Bill LaVista, Lucas Valentich, Victoria Warner, Brian Yoder
FEMA Disaster Declaration
13 September 2020

Predicting FEMA Disaster Declarations

Github: https://github.com/georgetown-analytics/Los-Ninos-Disaster-Declaration
Presentation: https://drive.google.com/file/d/1OHlADXrsgG7aNgh90IKIUkBASJyaf5Ec/view?usp=sharing

Introduction

The number and magnitude of natural disasters declared as federal emergencies in the United States during the past few years has increased precipitously. One estimate puts the amount of federal funds spent on natural disasters by the Federal Emergency Management Agency (FEMA) during the last four years to exceed the amount of federal funds spent on all national disasters during the prior forty years.

A majority of large, costly, natural disasters involve rainfall and flooding, especially in coastal areas. Currently, there is no known federal system or process that links real-time precipitation data to predict which counties will be included in a disaster declaration or to estimate the financial magnitude of a disaster with real-time weather data using data science.

Here lies an opportunity to use real-time precipitation data and data science to create predictive models that forecast which counties will be included in a disaster declaration and the magnitude of damage. Accurate, real-time predictions will give emergency managers additional hours and days to better allocate emergency resources during the outset of a disaster.

Research Question
Can we predict the likelihood of counties in Florida to enact disaster declarations for weather-related disasters in real-time based on past precipitation and disaster declaration data?

Hypothesis
Precipitation will be the primary indicator of a weather-related disaster declaration in Florida (i.e. hurricane, severe storm(s)). There is a correlation between precipitation, time, and disaster declarations.

Data Ingestion

We identified promising data sources from several Federal agencies to use in our project, including Census, FEMA, NASA, NOAA and USGS. After careful group deliberation, we decided to go with three data sources to keep the project manageable: NASA precipitation data, FEMA disaster declaration data, and Census data.

FEMA data – The FEMA dataset came from OpenFema and included summaries of disaster declarations during the past 40 years across the United States. Data included dates in which disasters were declared, the type of disaster (ie. hurricane, severe storm(s), fire, etc.), amount of

Federal funds spent on a disaster at the state level, the number of individual registrations for FEMA assistance, and much more. It also provided Booleans on whether or not certain provisions were enacted, like the Individuals and Household program, Public Assistance program, and more. Data was in the form of a CSV file, which was ingested using an API (https://www.fema.gov/about/reports-and-data/openfema).

NASA data – NASA's EarthData (https://urs.earthdata.nasa.gov) contains weather/climate related data that NASA collects and maintains. Accessing the data requires registration, and downloading the data requires some additional set-up using cURL or Wget to configure NASA's EarthData Login authentication. We decided to use the IMERG one-day dataset that provides precipitation estimates at 0.1 degrees using multiple satellites to estimate rainfall for the entire world. IMERG satellite estimates go back to the year 2000, providing us with 20 years of data.

Using NASA data is advantageous over NOAA data. NASA has 20 years of precipitation estimates that are created through measurements taken by satellites, so it provides consistent precipitation estimates over many years for the world by 1 tenth of a degree. NOAA had sporadic weather data on its public facing website that is collected primarily through land based weather stations. During severe storms, such as a hurricane, land-based weather stations can be inundated with water and do not provide precipitation data when precipitation is at its peak. Since NASA precipitation is collected through satellite estimates, data collection is not compromised during peak precipitation events. Also, the NASA data set was complete and did not require data cleaning or imputation of missing values.

We began downloading data from the NASA website, which was approximately 7500 HDF5 files. HDF5 is a hierarchical data format that is often used in the hard sciences and is good for storing data that includes multiple times and locations. It also compresses data so storage of HDF5 files requires less space.

As the HDF5 files provided too high of a resolution for the team to reasonably store, process and analyze the dataset, we decided to use daily averages from NASA that are in the NC4 file type. Similar to the HDF5, the data is compressed and organized.

Census data - Census data was used to link FEMA and NASA data. The Census data included polygons of counties as well as county FIPS codes.

We stored our data in a Digital Ocean remote server where we also built a PostgreSQL database for ease of access and use. In this stage, we also determined that we would focus solely on Florida and precipitation events, given the great quantity of data needed from NASA, storage limitations, and the prevalence of these events in Florida.

Munging and Wrangling

We used a python library to access multiple tables within the hierarchical data structure for the NASA data. The tables we accessed included latitude, longitude, time, and daily precipitation estimates and other information that was not important to this project. Latitude and longitude

were stored as arrays and daily precipitation estimates for the entire world were saved as a matrix. Time was saved as a single value.

