Team 13:
- John Lu
- Sai Prasanna Kumar Kumaru
- Sai Vennela Garikapati

Conducting Sentiment Analysis on Tweets

Abstract:

The project's objective is **to find out what percentage of Twitter users believe climate change exists**. The main task is to classify the tweets into two categories whilst also comparing the performance of several different classification algorithms. This is also a good project to learn how to conduct preprocessing for an NLP task. This project is significant since it is a useful way to gauge public opinion on this issue.

Dataset:

The dataset is found on the website data world and consists of 6,090 tweets in total about climate change or global warming [1]. The tweets are manually labeled by workers through the platform CrowdFlower [2]. The tweets are classified into three categories shown in the "existence" column: believing in climate change (yes), not believing in climate change (no), and not directly referencing climate change (null) [2]. The "existence_confidence" column shows the level of agreement for the judgment of that tweet and "the trust level of each of those workers" [2]. This third column will be ignored since we only care about the labels.

Tweets have also been web-scraped using Selenium and Beautiful Soup by searching for "climate change" or "global warming" on Twitter. The Twitter API was also used to retrieve tweets, but it had an upper limit of 10 tweets per request. These tweets will be used to test how well the model works on modern tweets using a Web interface.

Preprocessing:

An initial preprocessing step is removing duplicate tweets, which reduced the number of tweets down to 5,541. The existence column also needs to be encoded into binary classes (Yes/Y for 1 and unsure/No/N for 0).

The natural language preprocessing steps involve the spaCy pipeline, which comes pre-trained with various NLP preprocessing functionalities. spaCy provides capabilities such as removing stop words, lemmatizing words, checking for alphanumeric characters, and filtering out location names. Lemmatization is the process of reducing words to their base words ("eating" to "eat") so that different words in different forms can be recognized.

The next step is converting the words to lowercase and removing all words less than length three. The links, hashtags, and mentions do not contribute to any meaning, so they have also been removed. However, we may experiment with keeping the hashtags in the future to see how it affects the sentiment analysis. Tweets that contain less than 4 words are pruned because they will not have enough words to help make a decision. There are also words combined with punctuation or other words that need to be split (such as "act|Brussels" or "EarthDay"). Nonsensical words that are not mapped to the English dictionary are removed.

The tweets from the original dataset are also cut off or encoded improperly, leading to the presence of special non-alphanumeric characters that need to be removed. This also exposes a shortcoming of spaCy, which failed to filter out these characters.

One challenge in preprocessing is deciding which words to omit. The erroneous removal of misspelled words can remove necessary context from a tweet; this results from the 280-character limit being enforced on tweets. One example is the word “movement” shortened into the word “mvmt”. However, this is a computationally expensive and tedious task to cover all the cases.

The data is then transformed into a sparse matrix using the bag of words approach and properly weighted using the TF-IDF vectorizer. Since this is text data, the number of features will be based on the number of unique words in all of the tweets. I also removed words that had greater than 80% document frequency in order to remove common signals between vectors.

Exploratory Data Analysis:

The following are the insights from the dataset:

1) There are 1,536 mentions, 1,551 hashtags, and 2,699 URLs.
2) The words in the top 10 hashtags are similar in both positive and negative scenarios. This does not help the model in classification, so they have been removed.

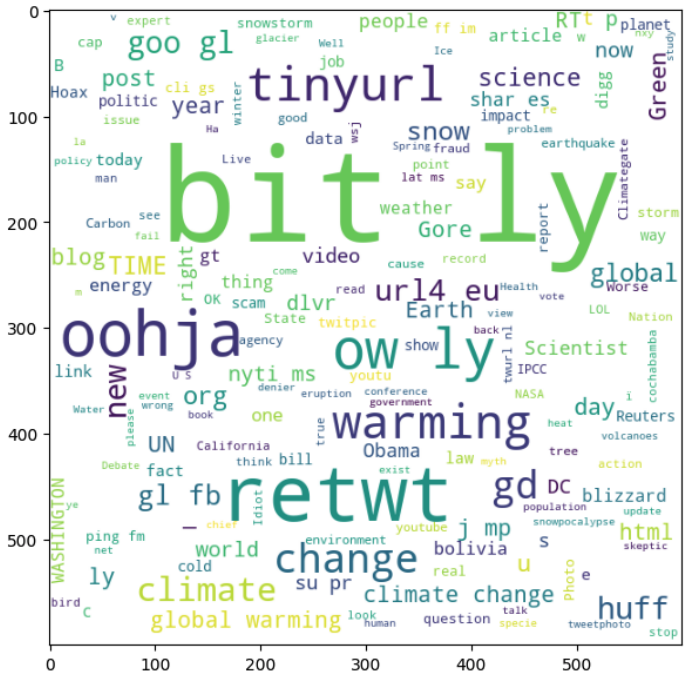
Positive

	hashtag	occurences
58	#climate	138.0
245	#tcot	69.0
119	#green	47.0
188	#p2	47.0
60	#climatechange	36.0
209	#saveterra	31.0
76	#earthday	24.0
80	#eco	23.0
113	#global	18.0
114	#globalwarming	15.0

Negative

	hashtag	occurences
123	#tcot	118.0
93	#p2	38.0
23	#climate	36.0
124	#teaparty	25.0
49	#global	18.0
25	#climategate	18.0
53	#gop	16.0
113	#sgp	15.0
127	#tlot	10.0
92	#ocra	10.0

