SCHOOL OF COMPUTER SCIENCE

AND ENGINEERING

CSE3020

DATA VISUALIZATION

PROJECT REPORT

Predicting and Analysing Trends in Natural Disasters

Members:

Sanjit Kumar (18BCE0715)

Nitish G (18BCB0151)

Tanya Warrier (18BCB0102)

Slot: L13, L14

Faculty: Prof. Nalini N

TABLE OF CONTENTS

ABSTRACT

Predictive analysis investigates previous occasions to recognize and remove inhabitants helpless against normal disasters. Huge number of administered and unaided methodologies can be utilized to recognize in danger regions and improve forecasts of future occasions. Furthermore, predictive examination strategies can likewise give knowledge to understanding the financial and human effect of regular disasters. Heavy precipitation forecast is a significant issue for meteorological division and it is firmly connected with the economy and human existence. Here the cataclysmic events we will zero in on are floods, landslides and earthquakes which are experienced by individuals across the globe consistently. In this study we have endeavored to utilize some established datasets from true sources like NASA's landslide index, earthquakes dataset from USG Earthquake Hazard Program and the month to month precipitation dataset from the Indian Government to contemplate the patterns and later use Machine Learning models to foresee the event and harm. All visualizations have been done through a Javascript frontend made utilizing ReactJS with libraries like Nivo, Leaflet and D3.js.

The forecasts were finished utilizing a python backend, with four ML-calculations were run on the equivalent dataset (Linear Regression, Decision Tree, Random Forest and XGBoost) and the best one for each dataset was picked, which for this situation ended up being Random Forest for every one of the three datasets. The frontend gets to Machine Learning models and gets back the expectations for the visual components through a Flask API. In doing this, we have made and examined data visualisation with the cataclysmic event information and utilized them for significant predictions.

Keywords - Linear Regression, Decision Tree, Random Forest, XGBoost, Mean Absolute Error, ReactJS, Nivo, Leaflet, D3.js

INTRODUCTION

I. Natural Disasters

A natural disaster is an event that is a prominent unfavorable occasion coming about because of regular cycles of our planet;A few types of natural disasters are floods, typhoons, twisters, volcanic ejections, tidal waves, storms, and other geologic cycles. A natural disaster event can harm property and also death tolls, and most commonly causes some financial losses afterward.

In current occasions, the split between regular, man-made and man-sped up is very hard to draw with human decisions like engineering, fire, asset the executives or even environmental change possibly assuming a part. An unfavourable occasion won't come down to the level of a calamity in the event that it happens in a region without a weak population. In a region that's weak, like Nepal during the 2015 seismic tremor, a very unfortunate occasion can have heartbreaking results and leave enduring harm, which could take a very long time to fix. The grievous results additionally sway the emotional wellness of affected networks frequently prompting post-awful manifestations. These expanded passionate encounters can be upheld through aggregate preparing, prompting flexibility and expanded local area commitment.

A landslide disaster is depicted as an outward and descending slant development of a bounty of slant shaping materials including rock, soil, counterfeit materials, or a blend of these. A tremor or earthquake is the consequence of an unexpected arrival of energy in the Earth's hull that makes seismic waves. At the Earth's surface, tremors show themselves by vibration, shaking, and now and then dislodging of the ground. Seismic tremors are brought about by slippage inside topographical deficiencies. A flood is a flood of water that 'lowers' land.The EU Floods Directive characterizes a flood as a transitory covering of land that is typically dry with water. In the feeling of 'streaming water', the word may likewise be applied to the inflow of the tides. Flooding may result from the volume of a waterway, like a stream or lake, getting higher than expected, making a portion of the water get away from its typical limits.

II. Objectives of the Project

The aim of the project is to predict and visualise:

1. Magnitude of fatality in landslide prone areas
2. Magnitude of tremor in earthquake prone areas
3. Measure of monthly rainfall for the consecutive month

III. Machine Learning for Prediction

AI is a use of man-made software that gives the framework the ability to consequently take in, learn and improve as a matter of fact without the necessity of being unequivocally customized. Machine learning centers around the advancement of PC programs that can get to information and use it to find out on their own.

Machine learning frameworks can be prepared with the assistance of seismic information to investigate the greatness and example of tremors and foresee the area of quakes and consequential convulsions, subsequently saving a large number of lives. The rainfalls in the earlier years can be recorded and an application can be created to recreate the floods. Machine learning can extraordinarily help in crisis and catastrophe the board endeavors. On the off chance that we can estimate the event of a debacle, it won't just be useful in saving a large number of lives but also additionally moderate the loss of cash and foundation. Continuous satellite pictures could be fed to the model which would prompt better reaction and help to the affected regions.

Linear Regression

Linear Regression is a calculation dependent on supervised learning. It plays out a regression task. This algorithm models an objective prediction point dependent on independent factors. It is more likely used to discover the connection among the various factors and perform an estimation. The varied regression models change based on how the connection among the different independent and the different dependent factors are constantly analyzed with quantity of independent variables being used.

Decision Tree Regression

Original set of nodes, S, is the root node. Every iteration uses the unused attributes to calculate the Entropy (H) using the formula,

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

and Information Gain (IG) using,

$$Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

Validate the results and append a new attribute to the model that stores whether or not the prediction is right. Pass the appended model to XGBoost as input.

Random Forest Regression

Each decision tree has extreme variance, yet when we consolidate every one of them together in equal then the resultant difference is low as every decision tree gets entirely prepared on that specific example information and consequently the output doesn't rely upon one decision tree however various decision trees. On account of a regression, the last output is taken by utilizing the majority casting a vote classifier. On account of a regression, the last result is the mean of the multitude of outputs. This part is Aggregation.

A Random Forest is a gathering procedure equipped for performing both regression and order jobs with the utilization of various decision trees and a method called Bootstrap and Aggregation, usually known as sacking. The essential thought behind this is to join numerous decision trees in deciding the last output as opposed to depending on singular decision trees.

Random Forest has different decision trees as base learning models. We arbitrarily perform row and feature sampling and include testing from the dataset, framing test datasets for each model. This part is called Bootstrap.

XGBoost Regression

XGBoost is an amazing methodology for building managed regression models. This model is an ensemble model that comprises a series of steps that help increase the accuracy in it’s predictions.

- Decision Tree Classification
- Bagging
- Random Forest Classification
- Boosting
- Gradient Boosting

The above steps are processed in order, but in order to prevent overfitting of the data and stopping at the right point, we use 3 parameters regularization parameter, convergence parameter, Similarity Score where $S.S = S.R^2/(N.R * \dagger)$. Hence the training time for this model is reduced significantly.

IV. Advantages of visualising data

Data visualisation helps developers and users dissect reports with respect to deals, promoting techniques, and item interest. From the perspective of investigation, one can zero in on the spaces where it is expected consideration regarding increment benefits, thereby making it more useful. Similar to the reference beforehand, a human cerebrum can handle visuals more effectively than a table with the reports. The representation of information permits chiefs to be advised rapidly of new information bits of knowledge and make fundamental moves for project development. A large amount of convoluted information can give many chances to experiences when we imagine them.

Visualisation allows the users to perceive connections between the information, It gives more prominent importance to it. Researching these models helps while focusing in on express locales that require thought in the data, so they can perceive the significance of those spaces to drive their undertaking forward. Visualising your information in your mind helps rapidly recognize any mistakes in the information. In the event that the provided information will result in recommending some unacceptable activities, the preconceived visualizations in our mind would help recognize wrong information sooner so it tends to be taken out from investigation.

