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

Conducting Sentiment Analysis on Tweets

Abstract/Intro: (John)

The project's objective is **to find out what percentage of Twitter users believe climate change exists**. The main task is to compare how well different classification models perform. This is also a good project to learn how to conduct preprocessing for NLP. This project is significant since it is a useful way to gauge public opinion on this issue.

Dataset: (John)

The dataset is found from the website data world and consists of 6,090 tweets in total [1]. The tweets are all relevant to climate change or global warming [1]. The creator of the dataset used a platform called CrowdFlower to get workers to manually classify the tweets into three categories [2]. The tweets are classified as [1, 2].

The tweets in the original dataset are cut off and replaced with ellipses at the end. Therefore, some words are cut off, and context is lost.

Tweets have also been web-scraped using Selenium and Beautiful Soup by searching for "climate change" or "global warming" on Twitter. These tweets will be used to test how well the model works on modern tweets using a Web interface.

Preprocessing: (John)

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 and checking for alphanumeric characters. Lemmatization is the process of reducing words to their base words ("eating" to "eat") so that different words can all 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 that need to be split (such as “act|Brussels”).

The data will be 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.

The data now has around 6000 attributes in a sparse matrix format. We used TruncatedSVD as our main form of dimensionality reduction. This method is favorable for sparse matrices since it does not center the data on its mean.

One challenge in preprocessing is deciding which words to omit. Misspelled words in the tweets will be removed when in fact they may be useful; 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.

Another observation is that symbols or punctuation were encoded incorrectly when the dataset creator was getting the tweets. There is no general rule that can cover all the cases perfectly. Furthermore, it is questionable whether city or country names would contribute to the analysis. It is also challenging to figure out how to expand contractions, which can add negations and further hone the sentiment analysis.

Exploratory Data Analysis: (Sai Prasanna, Vennela)

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	Negative

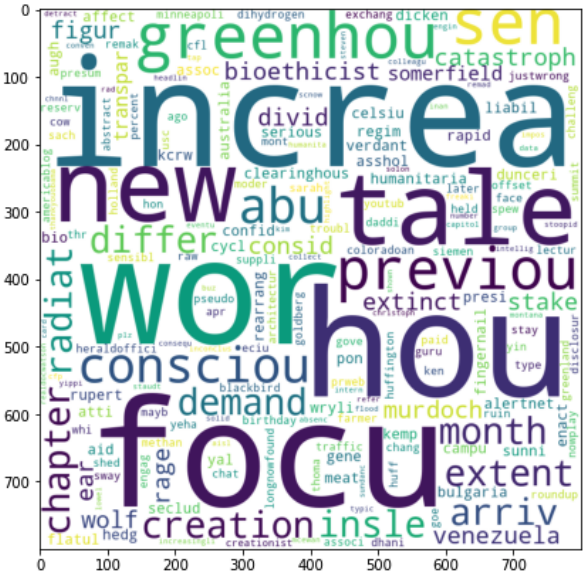
	hashtag	occurences
58	#climate	138.0
245	#tcot	69.0
119	#green	47.0
188	#p2	47.0
60	#climatechange	36.0
209	#saveterra	31.0
76	#earthday	24.0
80	#eco	23.0
113	#global	18.0
114	#globalwarming	15.0