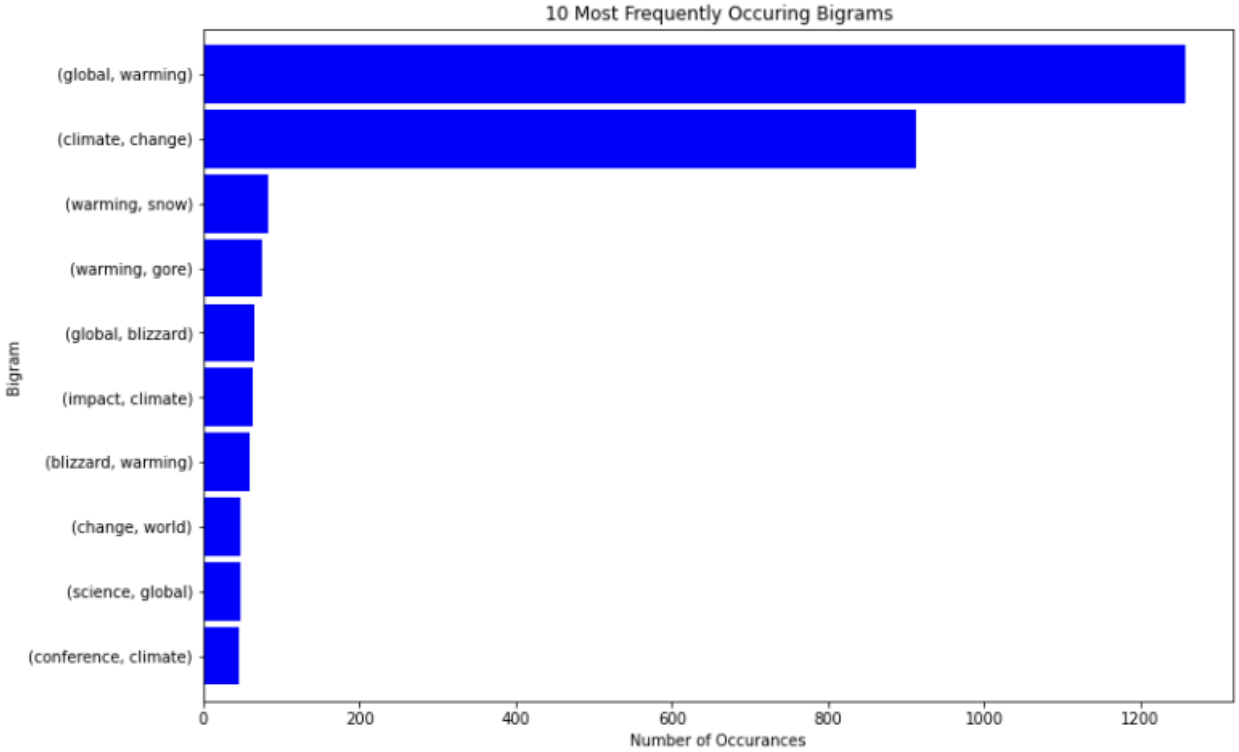
3) In the word cloud plot, larger-sized words indicate more frequent words, while smaller-sized ones indicate less frequent words. One interesting finding is that the words pertaining to links (bit, ly, tinyurl) show up as the most impactful prior to preprocessing.



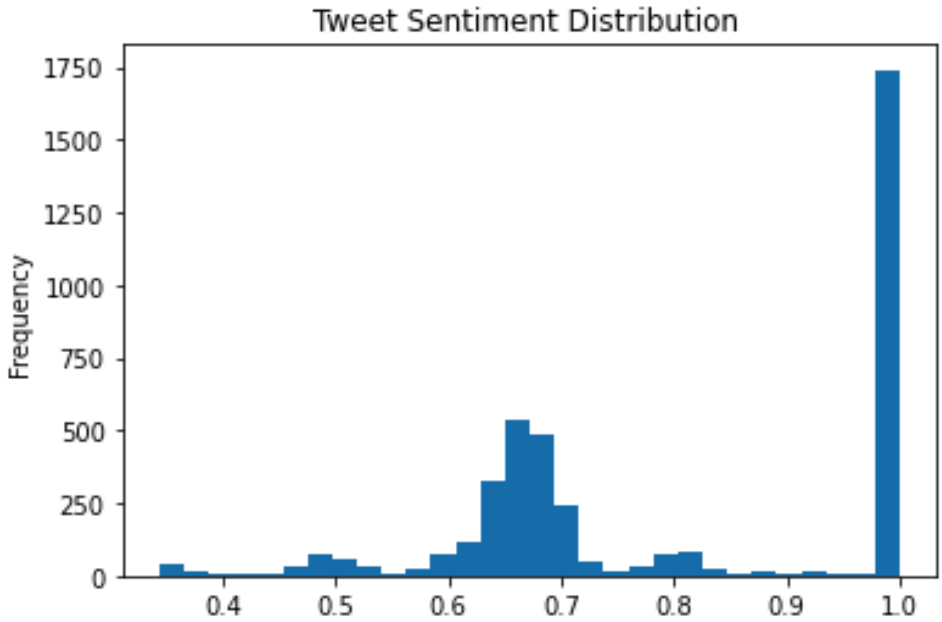
4) Words such as “earthquake", “confirm”, and “tsunami” contribute to positive tweets. These words have a correlation with climate change.
5) Words such as “myth” and “archive” contribute to negative tweets. These words belong to the group of words that do not believe in climate change.

Positive	Negative
	

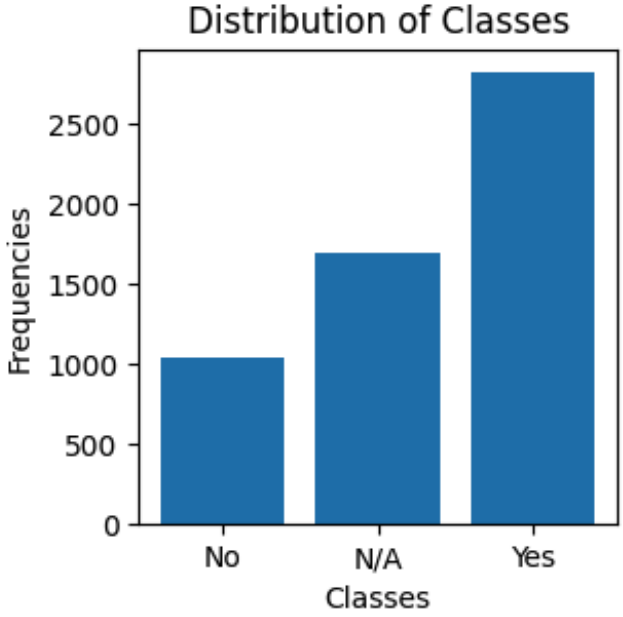
6) Word associations can be explored using N-grams, which provides further insight into understanding the context of the words.