Narration is the reason for a dashboard. By taking time and planning one’s visuals in a significant manner, one can help the intended group of experts handle the story in a solitary look.

Continuously make sure to pass on the story in the most straightforward manner, without unreasonable confounded visuals. In a real world scenario, finding information relationships by using visual portrayals is critical to differentiable experiences. Investigating the experiences is critical for users and developers to set the right way to accomplish the project's objectives. By using data visualization, we can find the most recent patterns in their data to give quality items and recognize issues before they come out. By maintaining a steady patterns, one can invest more energy into expanding benefits for the project.

Scatter Plot

A scatter plot is an illustration of a plot or numerical graph that utilizes Cartesian coordinates to illustrate values for ideally two factors of a dataset. One additional variable can also be displayed if the points are encoded.

Histogram

A histogram is represented by a bar graph-like representation of the dataset. This plot buckets a range of output values into columns along the horizontal x-axis. The vertical y-axis represents the frequencies or percentage of occurrences in the dataset for each value in the column. Histograms can be used to visualize distributions in the dataset.

Choropleth Map

This is a thematic map with predefined areas that could be patterned or coloured with proportion to a feature bound by mathematics is a choropleth map. The feature could stand for a cumulation of gross summary of certain geographical characteristics within separate regions, such as the per-capita income or the population density.

Pie Chart

A circular and statistical graph is termed as a pie chart. A pie chart is a circle that is divided into multiple slices to where each slice would represent a numerical percentage of a feature.The length of the arc of each and every slic present in the chart is directly proportional to the numeric feature that it represents.

World Map with Markers

A world map defines a map that consists of most or every part of the surface of Earth. World maps, due to their size and scale, have to deal with problems involving projection. Maps that are rendered in two dimensions necessarily have to distort the display of the real life three-dimensional surface of the globe of earth.

Line Chart

To show information that continuously changes with time, we implement a line graph. Line graphs are plotted with straight lines that connect points. It is also commonly known as a line chart.

LITERATURE SURVEY

In paper referenced [1], the authors address how landslides are a part of the course of geographical perils in a wide scope of geo-environments. In this study, researchers explore and analyze two best in class AI models, i.e Decision Tree (DT) and Random Forest (RF) ways to deal with the humongous precipitation-triggered landslide events in the Izu-Oshima Volcanic Island, Japan at a local scale.

From the output, a landslide inventory guide is arranged of 44 landslide polygons (10,444 pixels) from aerial photographs and field studies. Twelve causative elements including height, incline point, slant perspective, plan shape, complete curve, compound geographical file, stream power file, distance to waste organization, seepage thickness, distance to topographical limits, lithology and aggregate precipitation were chosen as indicators to carry out the landslide proneness model.

The outcome shows that the Decision Tree and Random Forest models accomplished surprising remarkable performance (AUC > 0.9), creating close to exact susceptibility maps. The general efficiency of Random Forest (AUC = 0.956) is found essentially higher than the Decision Tree (AUC = 0.928) results. Furthermore, the authors noticed that the presentation of Random Forest for demonstrating landslide proneness is powerful despite the fact that the preparation and approval tests are adjusted.

Thinking about the predictions, the authors recommend that both Random Forest and Decision Tree models can be utilized in other comparable non-eruption-related landslide research in the tephra-stored rich volcanoes, as they are able to do quickly creating precise and stable LSM maps for hazard alleviation, management practices, and dynamic decision making. In addition, the Random Forest based model is promising and enough to be prescribed as a strategy to plan local landslide vulnerability.

Visualizing large amounts of data in a single and big uniform static visualization, defeats the purpose and advantages of utilizing such tools. As a remedy to tackle the same an advanced aggregate computation methodology [2] has been proposed.

The methodology first uses an approach of visualization-based data separation as well as aggregation to transform large data. This is done while keeping intact operability of data. Next, the aggregated data are displayed with a minimum size of only one pixel. This ensures visibility of important but miniscule information in the visual analysis. The aggregated data are mapped to visual primitives thinner than one pixel.

After the computation a web-visualization system is explored. The authors used a D3-based implementation to improve the consecutive visualizations.The large dataset is divided into two categories for the purpose of visualization. The first category includes items that will be matched to visual primitives lesser than or equal to one pixel on the screen. The second includes data that are mapped to primitives larger than one pixel. It is important to note that the data in the first category are suitably aggregated such that they overlap on the screen. The latter category consists of non-overlapping primitives. The purpose of a minimum size limitation is essential for keeping important but small information visible for visualization analysis or exploration purposes.

This advanced methodology proposed can prove to be of great use in visualizing large volumes of data records in the meteorological systems.

Enclosure partitioning approaches like the Treemaps, have always proved their effectiveness for visualizing large hierarchical structures within a limited display area. A very high number of the Treemaps techniques that are being used to show the structural relations do not use node-links. This paper introduces a new visualization that is a tree based technique known as Drawer-Tree that can be used for organization, to present the structure, and interrelation of big data. This approach utilizes the existing display space with traditional node link visualization. The novel method implemented helps one to visualize the tree structures with high scalability.

The Drawer tree technique displays structures that resemble drawers in a large cabinet and also allows one to clearly distinguish between parent nodes and leaf nodes. The parent nodes are considered as “drawers”, similar to containers that hold leaf nodes.

The drawer tree is a 2D tree that expands into two orthogonal directions and it aims at addressing the following design criteria:

- To improve space criteria
- To provide a clear hierarchy presentation
- To enhance and help differentiate between parent and leaf nodes
- To display useful information such as space and colors with attributed information
- Provide interactive design to allow one to explore the hierarchy
- Allowing easy scalability by allowing a user to handle big datasets

The technical aspects of the drawer trees are:

1. Non-leaf Nodes - This represents a parent node
2. Leaf Nodes - This represents a leaf node
3. Data Visualization

This paper demonstrates a potential hierarchical visualization in 2D, called Drawer Tree. Experiments conducted by the authors show us that the drawer tree algorithm has the capability to visualize large data sets. The various provided interactions and attributed properties help improve readability and aspect ratio.This method utilized the fractal tree-based approach which allows the branches to expand in 4 directions. The drawer tree’s branches are well ordered and therefore the utilization of the existing space is higher. Furthermore, the tree structures that are presented in the traditional node-link visualizations are properly executed in this visualization as the branches at various positions are coded with color and are arranged in the right and desired orientation.

Visualising large amounts of data is often complex because data in extremely large volumes tends to behave a bit differently in software systems. Real Time analysis needs large amounts of data and they don't make sense on their own. So visualisation of key dimensions show the viewer the details that need to be highlighted upon. In [4] , A.Imawan et al. propose a timeline visualisation system that documents the events such as traffic congestion and traffic accidents in the order of occurrence.

The key context that can be derived out of such a visualisation is the relationship between two or more incidents. We can know which accidents happen together, and which accidents affected or caused other accidents. This visualisation is proposed to be a part of a web interface with menu, query and display panels. The LinkIDs are provided by the user as a list and the web interfaces makes user experience better by providing a dropdown suggestion based on the location of the user.