	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

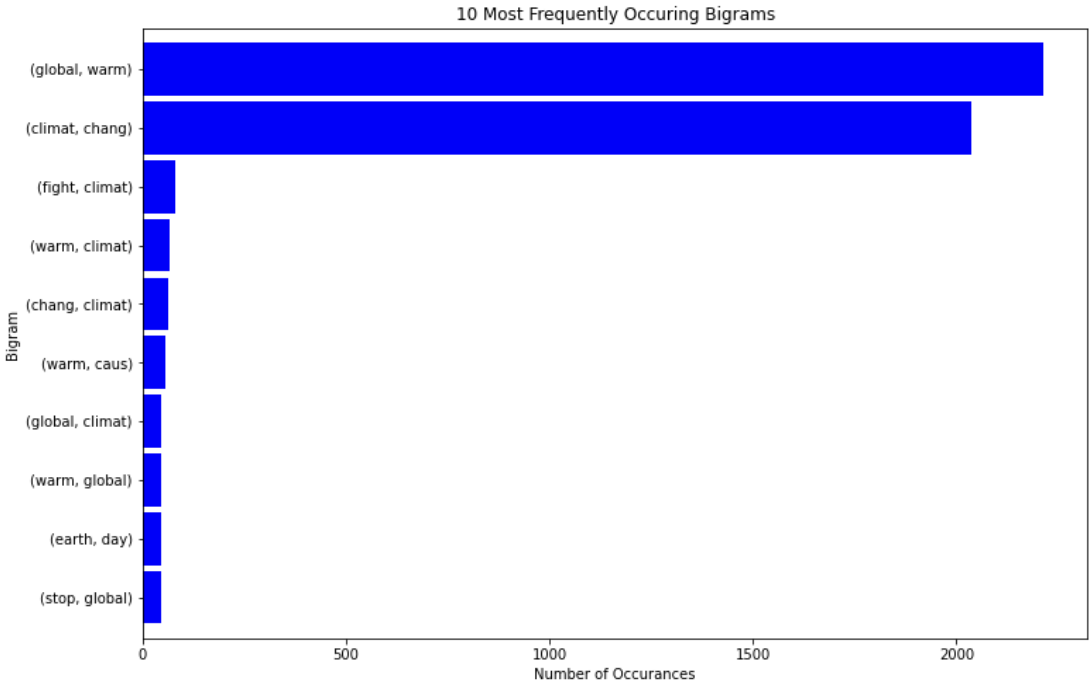
3) Bigger words indicate more frequent words, while smaller ones indicate less frequent words.

4) Words such as greenhou, focu, consciou, and previou contribute to positive tweets. These words have a correlation with climate change.

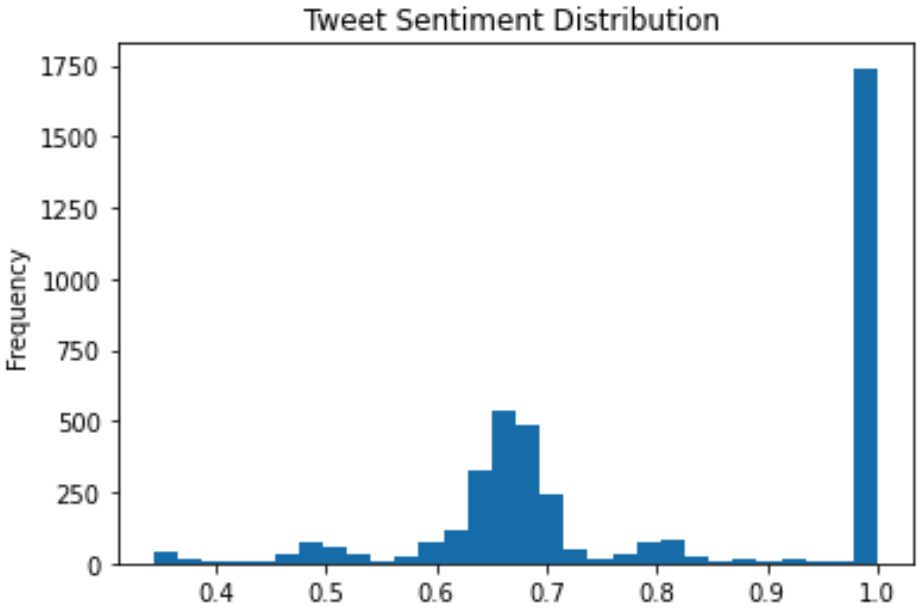
5) Words such as rage, tale, and month contribute to negative tweets.
These words belong to the group of words that do not believe in climate change. One interesting pattern is that the words “global warming” or “climate” change stay in the top twenty either way. Another interesting finding is that the words pertaining to links (bit.ly, TinyURL) are showing up as the most impactful prior to preprocessing. After preprocessing, words/phrases such as a greenhouse, conscious, and oil spill show up as the most impactful. This confirms that preprocessing helped in bringing out the most relevant words in the analysis.

