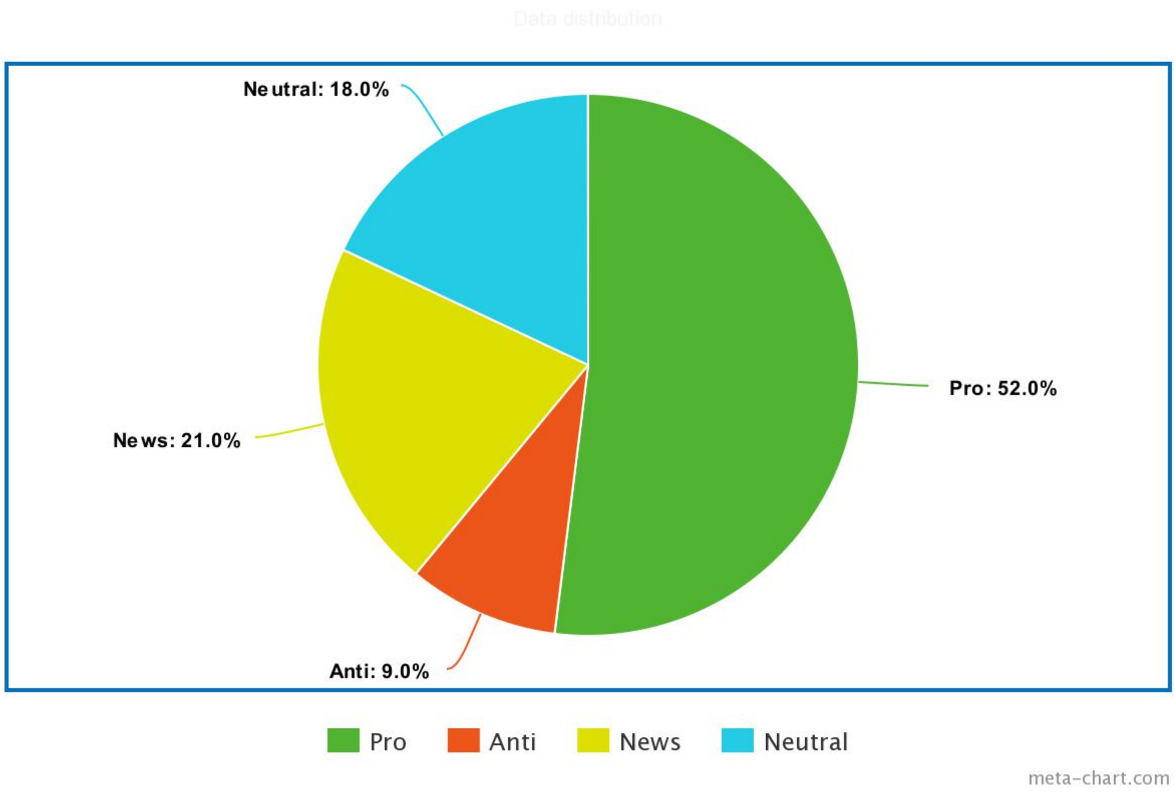
Our most important columns are, naturally, the *sentiment* and *message* columns, we can ignore *tweetid* — since it isn't relevant to a Tweet's sentiment.

3.3 Data Labelling

Classes -1, 0, 1 are self-explanatory, it's worth taking note of class 2. The 'News' class adds a secondary goal for our model, to sense if a tweet is sharing factual news.

-1	Tweet doesn't believe in climate-change
0	Tweet neither supports or disagrees with the belief of climate-change
1	Tweet supports belief of climate change
2	Tweet is factual news about climate change

3.4 Data distribution



3.5 Sample data from the dataset

TweetId66	Message	Sentiment
344354	@tiniebeany climate change is an interesting h...	-1
234554	RT @NatGeoChannel: Watch #BeforeTheFlood right...	1
678324	RT @AmericanIndian8: Leonardo DiCaprio's climate change documentary is...	0
789325	RT @cnalive: Pranita Biswasi, a Lutheran from ...	2

It is important to note that these classes **are not balanced —** we have unequal populations of each class — we'll address this later with under sampling the majority classes.

