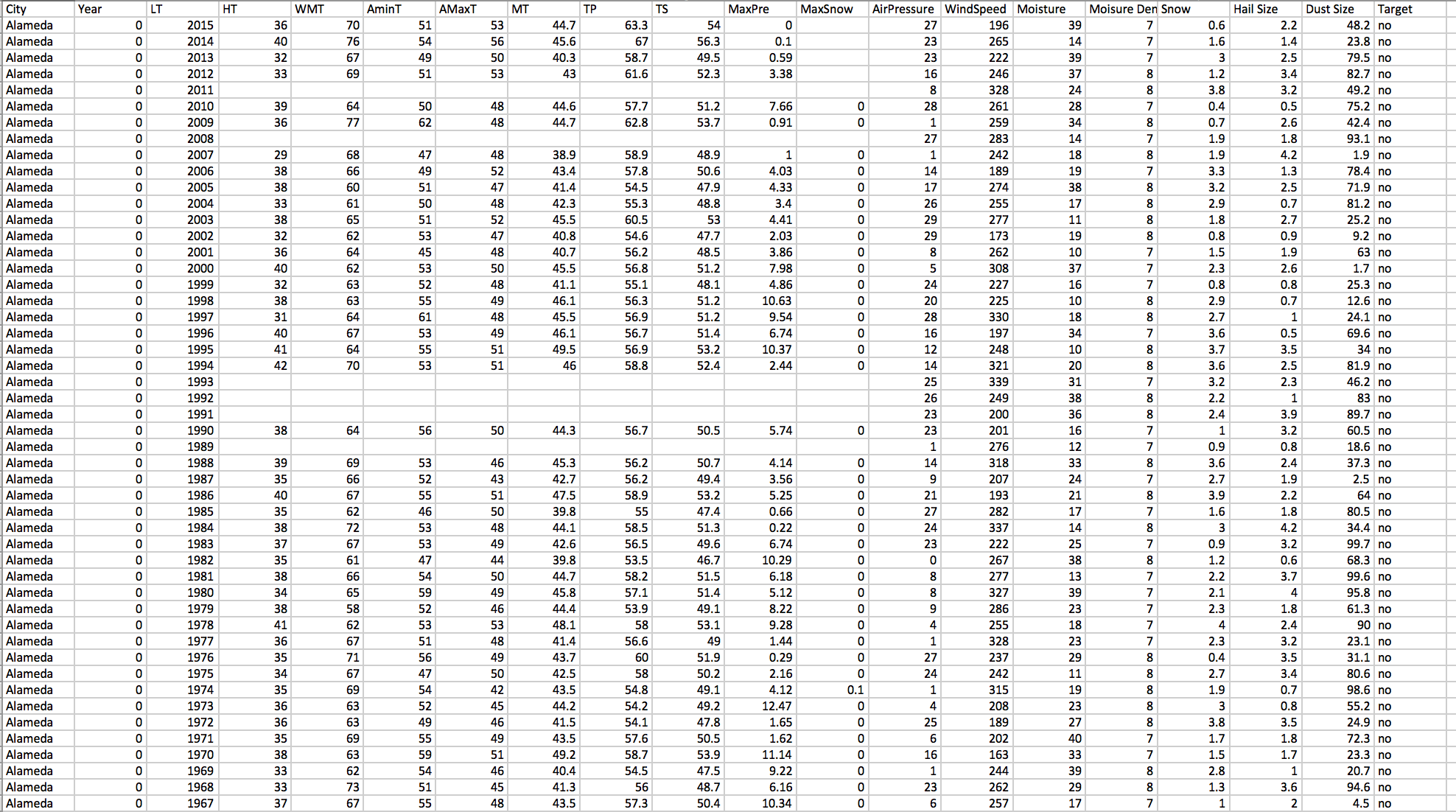
**Chapter 4. System Design**

**4.2 System data and database design**

4.2.1Following is the snapshot of the csv storm data we are using initially:



After reading the data into pandas dataframe, we are doing some modifications on it. Modifications include dealing with missing values, making a new column of target as 0 or 1, instead of yes or no where 0 stands for no and 1 stands for yes.

Following is the summary statistics related to the modified dataframe for storm data:

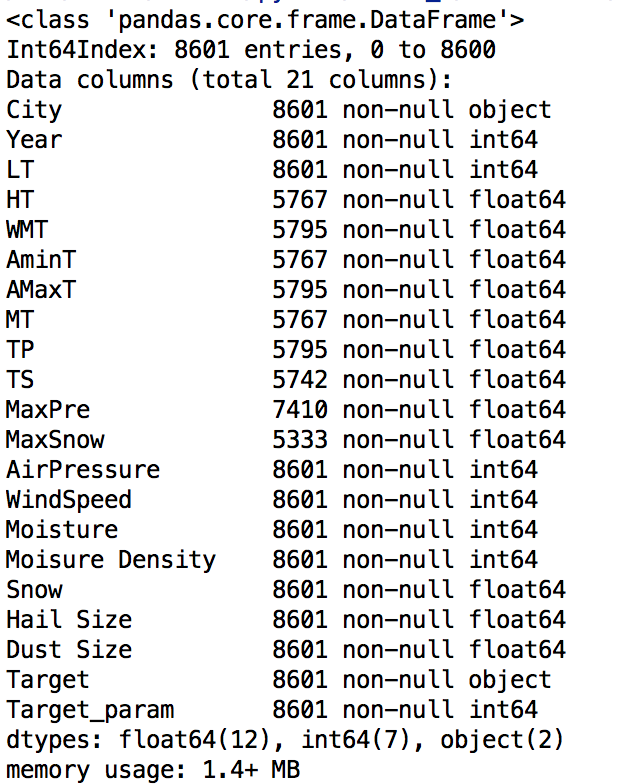


Fig: modified\_weather\_df.info()

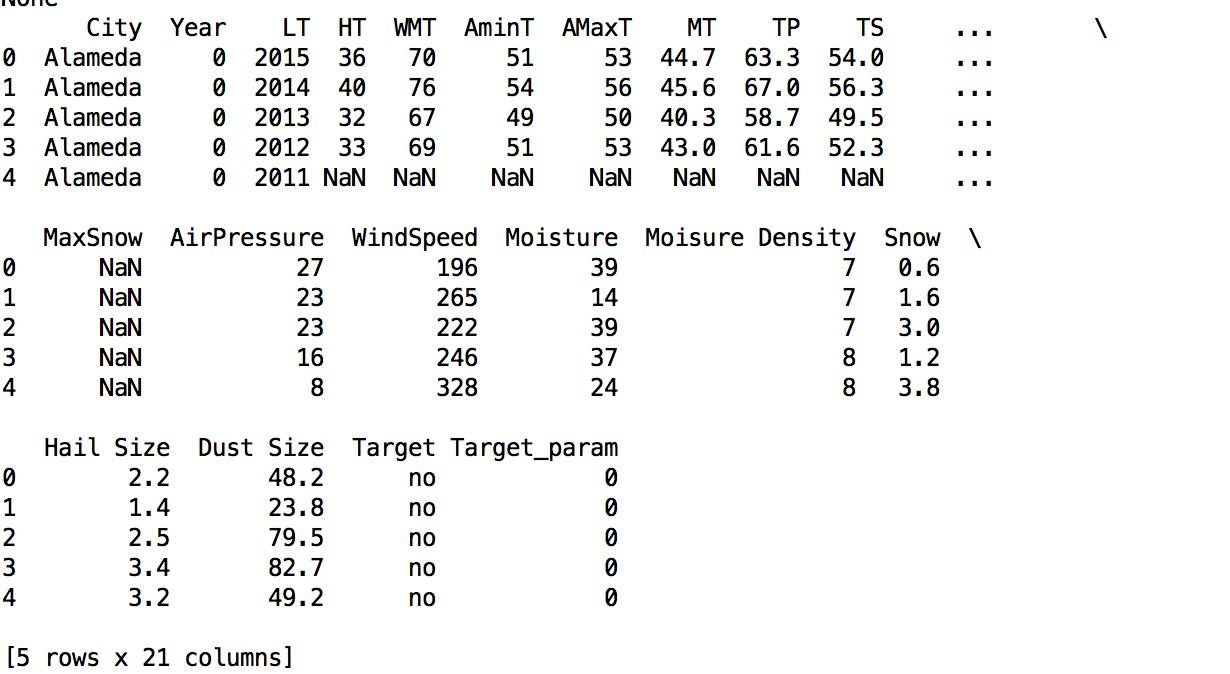


Fig: modified\_weather\_df.head()