We flattened the precipitation matrix and figured out which row matched with the appropriate latitude and longitude and created a date column into an array that repeated. The team repeated the latitude and longitude arrays so the arrays contained the same number of rows as precipitation and then concatenated the four arrays into a 4 by 6 million matrix. We identified the latitude and longitude for Florida and sliced that section out and then appended files for the daily precipitation into a large dataset which includes daily precipitation for each tenth of degree for the state of Florida for twenty years.

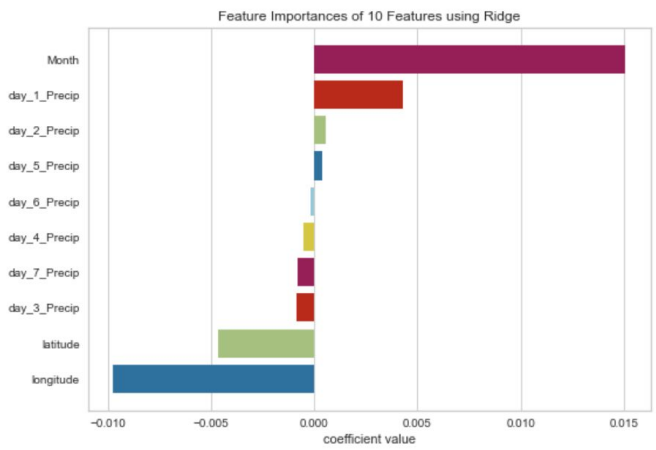
The latitude and longitude NASA data was paired with the Census data that matched each datapoint to a county in Florida and included a FIPS code, a numeric value from Census for each county in the United States is designated. Including the FIPS code to the NASA data allowed us to index with the FEMA data.

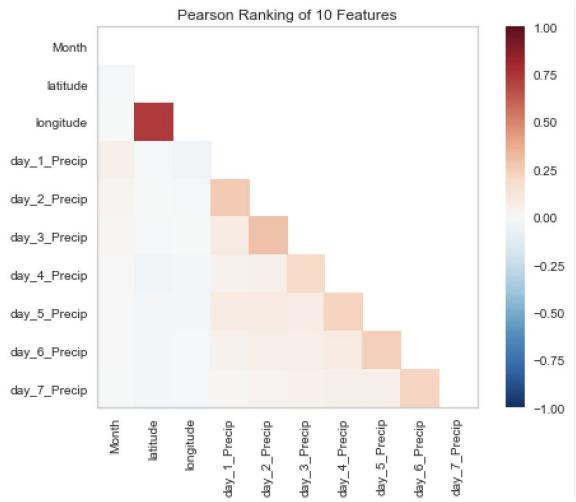
We converted dates in the FEMA data to POSIX time so we could easily link when disasters were declared to the corresponding time in the NASA data.

Computation and Analysis

Exploring the data, we found that disasters had been increasing over time, especially hurricanes. Severe storms, however, were on the decline. The average length of a disaster as identified by FEMA's disaster start and end dates was 20.83 days. However, this was greatly skewed right, or positively skewed, as the median was 13.0 days. Finally, we calculated the number of days after the disaster began for the declaration to be enforced. Hurricanes are usually declared soon after they occur with a relatively small distribution and a median at 3.0 days. The distribution of the declarations as a result of severe storms is much wider, with a median 9.5 days after the start of the disaster. This helped to inform our models in deciding the days of precipitation that should be taken into consideration.

At first, we used the mean value of the last 21 days as features, but concluded that this was not appropriate given our findings in our exploration of when disasters are declared. Therefore, we decided to use the last seven days of precipitation. As it can be seen in the graph below "Month" is the most relevant feature, followed by "day 1 precip". On the other hand "latitude" and "longitude" were not relevant but we used them because by predicting whether or not a latitude or longitude will declare a disaster we can aggregate up to the county level, meaning we can tolerate a less accurate model, and still get accurate results.





Module and application

We worked on two different classification models to shed light not only on the occurrence of disasters in Florida, but also on the types of assistance provided. The first classification model was a two-class classification, in which we wanted to predict disaster declarations in Florida counties. The second model focused on identifying whether the Individuals and Households Provision would be enacted in the instance of a disaster declaration.

Disaster Classification

To determine the most successful model we focused on the accuracy of the training data set, the accuracy of the test data set, and the f1-score for both outcomes. We first tried different models to see the results on each of them. The models we tried were: LogisticRegression, DecisionTree, RandomForest, GradientBoosting, KNeighbors, LinearDiscrimination, and GaussianNB.