Positive	Negative
	

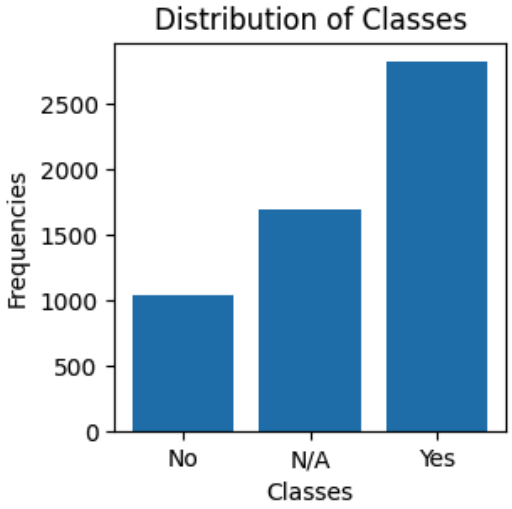
6) Word associations can be explored using N-grams, which provides further insight into understanding the context of the words.



7) 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.



8) We have an imbalance where there are more tweets believing in climate change compared to those that don't. This is further confirmed by the existence distribution histogram.



Method: (Sai Prasanna, Vennela, John)

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

on conducting binary classification. The best algorithm will be chosen by comparing their accuracies and confusion matrices.

Experiment/Analysis:

- **John**

1) The Perceptron is the first choice model to use. It achieves very high train accuracy but overfits. Therefore, it yields lower test accuracy. The reason is that the separating boundary may be very close to the points from either class, which increases the chance of test points being misclassified.
2) Logistic Regression is another basic linear model used for classification. It performs better compared to the Perceptron but still overfits.

Note: I conducted binary classification(combining N/A and no into one class), while Sai Prasanna and Sai Vennela conducted binary class classification with Yes, No (excluding N/A).

- **Sai Prasanna**

As mentioned above we trained our classification models on binary class (Yes, No) excluding N/A as the data suggests that the tweets are ambiguous. Removing the N/A label leads to class imbalance and will use the necessary approaches/methods to overcome this.

Methods to overcome imbalance dataset:

1. RandomOverSampling/ RandomUnderSampling - randomly duplicates minority class / randomly deletes samples in majority class. Both methods can lead to either overfitting the model or losing valuable information from the data.
2. SMOTE - Synthetic minority over-sampling technique - 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. [Drawing a line between the examples in feature space and drawing a new sample at a point along that line.]

1) KNN calculates the likelihood of a data point mapping to a respective class based on what class the nearest points belong to.
    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 (Because the majority of samples are from a single class) which results in misclassifying the minority class.

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 and efficient. They are scale invariant. Additionally, It is capable of capturing non-linear relationships.
    a) We achieve high train accuracy(99%) but comparatively less accuracy on tests (77%) indicating that the model overfitted on the data.
    b) Due to 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 the data when new data is added the tree structure completely changes.

3) Linear SVM is a simple algorithm that creates a line or hyperplane 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. LinearSVC minimizes the squared hinge loss while SVC minimizes the regular hinge loss. It has very less chance of overfitting.
    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, it performed far worse than a linear SVM. This indicates that the data might be linearly separatable.

    Linear SVM performs best among trained models (high accuracy, recall, precision, and F1 score).

- **Sai Vennela**
  Performed classification with two popular techniques for text classification naive bayes and Random forest :

1) Naïve Bayes comes under the statistical algorithm segment, and it 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 as naïve. One variant of Naive Bayes is Multivariate Event Model, which is also known as Multinomial Naïve Bayes is considere. The Multinomial Naïve Bayes uses a multinomial distribution.

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

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

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

The motives behind choosing this algorithm are:
1) It is very fast and uses less storage.
2) The dataset is not so large with a few thousand records with class labels.

Insights on Performance:

The testing accuracy is very poor when compared to the training accuracy. This is because of overfitting. Also, as the majority class is positive, the likelihood of classifying the data to positive class by naive bayes algorithm is high.

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

2) 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.