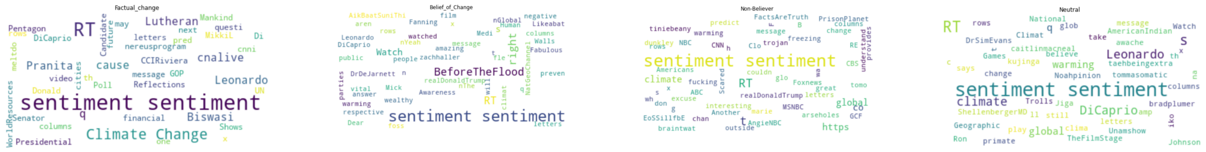
4. Method

4.1 Pre-processing

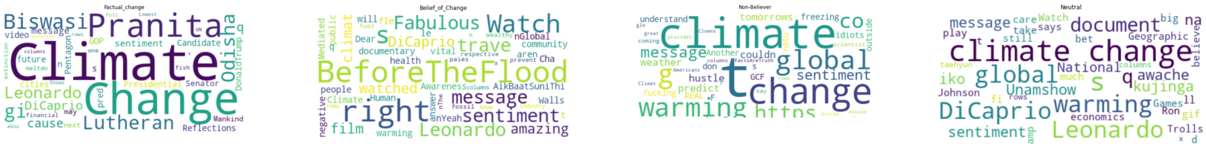
Prior to running our classification methods we performed exploratory analysis of the dataset and carried out the following pre-processing steps:

1. Hashtag and Mention Count
2. Word count and Punctuation count
3. Mentions and URL removals
4. Remove stopwords
5. Checking for null values, unique labels and plot distribution
6. Remove punctuation and non-alphabetical characters
7. Convert data to lowercase
8. Mean word length

To obtain visual context on the dataset we also plotted a word cloud to assess if preprocessing the dataset removed noise.



Post preprocessing the world cloud generated is shown below



From the wordcloud generated post pre-processing , we observe the data is denoised and common keywords have been removed(Eg, 'RT', 'Sentiment'). A key observation here is that there are common keywords used across tweets to identify them to be relevant to their class labels. This causes an inherent bias to the dataset and requires complex models to discriminate efficiently.

4.2 Learning Models Implemented

Multinomial Naive Bayes

We used CountVectorizer to come up with a Bag of Words representation after pre-processing the tweets. This was then passed to a TF-IDF transformer. The reason we did this is because we assumed that attaching an importance metric to the words would improve accuracy, and we wanted to capture both the raw word counts as well as their relative importance. This intuition was wrong.

This was then given as input to the Multinomial Naive Bayes classifier. Here, although the label is categorical, we cannot use Categorical Naive Bayes due to how the prediction is being made. Since we have converted out tweets into individual tokens, and the prediction is made as an aggregate based on the tokens within a "bag", we need to use Multinomial Naive Bayes to predict the probability of every label for every combination of tokens, not give an expectation of label by taking the aggregate probabilities of every token.

We also considered unigrams and bigrams in our Bag of Words, which improved the accuracy.

Random Forests

We implemented Random forests which are an ensemble method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

Before we feed our data into a model, we had to vectorize it. For this we used a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer from Scikit-learn.

Logistic Regression

Logistic Regression uses the probability of a data point to belonging to a certain class to classify each datapoint to its best estimated class. In our case we used multi-nominal logistic which is a simple extension of binary logistic regression that allows for more than two categories of the dependent or outcome variable. Like binary logistic regression, multinomial logistic regression uses maximum likelihood estimation to evaluate the probability of categorical membership.

LSTM

We chose this RNN architecture because we saw that the words "Climate", "Change" were present with very high frequency in all tweet classes, and we did not want the bias in the data to explode the coefficient for these words and other similar words in favor of class '1'. The special architecture of the LSTM was best equipped to handle this exploding coefficient problem, and would be able to learn other less significant features in the data as well. This intuition proved to be correct.

We also saw that the best utilization of computational power was with 64 dimensional features, and we were able to reduce overfitting by using a recurrent dropout. We used normal Tokenizer embeddings and built a simple architecture to extract 64 dimensional feature sets for the 2000 dimensional embeddings, which we used to predict the label. We also extracted the probability values to get an intuition for the baseline bias in the data in terms of percentage of wrong class mappings.