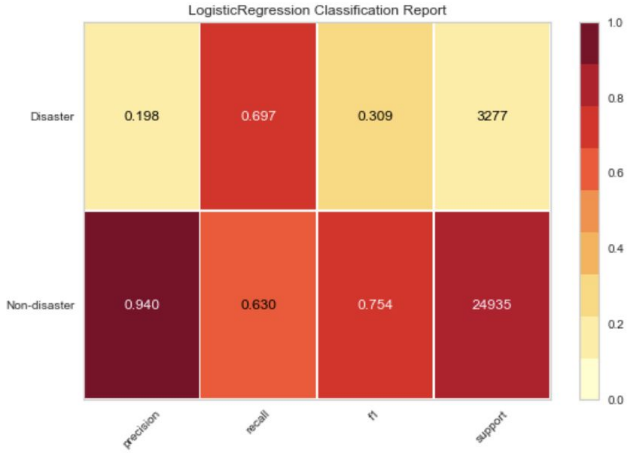
In the beginning, the main problem that we got was the imbalance of our data. We had 99% of the data that were non-disaster declaration rows. Therefore, all our models gave as 100% accuracy because on the training and test set there was only data from non-disaster declarations. To solve that problem, we tried two solutions. One was to select a sample from the non-disaster declaration and use all the disaster declarations. The other solution for some of the modules was to use a parameter “class_weight = balanced” to have a balance between the two outcomes.

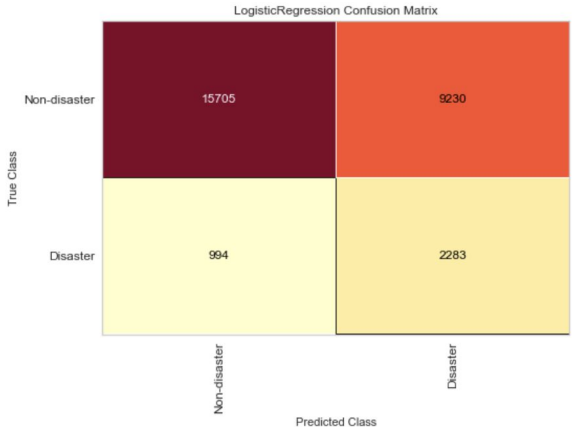
After we balanced the data, we started to get more reasonable values for accuracy and f1-scores. We set a baseline from our first try that was using a sample of 50,000 for the non-disaster days. The baseline was: 0.82 of test accuracy, and a f1-score of 0.89 for non disaster declaration and 0.31 for disaster declaration. The features that we used that time were: Geometry (longitude and latitude), County, and Precipitation of that day. As the f1-score of the disaster declaration was so low, we decided to try adding more features in two different ways. The first one was adding the mean precipitation of the last 20 days for each day. The second one was to add all 21 days of prior precipitation. We got better results from the second case rather than the first one.

Although we tried seven different models, there were only three that had a good performance and where we focused our analysis. These modules were: LogisticRegression, DecisionTree, and RandomForest.

- Logistic Regression

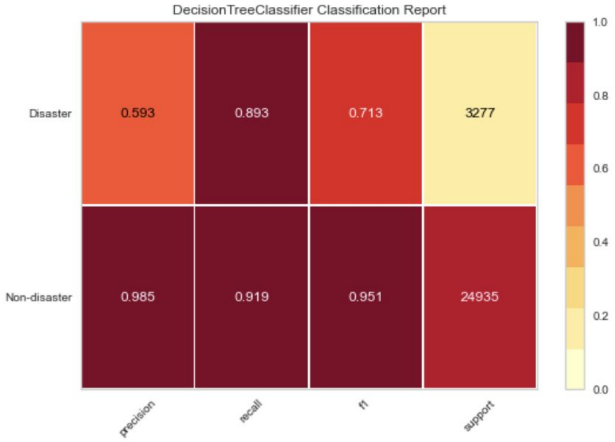
For this model we got an accuracy of 0.63 on the training and test set. As we can see in the Classification report, the f1-score of the non-disaster declaration was 0.754 and 0.309 for the disaster declaration. This was above our baseline, but it was not the f1-score that we were expecting.

