FOODEASE

smart relief solutions.

Tufts University • Project Report

Applied Data Science, Spring 2020

Abstract:

Victims of natural disasters have difficulty receiving and sharing fresh food and clean water, because leaders and first responders have difficulty interacting directly with or optimizing information directly from victims through the most accessible channels, e.g. Facebook, Twitter, WhatsApp. FoodEase is currently pursuing a project to optimize resource distribution for disaster relief, by asking three key questions:

- How can social media validate or disprove news of a crisis?
- How large is a crisis event (number of people affected)?
- How can the number of mentions indicate which locations are most at risk?

By asking these questions, we could use these data as a deciding factor on locations where we could have the most impact and/or understand better what demographic groups are most at risk.

In this project, we have used a series of datasets pulled from social media (Tweets, Geotag information) and then applied supervised learning in order to analyze if a social media tweet refers to an actual disaster event and where. The Python model uses a TfiD Vectorizer and a PassiveAggressiveClassifier to classify tweets into “Real” and “Fake” or “True” and “False.” Finally, two classification models were tested — Logistic Regression and Decision Tree — due to their strong application to binary evaluations like this. Ultimately, we found the logistic regression and TfiD vectorizer to provide the greatest accuracy. We further evaluated the accuracy of the model through a series of heatmaps, an interactive map of individual markers on prediction and actual layers, and an interactive map with automated clustering and color-coding by the relative amount of markers in a location region.

The resulting visualizations led to three insights. First, additional conditions are needed to ensure that the location of the user matches the location of the crisis mentioned in the tweet. Secondly, additional models or classification are needed to evaluate the timing of crisis events. Some tweets reference very recent (urgent) events while others reference recovery activities after an event has long passed. Finally, more robust training is needed to ensure that tweets predicted are indeed referencing real disasters, reducing the risk of false positives.

In conclusion, the project allowed us to confirm that we can validate or disprove news of a crisis through social media — and, with refinement, can use it as a proxy for impact estimates.

• • •

Special thanks to our advisor Kyle Monahan, Senior Data Science Specialist at Tufts University.

Introduction:

Globally, 28M people were newly displaced because of conflict and disaster in 2018. In the US, 1.3M were newly-displaced, while at least 850,000 people were displaced in the Southeast region during hurricane season. As natural hazards increase with global climate change, there is an increasing need for communities, NGOs, and governments to plan and coordinate responses that enable efficient and effective relief.

For this assignment, we analyzed over 10,000 tweets that used keywords such as "ablaze", "quarantine", and "pandemonium." To analyze and clean through this information, three related datasets were used (see references):

- Crowdflower's Disasters on Social Media
- Kaggle's Real or Not? NLP with Disaster Tweets (for training and testing)
- Kaggle's Disaster Tweets Geodata

The datasets consist of social media tweets (with locations). We used supervised learning to analyze if a tweet refers to an actual disaster and where: the model uses a TfiDf Vectorizer and a PassiveAggressiveClassifier to classify tweets into "Real" and "Fake" or "True" and "False."

Method:

01. Data Cleaning

In order to test the accuracy of our algorithm on information not used to train on, we choose to split our information into a training and validation set. This training set we will use to fit our model, and the validation set will be used to predict how our algorithm will work on the test set:

- x_train, y_train will be our training dataset
- x_valid, y_valid will be our validation set

As well, note that the input to our function is x_train_and_validation.text.values. This provides a numpy array with all the text entries of the data. This does not reorder the rows, so the y and x arrays are still in sync. Also, we are keeping `x_valid_df` and `x_train_df` as data frames so we can use the validation set locations later. Here we reduce them to just their text values.