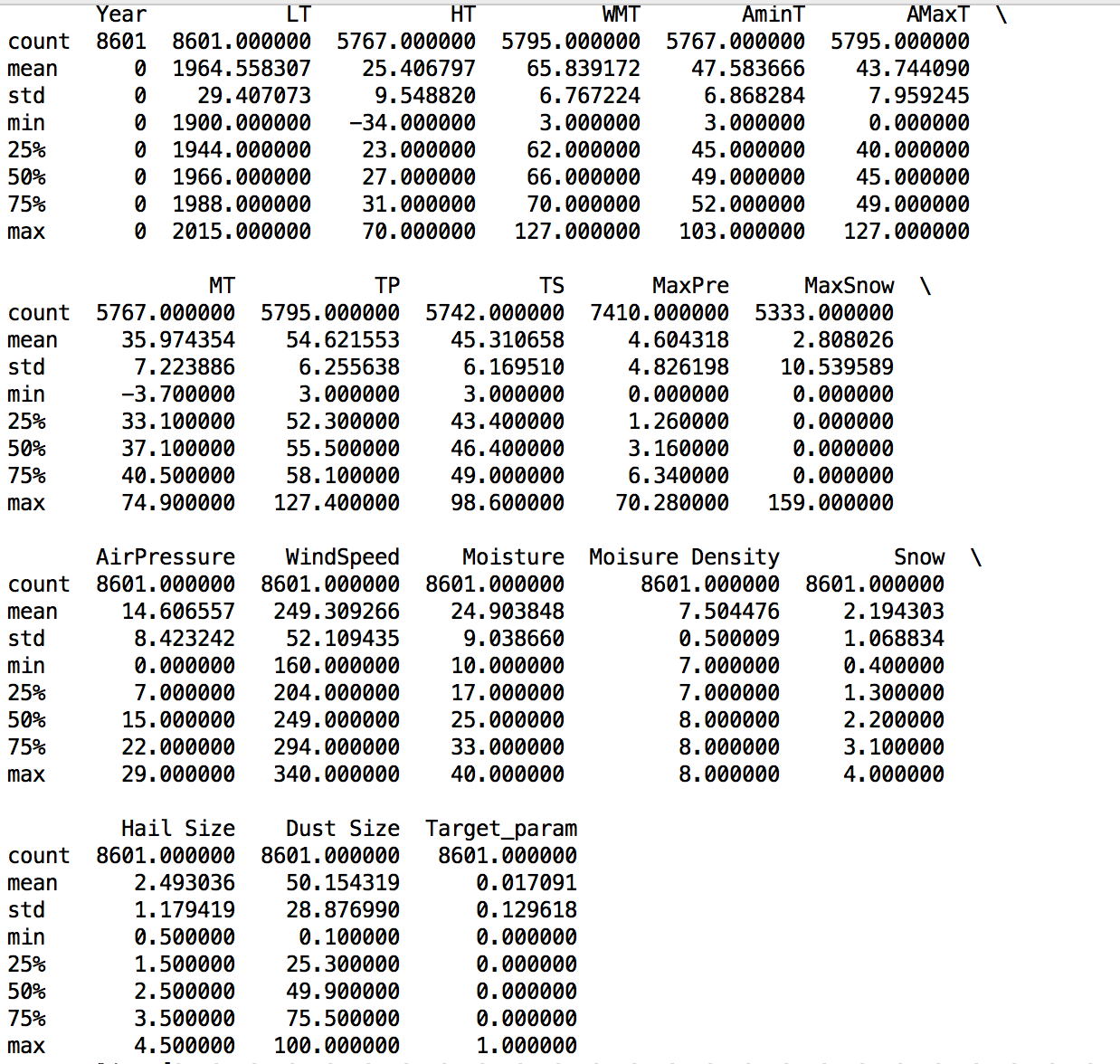


fig: modified\_weather\_df.describe()

4.2.2. Following is the snapshot of dataset to predict earthquake destructiveness, which we are loading into pandas dataframe:

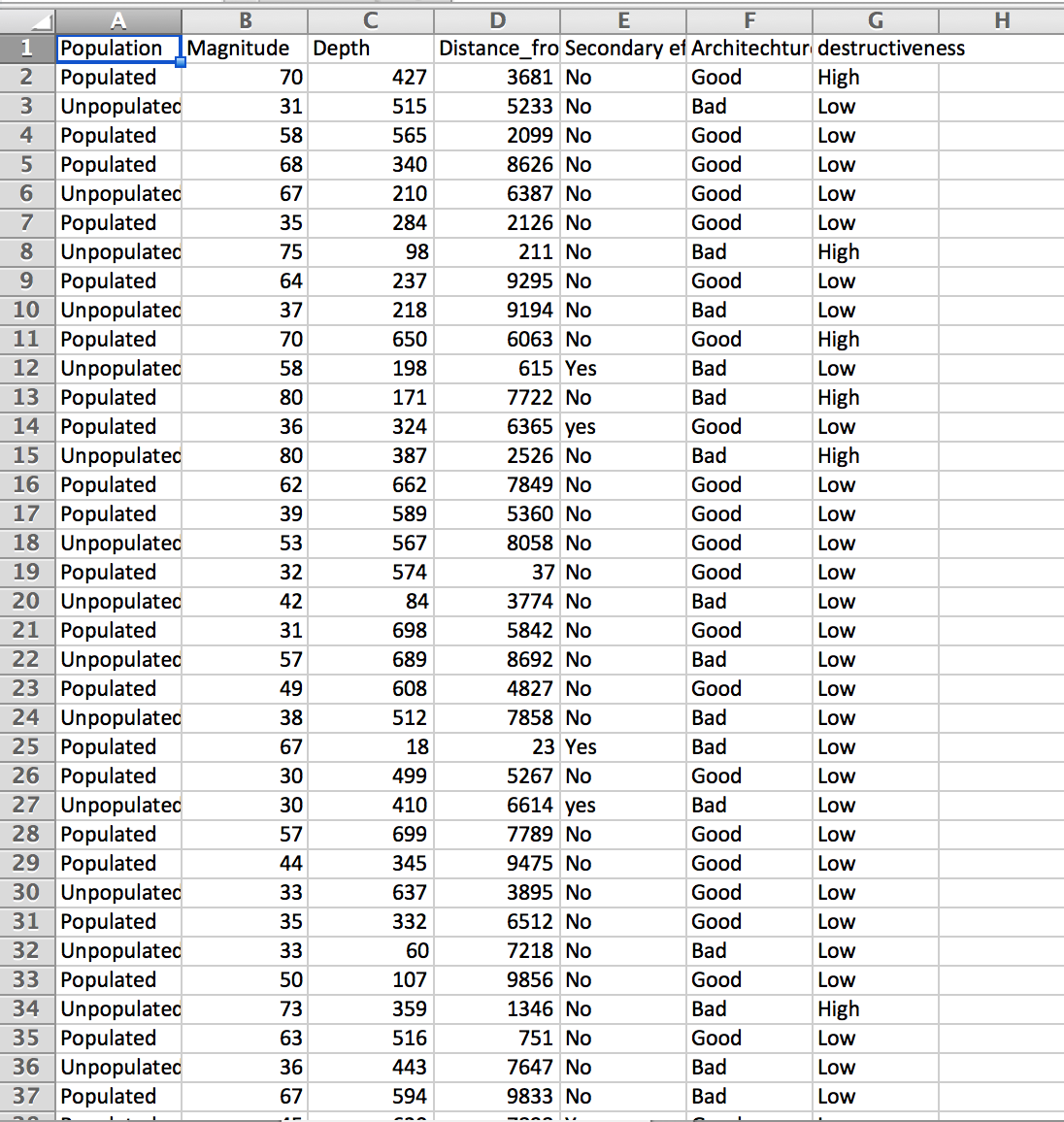


Fig: Earthquake destructiveness dataset

We modified the magnitude column by multiplying it by 10. So for example where the magnitude value is 6.8 we have 68.

We are modifying the dataframe to get the format, which can be used to fit the model. We are adding the columns for categorical columns as we did in the first dataset for column target. We add column for population, secondary effect, Architecture and destructiveness since they all had categorical data.

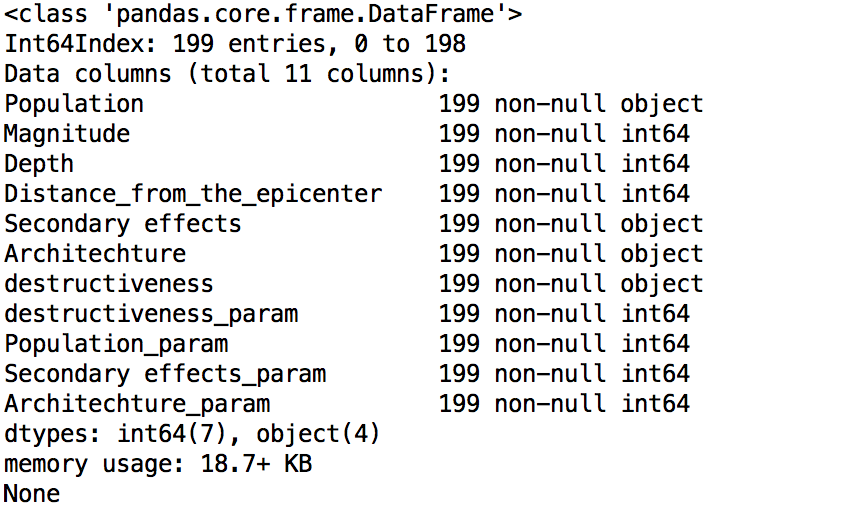


Fig: modified\_earthquake\_destructiveness\_df.info()

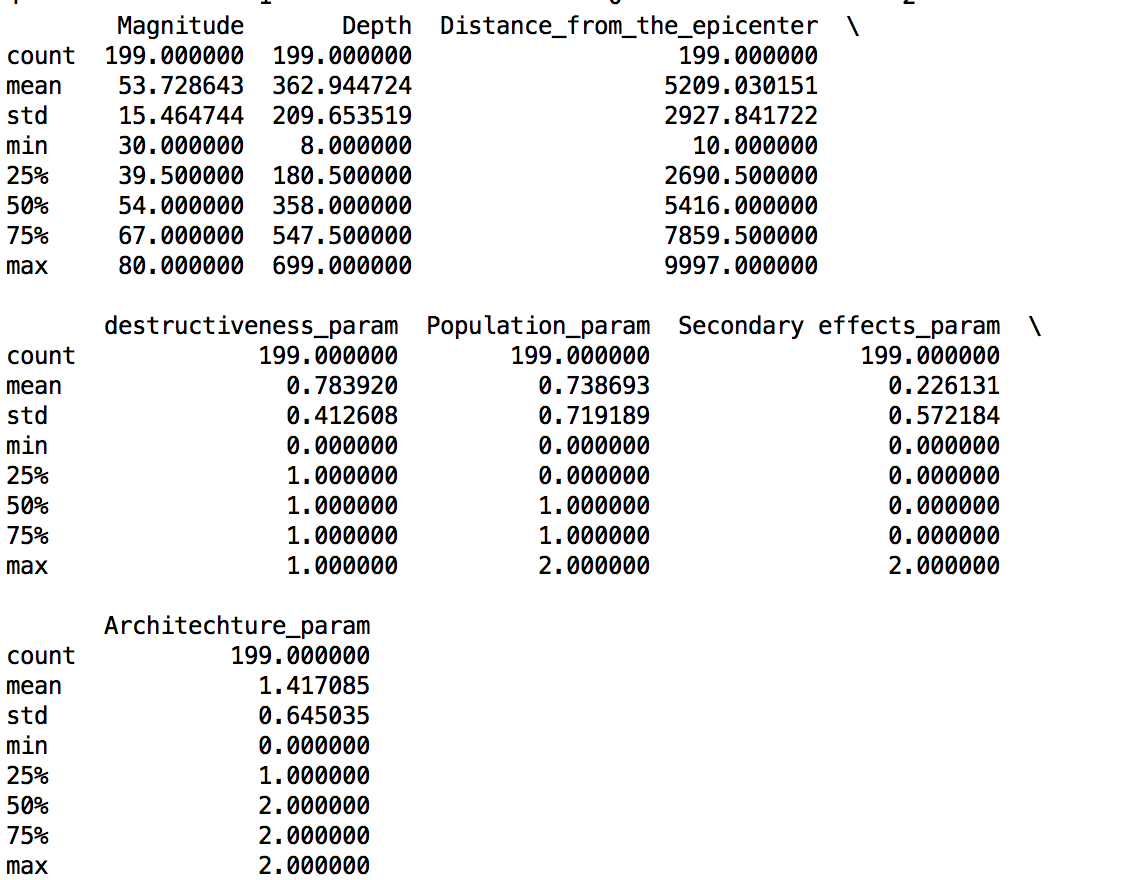


Fig: modified\_earthquake\_destructiveness\_df.describe()

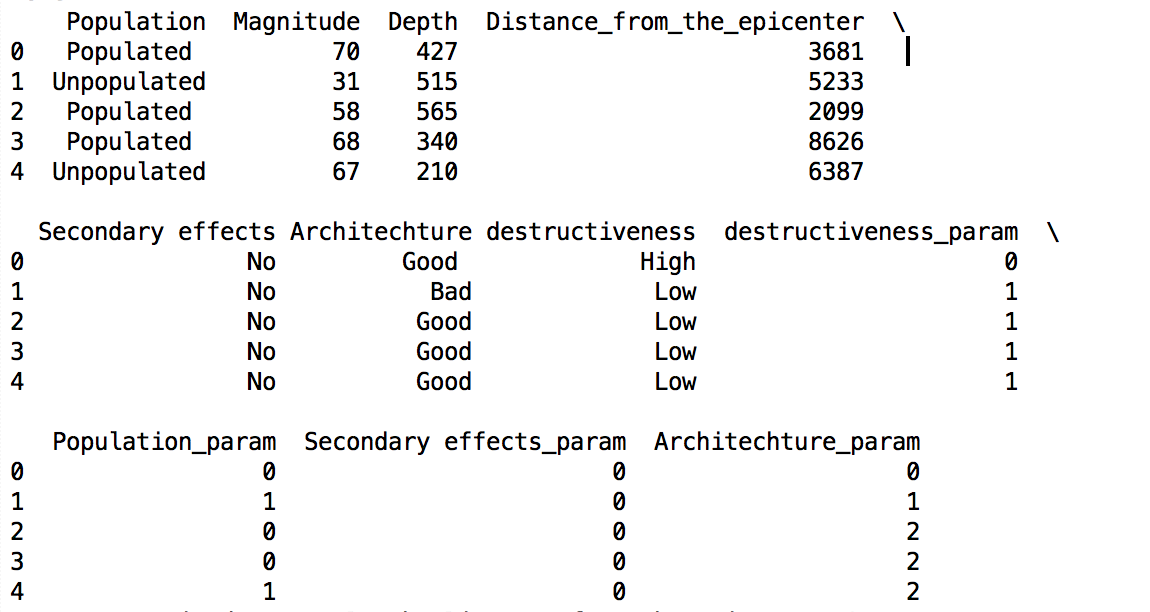
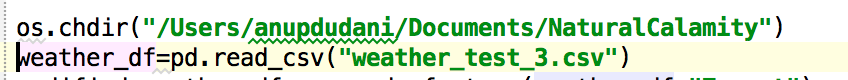