Adil Usman [5] reviews and analyses the disaster mitigation and safety situation in India. He talks about the Natural disaster Management field in India in the past 3 decades. He further talks about symptomatic climate change that is visible throughout the subcontinent. This is marked by increasingly hot and extended summers in the plains, faster melting glaciers and decreasing plant and animal life in the Himalayas. The NDMF in India is comprised of

1. National Executive Committee
2. Natural Disaster Response Force
3. Cabinet Committee on Security
4. Natural Institute of Disaster Management

He goes on to cite the various key data points released by the above institutions via visualisations - bar graphs, pie charts and more. The systematic approach to reduce natural disaster has to do with reducing hazardous output into the environment from artificial production and processing units. The political and administrative efforts to raise awareness and mitigate these problems are taken by the Ministry of Environment that has ties with the Ministry of Agriculture, Ministry of Health and Family and the Ministry of Science and Technology.

Data mining visualisation is a significant component of big data visualisation. The paper by Praveen Kumar et al. studies the impact of nature-inspired algorithms in machine learning like the artificial immune system, the CLONALG etc, in the context of mining data for big data analysis. The aspects of data allocation and resource sharing are important steps before data can be extracted and visualised.

Organisational big data is a part of a larger business model where information is indexed and utilised in an expansive manner. These techniques are replicated in open source tech like the Apache Hadoop framework and file system. Visualising the data with data and other empirical aspects and integrating them with the IoT domain. The scope of IoT integration is practically unlimited, from healthcare, wearable devices, sensor technology, industrial devices, smart cities to smart homes.

The paper discusses the security enhancement and analysis based on level. Security trends in terms of hybridisation and data sharing are studied. To find out the current trend in secure data handling and features of adaptability. The client server architecture for computational capability is researched and discussed for adaptable data chunking. This adaptability is a key concept in nature-inspired genetic algorithms such as CLONALG.

The key idea in most general natural algorithms is to draw from the evolutionary mechanisms of adaptive selection based on the environment (here is the cloud server landscape). Sharma et al. in [6] have discussed the methods to use cloud as a depository. The cyber data is stored and applied as a service. Here data security is crucial since the data is stored from different environments. And to this end data encryption algorithms can be used for security and privacy.

PROPOSED WORK

I. Datasets

a. Landslide Fatality Prediction: NASA Global Landslide Catalog

The Global Landslide Catalog (GLC) was developed with the goal of identifying rainfall-triggered landslide events around the world, regardless of size, impacts or location. The GLC considers all types of mass movements triggered by rainfall, which have been reported in the media, disaster databases, scientific reports, or other sources.

b. Earthquakes Dataset from USG Earthquakes Hazards Program

This data set is taken from USGS (U.S Geological Survey). The USGS provides reliable scientific information to describe and understand the Earth; minimize loss of life and property from natural disasters; manage water, biological, energy, and mineral resources; and enhance and protect our quality of life. It also monitors and reports on earthquakes, assesses earthquake impacts and hazards, and conducts targeted research on the causes and effects of earthquakes.

c. Sub Division Monthly Rainfall Dataset from IND GOV

Contains month wise all India rainfall data. The subdivision wise rainfall and its departure from normal for each month and season has been provided in the data. Released under NDSAP by the ministry of earth sciences and department of meteorology.

II. Machine Learning Algorithms

In this project, the preprocessing methods adopted are as follows.

a. Drop the irrelevant features columns of the dataset
b. Drop the rows consisting NaN values in the dataset
c. Drop the rows whose value categories are 'unknown'
d. Perform Label Encoding on all the categorical data columns
e. Normalize the continuous values of the dataset for better performance

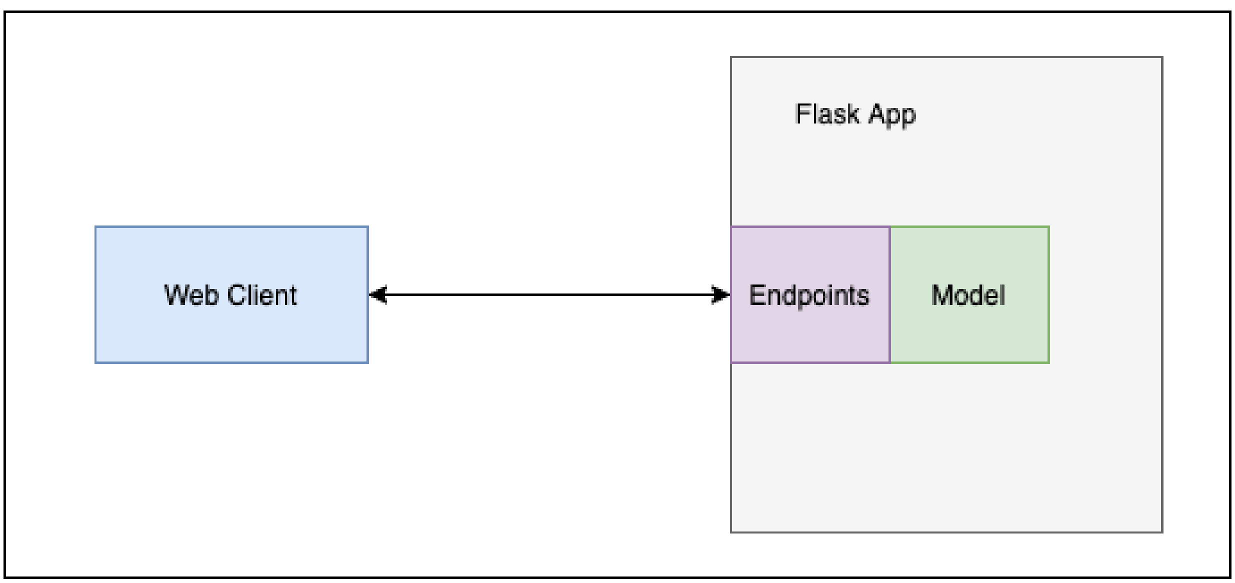
Four machine learning regression algorithms were implemented on each of the three datasets to predict landslide fatality, earthquake magnitude and monthly rainfall.

a. Linear Regression
b. Decision Tree Regression
c. Random Forest Regression
d. XGBoost Regression

Each model was evaluated based on the mean absolute error. The model evaluation that resulted in the least error was selected for visualization in all three cases.

III. Application Programming Interface

The API implements a simple random forest regression classifier on the above datasets to predict the future disaster levels. The model is used to compute and deliver the data via a Flask API.



IV. Data Visualization

The visualisations are a part of a Javascript dashboard web interface. It is created using the React.js library. The dashboard visualises the various trends and characteristics of the dataset to study what kind of predictions would make the most sense. Later, once the predictions are determined, the data is again visualised in various forms to create interactive visual elements. The react dependencies used for the purpose of visualisations are Nivo Charts and Leaflet that are actually layered on D3.js. The inferences from the visualisations are mentioned in the result section.

The dashboard has the following multiple components:

1. Static Visualisations
   a. Landslide Data Visualisations
      - Bar Graph - Landslide Triggers vs Fatalities - Provides insight about what natural or artificial events lead to the most fatalities.
      - Pie Chart - Total Fatalities vs Country - Studies the composition of deaths caused by major landslides across the world and their distribution based on countries.
      - Scatter Plot - Population of affected region vs fatality count vs Size of Landslide - This plot helps find outliers from the trend where high fatality rates are from affected highly populous regions.

   b. Rainfall Data Visualisation
      It is an interactive GeoViz of the Indian State Map. On clicking the individual states, they are coloured based on the relative amount of rainfall they received in the designated time frame. The time frame can be changed via a slider component in the top left.

   c. Earthquake Data Visualisation
      Another interactive GeoViz of the World Political Map. It highlights the regions that are earthquake prone. By changing the checkbox parameters in the top right - Google Streets, Google Satellite and a Black and White map.