02. Data Modeling

The classification algorithms we covered in class took numeric data and used that information to classify different entries based on which numbers were the most meaningful. Unfortunately when it comes to sentences or strings like the tweets we're looking at in this project, the literal strings of text individually contain very little classifiable meaning.

a. Natural Language Processing

For this project we're using vectorizers for natural language processing. Vectorizors take a list of inputs and based on that list comes up with a list of useful numbers for each entry that help differentiate them in a meaningful way. To begin, we defined a function to take a training and validation dataset, run those against a model and vectorizer, and return a set of basic analytics of the model on the dataset:

- train_log_loss — Log loss of the model on the training set
- train_score — Accuracy score of the model on the training set
- valid_log_loss — Log loss of the model on the validation set
- valid_score — Accuracy score of the model on the validation set

The function returns a 2D array as the model_confusion_matrix in the form [[TN, FP], [FN, TP]]:

- TN stands for True Negative,
- FP is False Positive,
- FN is False Negative, and
- TP is True Positive.

The matrix also presents a precision score (the ratio of TP to the sum of TP and FN) and recall score (the ratio of TP to the sum of TP and FP).

b. Sentence Vectorizers

To classify our disaster tweets, we chose to test out two different vectorizers and see how they handle the dataset against two different classification methods as well. The first vectorizer we chose to look at was the term frequency - or an inverse document frequency vectorizer. This works based on the principle that words that show up a good deal are useful, but not useful if they show up too much.[1] The second vectorizer we chose to use is a simpler vectorizer — a count vectorizer — as it counts the number of times each word appears.

c. Classification Models

The two classification models used were a logistic regression model — which we have explored in class — and a decision tree classification model, which works well for this type of binary problem. With logistic regression, we are able to use a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). Hence, our logistic regression model predicts P(Y=1) as a function of X. Similarly, a decision tree is a simple representation for classifying examples with

[1] en.wikipedia.org/wiki/Tf%E2%80%93idf

supervised learning. the data is continuously split into subsets according to a certain parameter by recursive partitioning.

Both classification models were then applied to the two sentence vectorizers (inverse document frequency vectorizer and count vectorizer), and we were able to compare the results on three key questions:

- What are the differences between training score and validation score?
- How do they compare?
- What do we notice about the confusion matrix (e.g. precision and recall)?

03. Data Visualization

The classification model that provided the most convincing results was the logistic regression classifier using the term frequency - inverse document frequency classifier. Using the results we got from that we'll do our best to figure out where we believe the disasters are based on only those tweets, and only the ones that we believe are disasters. We began by defining a function called locations_to_heatmap, which converts a 2D array of latitude and longitude values into digitized versions.

More so, in order to compare the quality of the disaster prediction against where the disasters actually were, we create two different datasets: one based on the actual disaster locations, and the other based on where the tweets that we predicted were disasters were located. We developed a contrasting heatmap of the locations found through tweets referring to actual disaster events.

Contrasting these two datasets, we developed three visuals:

- Pixelated heatmap for both predicted and actual disasters globally, in the United States, and in the state of Florida.
- ESRI (GIS) map with markers for individual 'predicted' tweets that overlay the markers for individual 'real disaster' tweets.
- ESRI (GIS) map that clusters individual 'real disaster' tweets by geographic zone and indicates the size of clusters compared to other zones' clusters.

Results:

01. Logistic Regression Classification

The logistic regression gave the best predictions using the term frequency vectorizer, which matched our prediction: This incorporates potentially more meaningful information than simply the count of each word in the tweets. Using those three key questions, we arrived at a few insights for the pairings (See below).

Of course, the training score is better than the validation score since the model was taught on the training data. The confusion matrix counts were comparable, with the logistic regression on term frequency producing 450 true positives, 204 false negatives, 772 true negatives, and 97 false positives. In contrast, the count vectorizer produced 439 true positives, 215 false negatives, 778 true negatives, and 91 false positives. Also, the inverse document frequency classifier produced a recall of 0.6881 and precision of 0.8227, while the count vectorizer produced a recall of 0.6713 and precision of 0.8283. Hence, the two models appear to be comparable in precision and recall.