Fig: modified\_earthquake\_destructiveness\_df.head()

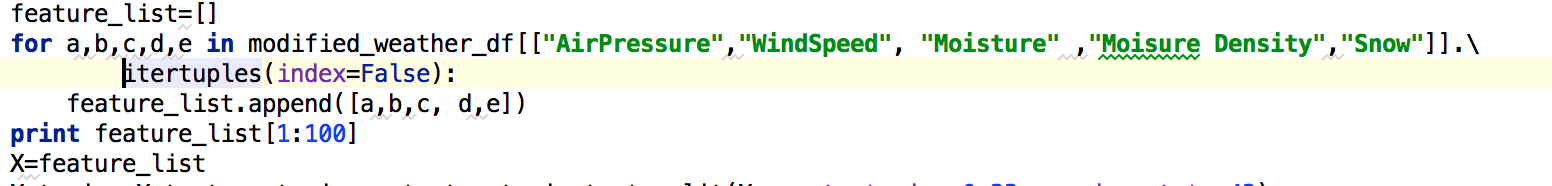
4. 3. System Implementation and Connectivity design

4.3.1. Data Loading:

We are loading the data from the csv files in pandas dataframe. We selected pandas dataframe since it makes easier to analyze data by providing inbuilt methods.



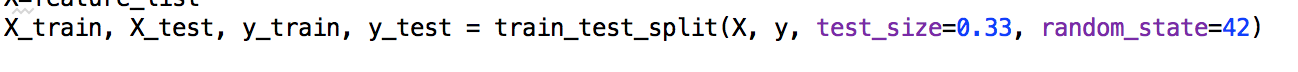
4.3.2. Selecting the features:



We are adding the features as shown in above code snippet.

4.3.3. Running algorithms:

Before running the algorithm we are dividing the feature set and target into training and test data as follows:



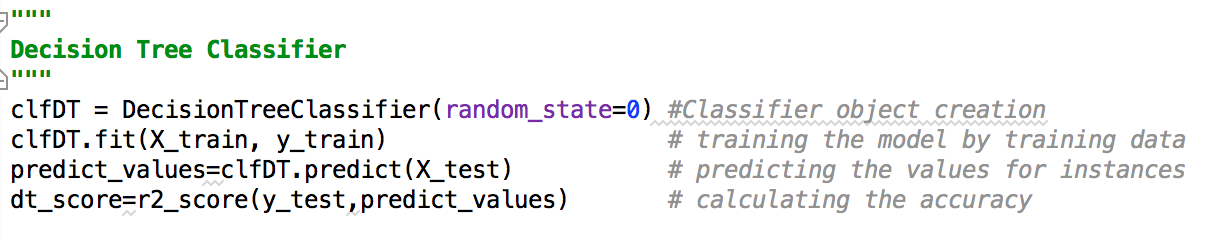
We are using sklearn library for the algorithms, we are using 5 classification algorithms:

1. Decision Tree Classifier
2. SVM
3. Svm.Linearsvc
4. Naïve bayes
5. Knn classifier

All the algorithms implementation in sklearn requires following steps to do:

1. Creating an object of classifier
2. Training the data with training data
3. Getting the predicted values
4. Checking the accuracy