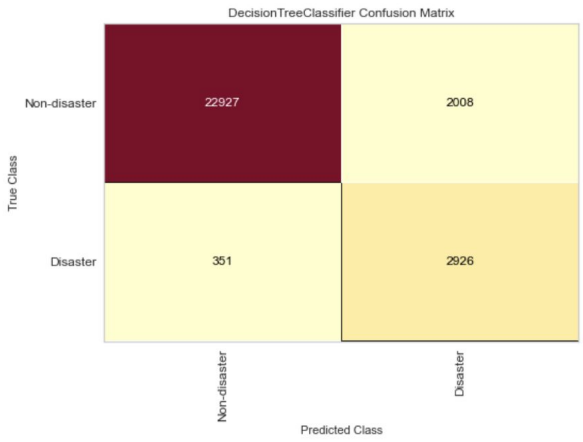




- Decision Tree

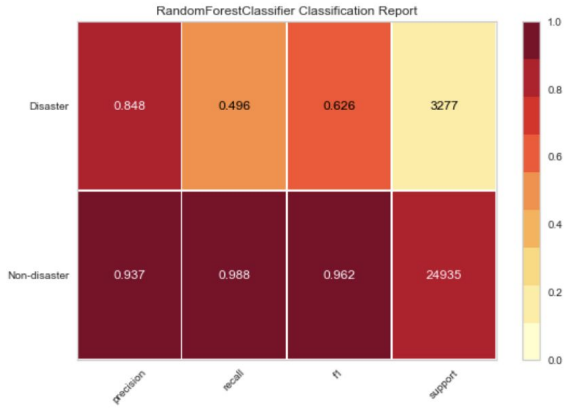
For this model we got an accuracy of 0.94 on the training set and 0.92 on the test set. As we can see in the Classification report, the f1-score of the non-disaster declaration was 0.951 and 0.713 for the disaster declaration.

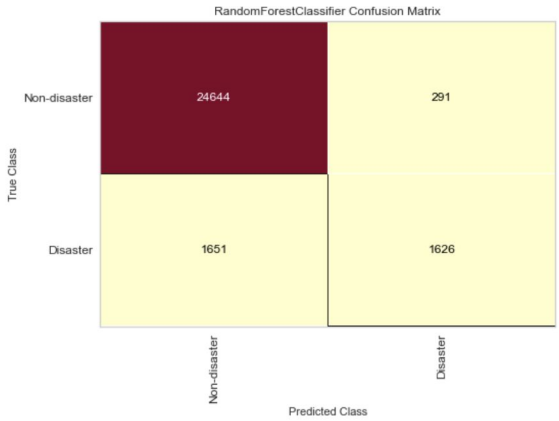




- Random Forest

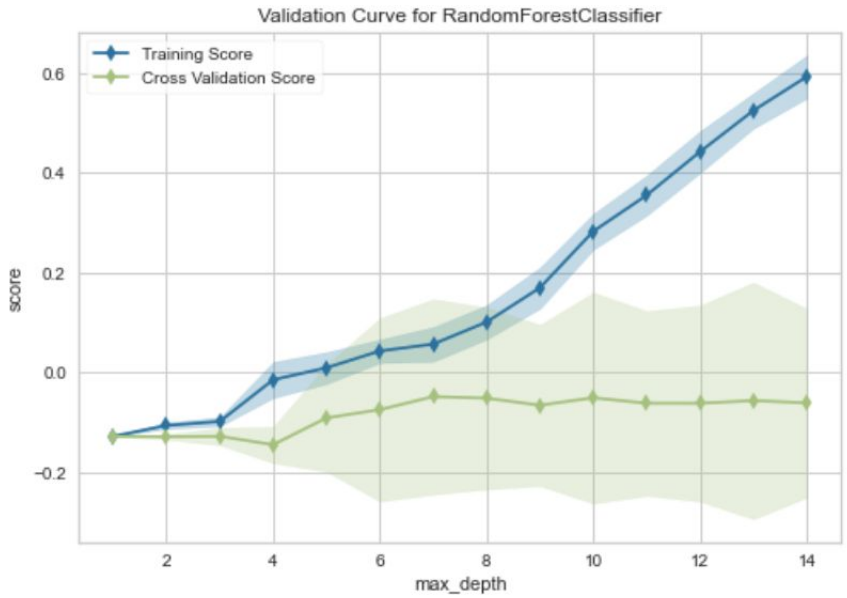
For this model we got an accuracy of 0.943 on the training set and 0.931 on the test set. As we can see in the Classification report, the f1-score of the non-disaster declaration was 0.962 and 0.626 for the disaster declaration.





