CMPE 255-01: Data Mining

A Project Report on

Natural Disaster Prediction

Group 9

Emma Peatfield – 009192534

Trung Huynh – 011816337

Saurabh Ghotane – 011451427

Fall 2018

Introduction

- **Motivation**

  Within just the past year, we have seen intense weather events that have created so much damage to our cities and towns throughout the United States alone. Each time a storm hits, it seems to get worse and worse. When it comes to future storms, our country needs to be more prepared for these storms. One way to help with preparedness is quicker and more efficient predictions for storms. The sooner a town or city knows about these storms, the sooner they can begin preparations. With more preparation, cities will save so many more lives, much more money, and a lot more infrastructure from being destroyed.

- **Objective**

  The objective of our project was to take current and past weather data that was available online and use it to predict what type of storm could occur from certain weather variables. We chose to use weather data from the National Climatic Data Center, which was available to download from the National Oceanic and Atmospheric Administration website. This weather data has thousands of stations that have been collecting local climate data throughout the United States since early 1940's. It contains features, such as dry and wet bulb temperatures, wind speed, wind direction, humidity, visibility, precipitation, etc. We wanted to use this data to create a model that, when given values for these features, it could predict if there will be a storm or not, and if so, what type of storm.

System Design & Implementation details

- **Algorithms**

  For the start of the project, we had a few algorithms in mind to try. Each of us chose one or two and worked with our data to implement them individually. These algorithms were chosen based on each person's preferences and which algorithm they thought would be a good choice for the kind of data that we had.

  Gaussian Naive Bayes and **k-Nearest Neighbors** were the algorithms that Emma used. Gaussian Naive Bayes was chosen because it is fast and works well with less training data. There is a lot of data now, but when the project was started, there was only some sample data that was downloaded. So to start with the project, she wanted a general idea of what kind of results she could produce. Since it produced good results, she kept it for testing with more data. Another benefit to using Gaussian Naive Bayes is that is works really well with multi-label classification and is not sensitive to irrelevant features, which meant that it was a good fit for the data the group used. K-Nearest Neighbors was another good fit because it worked for data that has multiple features and for finding the prediction based on the closest neighbors, i.e. the most similar entries to that one. Depending on the input parameters, it worked just as well as the Gaussian Naive Bayes algorithm.

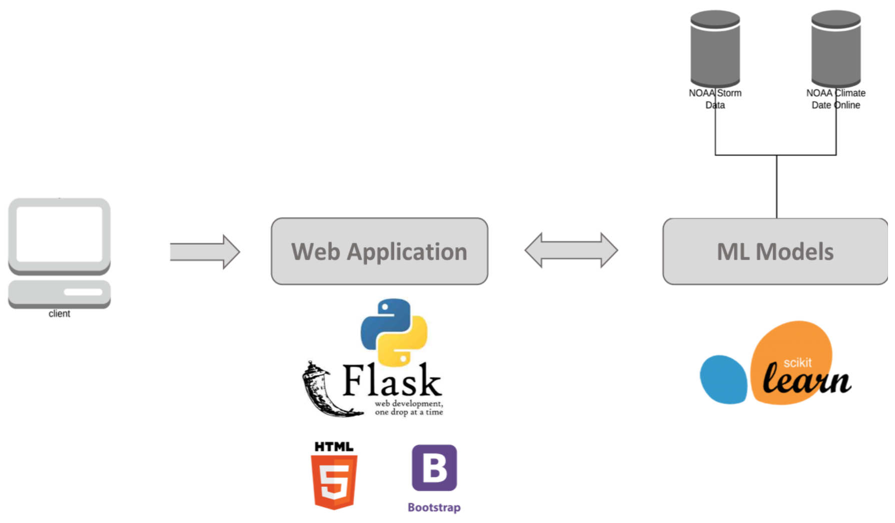
AdaBoostClassifier: Ensemble technique that iteratively train multiple classifiers with different sample of the dataset. It put more weight on the hard classes so that these instances will be picked for subsequence iterations. even though adaboost took more time to train, it is popular for imbalanced dataset.

Random Forest is one of the most widely used and efficient supervised learning algorithm. It is an ensemble algorithm and works by building decision trees and then by using voting mechanism to predict the final class. This algorithm was selected because it is efficient for dealing with large amounts of data. One of the key benefits of using this algorithm was that it gave a good estimate of which features are important for the classification. Random forest algorithm is also known to be effective at handling missing data and is well suited for multi-label classification problems.