Ex. Following is the example of Decision tree classifier

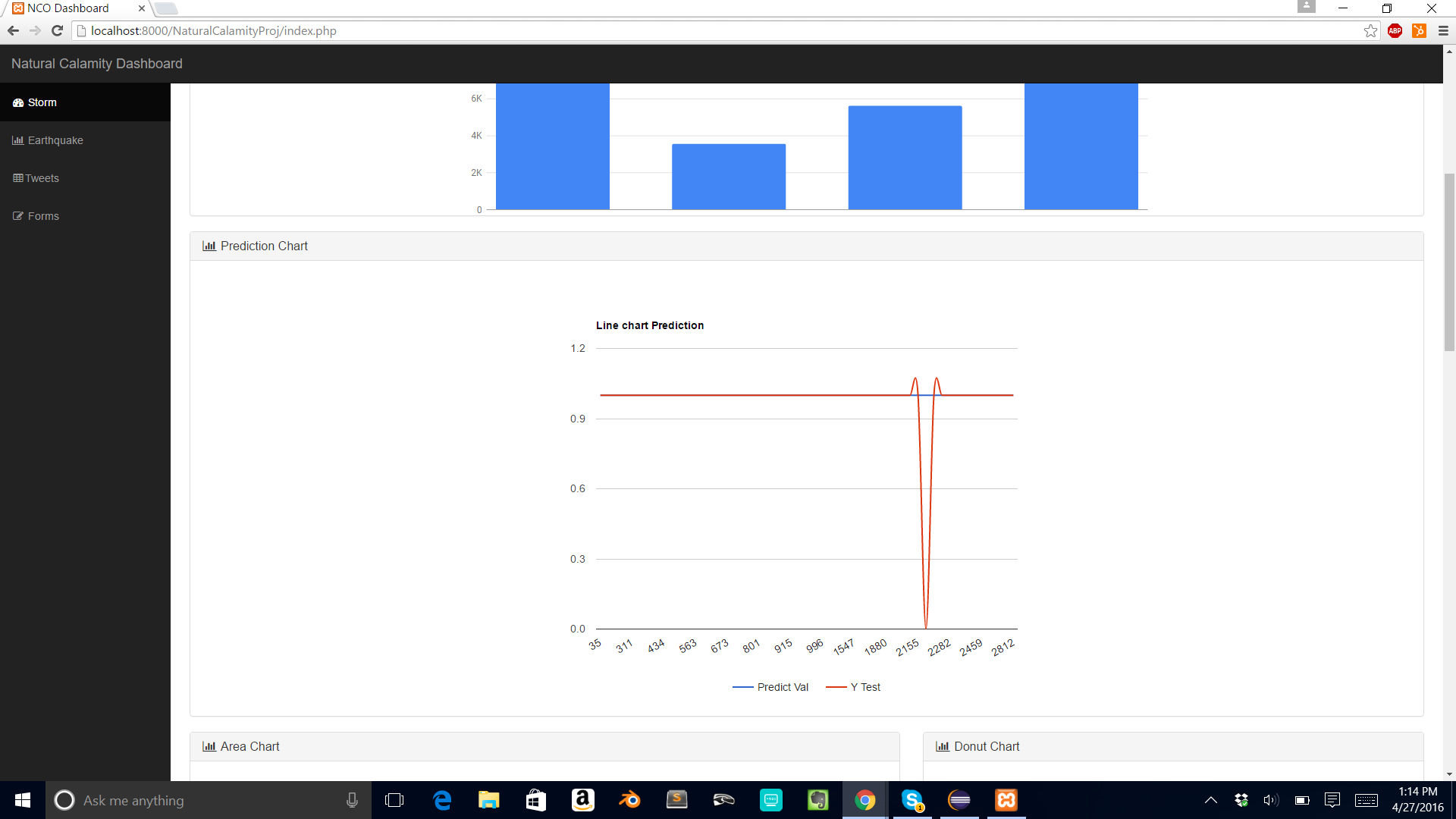


4.3.5. Visualizations:

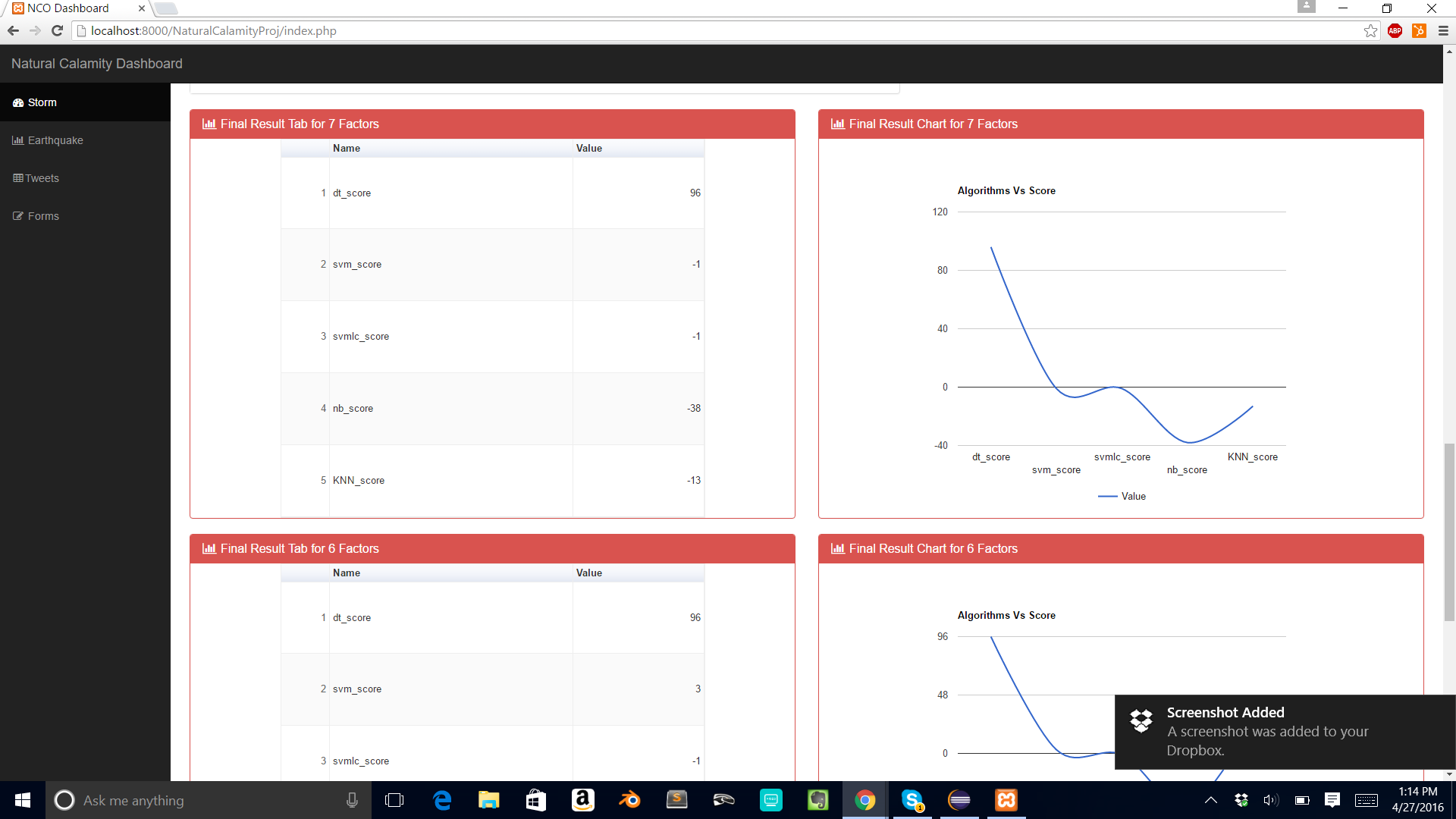
We are using PHP to get data from MySQL and then visualize