8) We have analyzed the frequencies and distributions of the **existence** and **existence_confidence** columns. The distribution of the scores for **existence_confidence** is most frequent around 0.6 to 0.75 and highest at 1.0.



9) We have an imbalance where more tweets believe in climate change than those don't. This is confirmed by the existence distribution histogram.



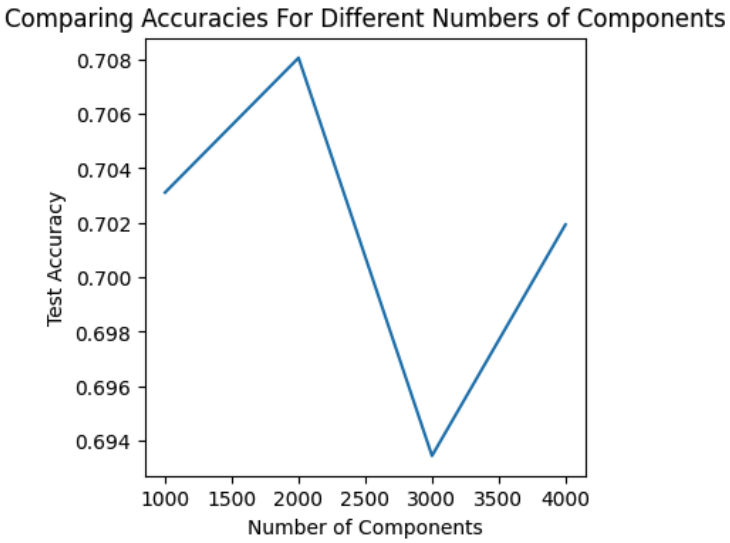
Method:

The data will be segmented into train and test sets with an 80/20 split. We have also used cross-validation sets with 10 folds to see how well the model generalizes. We are planning on conducting binary classification.

We took two approaches regarding the labels. One approach combined the "No" and "N/A" labels into one class, making them relatively balanced. The other approach excluded labels with N/A since the data suggests that those tweets are ambiguous. The removal of the N/A label leads to class imbalance. Therefore, the following approaches are necessary to overcome this:

1. Random Oversampling/Undersampling - samples from the minority class are picked with replacement, and random samples from the majority class are represented less. Both methods can lead to either overfitting the model or losing valuable information from the data.
2. SMOTE (Synthetic Minority Over-Sampling Technique) - a type of data augmentation technique for tabular data. It synthesizes new data for the minority class by selecting examples that are close to the minority class in the feature space. It draws a line between the examples in the feature space and picks a new sample at a point along that line.

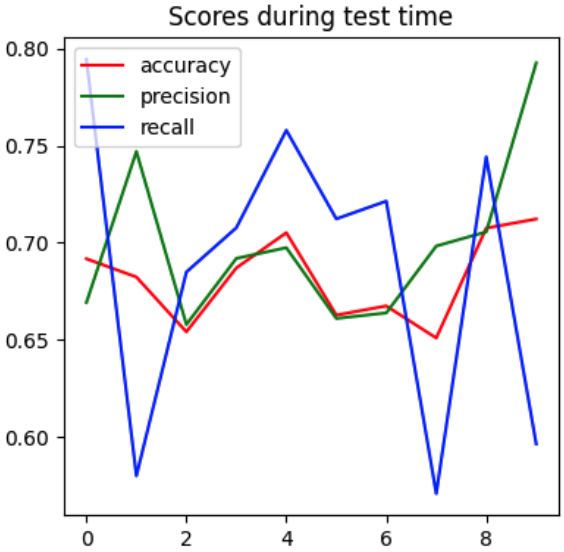
For both algorithms, the accuracy, precision, and recall were plotted across 10 folds using k-fold cross-validation. The best algorithm will be chosen by comparing the means of the accuracy, precision, recall, and F-1 scores. An additional preprocessing step was using TruncatedSVD to perform dimensionality reduction. This method is favorable for sparse matrices since it does not center the data on its mean. By plotting the number of components to the accuracy (shown below), it was shown that ~2000 components provided the best accuracy score.



Experiment/Analysis:

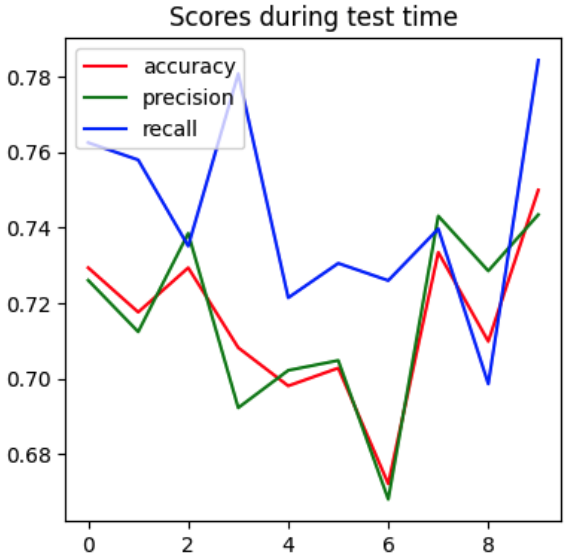
Balanced Dataset:

1) The Perceptron is a good baseline linear model to start with. It can help establish an initial understanding of the problem based on its accuracy scores.
    a) The model exhibits high variance across the different cross-validation folds. The training accuracy is 90% while the test accuracy is 67%. This means the model is performing slightly better than randomly guessing in test time.