Overall, the logistic regression on term frequency produced a training score of 0.874 and a validation score of 0.802, while the logistic regression on the count vectorizer produced a training score of 0.995 and validation score of 0.799. Therefore, we concluded that the best predictions under logistic regression use the term frequency vectorizer.

```
logistic regression on term frequency - inverse document frequency
        log loss on training set 0.396
        training score 0.874
        log_loss on validation set 0.471
        validation score 0.802

        confusion matrix count: TP:  450, FN:  204, TN:  772, FP:   97
        recall: 0.6881 | precision: 0.8227 | F1 score: 0.3747

 logistic regression on count vector
         log loss on training set 0.083
         training score 0.995
         log_loss on validation set 0.452
         validation score 0.799

         confusion matrix count: TP:  439, FN:  215, TN:  778, FP:   91
         recall: 0.6713 | precision: 0.8283 | F1 score: 0.3708
```

02. Decision Tree Classification

For the second classification model (see below), the confusion matrix for the decision on term frequency vectorizer produced 282 true positives, 372 false negatives, 800 true negatives, and 69 false positives. Yet, the count vectorizer produced 312 true positives, 242 false negatives, 796 true negatives, and 73 false positives. Hence, it appears that the decision tree on count vectorizer classification is a slightly stronger model.

Additionally, the decision tree classification on term frequency produced a training score of 0.814 and a validation score of 0.710, while the decision tree classification on count vector produced a training score of 0.821 and a validation score of 0.728. The inverse document frequency vectorizer also produced a recall of 0.4312 and precision of 0.8034, and the count vector

vectorizer produced a recall of 0.4771 and precision of 0.8104. Both pairs' scores are weaker than the recall and precision of the logistic regression classification. Overall, these metrics are slightly inferior to those produced by the logistic regression classification.

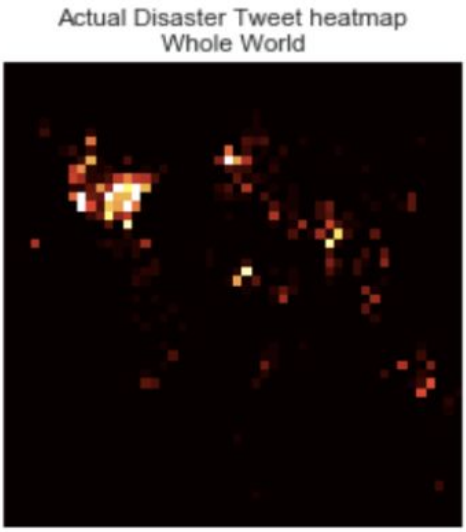
```
decision tree on term frequency - inverse document frequency
        log loss on training set 0.370
        training score 0.814
        log_loss on validation set 2.907
        validation score 0.710

        confusion matrix count: TP:  282, FN:  372, TN:  800, FP:   69
        recall: 0.4312 | precision: 0.8034 | F1 score: 0.2806


decision tree on count vector
        log loss on training set 0.379
        training score 0.821
        log_loss on validation set 2.632
        validation score 0.728

        confusion matrix count: TP:  312, FN:  342, TN:  796, FP:   73
        recall: 0.4771 | precision: 0.8104 | F1 score: 0.3003
```

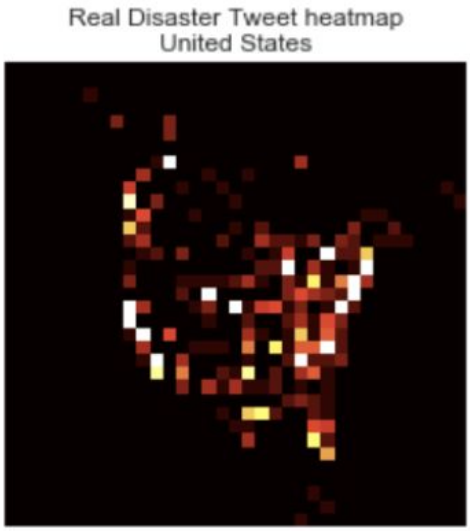
03. Geolocation Prediction and Visualization