4.4. System User Interface Design

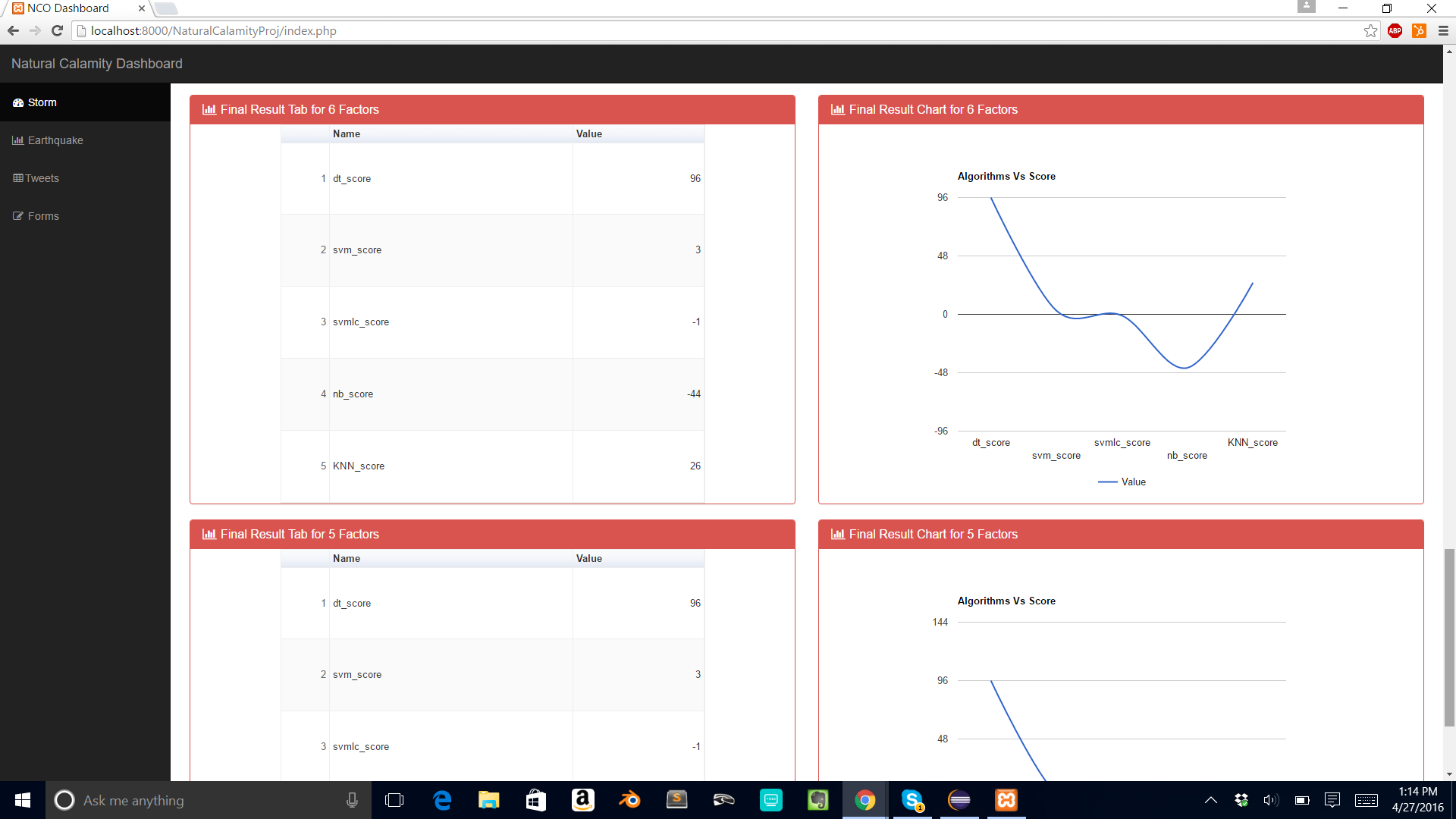
1. Screenshots of the application



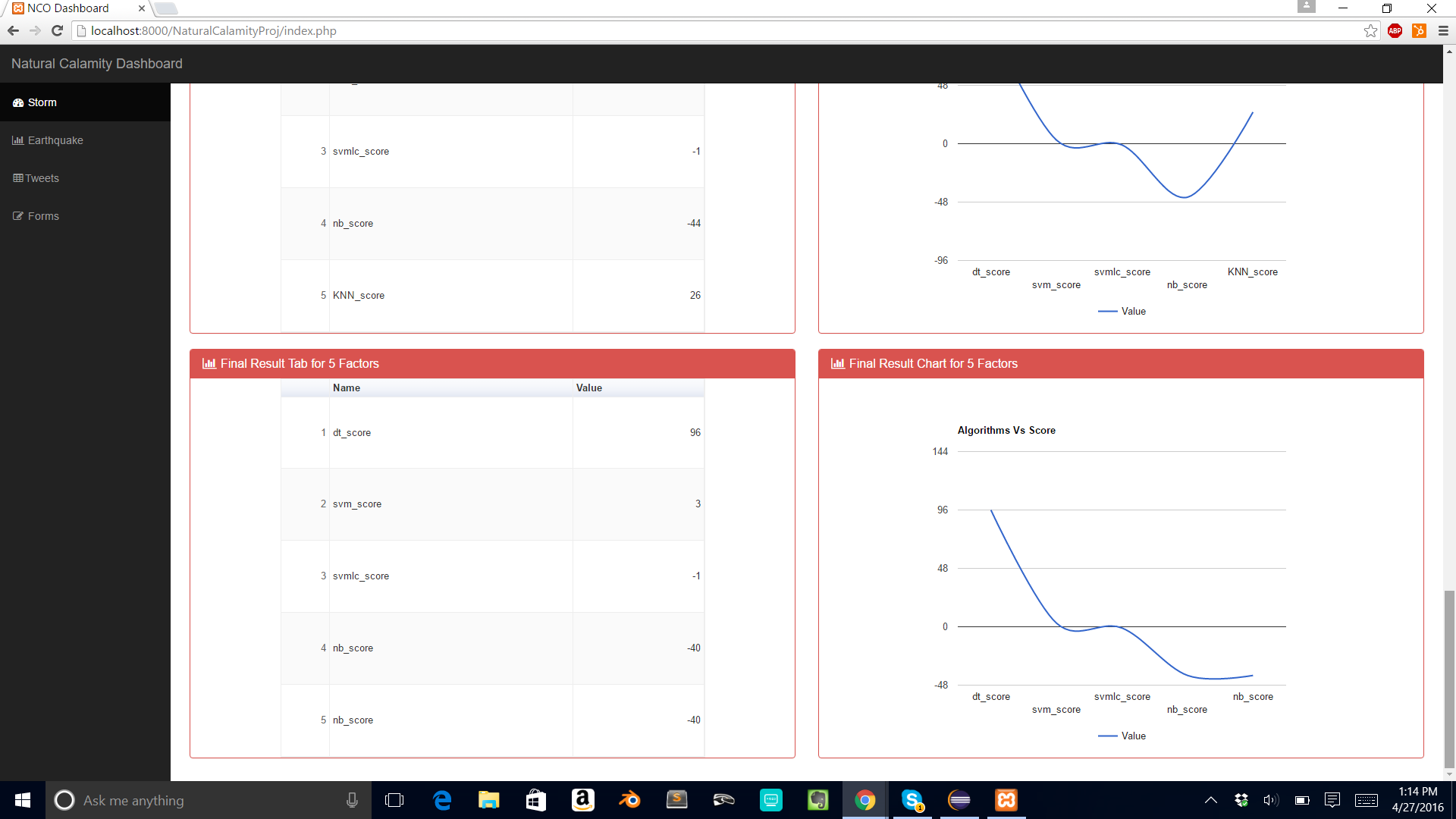
Screenshot 1: Predicted values vs Actual values



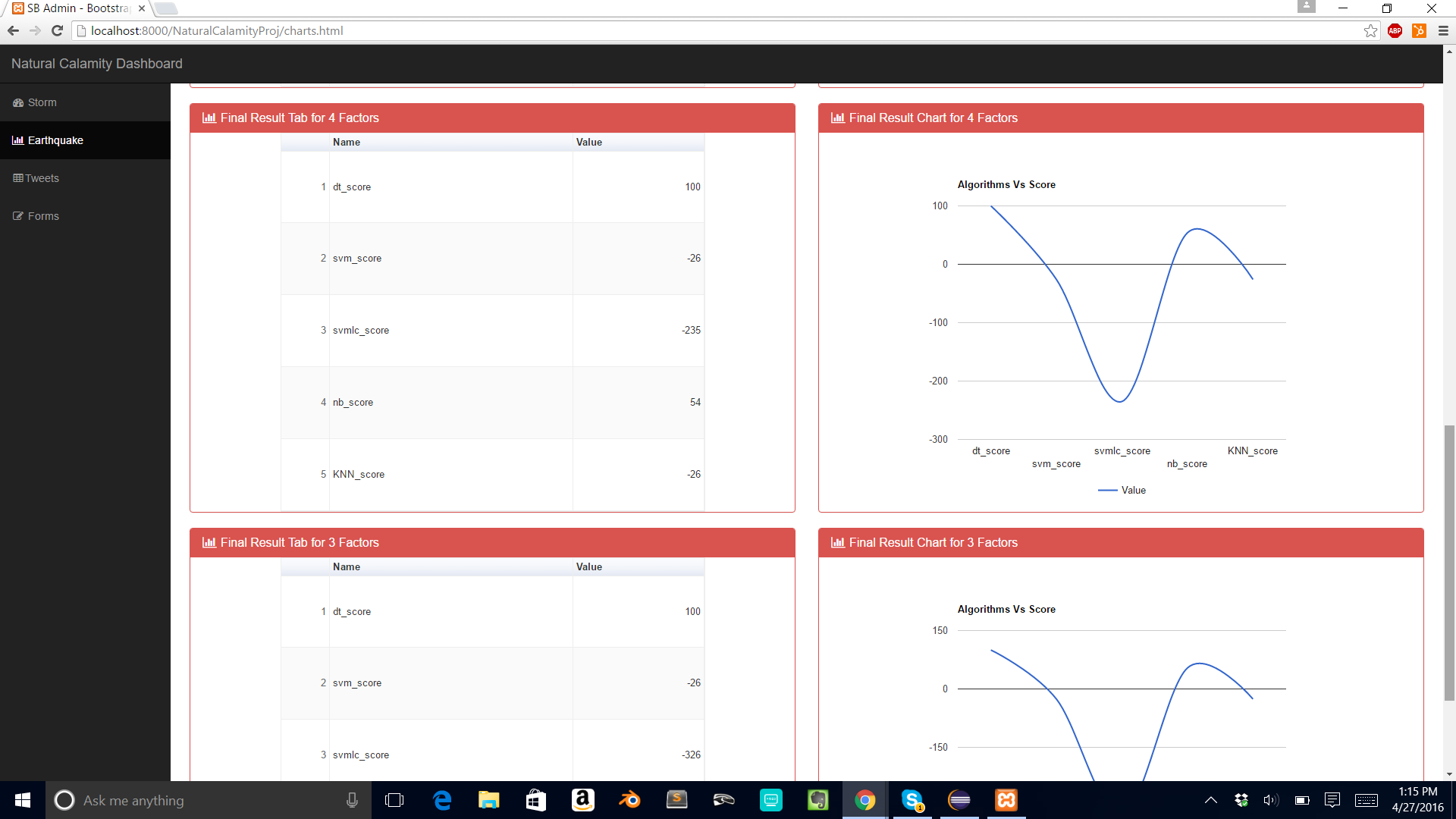
Screenshot 2: Final Results for 7 factors