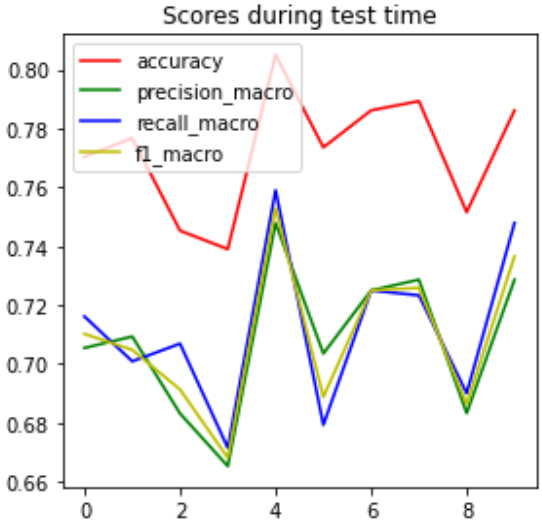
    b) The high training accuracy means that the data is almost linearly separable. Additionally, the separating hyperplane is very close to the boundaries of both classes, which leads to various misclassifications during test time.

2) Logistic Regression is another basic linear model used for classification. This model classifies tweets using a soft threshold with the sigmoid function. It performs better compared to the Perceptron but still overfits.
    a) The training accuracy is 86% while the test accuracy is 70%. The model also still exhibits high variance.

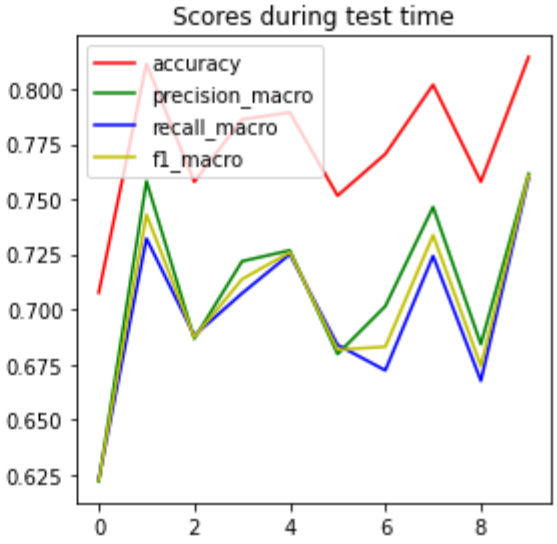


Imbalanced Dataset:

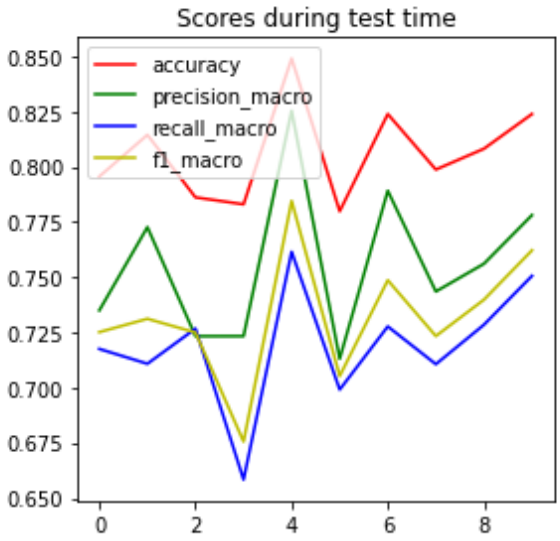
1) KNN calculates the likelihood of a data point mapping to a respective class based on the class of the points closest to it.
    a) When compared to other models, KNN achieves low test accuracy and F1 scores.
    b) Due to the imbalance in the dataset, KNN will give a lot of preference to the majority class which results in misclassifying the minority class. The reason for this is that the majority of the samples are from a single class.
    c) These are the cross-validation scores obtained after performing SVD with 2000 components.



2) Decision Trees use a tree-like model to make a decision. In comparison to KNN and other basic classification algorithms, decision trees are very fast, efficient, and scale invariant. Additionally, it is capable of capturing non-linear relationships.
    a) We achieve high train accuracy (99%) but comparatively less test accuracy (77%), indicating that the model overfitted the data.
    b) A large number of feature vectors results in a large number of splits which in turn generates complex trees that can result in overfitting.
    c) One of the disadvantages of decision trees is that it is highly sensitive to new data points, which change the tree structure completely.
    d) These are the cross-validation scores obtained after performing SVD with 2000 components.



3) Linear SVM is a simple algorithm that creates a line or hyperplane (decision boundary) which separates data into classes using a linear kernel. It is fast and uses less computation power. It tries to maximize the margin between the support vectors. Linear SVC minimizes the squared hinge loss while SVC minimizes the regular hinge loss. It has less chance of overfitting compared to other models.
    a) Among the models, SVM achieved high accuracy for train and testing with 95% and 80% respectively.
    b) This model outperforms KNN and Decision Tree in terms of F1 score.
    c) I trained the model with a non-linear kernel, which performed far worse than a linear SVM. This indicates that the data might be linearly separatable.
    d) Linear SVM performs the best among the other models (high accuracy, recall, precision, and F1 score).
    e) These are the cross-validation scores obtained after performing SVD with 2000 components.