2. ML-predictions Visualisations

   a. Landslide Fatality Prediction

      - Predicts the average number of fatalities resulting from the different categories of landslides.
      - Provides insight into what kind of triggers might cause the respective categories of landslides and what locations are vulnerable to them.
      - Complex and Creep Landslides seem to be very fatel with a score of 8.48 each.

   b. Rainfall Prediction

      - The rainfall of any location for any month can be predicted given the rainfall of the previous three months in the same location
      - The prediction is plotted as a line chart with months in the x axis and mm of rainfall in y axis
      - The input is obtained via text boxes from the user

   c. Earthquake Prediction

      - The magnitude of an earthquake of any region is predicted using the geographical location, RMS and type of earthquake.
      - The obtained result is the magnitude in the Richter scale.
      - The event type location source, mag source, event type, location and RMS are taken from the user and a magnitude is given back. The predictions are plotted as a line graph.

RESULTS

I. Mean Absolute Error Observations

In statistics, mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

Magnitude of fatality of landslide prone areas -

Linear Regression:	8.329373033190977
Decision Tree Regression:	3.6047164514317798
Random Forest Regression:	3.556945536215609
XGBoost Regression:	3.719126223664415

Magnitude of tremor in earthquake prone areas -

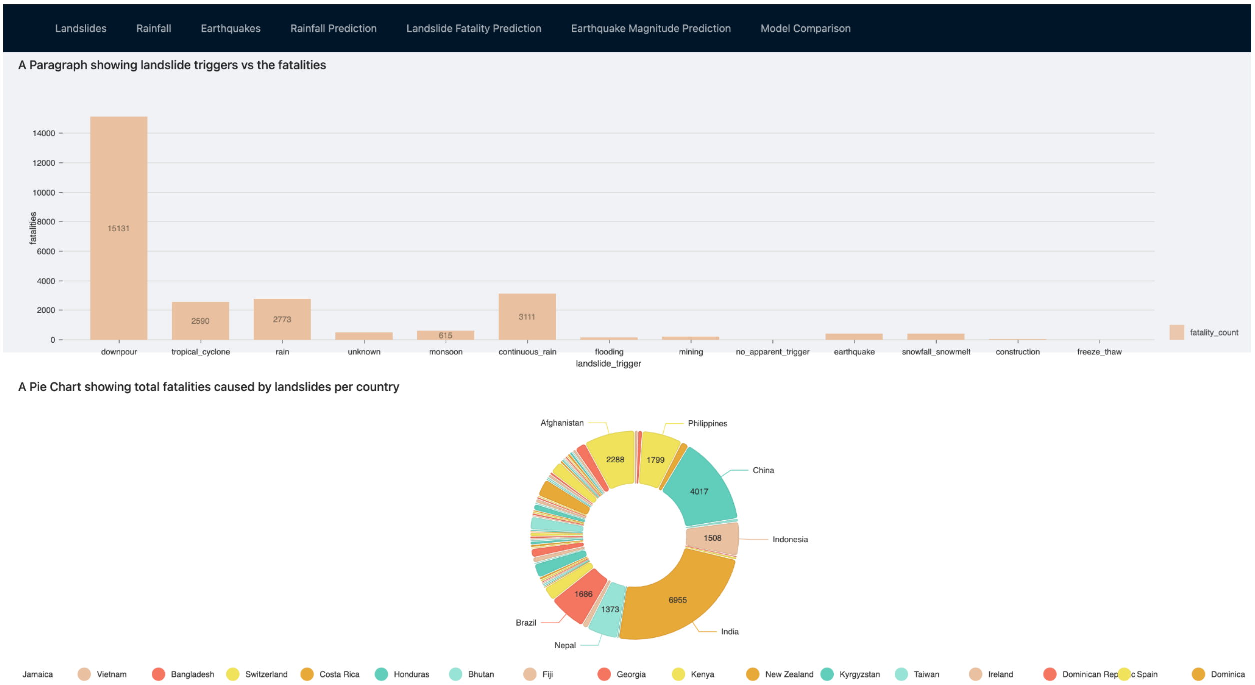
Linear Regression:	0.5057420205050159
Decision Tree Regression:	0.40604938288224124
Random Forest Regression:	0.3003447294107278
XGBoost Regression:	0.34667492567787556

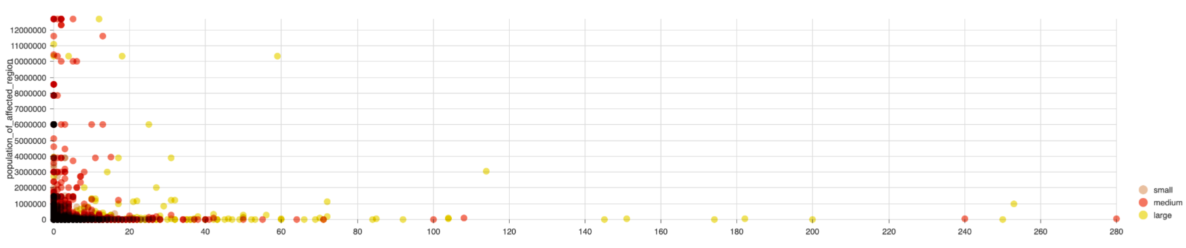
Measure of monthly rainfall for the consecutive month -

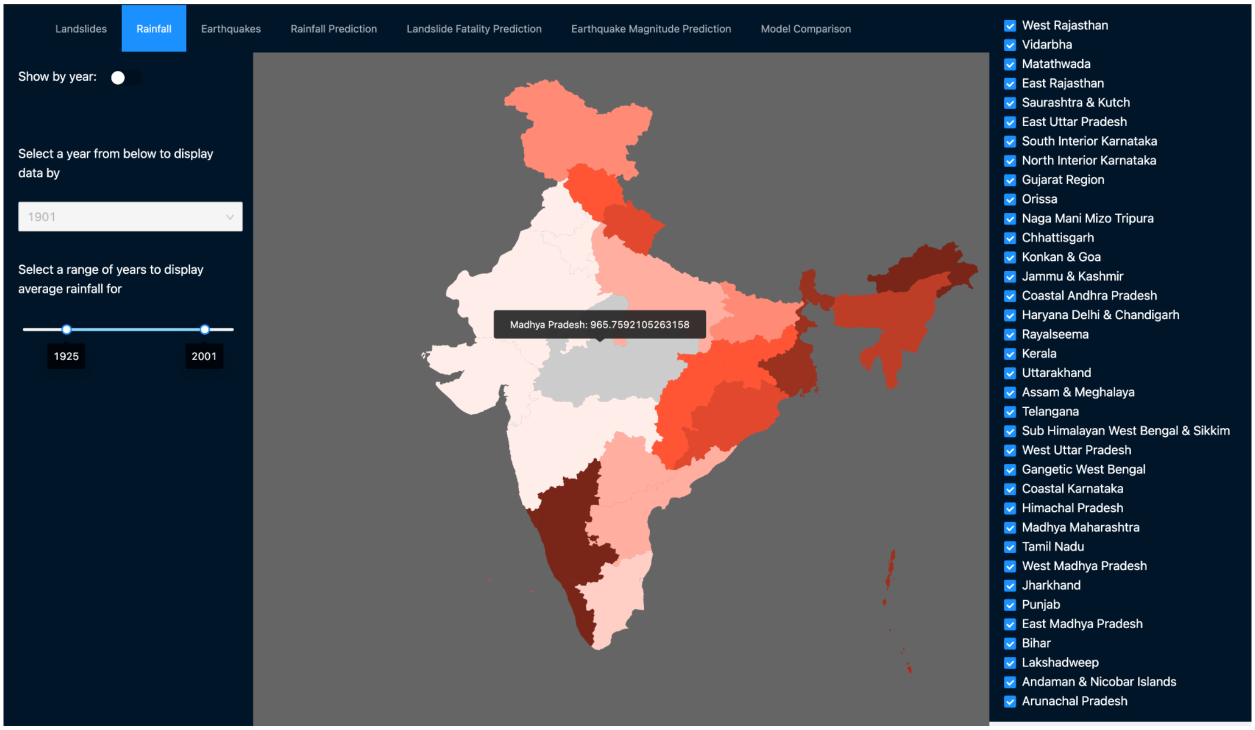
Linear Regression:	0.5341147443675478
Decision Tree Regression:	0.7338735145768813
Random Forest Regression:	0.5252136029181004
XGBoost Regression:	0.527699950263565