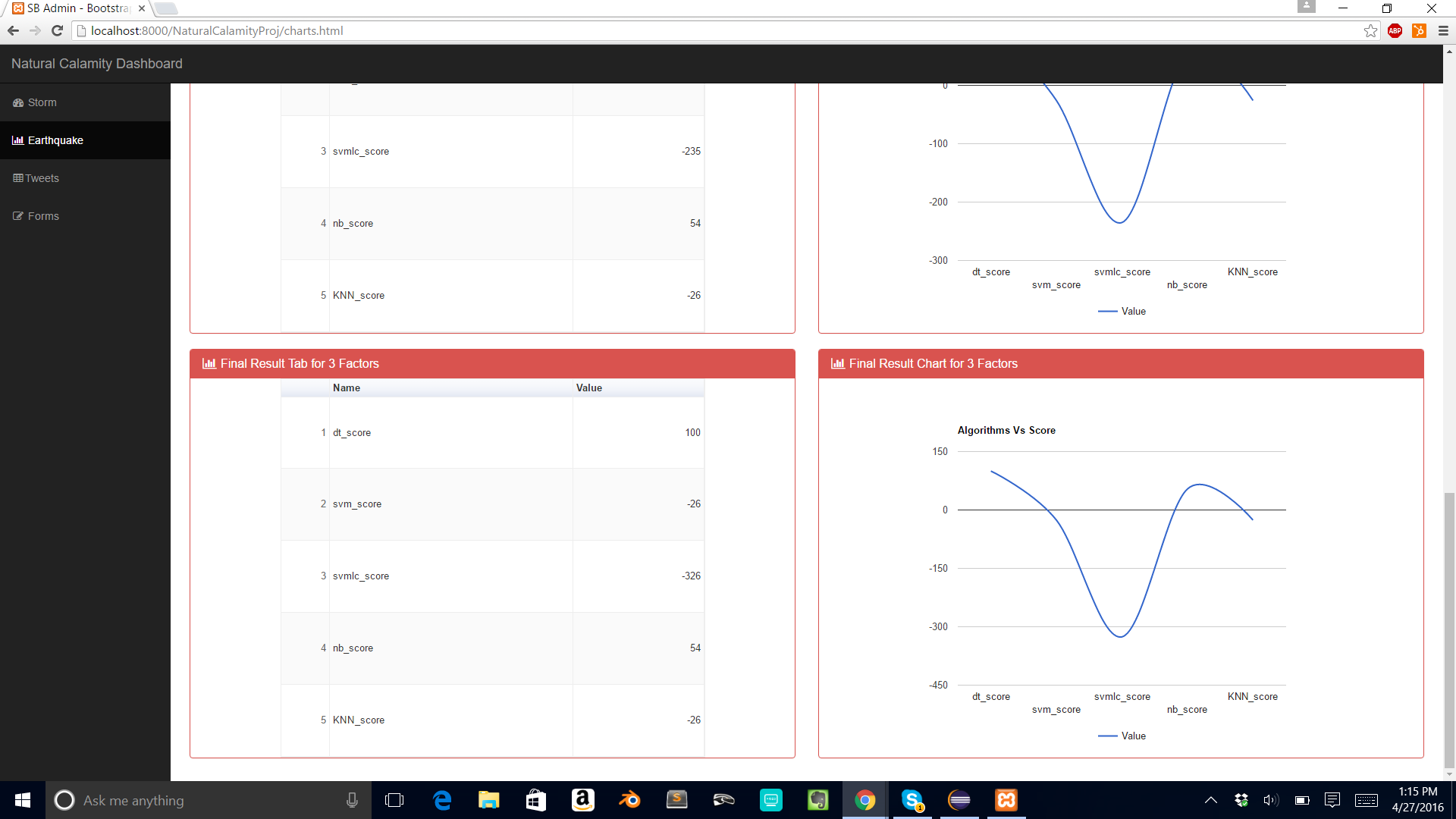
Screenshot 3: Final Results for 6 factors



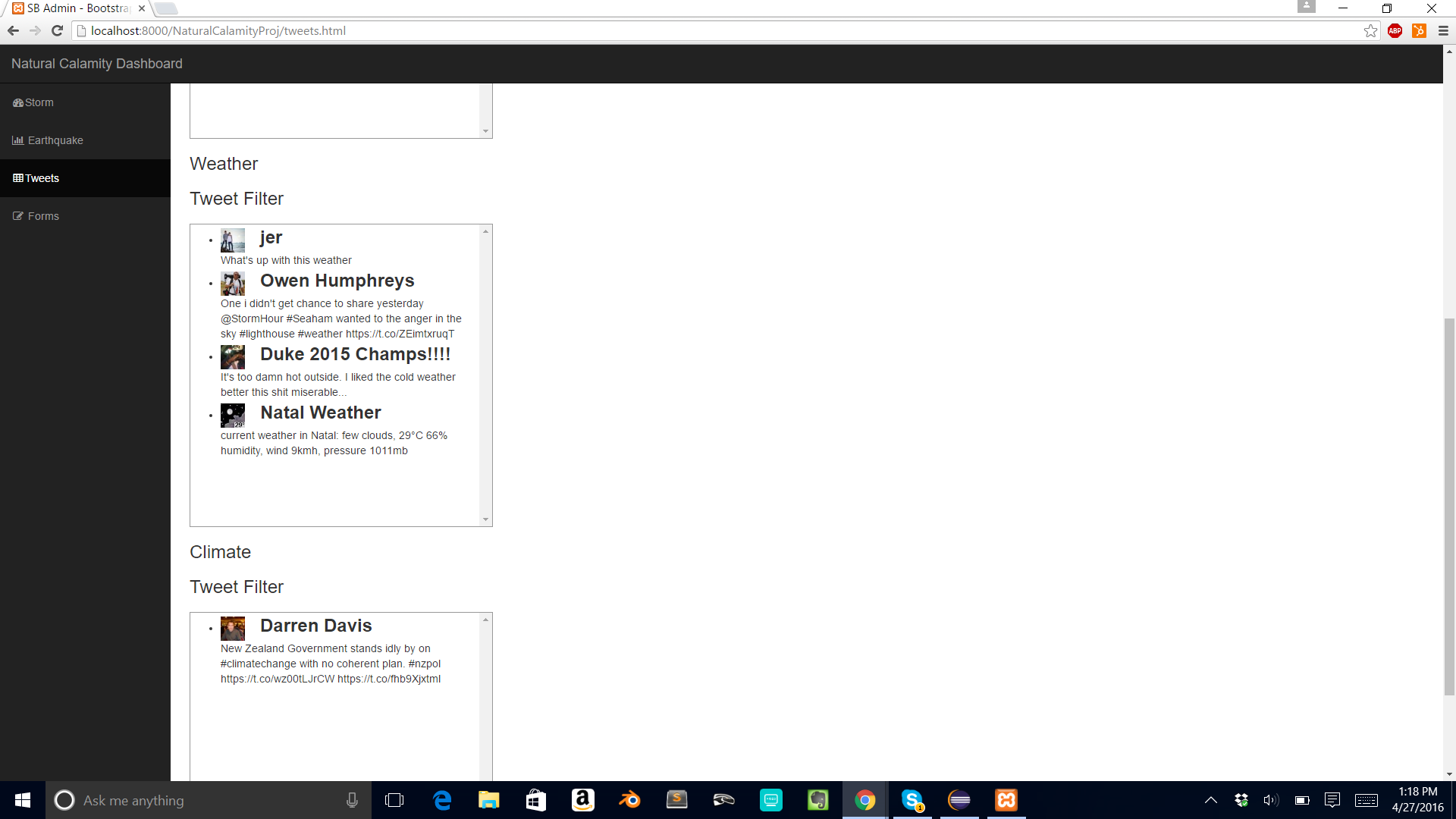
Screenshot 4: Final results for 5 factors