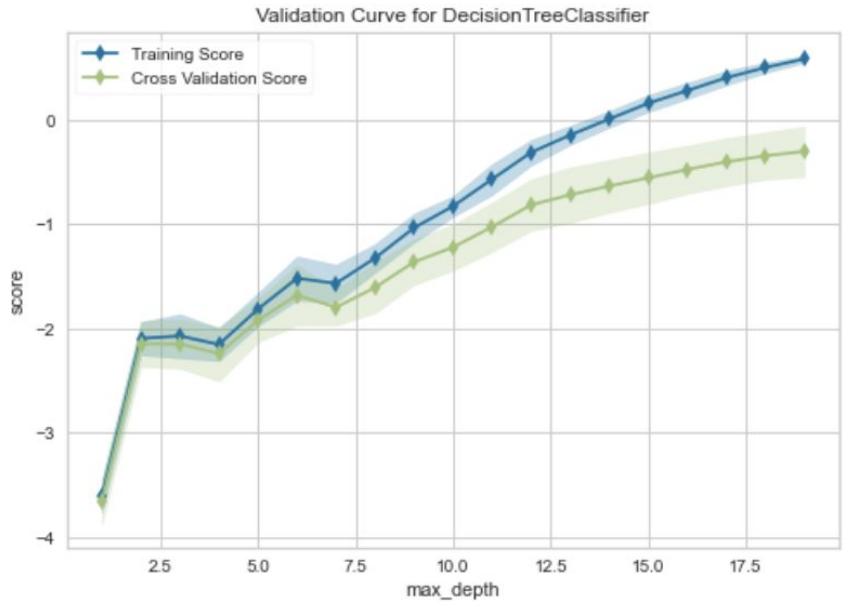
Model Evaluation

- Random Forest : As it is shown in the chart below, there is a big difference between the training score line and the cross validation score line. This means that the model is overfitted. That is one of the reasons why we did not choose this model. Another reason is that the recall value was too low (0.49) , in other words, it did not predict half of the disaster declaration.



- Logistic Regression: This was not the best module because it has a low accuracy on the training set, test set, and f1-score for disasters.

- Decision Tree: This was the best module because it has the best accuracy on the recall and f1-score for disasters. Moreover, in the graph below, we can see that there is a small gap between the training score line, and the cross validation line meaning that the model is neither overfitted nor underfitted.

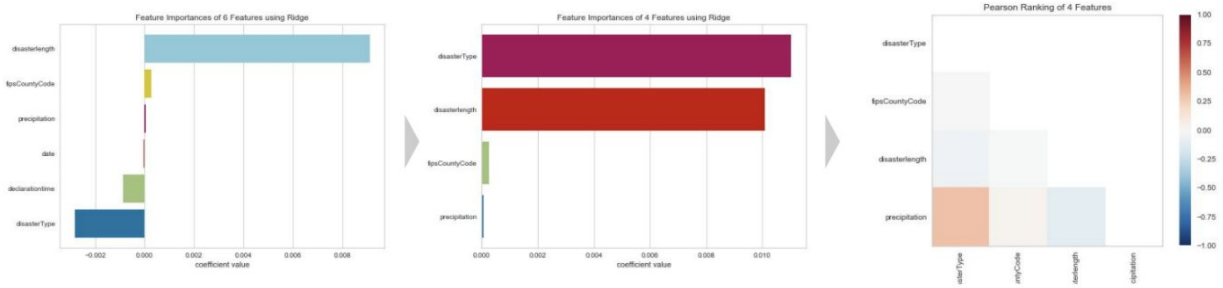


Individuals and Households Provision Enactment Prediction

The Individuals and Households Housing Assistance Provision includes both financial housing (i.e. rental assistance, home repair) and direct housing assistance (i.e. temporary housing units, permanent housing construction). The FEMA dataset that we used identifies the enactment of this provision as a boolean. Therefore, we pursued a classification model.

First, we created two features to add to the model to analyze. First, we calculated the length of the disaster as reported from FEMA. We then calculated when the disaster was declared as the number of days after the start of the disaster. We merged the datasets using the county code and the initial incident/disaster date. One possibility to improve the model would be to include all the precipitation data from the initial incident date to the end of the incident. We then performed several transformations on the data, including utilizing a label encoder to assign a unique value to the categorical variable *disaster type*. This is due to the fact that in our exploratory data analysis, we saw differences in the distribution of features based on disaster type, so we wanted to explore it as a potential feature.

We then went through extensive feature selection using regularization techniques, like LASSO, ridge regression, and ElasticNet. We also used transformer methods to select features based on their subsequent importance weights. The Lasso transformer method identified disaster start date, disaster length, and precipitation as key features. The Ridge and ElasticNet transformer methods both identified disaster length. We also completed a correlation matrix of our features to evaluate the correlations between each of the features that we were evaluating. Ultimately, this narrowed our features down from 6 to 4. We then created a two-dimensional ranking of the features utilizing a ranking algorithm that took into account the different pairs of features at the same time. This model utilized the Pearson correlation score to detect collinear relationships.