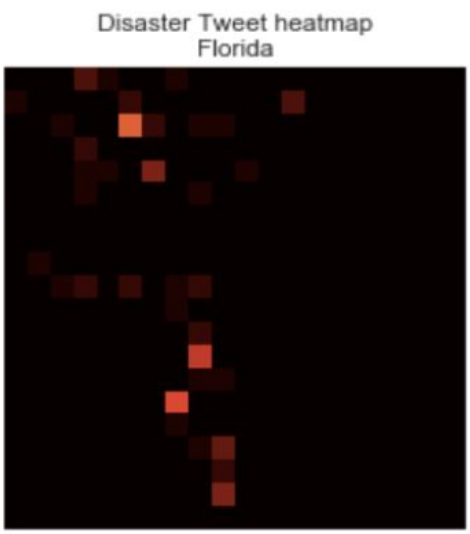
Using one dataset is based on the actual disaster locations and another based on where the tweets that we predicted were disasters were located, we visualized three subsets of the data on a heat map and compared each pairing: The World, United States, and the state of Florida. (See below.)

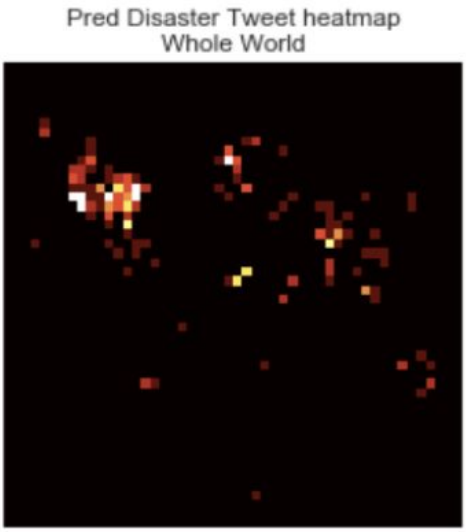
Additionally, using the Leaflet library and folium plugins, we were able to create two interactive maps that show at high-resolution where tweets were made. The first map (below) combined the predicted disaster tweets in one layer (blue pins) and the real disaster tweets (purple dots) in another. By toggling between the layers, we grasp the amount of real disasters that were missed by the model. The second map (below) clusters and color-codes real disaster tweets by the relative region they are located in. Such a visual provides quantitative and spatial information that could improve estimates for unmet need or priority ranking between crisis events.

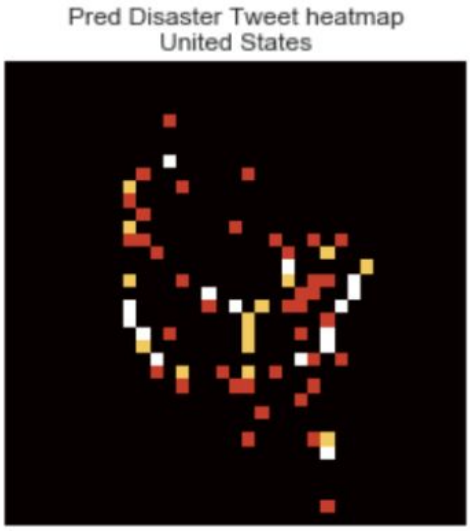
Heat Maps — Subset Locations of Tweets by Prediction or Real