5. Experimental Analysis

5.1 Intuition

With the data that we had, we were confident that the existing data labels were accurate as they came from a 3-person consensus after independent hand labelling. Looking at the distribution of the data, we saw that it reflected the real world pretty accurately, but since this is not a toy dataset, and tweets contain a lot of contextual information that is not important to the task, but cannot be removed, we expected a high level of bias even after we reduce the noise in the data (removing @ mentions, tweet ids, etc).

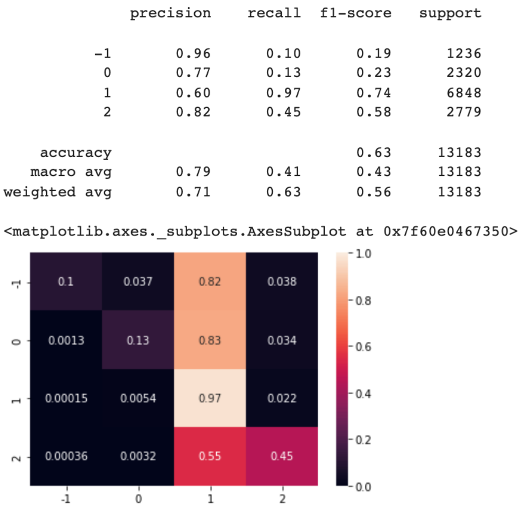
5.2 Model Results

Naive bayes

Model Scenarios Run:

- CountVectorizer was run without n-grams with pre-processed data
- CountVectorizer was run with n-grams (1 and 2) with pre-processed data
- We also tried using a TF-IDF transformer before training the data, but due to an implicit skew in the data distribution, we noticed that it was driving all the predictions to the max class

(Best Model: Naive Bayes with a simple CountVectorizer representation of the raw data after pre-processing, including unigrams and bigrams. Results on the right above.)



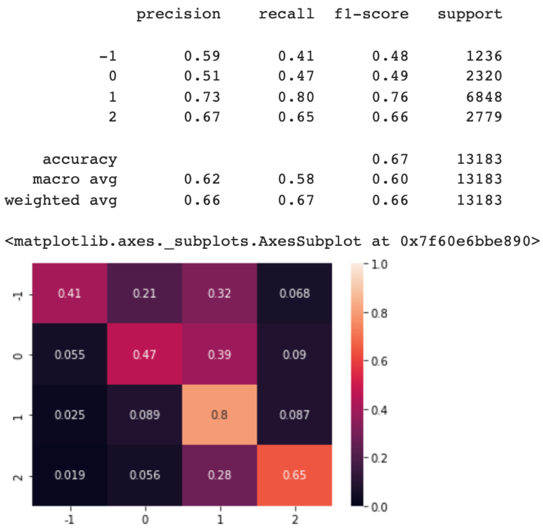
```
              precision    recall  f1-score   support

          -1       0.96      0.10      0.19      1236
           0       0.77      0.13      0.23      2320
           1       0.60      0.97      0.74      6848
           2       0.82      0.45      0.58      2779

    accuracy                           0.63     13183
   macro avg       0.79      0.41      0.43     13183
weighted avg       0.71      0.63      0.56     13183

<matplotlib.axes._subplots.AxesSubplot at 0x7f60e0467350>
```

Logistic Regression

Model scenarios run:

- With Regularization, and including unigrams and bigrams in a Bag of Words model
- With Regularization, but no n-grams in the Bag of Words model
- Without Regularization, and including unigrams and bigrams in a Bag of Words model

  (Best Model: with Regularization, and including unigrams and bigrams in a Bag of Words model. Results on the right above)

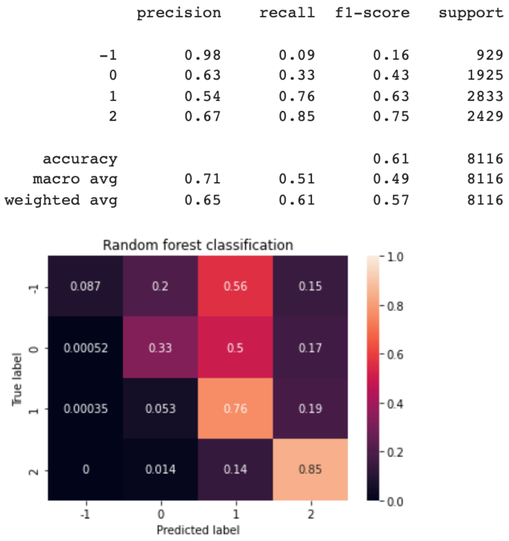


```
              precision    recall  f1-score   support

          -1       0.59      0.41      0.48      1236
           0       0.51      0.47      0.49      2320
           1       0.73      0.80      0.76      6848
           2       0.67      0.65      0.66      2779

    accuracy                           0.67     13183
   macro avg       0.62      0.58      0.60     13183
weighted avg       0.66      0.67      0.66     13183

<matplotlib.axes._subplots.AxesSubplot at 0x7f60e6bbe890>
```