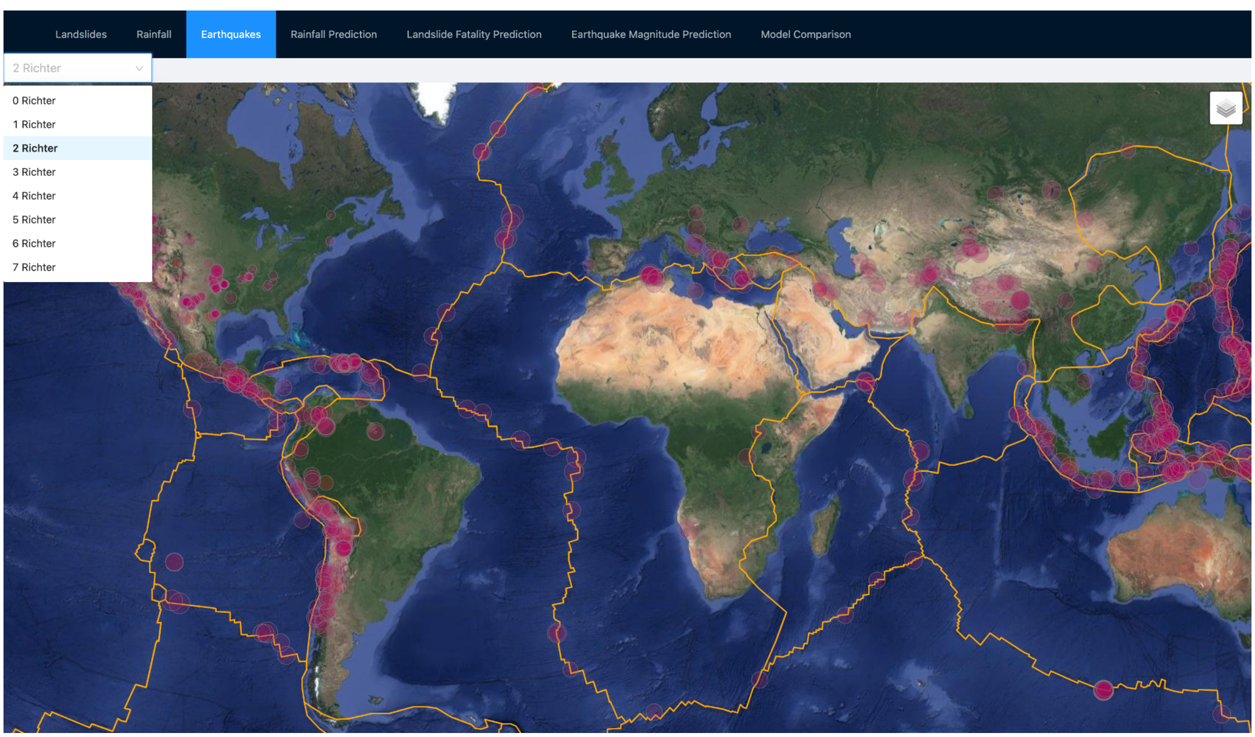
It is observed that for all the sub components of the project, Random Forest Regression gives the least Mean Absolute Error, and fits best for the prediction model.