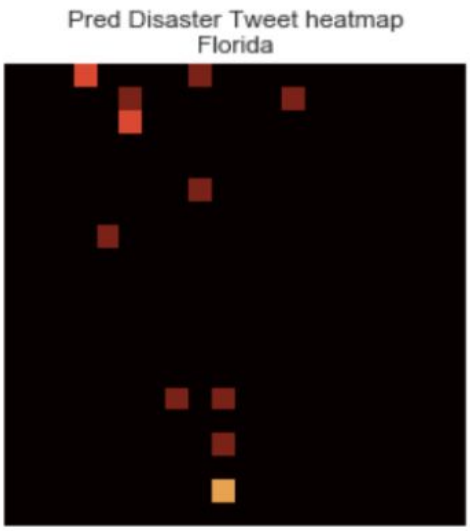


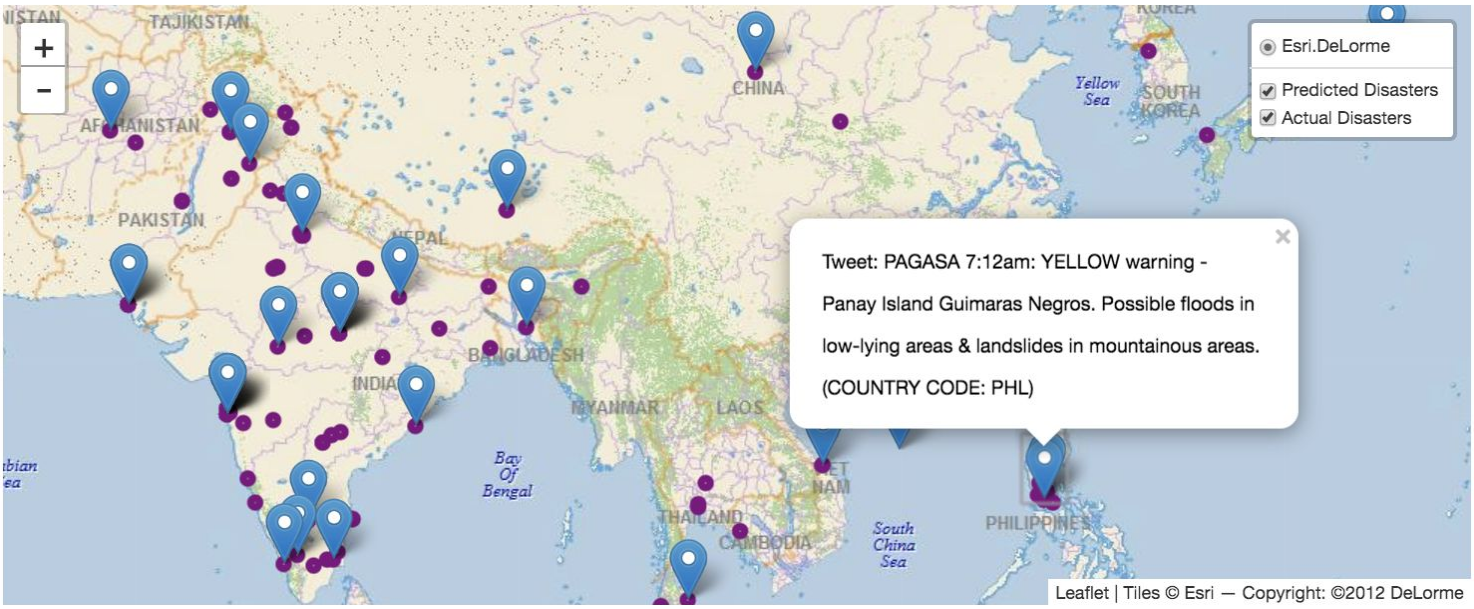




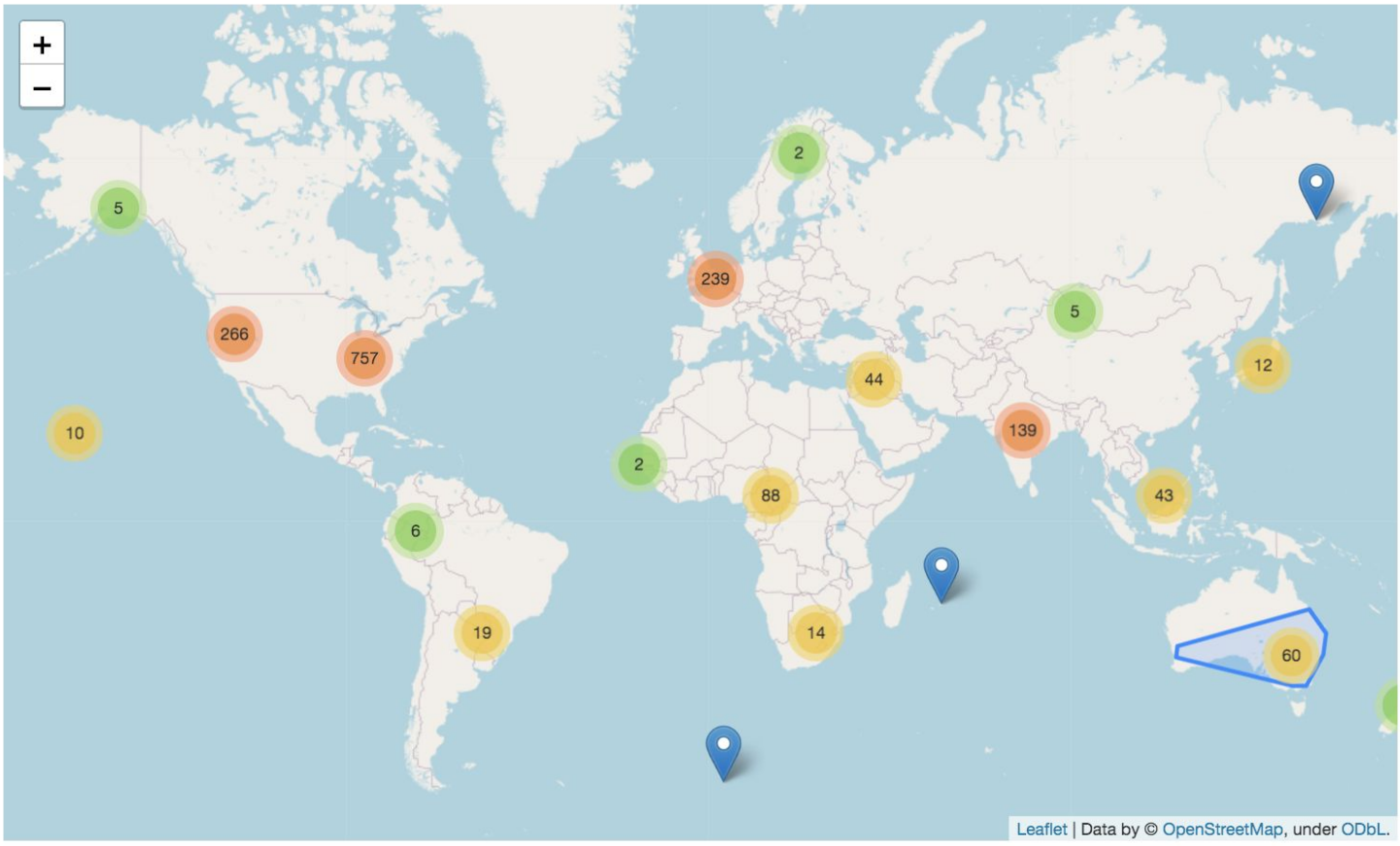








Interactive No. 1 — Locations of Tweets by Prediction or Real Dataset



Interactive No. 2 — Clusters of Real Disaster Tweets by Regional Location

Conclusion

As a result of this project and early prototype, the FoodEase team was able to answer those beginning questions. First, we confirmed that we can — by using logistic regression to predict term frequency vectorizer on social media and contrasting it to actual data — we can validate or disprove news of a crisis. Secondly, it is also possible to visualize the size of a crisis through clustering by geographic zones. And, finally, we can also identify which locations are most at risk if tied with more data sources that indicate geographic economic and structural disparities.

The three visualizations allowed us to spatially compare the accuracy of our model at different levels. The first series of heat maps indicated that our model matches the general shape of the real disaster data but lacked density. To verify this inference, we looked further at the two interactive maps. As a result, we quickly found the location of the user might not match the location of the crisis mentioned in the tweet. We were also able to read tweets and understand that our model doesn't provide insight into timing. Additionally, a few tweets predicted as disaster did not seem relevant at all, meaning that false positives are still a risk.