Random Forests

Model Scenarios Run:

- Without under sampling the majority class of pro climate change tweets and with pre-processed tweets
- With under sampling the majority class of pro climate change tweets and with pre-processed tweets
- Without under sampling the majority class of pro climate change tweets and without pre-processing tweets
- With under sampling the majority class of pro climate change tweets and without pre-processing tweets



```
              precision    recall  f1-score   support

          -1       0.98      0.09      0.16       929
           0       0.63      0.33      0.43      1925
           1       0.54      0.76      0.63      2833
           2       0.67      0.85      0.75      2429

    accuracy                           0.61      8116
   macro avg       0.71      0.51      0.49      8116
weighted avg       0.65      0.61      0.57      8116
```

With under sampling the majority class of pro climate change tweets and without pre-processing tweets gave us the best accuracy in the case of Random Forests. This process was repeated over different train & test splits and over different ranges of max_depth and number of decision trees to get the average performance.

Observations:

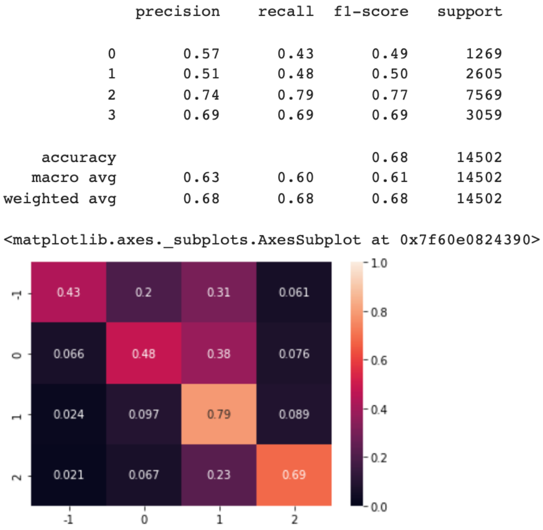
- The f1-scores of anti & neutral class tweets are low, especially for the anti climate change class tweets.
- Random forests performs poorly on classes -1 & 0. The model pre-dominantly classifies anti climate change class tweets as pro climate change tweets.
- The overall accuracy of the model is 0.61.

(Best Model: With under sampling the majority class of pro climate change tweets and without pre-processing tweets. Results on the right above)

LSTM

Model Scenarios run:

- Ran a tokenizer to create a simple Bag of Words representation and convert tweets to pre-padded token list representations
- Ran a tokenizer to create a simple Bag of Words representation and convert tweets to post-padded token list representations. (Dropped due to lower performance)



```
              precision    recall  f1-score   support

           0       0.57      0.43      0.49      1269
           1       0.51      0.48      0.50      2605
           2       0.74      0.79      0.77      7569
           3       0.69      0.69      0.69      3059

    accuracy                           0.68     14502
   macro avg       0.63      0.60      0.61     14502
weighted avg       0.68      0.68      0.68     14502

<matplotlib.axes._subplots.AxesSubplot at 0x7f60e0824390>
```

To build the model, we added a Sequential Layer, added the Token Embeddings as inputs, fed it into an LSTM Layer with recurrent dropouts, and finally, added a Dense layer for the 4 class prediction.

For the LSTM, we decided that since the RNN architecture would take care of feature isolation and importance, so we did not need to use a TF-IDF network for the Embeddings. The LSTM architecture was able to handle the predictions the best, mostly because it was not limited to:

- only unigrams and bigrams
- was able to do better feature isolation and importance

Additionally, this performance also gives us an intuition for the baseline bias in the data, which is between 0.23 and 0.38 in favor of the class '1'.