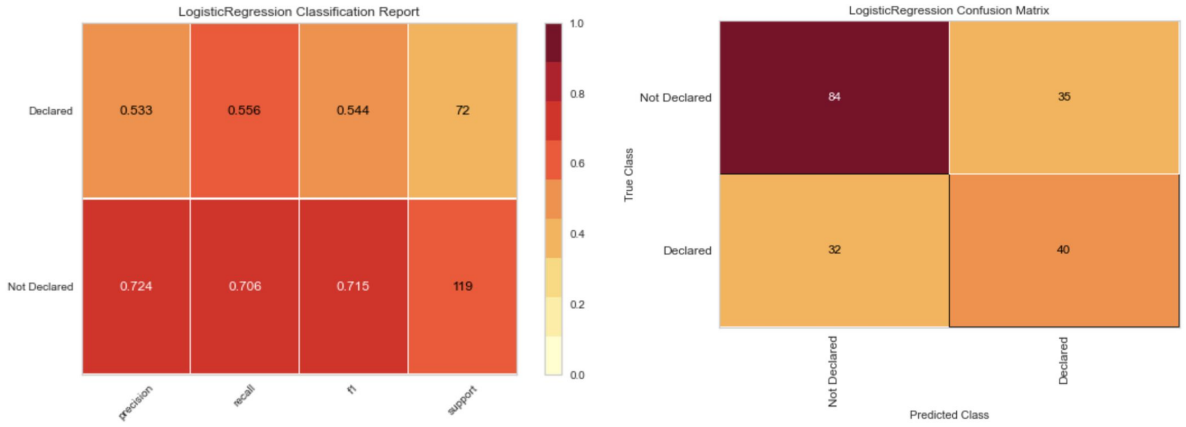
We also completed hyperparameter tuning to determine parameters that are not directly learned within an estimator, like alpha. We found the alpha values for each of the regularizers and also performed a Gridsearch to see whether we needed to tune any of the Ridge parameters. In this instance, we would normalize before regression, not include an intercept, and use a least squares solution. We also completed visual Grideseach through the AlphaSelection Visualizer. Because the alpha was high, the more the regularization parameter will influence the final model.

We then built a way to evaluate multiple estimators to decide which models to pursue. Ultimately, we focused on a decision tree and random forest with the random forest better

predicting the declaration of the IHP provision and the decision tree better predicting the absence of declaration.

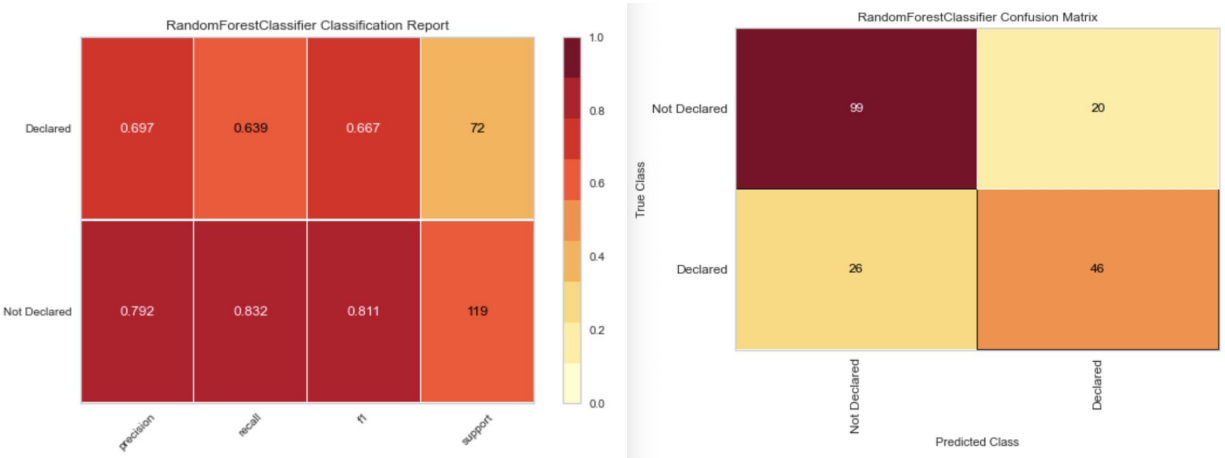
- Logistic Regression

For this model we got an accuracy of 0.65 on the training and test set. As we can see in the Classification report, the f1-score of the usage of the provision was 0.544 and 0.715 for the non-usage of the provision. This was above our baseline, but it was not the f1-score that we were expecting.