II. Data Visualization

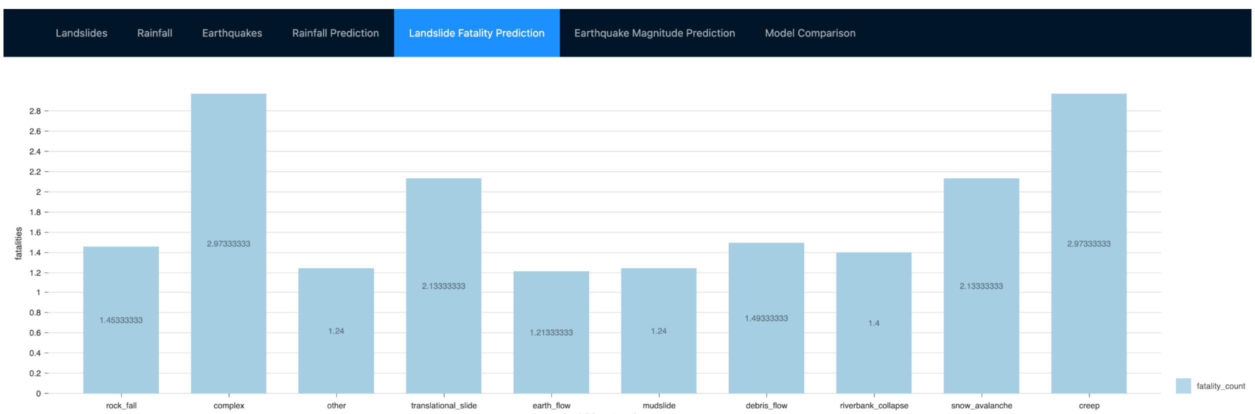


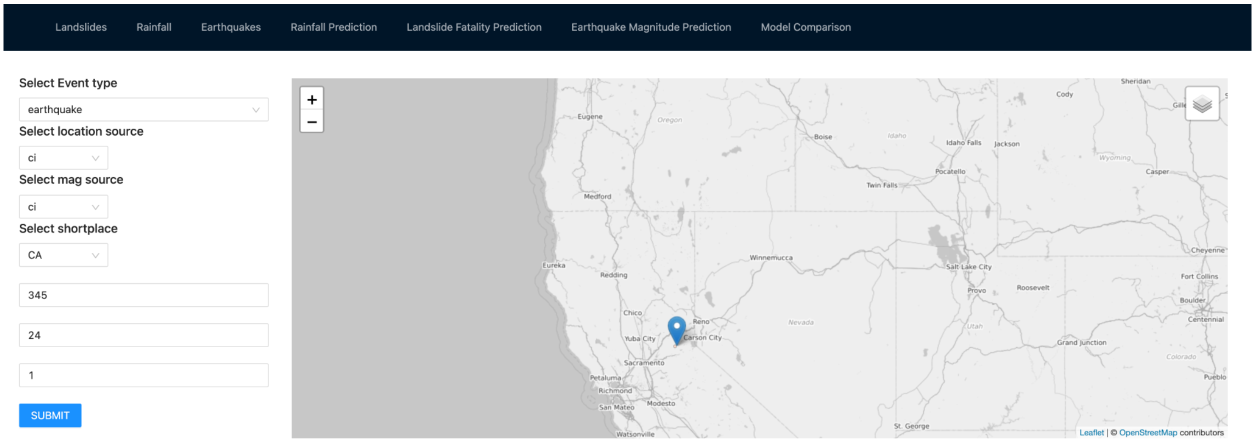


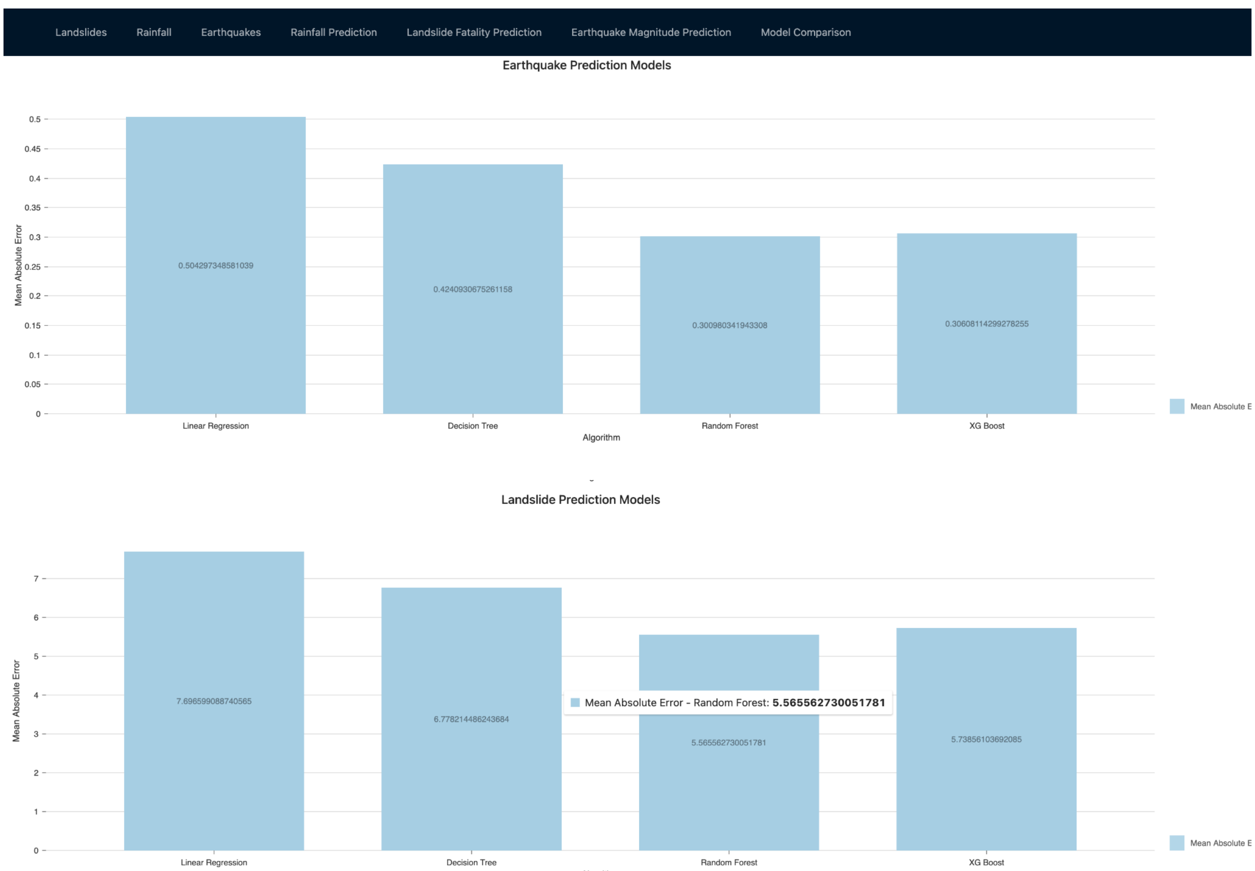

Landslides
Rainfall
Earthquakes
Rainfall Prediction
Landslide Fatality Prediction
Earthquake Magnitude Prediction
Model Comparison
Show by year:
Select a year from below to display data by
1901
Select a range of years to display average rainfall for
1925
2001
Madhya Pradesh: 965.7592105263158
West Rajasthan
Vidarbha
Matathwada
East Rajasthan
Saurashtra & Kutch
East Uttar Pradesh
South Interior Karnataka
North Interior Karnataka
Gujarat Region
Orissa
Naga Mani Mizo Tripura
Chhattisgarh
Konkan & Goa
Jammu & Kashmir
Coastal Andhra Pradesh
Haryana Delhi & Chandigarh
Rayalseema
Kerala
Uttarakhand
Assam & Meghalaya
Telangana
Sub Himalayan West Bengal & Sikkim
West Uttar Pradesh
Gangetic West Bengal
Coastal Karnataka
Himachal Pradesh
Madhya Maharashtra
Tamil Nadu
West Madhya Pradesh
Jharkhand
Punjab
East Madhya Pradesh
Bihar
Lakshadweep
Andaman & Nicobar Islands
Arunachal Pradesh

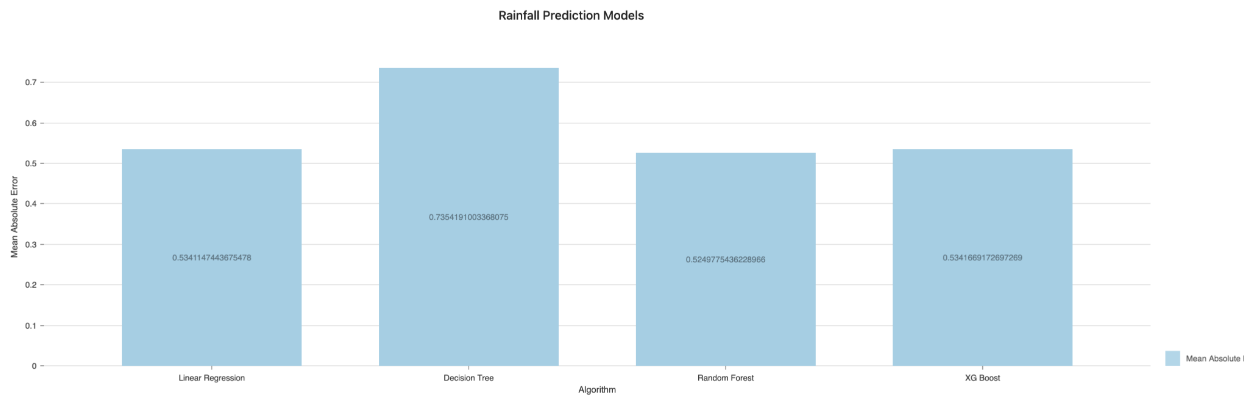

Landslides
Rainfall
Earthquakes
Rainfall Prediction
Landslide Fatality Prediction
Earthquake Magnitude Prediction
Model Comparison
2 Richter
0 Richter
1 Richter
2 Richter
3 Richter
4 Richter
5 Richter
6 Richter
7 Richter


Landslides
Rainfall
Earthquakes
Rainfall Prediction
Landslide Fatality Prediction
Earthquake Magnitude Prediction
Model Comparison
Month 1:
123
Month 2:
345
Month 3:
234
SUBMIT
Rainfall in mm
340
320
300
280
260
240
220
200
180
160
140
month_1
month_2
month_3
month_4
Months
rainfall