- **Technologies & Tools**

A few of the tools that we used on our own machines were Jupyter Notebook and Microsoft Excel. These were used to help us code, debug, and visualize our data. Along with those tools, while coding in Python, we used quite a few libraries that helped us with preprocessing data and running our algorithms. These were the sklearn, numpy, pandas, and the matplotlib libraries. Our own machines that were accessible to us were Windows and MacOS machines and through those we also accessed the HPC from SJSU. The HPC was used to help us run large amounts of data, so it helped us speedup our preprocessing and training. It also was very useful for file sharing. For the web application, Saurabh utilized Flask, HTML5, CSS, and Bootstrap, which are all popular frameworks for developing a web-application.

- **System Architecture**



- **Web application UI**

  Given below is the screenshot of our web-app form in which user can input certain parameters like temperature, humidity, wind speed, pressure. This data is then passed as input features to the underlying pre-trained model that will predict the type of event that may occur. The input and output screens of the application are shown below:

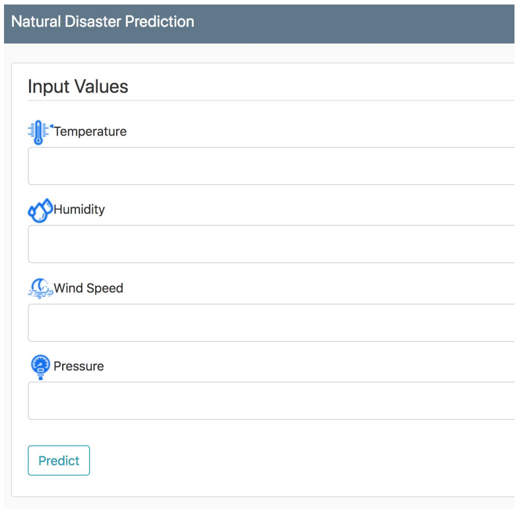
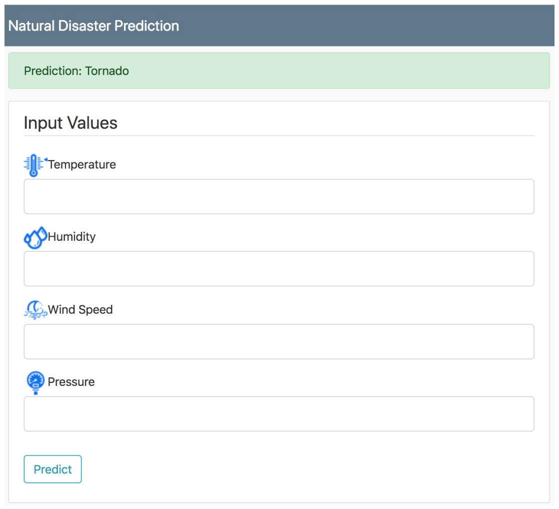


Fig-2 Home Screen



Prediction: Tornado

Fig-3: Result Screen

Experiments/ Proof of concept evaluation

- **Dataset**

  Storm Events: obtained from NOAA website. The data are tables of 265576 storm event entries from 2014 to 2018 with 52 columns. The size is approximately 237 MB
  Link: https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/

  Local Climatological Data: obtained from NOAA website. The data are tables of 316426 weather entries from 2014 to 2018 of 6 weather stations from 6 cities in Florida: Miami, Jacksonville, Tallahassee, Orlando, Daytona Beach, Boca Raton. The data has a total of 92 columns and the size is approximately 88 MB
  Link: https://www.ncdc.noaa.gov/cdo-web/datatools/lcd

  Enhanced Master Station History Report: obtained from NOAA website. This dataset was just a text document that included information on each station. The main information that we used was the state and county listings per station.

Link: https://www.ncdc.noaa.gov/homr/reports/mshr

File	Non-Zeros (Approx.)
StormEvents_Details	13.8 million
Local Climate Data	29 million

To get the weather data ready to run through the algorithms, we needed to do quite a bit of processing. To start, the weather data was downloaded from different stations throughout a specified region. The storm data was downloaded from the same website, which included the State names, whereas, the weather data only included the station names. To be able to merge these two, we had to find another dataset that included information for each station, i.e. the state and county. This was from the Enhanced Station History Report, which was a just a basic list of information for each station. Using this, I was able to merge the weather data with the locations based on the station ID, or WBAN. This worked well, but the states in the two datasets were still not matching up because the storm file had states listed as their full name, but the weather data had the states' abbreviations. To get around this, a states.csv file was created by Emma, which was merged with the weather data. Then once this was complete, there were a few other processing techniques that had to be done, like adding an end time to the weather data, and a few formatting steps. Finally, the two datasets were ready to be merged. We used State, County, and Date columns to merge the two datasets, so that the final data-frame consists of the hourly data of 6 locations along with the weather readings and the event type associated with that readings. The first step after we obtain the final data-frame is to drop the columns that have too many missing values, after that we fill the NaN values in the remaining columns with the previous value, since our data is ordered by time, and the weather readings are dependent on the previous readings, therefore, it makes more sense to impute missing value with the previous one.