4) Naïve Bayes is a type of statistical algorithm and is mostly used in text classification and analysis. This begins with an assumption that the features in the dataset are mutually independent. Due to these independent assumptions, the algorithm is considered naïve. Another variant of Naive Bayes is called the Multivariate Event Model or the Multinomial Naïve Bayes. The Multinomial Naïve Bayes uses a multinomial distribution.
    a) The motives behind choosing this algorithm are:
        - It is very fast and uses less storage.
        - The dataset is not large with a few thousand records with class labels.

$$P_{x_1,x_2\ldots\ldots x_k}(x_1,x_2\ldots\ldots x_k)=\frac{n!}{x_1\ *\ x_2\ldots\ldots\ldots x_k}*\ p_1^{x_1}*p_2^{x_2}\ldots\ldots*p_k^{x_k}$$

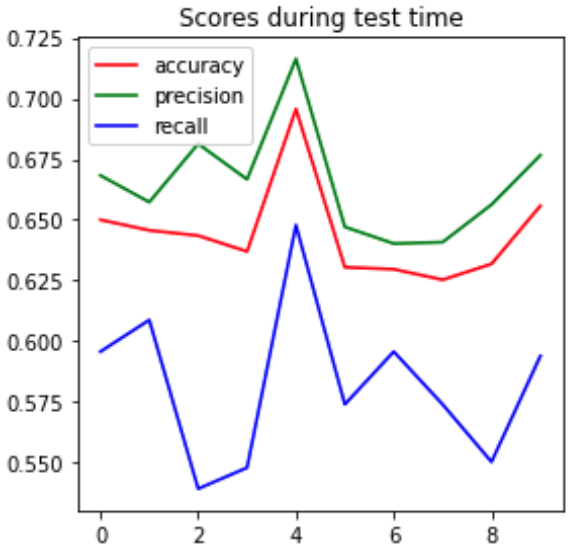
$$with\ \sum_{i=1}^{k}x_i=n\ and\ \sum_{i=1}^{k}p_i=1$$

$x_1,\ldots\ldots,x_k$=Number of occurrences of the word.
$p_1,p_2\ldots\ldots p_k$=Probability of each occurrence.
n=The total number of occurrences.

b) Insights on Performance:
- The testing accuracy of 78% is very poor when compared to the training accuracy of 86%. The majority class is positive, so the Naive Bayes algorithm has a higher likelihood of classifying the data into a positive class. Also, the precision is 98% when the recall is only 78%. This is due to overfitting.
- One possible reason for this is there are variables in the training set that have very high or low probabilities influencing the predictions. The probabilities differ from what we get from outside training data.

This results in higher accuracy at training time but not during prediction time.

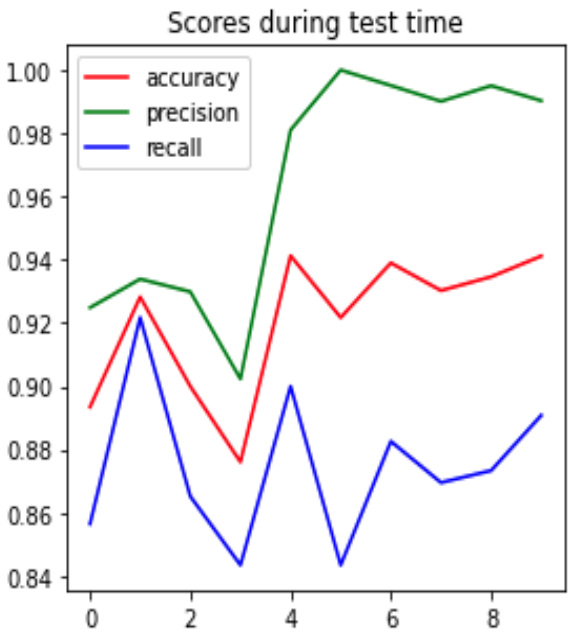
- After performing random over-sampling, it is observed that there is no significant improvement in both precision and recall scores. Moreover, accuracy dropped to 64%, and precision dropped to 60%.



5) Random Forest as the name forest implies a combination of trees, this algorithm is based on the concept of decision trees. First, it identifies a random set of features and grows the decision trees. These trees have their own error rate (out-of-bag error). Finally, all the decision trees are compared to produce a strong classification method.
   a) The reasons for considering Random Forest are:
      - There is a chance of overfitting of training data during Decision Trees classification, which can be resolved using random forest.
      - The features are randomly selected for the bagging method.

   b) Insights on Performance:
      - Random Forest **before** random-over sampling:
        - Initially, the accuracy is high for the model with 98% training accuracy and 80% testing accuracy, but that did not seem to be a reliable metric for gauging the performance of the model.
        - When considering other metrics, we observed a low score for precision and a high score for recall.
        - This indicates that most of the values predicted by the classifier are incorrect.
        - This has occurred due to the imbalance in the dataset.
      - Random Forest **after** random-over sampling:

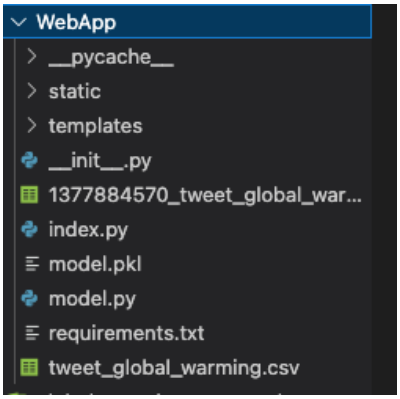
- After performing random over-sampling, it is observed that there is a **significant improvement** in both precision and recall scores, which are 86% and 85% respectively.
- The model has a testing accuracy of 80%.
- These are the cross-validation scores obtained after performing SVD with 2500 components.