Landslides
Rainfall
Earthquakes
Rainfall Prediction
Landslide Fatality Prediction
Earthquake Magnitude Prediction
Model Comparison
fatalities
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1.45333333
2.97333333
1.24
2.13333333
1.21333333
1.24
1.49333333
1.4
2.13333333
2.97333333
rock_fall
complex
other
translational_slide
earth_flow
mudslide
debris_flow
riverbank_collapse
snow_avalanche
creep
landslide_categories
fatality_count


Landslides
Rainfall
Earthquakes
Rainfall Prediction
Landslide Fatality Prediction
Earthquake Magnitude Prediction
Model Comparison
Select Event type
earthquake
Select location source
ci
Select mag source
ci
Select shortplace
CA
345
24
1
SUBMIT
Leaflet | © OpenStreetMap contributors


Landslides
Rainfall
Earthquakes
Rainfall Prediction
Landslide Fatality Prediction
Earthquake Magnitude Prediction
Model Comparison
Earthquake Prediction Models
Mean Absolute Error
0.504297348581039
0.4240930675261158
0.300980341943308
0.30608114299278255
Linear Regression
Decision Tree
Random Forest
XG Boost
Algorithm
Landslide Prediction Models
Mean Absolute Error
7.696599088740565
6.778214486243684
Mean Absolute Error - Random Forest: 5.565562730051781
5.565562730051781
5.73856103692085
Linear Regression
Decision Tree
Random Forest
XG Boost
Algorithm



CONCLUSION

In this project we have attempted to use some well established datasets from authentic sources like NASA's landslide catalog, earthquake dataset from USG Earthquake Hazard Program and the monthly rainfall dataset from the Indian Government to study the trends and later use Machine Learning models to predict the occurrence and damage. All visualisations have been done via a Javascript frontend created using ReactJS with visualisation libraries like Nivo, Leaflet and D3.js.

The predictions were done using a python backend, where 4 ML-algorithms were run on the same dataset (Linear Regression, Decision Tree, Random Forest and XGBoost) and the best one for each dataset was chosen, which in this case turned out to be Random Forest for all 3 datasets. The frontend accesses ML models and gets back the predictions for the visual elements via a Flask API that runs on the port 4000. In doing this, we have created and analysed visualisations with the natural disaster data and used them for meaningful predictions.

FUTURE WORK

Future work in this project could be to improve the prediction and compare existing models with deep neural networks to improve accuracy. Deployment of the application to any popular hosting service is also within the radar. More interactive visualisations can be done with React. We have made the project open source and welcome any improvements wherever possible.

REFERENCES

1. Dou, J., Yunus, A. P., Bui, D. T., Merghadi, A., Sahana, M., Zhu, Z., ... & Pham, B. T. (2019). Assessment of advanced random forest and decision tree algorithms for modeling rainfall-induced landslide susceptibility in the Izu-Oshima Volcanic Island, Japan. *Science of the total environment*, *662*, 332-346.

2. Bao, F., & Chen, J. (2014, May). Visual framework for big data in d3. js. In *2014 Ieee Workshop on Electronics, Computer and Applications* (pp. 47-50). IEEE.

3. Yang, Y., Dou, N., Zhao, S., Yang, Z., Zhang, K., & Nguyen, Q. V. (2014, March). Visualizing large hierarchies with drawer trees. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing* (pp. 951-956).

4. A. Imawan and J. Kwon, "A timeline visualization system for road traffic big data," 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 2015, pp. 2928-2929, doi: 10.1109/BigData.2015.7364125.

5. A. Usman, "Integrated disaster risk management in Indian environment: Prediction, prevention and preparedness," 2017 IEEE Global Humanitarian Technology Conference (GHTC), 2017, pp. 1-6, doi: 10.1109/GHTC.2017.8239246.

6. P. K. Kotturu and A. Kumar, "Data Mining Visualization with the Impact of Nature Inspired Algorithms in Big Data," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 664-668, doi: 10.1109/ICOEI48184.2020.9142979.

APPENDIX A

I. Earthquake Magnitude Predictor Script

```
# Read dataset
earthquake_df = pd.read_csv('Datasets/earthquake-all-month.csv')

earthquake_df['short place']=[re.findall(r'\w+',i)[-1] for i in earthquake_df['place']]
earthquake_df.dropna(subset=['mag'],inplace=True)

# Feature vector
features=[i for i in earthquake_df.columns if earthquake_df[i].isna().sum()==0] # features
include place, type and source

for i in ['mag','place','time','id','updated','net','magType']:
    features.remove(i)

X=earthquake_df[features]
y=earthquake_df['mag'] # predict magnitude

#Normalize the depth feature
X[['depth']] = StandardScaler().fit_transform(X[['depth']])

# Segregate categorical data
categorical = []
for i in features:
    if earthquake_df[i].dtype=="object":
        categorical.append(i)

# Encode the data
from sklearn import preprocessing
label_maps = {}
```

```
for i in categorical:
    le = preprocessing.LabelEncoder().fit(X[i])
    X[i]=le.transform(X[i])
    d = dict(zip(le.classes_, le.transform(le.classes_)))
    label_maps[i] = d

# Train and test split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.10)

# Random forest regressor
clf = RandomForestRegressor(n_estimators=100, criterion='mse', max_depth=None)
clf.fit(X_train, y_train)

def predictor(my_array):
    enc_array = my_array[0:4]
    labels = ['type', 'depthError', 'status', 'locationSource', 'magSource', 'short place']
    i = 4
    for label in labels:
        if label == 'depthError':
            enc_array.append(my_array[5])
            i +=1
            continue
        t = label_maps[label][my_array[i]]
        i += 1
        enc_array.append(t)

    y_pred = clf.predict([enc_array])
    return y_pred[0]

def get_mae():
    reg = LinearRegression()
```

```
    reg.fit(X_train, y_train)
    y_pred_reg = reg.predict(X_test)

    dec = DecisionTreeRegressor()
    dec.fit(X_train, y_train)
    y_pred_dec = dec.predict(X_test)

    clf = RandomForestRegressor(n_estimators=100, criterion='mse', max_depth=None)
    clf.fit(X_train, y_train)
    y_pred_clf = clf.predict(X_test)

    xgb = XGBRegressor()
    xgb.fit(X_train, y_train)
    y_pred_xgb = xgb.predict(X_test)

    return [mean_absolute_error(y_test, y_pred_reg), mean_absolute_error(y_test, y_pred_dec), mean_absolute_error(y_test, y_pred_clf), mean_absolute_error(y_test, y_pred_xgb)]
```