Next Steps:

01. Reverse Geocoding

Reverse geocoding is the process of back-coding a point location with coordinates (latitude, longitude) to a readable address or place name. Because of rate limits, oftentimes the function we have defined for finding addresses of point locations is met with a session time from the free API through OpenStreetMap.[2] For robust datasets, we would have to pay a premium for the OSM or other geocoding services. From there, it is possible that FoodEase could begin exploring algorithms for route optimizations between different physical addresses and infrastructure.

02. Application for COVID-19 or Pilot

We would like to pilot this in the context of an actual food disaster, particularly during this COVID-19 pandemic. We would like to use this model to collect and process real-time data from tweets to identify where food resources are needed most. It would be interesting on how we could incorporate more formalized data sources that currently track coronavirus cases at a national or regional level, like John Hopkins Public Health.[3]

[2] operations.osmfoundation.org/policies/nominatim
[3] data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases

References:

01. Sources of Data

original.csv — data.world/crowdflower/disasters-on-social-media
This is the original tweet data that was used to generate the kaggle dataset. Contributors looked at over 10,000 tweets culled with a variety of searches like "ablaze", "quarantine", and "pandemonium", then noted whether the tweet referred to a disaster event (as opposed to a joke with the word or a movie review or something non-disastrous).

train.csv and test.csv — kaggle.com/c/nlp-getting-started/data
The direct translation of the information in original.csv such that it's formatted a bit more easily and can be submitted for the kaggle competition. Each sample in the train and test set has the following information: the text of a tweet; a keyword from that tweet (if available); and the location the tweet was sent from (if available).

geodata.csv — kaggle.com/frednavruzov/disaster-tweets-geodata
The tweet dataset from original.csv but with actual geo locations that have been cleaned up. Each sample has the following information: initial tweet id, latitude coordinate, longitude coordinate, ISO 3166-1 alpha-3 country code, and city name.

02. Python Libraries

Numpy is the fundamental package for scientific computing, e.g. matrix manipulation.

Pandas offers data structures and operations, primarily for dataset management.

Matplotlib is a comprehensive library for creating static, animated, and interactive visuals.

Scipy provides user-friendly and efficient numerical routines (for scientific computing).

Sklearn features various classification, regression and clustering algorithms. It offers metrics functions like import log_loss and confusion_matrix; vectorizers; and splitting train and test data.

Ipyleaflet is an interactive widgets library used for map visualizations like marker layers and clusters. Combined with folium and ipywidgets , the library is often used for HTML dashboards.

Geopy locates the coordinates of addresses, cities,, and landmarks using 3rd-party geocoders.

Geopandas extends the datatypes used by pandas to allow spatial operations on geometric types.

Team:

Christina Holman

is currently in the MS in Innovation and Management, with a Bachelor's in Economics and a Certificate of Engineering: Design, focused on international development and human-centered design. Additionally, Christina is an Analyst for the global corporate innovation consultancy at the Cambridge Innovation Center. • christina.holman@tufts.edu

Earn Khunpinit

is currently in the MS in Innovation and Management, with a Bachelor's in Operation Management. She has assisted manufacturing firms in improving water management systems and the Royal Thai Navy SEAL during severe flooding. • tanyanit.khunpinit@tufts.edu

Lee Ann Song

is currently in the MS in Innovation and Management, with a Bachelor's in Neuroscience and a Minor in Global Health and Health Policy. She completed the culinary arts program at Boston University and also volunteered on humanitarian missions in Tanzania, China, Guatemala, Costa Rica, and Haiti. • lee_ann.song@tufts.edu

Sai Wang

is currently in the MS in innovation and Management, with a Bachelor's in Civil Engineering. He has experience in structural design, infrastructure development, and urban planning. Sai also volunteered on humanitarian missions in China and Sri Lanka. • sai.wang@tufts.edu