The reasons for considering Random Forest is:

There is a chance of overfitting of training data during Decision Trees classification, which can be resolved using random forest. As the features are randomly selected and the bagging method is applied.

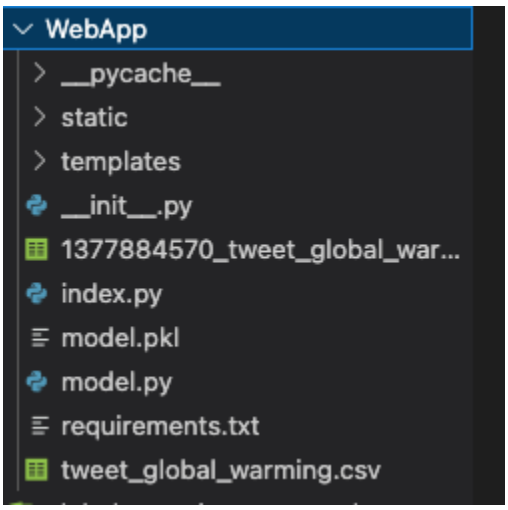
Insights on Performance:

Initially, the accuracy is high for the model with approx. 82% testing accuracy and 97% training accuracy , but that did not seem reliable metric for gauging the performance of the model. When considering other metrics, we observed a high score for precision and a low score for recall. This indicates that most of the values that are predicted by the classifier are not correct. This has occurred due to the imbalance in the dataset. To overcome this, **Random-Over Sampling((increasing the number of minority classes)** is performed on the dataset.

After performing random over-sampling, it is observed that there is significant improvement in the precision score and both precision and recall are closer to 86% and testing accuracy of 80%.

- **Web App Integration**:

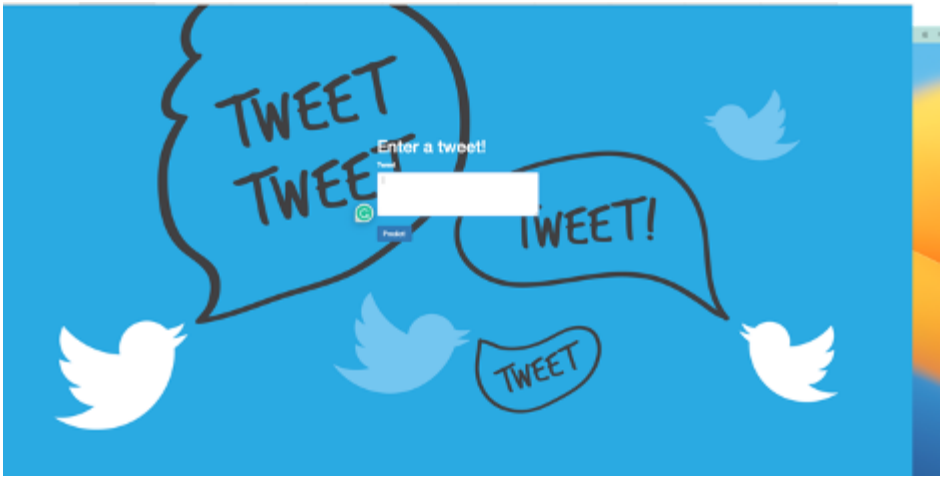
1) Install Flask with this command: pip3 install flask
2) Create the folder structure as follows:



3) Create a pickle file for the generated model.