- **Methodology**

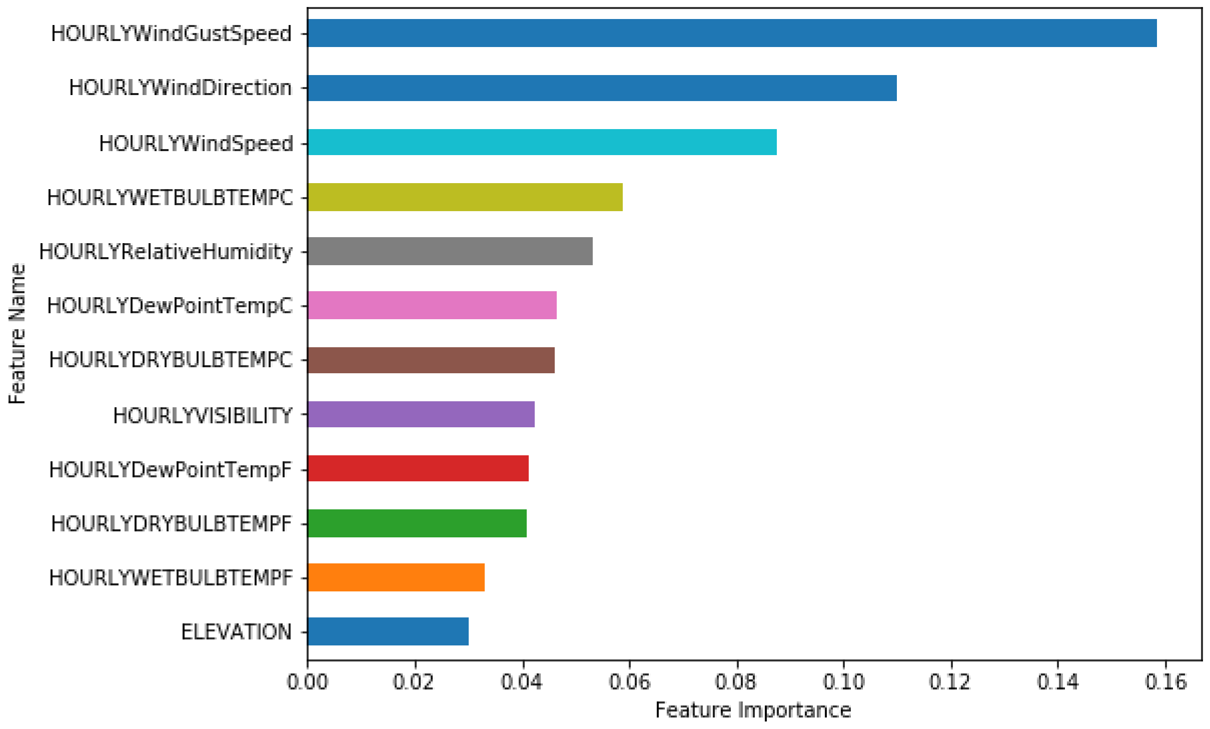
Since we are using different algorithms, we decided that it would be better for each of us to approach the problem individually.
Emma performed a 25:75 split on the data for train and test sets. The data was then fit to the specified algorithm (Gaussian Naive Bayes and K-Neighbors Classifier) and then the training split was used for predictions. To evaluate the performance, Emma used the accuracy score for each algorithm.

Trung: performed a 2:1 stratified split on the data into train and test sets to reserve the class distribution. The train set is splitted again in 5-fold-cross validation for parameter tuning and model selection. The test set is used to evaluate the performance of the model.

Saurabh: K-Fold cross validation technique was used to split the dataset into 10 consecutive folds with shuffle attribute enabled to shuffle the data before splitting into batches. Each fold was used once as a validation set while the k - 1 remaining folds form the training set. This K-Folding technique was used to evaluate the model performance for different parameters and F1 Score and Accuracy metric for each iteration were logged and evaluated. In order to identify optimal values for different parameters of Random Forest algorithm, an iterative approach was followed wherein a range of different values for a particular parameter were tried and plotted on a graph to identify the optimal values.