Web App Integration:

- **Instructions:**

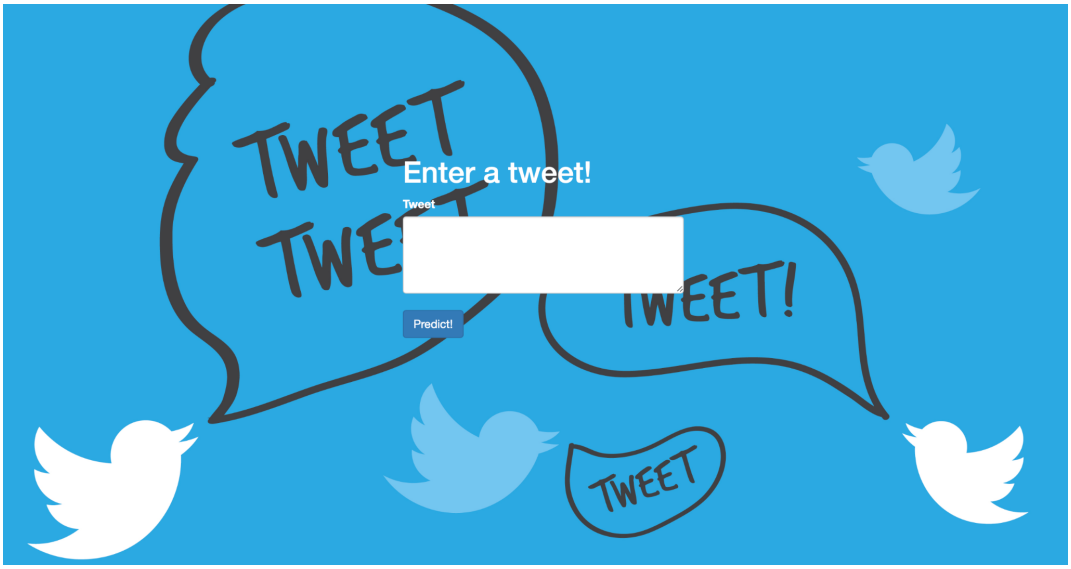
1) Install Flask with this command: pip3 install flask
2) Create the folder structure as shown below:



3) Create a pickle file for the generated model.
4) Create model.py and restore the generated model using the pickle file.

- **UI Screenshots and Use Cases**

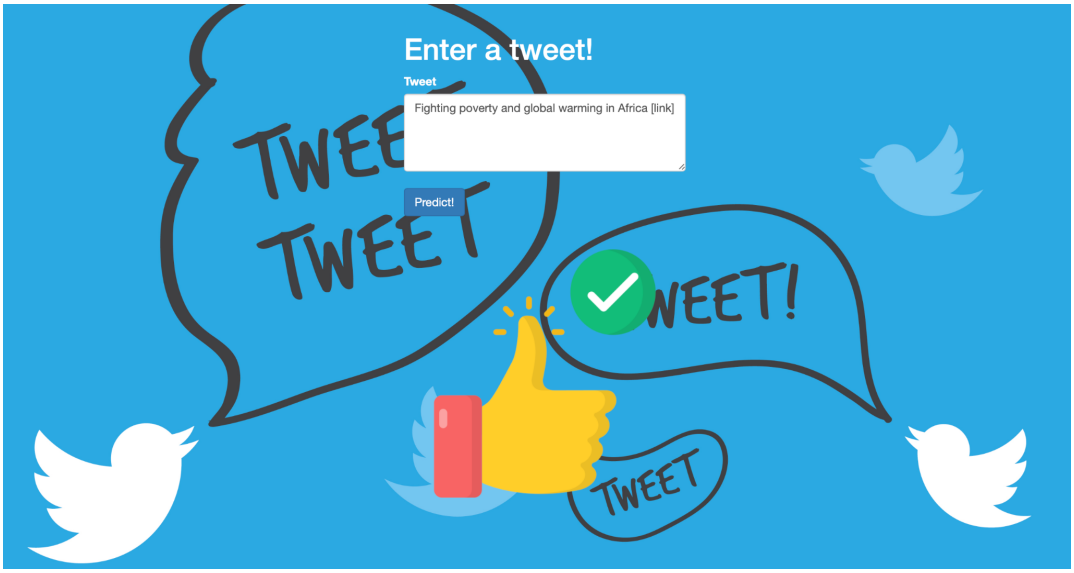
We have created a web page for testing the scraped tweets from Twitter. The screenshots of the UI are shown below.



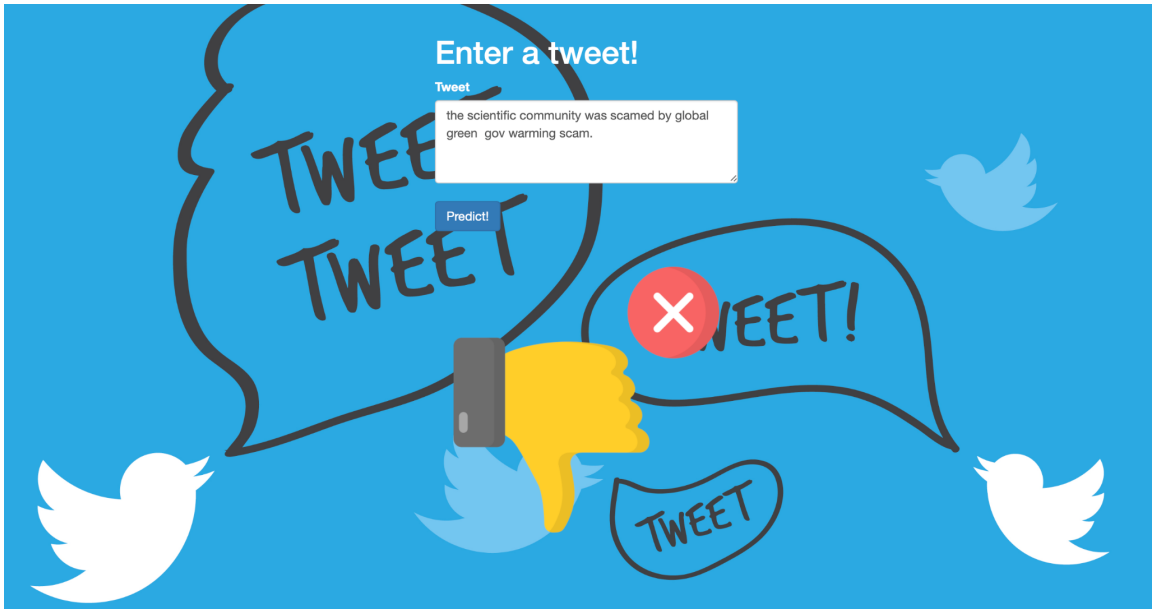
Using the two APIs created from Flask:

1) **Index**: This will redirect the user to a webpage where the user can enter a tweet. An AJAX request is made once the user clicks the predict button. Based on the response from the model, an emoji is displayed based on the sentiment.
2) **Sentiment**: The entered tweet will be passed on to this API which returns the sentiment of the tweet based on the model.