Screenshot 5: Final Results for 4 Factors



Screenshot 6: Final results for 3 Factors



Screenshot 7: Twitter Feed

1. User flow

There are two scopes to the user flow:

1. User can log on to the web application and see the analytics regarding different algorithms implemented. That analytics include the accuracy comparison for different algorithms combined with different factors. Also we are showing comparison related to the predicted values and actual values.
2. The second scope is the Rest API consumption. The different parameters and results are shown in section 4.5 System API design and logic design. For our application we are taking the threshold magnitude value and the city name. The result of this API call is:

* Place – The same place with detailed information
* Radius – the default radius is 100 km
* Window – The window is the period for which the forecast is the result
* Probability – Probability is the probability of the earthquake at least once.

4.5 System API Design and logic design

Get EarthquakeProbability API

For a specific location the Earthquake probability API returns:

1. Latitude and longitude of the location
2. The current expected rate of earthquakes of specified magnitude over a specific time window
3. The current expected probability of experiencing at least one such earthquake

The parameters to the API are:

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Required |
| q | Geographical location to the query, this can be city name, street name etc. Parameter type is String | Yes |
| w | The calculation window (in days) for the forecast. The parameter type is an int. Default value is 1 year i.e 365 days. | No |
| m | The magnitude threshold for the forecast. The parameter type is float. Default value is 6.0 | No |
| r | The calculation radius(in km) for the forecast. The parameter type is integer. Default value is 50 km (approximately 31 miles) | No |

The results returned by API are:

|  |  |
| --- | --- |
| Parameter | Description |
| Location | The resulting parsed location for the calculation:   * Place: * Lat * Lng * radius |
| Forecast | The calculation results:   * Window: * Mag: * Rate: * Prob: |

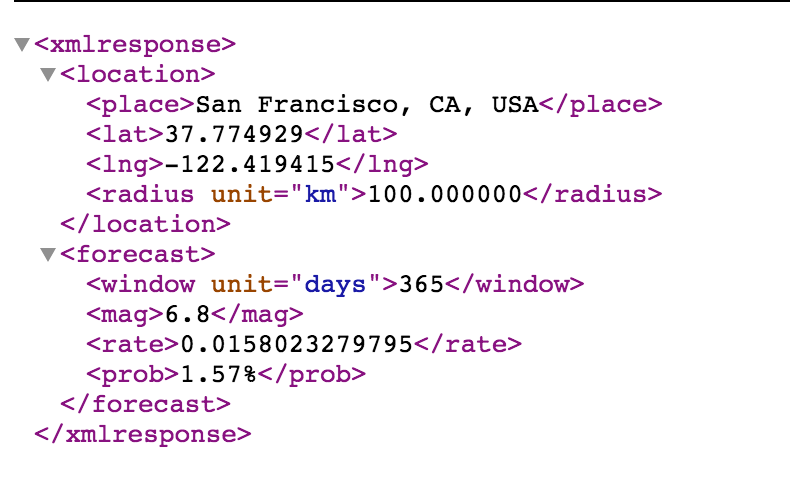
Messages and Error codes:

The following codes give the error codes returned by API:

|  |  |  |
| --- | --- | --- |
| Error code | Description | Resolution |
| 0 | No location was specified | Check if you have provided a geographical location in your API |
| 1 | Location not be found | Check the provided geographical location for spelling mistakes |

**Example**

The following request will return the probability of a magnitude 6.8 earthquake occurring within 100km of San Francisco over the next 365 days



5. System Implementation

5.1. System Implementation Summary

5.1.1 System Design

* Completed %: 100%
* Open issues: No Issues
* Recommended solutions: NA

5.1.2 System Implementation

* Completed %: 90%
* Open issues: Integration in progress, Apache Spark Features implementation
* Recommended solutions: All the team members working to integrate their component as a system

5.1.3 System Testing and Experiment

* Completed %: 70 %
* Open issues: Performance testing and Regression Testing
* Recommended solutions: Selenium, Apache Jmeter and pytest will be used for testing. Team members will verify the end to end flow

5.1.4. Demo

* Completed %: 80%
* Open issues: Apache Spark Features and connecting php with python
* Recommended solutions: Team members are working on the flow to make it work.

5.2. System implementation issues and resolutions:

While developing the project we faced quite a few issues, some of them are mentioned below with their resolution.

1. Data:

Problem: The main issue regarding building the project was to find the proper dataset required for prediction of calamity, which has features and the target required for prediction

Solution: We contacted various organizations namely openhazards.com, National Oceanic and Atmospheric Administration, European-Mediterranean Seismological Centre. Though we got a reply from them, but the datasets where not proper for model building. So we did two things:

1. We simulated datasets for two calamities, earthquake and storm
2. We approached one of the organizations, which gave us access to their REST API, which predicts the earthquake probability.

2) Algorithmic implementation:

Problem: There are different algorithms which algorithm to choose was a difficulty and what library to use was a problem considering the data we have and problem we trying to solve.

Solution: We decided to go with sklearn machine learning library, which is widely used in python, since it provides the algorithms, which we were planning to use. Also python has few good packages for data processing, data cleaning and data analysis. So we choose python as a language of choice and sklearn for machine learning algorithm. We are trying to solve, classification problem in supervised learning.

5.3. **Used technologies and tools**

1. **Python:** We used python as a language of choice, since it provides many packages for data mining and machine learning. We used python for data analysis and model building.
2. **Apache Spark:** Apache spark is an open source cluster-computing framework. We are using Apache spark for processing large data.
3. **Sklearn:** Sklearn is a machine learning library in python which provides functionalities to process and build the models. We used sklearn to implement 5 algorithms:
4. Decision Tree classifier
5. SVM
6. SVM.linearsvc
7. Naïve bayes
8. KNN

1. **Google Charts API:** Google charts API is a tool, which allows you to create charts from some data and embed it with a web page.
2. **PHP:** PHP is server side scripting language used for web development. We used it to fetch data and visualize it through angularjs and google charts.
3. **MySQL:** MySQL is a widely used relational database. We used MySQL to save the result of the algorithms and then visualize though Angular.js and google charts.
4. **Pandas:** We used python pandas library to load the data and analyze it. We can perform various operations like summarize, join, merge etc. on dataframe easily.

1. **Numpy:** Numpy is python library, which can be used for summary stats in a simple way. We used numpy for performing some summary stats.