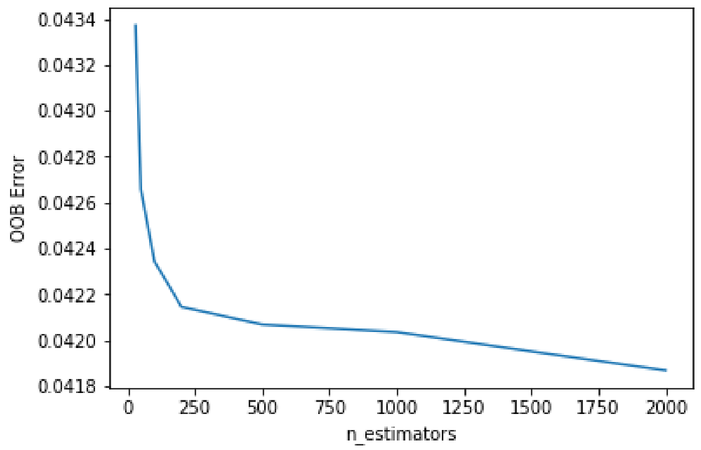
- **Feature Selection**

  The Random forest classifier allows us to identify features that are more helpful for the model by providing feature_importances_ attribute. We plotted features in decreasing order of their importance as shown in the figure below:



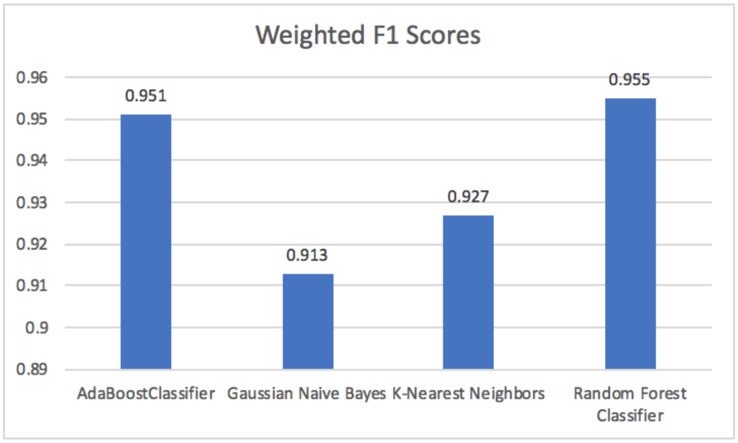
- **Parameter estimation**

  In order to identify the optimal value for different parameters, a for loop was used to iterate over a list of possible values for each parameter and then the result of the classifier like F1 score and Accuracy were plotted on a graph to identify the best value for the corresponding parameter. The figure below shows the drop in Out of bag error with the increase in n_estimators:



- **Analysis of results**

The table below shows the results of different classification algorithms used:



Since our dataset is highly imbalanced, therefore, we chose F1-score to evaluate our models. Each of the above mentioned models was optimized following an iterative approach to identify the optimal values for the model. We observed that ensemble classifiers performed better compared to base classifiers.

Discussion & Conclusions

- **Decisions made**
  - Decided to start with a only a few stations in 2018 in order to move forward with pre-processing. We wanted our model to be able to process data efficiently once we used the dataset as a whole.
  - We chose the cities that have many storm activities to ensure that we could have as much data as possible for predicting storms.
- **Difficulties faced**
  - Data was very disorganized and everything seemed to be in different formats, even though it was from the same place, which made it all very messy to work with.
  - All data was in a different format so things had to be changed, such as state names, time/date formats, and even collecting outside data for locations and full state names.
  - Had to decide how much data to use, which stations to pick, etc.
  - The climate dataset is not easily accessible. We had to download each station data one at a time, and each download had to be under a certain size.
  - Original data-type of each columns are string, therefore, we need to parse them into float. Note that aside from many NaN values, some of the values are incorrect so it can not be casted to float.
- **Things that worked**
  - The different algorithms that each of us tried.
  - Preprocessing the data enough to get it all to a merging point for training.
  - Including all data, even non-storm entries to improve our test results.
  - One hot encoding for the month improve the model f1-score
- **Things that didn't work well**
  - At first, merging all of the data
  - We decided to use SVM at first but it was too slow due to high number of rows.
  - One hot encoding for the station ids doesn't seem to help.
- **Conclusion**
  - We managed to develop a system for predicting the natural disasters based on several input features. However, we observed that while the models performed well on predicting some types of events like 'Thunderstorm Winds', 'Flash Flood', 'Heavy Rain' the models did not work well with certain other types of events like 'Floods', 'Funnel Clouds' and 'Lightning'.
  - In future we can utilize some other type of data and features which would be helpful in covering the remaining type of disaster events.

Project Plan / Task Distribution

- **Data Collection**
  - All - separately and together/shared
- **Data Pre-processing**
  - Initially- Emma
  - Final decisions/additions- Trung/Saurabh
- **Algorithm Implementation**
  - kNN/Gaussian - Emma
  - AdaBoostClassifier - Trung
  - Random Forest - Saurabh
- **Final Project Implementation**
  - All
- **Python Flask Web Application**
  - Saurabh
- **Project Report**
  - All
- **Project Presentation Slides**
  - All

References

https://www.ncdc.noaa.gov
https://scikit-learn.org/stable/
http://flask.pocoo.org/
https://medium.com/machine-learning-101/https-medium-com-savanpatel-chapter-6-adaboost-classifier-b945f330af06