(Best Model: First scenario as mentioned above. Results on the right above)

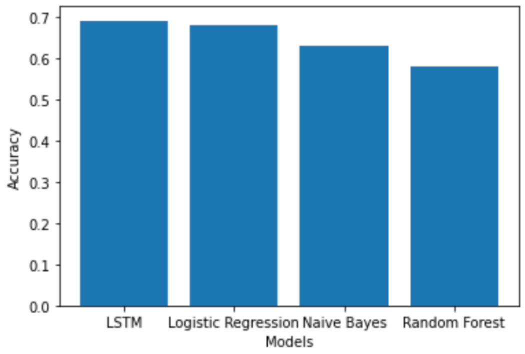
7. Discussion

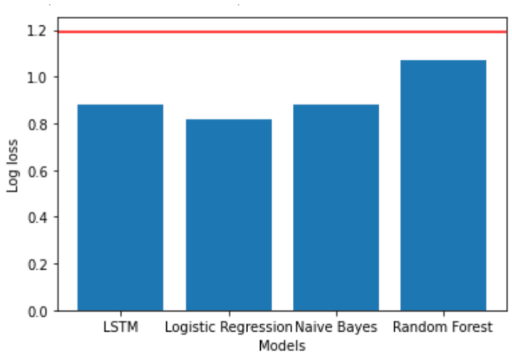
As observed from the performance of the above models we can conclude that LSTM and Multinomial Logistic regression performed the best. From the confusion matrices of these models as well as from our initial pre-processing of the dataset we can see that the dataset contains an inherent bias towards class 1(Belief of Change).

The dataset contained approximately 44,000 tweets of which 50% were labeled as containing belief of change. The inherent bias proved to be a major challenge to optimize our models. To offset this, we found that under-sampling the dataset to provide a more even spread for our labels proved only to marginally improve performance. Upon further pre-processing it was observed that due to the phrasing and keywords present in tweets labelled as Neutral and Non-Believer the performance of the models were impaired.

The log loss as well as model accuracy have been plotted below. It is observed from the confusion matrices as well as the below plots LSTM provided the best test accuracy though marginally higher than multinomial logistic regression which had the lowest log loss, providing a minor improvement in generalizability by offsetting the inherent bias relatively better.

Model	LSTM	Logistic Regression	Naive Bayes	Random Forest
Log Loss	0.88	0.82	0.88	1.07
Test Accuracy	0.69	0.68	0.63	0.58





8. Conclusion & Summary

Model Evaluation

Our problem was to be able to determine sentiment behind Climate Tweets, which is useful in understanding where people may stand on support for potential environmental policy or action.
After some pre-processing and resampling, we used a variety of models and found *Multinomial Logistic Regression* & *LSTM* to be the best models out of those examined.

Feature selection

Feature selection was done by using a numerical representation of the given tweets. In some cases, we found a simple Bag of Words representation to work best, in some cases, we needed to use an additional TF-IDF Transformer to capture importance data as well, but where we did not need it, we had a different Neural Network anyway. N-grams always worked better than just choosing unigrams, and hyperparameter tuning led us to the values that we have reported above.

With respect to the limitations in terms of the methodology employed, we recognize that in the current approach, though robust , require additional preprocessing to extract relevant information. An additional improvement of the performance of our models can be brought by the utilization of a superior dataset with uniform label distribution and absence of bias.

Overall, we conclude that for this dataset, we were able to identify a tangible bias in favor of one class, we were able to confirm that a numerical representation, especially a vector embedding, followed by some kind of Neural Network architecture, works best in classifying tweets to our desired classes.

9. References

1. https://athena.explore-datascience.net/student/content/train-view/38/100/1783
2. Sperandei, S., 2014. Understanding logistic regression analysis. Biochemia medica: Biochemia medica, 24(1), pp.12-18.
3. Dr. Sebastian Raschka. (2014). Turn Your Twitter Timeline into a Word Cloud. [online] Available at: https://sebastianraschka.com/Articles/2014_twitter_wordcloud.html
4. Zhou, L., How to Build a Better Machine Learning Pipeline, 2018. URL https://www.datanami.com/2018/09/05/how-to-build-a-better-machine-learning-pipeline.
5. The Independent. (2019). More than half of people say climate change will influence how they vote in general election. [online] Available at: https://www.independent.co.uk/environment/climate-change-crisis-latest-general-election-green-party-vote-boris-johnson-a9175756.html