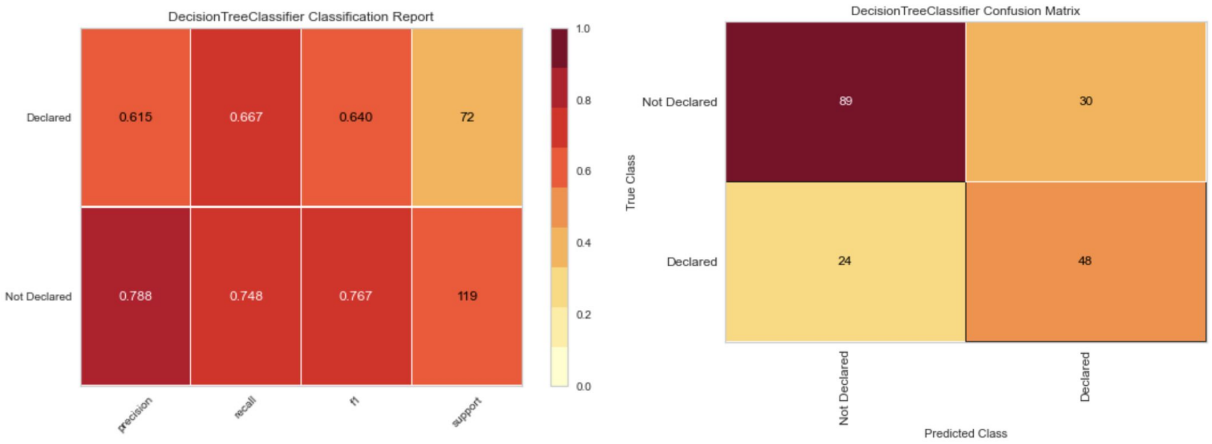
- Random Forest

For this model we got an accuracy of 0.963 on the training set and 0.759 on the test set. As we can see in the Classification report, the f1-score of the non-enactment was 0.81 and 0.67 for the enactment of the provision. The feature importance for this model was 0.034 (disaster type), 0.222 (county code), 0.426 (disaster length), and 0.318 (precipitation).



- Decision Tree

For this model we got an accuracy of 0.96 on the training set and 0.72 on the test set. As we can see in the Classification report, the f1-score of the non-enactment was 0.77 and 0.64 for the enactment of the provision.



Finally, we used Yellowbrick to evaluate our models, class prediction error, class balance, and cross validation. From this, we found that the model is worse at predicting non-enactment of the provision. We also saw that the classes were relatively balanced, and the mean cross validation scores for MultinomialNB was just under 0.5.

In conducting ROC curves for each of the classifiers, we found the training score is much higher than the cross validation score indicating some model overfitting. We then evaluated the tradeoff between the classifier's sensitivity and specificity. Ultimately, the model did relatively well but there is room for further evaluation. In this instance, false positives are not a bad thing as resource mobilization is the target of our model. It is better to have the resources available, especially when it comes to availability of a place to live, than to not be prepared.

Finally, this model would be improved through the expansion of the scope to the entire US, the identification of missing variables to eliminate non-random residual plot patterns, and the expansion of cross validation to address the model overfit. We see the most pressing matter to fix is the non-randomness of the residual plots.

Final Product

<u>Data Prediction:</u>

Once training and testing the model was complete, the team worked on implementing the model on new data (i.e. real time dataset). To process the data quickly and efficiently without retraining the model. we transferred the models using pickle into a .sav file type.