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

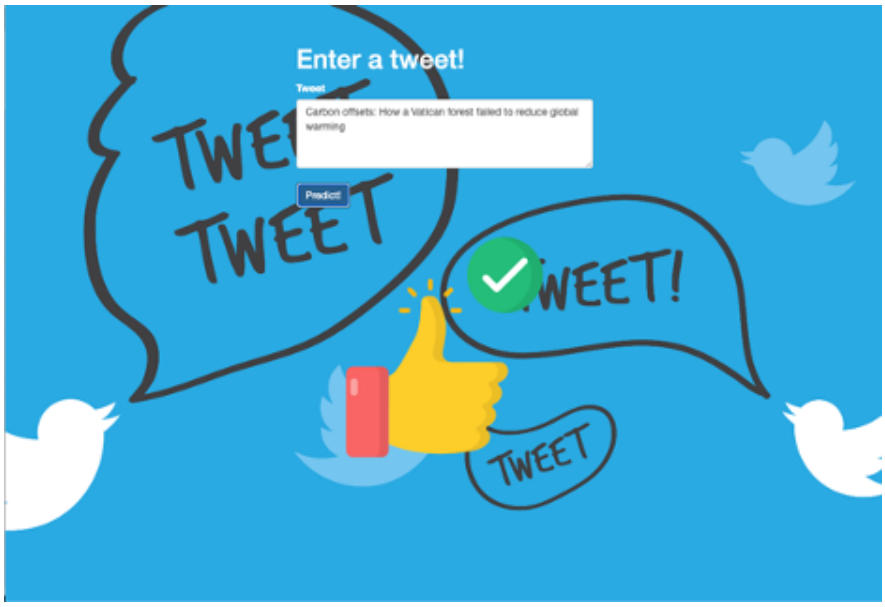
We have created a web page as follows for testing the scraped tweets from Twitter.



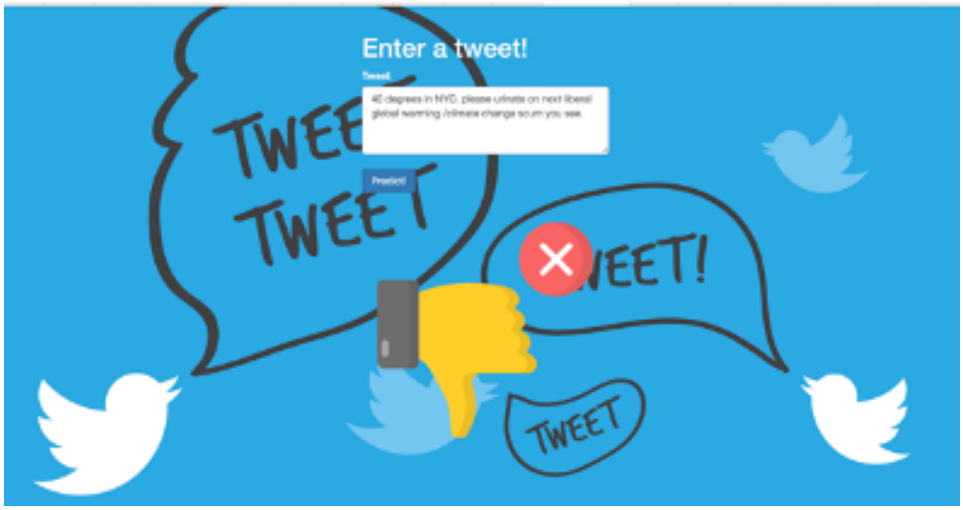
Using the two APIs created from Flask:

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 positive, the screen below is displayed.



Scenario 3: Added a few checks before processing the test tweet:

Sends an alert box:

Insights:

The model requires more data to give the sentiment of the tweet accurately.

- **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.

- **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.

Comparisons:
The performances of the various algorithms are evaluated on 5 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	0.68	0.74	0.61	N/A
Logistic Regression	0.71	0.72	0.74	N/A
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	0.60	0.86	0.71
Random Forest	0.80	0.86	0.85	0.86

Conclusion: (Sai Vennela)

In conclusion, this project aims to find the sentiment of users towards climate change using quality data obtained from Twitter and judgments obtained from Crowd Flower. With the analysis, it has been identified that the dataset is imbalanced with the majority inclined toward the positive class with 3029 records. So, random under-sampling and sampling are performed using SMOTE to ensure the classes are balanced 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 efficiency of seven classifying models was compared to get the accuracy scores over the preprocessed dataset. In this, SVM has given the highest F1-score with 86% followed by Random Forest with 86% and Decision Tree 84%. On the other hand, algorithms such as the simple perceptron, logistic regression, and Naive Bayes did not yield good results on the dataset due to overfitting. However, an increasing number of records in the current dataset would enhance the performance of these models.

Additionally, we have implemented a web interface using Flask that runs the SVM model. This takes the tweet as input and displays the sentiment of the tweet.

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