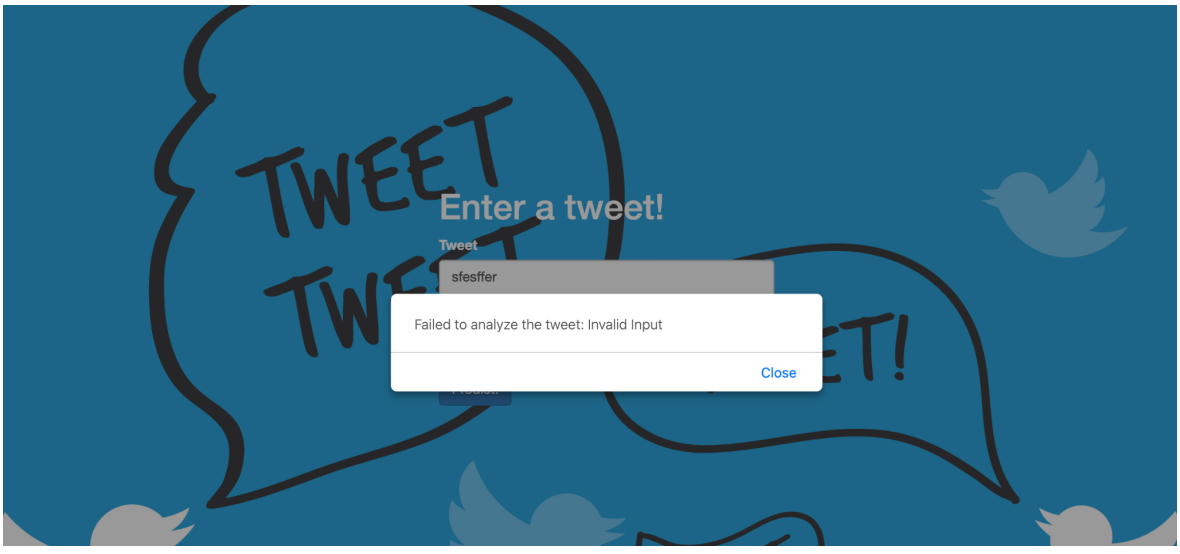
Scenario 1: Given a tweet in the textbox, the predict button then predicts the class.



Scenario 2: If the tweet is classified as negative, the screen below is displayed.



Scenario 3: Whenever a gibberish tweet is entered. I have validated it using a spellchecker, to know whether the entered tweet is in the English dictionary. If the word is not an English word, send an alert box displaying invalid input.



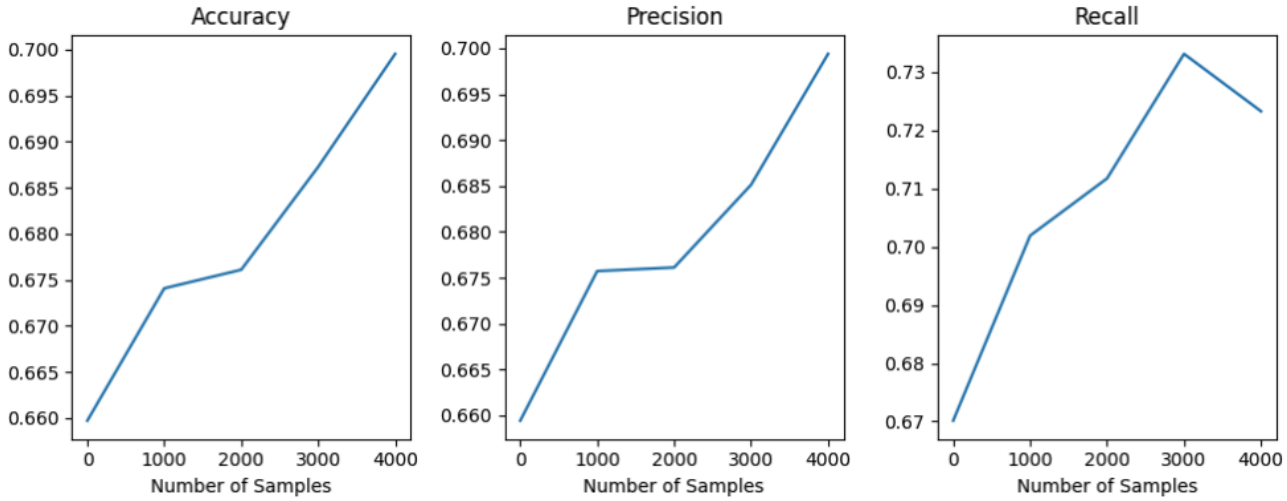
- **Insights:**
  - The model requires more training data to provide accurate sentiment analyses.

<u>Two Step Learning Analysis:</u>

- **How do we know we have enough data to learn?**

- K-fold cross-validation allows us to use the limited dataset we have most efficiently by switching out the data we use for training and testing.
- We need a lot more data since the preprocessed tweets become much smaller than their original size. Given the nature of tweets, people express themselves in lesser words, so there need to be more tweets to balance out the minimal number of words per tweet. Furthermore, all the data is concentrated on the month of April 2011, which may bring some inherent bias into the model [2]. However, we do not have the means to reproduce the experiment ourselves and obtain labels for additional data. We can only do the best we can with the limited amount of data we have.