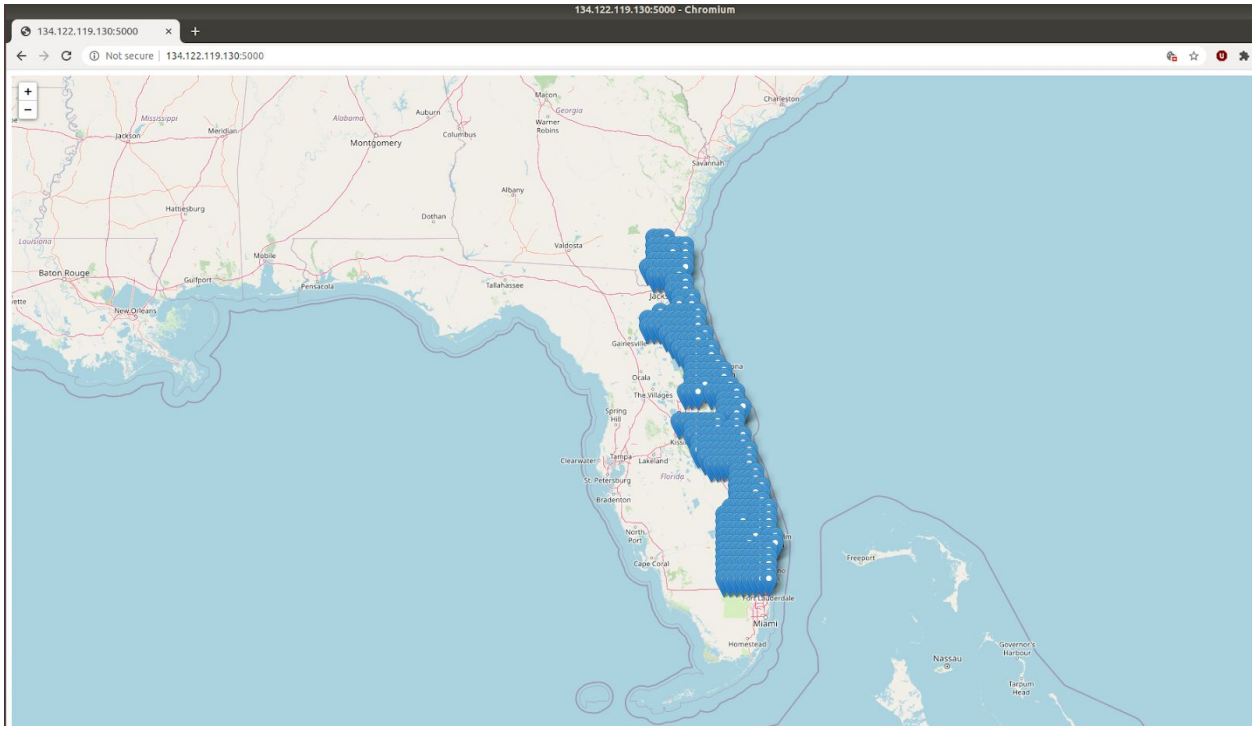
The .sav file can be unloaded to another computer and output a prediction based on the inputs the model is given. For our model, we needed to provide latitude, longitude, month, and seven days worth of historical data (mm/hour average). The model will then output a zero or one value based off of the trained data.

Implementation:

To deploy the code, the pickle file was moved to the digital ocean server and the Flask library was used as a web framework. Flask is similar to Node.js in providing a back end server utility and was chosen to maintain the python language throughout the project.

To request the page, the server must be accessed via the ip address and port 5000. Upon the http request, the function will read the longitude, latitude, month and seven days of precipitation data, input it into the pickle model, add a column to the data indicating if a disaster is predicted.

Once the model processes the data, the server will then create a map using the Folium library. Folium uses OpenStreetMaps and creates an html file that users can interact with such as zoom and pan. Flask will iterate through the declared disaster and provide points on predicted disasters. The points can be clicked and provide the latitude and longitude of the area declared a disaster. Below is an example of how the product is executed.



For presentational purposes, we used previous historical data to provide what the tool would look like if a disaster was predicted. Real time data could have been used by DarkSky API, which provides a seven day forecast based on latitude and longitude and the time of the API request. The data would have to be parsed accordingly for the pickle file, but the processing after inputting the model would be the same.

For future work, the team would like to add a domain name for easier access. Having a subdomain on the FEMA website would be the most convenient. Implementing real time data would also benefit the tool for future data. From a visual perspective, we attempted to add a layer of counties and a heat map providing an instantly recognizable designation for counties in need of aid.

The scope of the project can be further expanded to other states too. Considering the geography, it would require relatively few changes to retrain the model with other southern states such as Texas, Louisiana, Alabama, Mississippi, Georgia and South Carolina. Expanding to further states may be possible, but would probably require further analysis and model exploration due to geographic and meteorologic differences.

Lessons Learned

Data Storage

Drive space can affect the scope of your analysis and the model that can be built. Organizing the data appropriately can make modeling much easier. This is a tedious yet necessary step in data science. This component of our project took a lot longer than expected as we grappled with the level at which we wanted the precipitation data. With this came many discussions about the scope of what our project would be. Ultimately, this is a really important lesson in data science, but we also must be aware of the biases that can come with reduced scope and the transferability of a model as a result.

Feature Selection

There are many models that can be used to determine features to be used, including visualizations. Ultimately, this allows for reduced complexity and bias, lower dimensional space and less computation time, and better interpretability given fewer variables. In our case, lower dimension space was key given the size of the NASA data set. We utilized both domain expertise and the regularization techniques, transformer methods, and dimensionality reduction to hone in on the appropriate features to utilize for both of the models. At a high level, we wanted to get the smallest number of features so that the model would reach the maximal predictive value.

Model Evaluation and Tuning

There are many tools that enable the evaluation of the quality of a model. When addressing new problems in unfamiliar domains, it is often not obvious what models to use and what will work well. This class expanded our knowledge of cross validation, hyperparameter tuning, and other reporting that can influence the final model that you choose. Issues like class imbalance were prevalent in our disaster prediction model given that disasters are anomalies. Thus, we had to determine ways to overcome this, like taking a random sample of non-disaster days as well as using the class weight parameter to balance the data.