II. Landslide Fatality Predictor Script

```
# Read Dataset
landslide_df = pd.read_csv('Datasets/NASA_Global_Landslide_Catalog.csv')

# Drop unwanted features
landslide_df = landslide_df.drop(['source_name', 'source_link','event_id', 'event_date','event_time', 'event_title', 'event_description', 'location_description','storm_name','photo_link', 'notes', 'event_import_source','event_import_id','country_code','submitted_date', 'created_date', 'last_edited_date','admin_division_name','gazeteer_closest_point', 'gazeteer_distance','injury_count'], axis = 1)

# Drop unknown categories

to_remove = landslide_df[ (landslide_df['landslide_category'] == 'unknown') ].index
landslide_df = landslide_df.drop(to_remove)
to_remove = landslide_df[(landslide_df['location_accuracy'] == 'unknown')].index
landslide_df = landslide_df.drop(to_remove)

# Replace or drop unknown/NaN values

landslide_df = landslide_df.dropna(subset=['location_accuracy', 'landslide_category','landslide_trigger','landslide_size','landslide_setting','country_name'])

# Determine feature and target vectors
X_features = list(landslide_df.columns)
X_features.remove('fatality_count')
X = landslide_df[X_features]
y = landslide_df['fatality_count']
y = y.fillna(y.median()) # deal with na
```

```
# Encoding of categorical data
categorical = []
for i in X_features:
    if landslide_df[i].dtype=="object":
        categorical.append(i)
label_maps = {}
for i in categorical:
    le = preprocessing.LabelEncoder().fit(X[i])
    X[i]=le.transform(X[i])
    d = dict(zip(le.classes_, le.transform(le.classes_)))
    label_maps[i] = d

# Train and test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

# Perform regression
clf = RandomForestRegressor(n_estimators=150, max_depth = None, criterion='mse')
clf.fit(X_train, y_train)

def predictor(my_array):


    enc_array = [[]]
    print('MY ARRAY: ',my_array)
    labels = ['location_accuracy',        'landslide_category',    'landslide_trigger',
'landslide_size', 'landslide_setting', 'country_name',
'admin_division_population','longitude','latitude']
    i = 0
    for label in labels:
        if(label=='admin_division_population' or label=='longitude' or label=='latitude'):
            enc_array[0].append(float(my_array[i]))
            i += 1
```

```
        else:
            t = label_maps[label][my_array[i]]
            i += 1
            enc_array[0].append(t)

        print('ENCODED ARRAY: ',enc_array)

    y_pred = clf.predict(enc_array)
    return y_pred

def get_mae():
    reg = LinearRegression()
    reg.fit(X_train, y_train)
    y_pred_reg = reg.predict(X_test)

    dec = DecisionTreeRegressor()
    dec.fit(X_train, y_train)
    y_pred_dec = dec.predict(X_test)

    clf = RandomForestRegressor(n_estimators=150, max_depth = None, criterion='mse')
    clf.fit(X_train, y_train)
    y_pred_clf = clf.predict(X_test)

    xgb = XGBRegressor()
    xgb.fit(X_train, y_train)
    y_pred_xgb = xgb.predict(X_test)

    return [mean_absolute_error(y_test, y_pred_reg), mean_absolute_error(y_test, y_pred_dec), mean_absolute_error(y_test, y_pred_clf), mean_absolute_error(y_test, y_pred_xgb)]
```

III. Monthly Rainfall Predictor Script

```
# Read dataset
rainfall_df = pd.read_csv('Datasets/rainfall_india_1901-2017.csv')

# Deal with NaN values
rainfall_df.fillna(value = 0, inplace = True)

#Normalize the continuous values:
rainfall_df[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT',
'NOV', 'DEC', 'ANNUAL', 'JF', 'MAM', 'JJAS', 'OND']] =
StandardScaler().fit_transform(rainfall_df[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',
'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'JF', 'MAM', 'JJAS',
'OND']])

# Split train and test sets
div_data = np.asarray(rainfall_df[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
'AUG', 'SEP', 'OCT', 'NOV', 'DEC']])

X = None; y = None
for i in range(div_data.shape[1]-3):
   if X is None:
      X = div_data[:, i:i+3] # Three consecutive months
      y = div_data[:, i+3] # Next (fourth) month
   else:
      X = np.concatenate((X, div_data[:, i:i+3]), axis=0) # Three consecutive months
      y = np.concatenate((y, div_data[:, i+3]), axis=0) # Next (fourth) month

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)

# Perform Regression
rf = RandomForestRegressor(n_estimators = 200, max_depth=10)
```

```
rf.fit(X_train, y_train)

def predictor(my_array):
    y_pred = rf.predict(my_array) #array of 1 value
    return y_pred[0]

def get_mae():
    reg = LinearRegression()
    reg.fit(X_train, y_train)
    y_pred_reg = reg.predict(X_test)

    dec = DecisionTreeRegressor()
    dec.fit(X_train, y_train)
    y_pred_dec = dec.predict(X_test)

    clf = RandomForestRegressor(n_estimators=200, max_depth = 10, criterion='mse')
    clf.fit(X_train, y_train)
    y_pred_clf = clf.predict(X_test)

    xgb = XGBRegressor()
    xgb.fit(X_train, y_train)
    y_pred_xgb = xgb.predict(X_test)

        return [mean_absolute_error(y_test, y_pred_reg), mean_absolute_error(y_test,
y_pred_dec), mean_absolute_error(y_test, y_pred_clf), mean_absolute_error(y_test,
y_pred_xgb)]
```

IV. Source File

```
import React from 'react';
import { BrowserRouter as Router, Switch, Route, Link } from "react-router-dom"
import Visualisations from './Components/Visualisations/Visualisations'
import RainfallViz from './Components/RainfallViz/RainfallViz';
import EarthquakeViz from './Components/Earthquake/Earthquake';
import MlViz from './Components/Visualisations/MlVisualization';
import MlVizFatality from './Components/Visualisations/MlVisualisationFatality';
import EarthquakePred from './Components/Visualisations/earthquakePred';
import ModelComparison from './Components/Visualisations/modelComparison';
import { Layout, Menu, Breadcrumb } from 'antd';
import { UserOutlined, LaptopOutlined, NotificationOutlined } from '@ant-design/icons';
const { SubMenu } = Menu;
const { Header, Content, Footer, Sider } = Layout;

function App() {
  return (

    <div className="App" style={{ height:"100vh", width:"100vw"}}>

     <Router>
      <Layout>
     <Header style={{ position: 'fixed', zIndex: 1, width: '100%' }} className="header">
     <div className="logo" />
     <Menu theme="dark" mode="horizontal" defaultSelectedKeys={['0']}>
      <Menu.Item key="1">
        Landslides
        <Link to="/" />
      </Menu.Item>
      <Menu.Item key="2">
```

```
      Rainfall
      <Link to="/rainfall" />
    </Menu.Item>
    <Menu.Item key="3">
      Earthquakes
      <Link to="/earthquake" />
    </Menu.Item>
    <Menu.Item key="4">
      Rainfall Prediction
      <Link to="/ml" />
    </Menu.Item>
    <Menu.Item key="5">
      Landslide Fatality Prediction
      <Link to="/mlfat" />
      </Menu.Item>
      <Menu.Item key="6">
      Earthquake Magnitude Prediction
      <Link to="/earthpred" />
    </Menu.Item>
    <Menu.Item key="7">
      Model Comparison
      <Link to="/modelcomparison" />
    </Menu.Item>
  </Menu>
</Header>
<Content>
    <Switch>
      <Route exact path="/">
        <Visualisations />
      </Route>
      <Route exact path="/rainfall">
```

```
        <RainfallViz />
      </Route>
      <Route exact path="/earthquake">
        <EarthquakeViz/>
      </Route>
      <Route exact path="/ml">
        <MlViz/>
      </Route>
      <Route exact path="/mlfat">
        <MlVizFatality/>
        </Route>
      <Route exact path="/earthpred">
        <EarthquakePred/>
      </Route>
      <Route exact path="/modelcomparison">
        <ModelComparison/>
      </Route>
     </Switch>
     </Content>
     </Layout>
    </Router>


  </div>
 );
}

export default App;
```