- **How do we know if we have learned?**
  - The validation scores and test scores all demonstrate how well the model can generalize to unseen data. The Perceptron can be applied to any problem and will yield good scores if the data is almost or completely linearly separable. In the theory, the VC dimension for the Perceptron, Logistic Regression, and Linear SVM is d+1. The other non-parametric algorithms (KNN, Decision Trees, Naive Bayes) should theoretically be able to handle all possible dichotomies.
  - The plots below show how accuracy, precision, and recall change as the number of samples increases. Each set of samples was randomly sampled. The algorithm used below is Logistic Regression. This shows that we have indeed learned as the number of samples increases.



<u>Comparisons:</u>

The performances of the various algorithms are evaluated on 10 different folds for k-fold cross-validation on the entire dataset. The scores recorded here are the mean of the scores across the different folds at test time.

	Accuracy	Precision	Recall	F-1 Score
Perceptron	0.67	0.71	0.74	0.67
Logistic Regression	0.71	0.71	0.74	0.73
KNN	0.56	0.96	0.39	0.55
SVM	0.80	0.86	0.85	0.86
Decision Tree	0.77	0.82	0.85	0.84
Naive Bayes	0.64	0.66	0.58	0.66
Random Forest	0.92	0.96	0.87	0.92

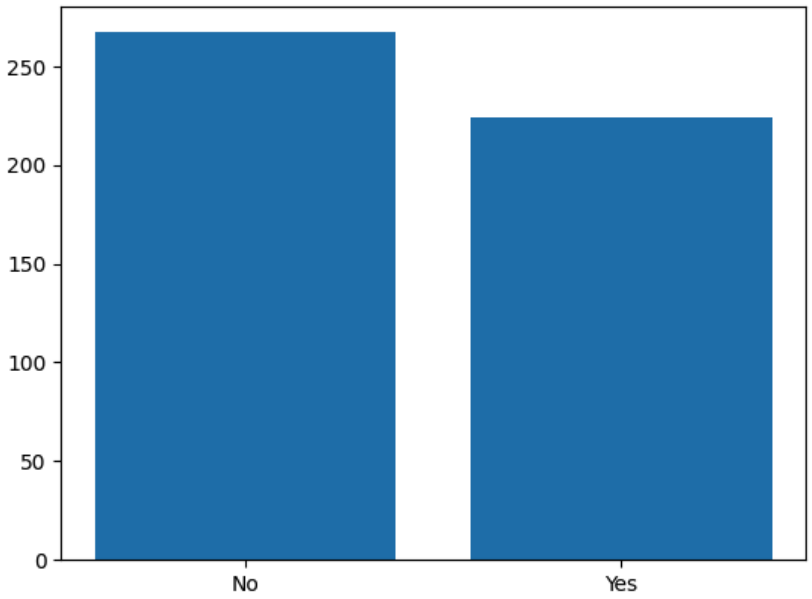
Conclusion:

In conclusion, this project aims to find the sentiment of users towards climate change using quality data obtained from Twitter and judgments obtained from CrowdFlower. With the analysis, it has been identified that the dataset is imbalanced with the majority inclined toward the positive class with 3029 records. Therefore, random under-sampling and sampling have been performed using SMOTE. This ensured balanced classes by generating more samples of the minority class. Also, the hashtags and mentions were removed as they are not contributing to the classification improvement.

Furthermore, the accuracy scores of seven classification models were compared over the preprocessed dataset. Among these models, Random Forest has given the highest F1-score with 92% followed by SVM with 86% and Decision Tree with 84%. On the other hand, algorithms such as the simple Perceptron, Logistic Regression, and Naive Bayes yielded worse results on the dataset due to overfitting. The solution to increasing the performance of the models would be increasing the number of records in the current dataset.

Additionally, we have implemented a web interface using Flask that runs the Random Forest model.

We have also run our model on the web scraped tweets from 2022 using the Random Forest model—our best-performing model. Our conclusion is that a higher number of tweets deny climate change. The figure shown below displays the distribution of the classes on these tweets. However, the distribution is close to half and half, which means our model can be made more accurate with more data and more model training.



Works Cited

1. https://data.world/xprizeai-env/sentiment-of-climate-change
2. https://kbares.quora.com/Can-We-Figure-Out-If-Twitter-Users-Who-Discuss-Global-Warming-Believe-It-Is-Occurring
3. https://stackoverflow.com/questions/19125722/adding-a-legend-to-pyplot-in-matplotlib-in-the-simplest-manner-possible
4. https://www.datarobot.com/blog/how-much-data-is-needed-to-train-a-good-model/
5. https://stackoverflow.com/questions/3788870/how-to-check-if-a-word-is-an-english-word-with-python
6. https://stackoverflow.com/questions/18057962/regex-pattern-including-all-special-characters
7. https://stackoverflow.com/questions/27016904/matplotlib-legends-in-subplot
8. https://stackoverflow.com/questions/61016110/plot-multiple-confusion-matrices-with-plot-confusion-matrix
9. https://stackoverflow.com/questions/61325314/how-to-change-plot-confusion-matrix-default-figure-size-in-sklearn-metrics-packa
10. https://spacy.io/usage/linguistic-features
11. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/
12. https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html
13. https://scikit-learn.org/dev/modules/model_evaluation.html#defining-your-scoring-strategy-from-metric-functions
14. https://elf11.github.io/2018/07/01/python-decision-trees-acm.html
15. https://www.freecodecamp.org/news/how-to-scrape-websites-with-python-2/
16. https://www.selenium.dev/documentation/webdriver/waits/

17. https://www.selenium.dev/documentation/webdriver/elements/finders/
18. https://stackoverflow.com/questions/73454187/how-to-get-tweets-in-twitter-using-selenium-in-python
19. https://pythonbasics.org/selenium-scroll-down/
20. https://javascript.info/size-and-scroll-window