

# **FLOOD FORECASTING USING MACHINE LEARNING**

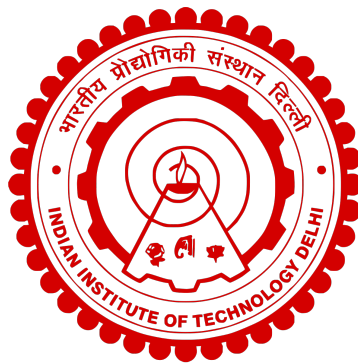
Submitted By

**Saurabh Tiwari (2020CE10295)**

Supervised by

**Prof A.K. Keshari**

A report of CVD411 - Project Part 1 submitted in partial  
fulfillment of the requirements of the degree of Bachelor of  
Technology



Department of Civil Engineering  
Indian Institute of Technology Delhi

November, 2023

# **CERTIFICATE**

I do certify that this report explains the work carried out by me in the Course CVD411 - Project Part 1 under the overall supervision of **Prof A.K. Keshari**. The contents of the report including text, figures, tables, computer programs, etc. have not been reproduced from other sources such as books, journals, reports, manuals, websites, etc. Wherever limited reproduction from another source had been made, the source had been duly acknowledged at that point and also listed in the References.

**Saurabh Tiwari**

November 27, 2023

This is to certify that the report submitted by **Saurabh Tiwari** describes the work carried out by him in the Course CVD411 - Project Part 1 under my overall supervision.

**Prof A.K. Keshari**

Professor, Department of Civil Engineering

Indian Institute of Technology Delhi

November 27, 2023

# ACKNOWLEDGMENT

I would like to express my sincere gratitude to my supervisor, **Prof A.K. Keshari**, for his guidance, support, and valuable feedback throughout the course of this project. His expertise and insights have been invaluable in shaping this work. I am also thankful to the **Department of Civil Engineering, Indian Institute of Technology Delhi**, for providing the necessary resources for the completion of this project.

**Saurabh Tiwari**

November 27, 2023

# **ABSTRACT**

This report presents a novel approach to flood prediction and mitigation, specifically tailored for the challenging context of the Indian Subcontinent. Recognizing the devastating impact of floods on communities, the project addresses the scarcity of comprehensive data in India by pioneering the extraction of information from local news reports, creating a robust dataset that sets it apart from traditional flood forecasting models.

Motivated by the urgent need to address the pervasive and severe impact of floods in India, the project develops a proactive tool for predicting and visualizing potential flood scenarios before they occur. By synthesizing complex data into an interactive and comprehensible format, the web application named "Flood Predictor" empowers individuals and governments to make informed decisions in the face of imminent threats.

The project's significance is rooted in the alarming statistics surrounding flood-related disasters in India, with over 80% of the population at risk. Beyond the immediate threat to life and property, floods exert a profound socioeconomic toll, contributing to an annual economic loss of nearly 15 billion USD in India alone. In response to this imperative, the Flood Predictor application emerges as a cost-effective and accessible solution, facilitating equitable distribution of resources and minimizing the impact of floods on vulnerable communities.

The objectives of the project include innovative data collection methods, implementation and optimization of a cutting-edge machine learning model, advanced data visualization techniques, and the development of an intuitive web application. The machine learning model, based on the Random Forest Classifier, achieves an impressive accuracy of 98.71% on the test set. The web application, hosted on AWS Elastic Beanstalk, provides real-time flood predictions for any city in India, contributing to proactive measures and informed decision-making.

The report concludes with a reflection on the journey, acknowledging the challenges overcome, and outlines future directions for expanding the application globally and incorporating satellite data for enhanced flood detection. The Flood Predictor project stands as a technological

achievement with the potential to make a significant impact on flood preparedness and resource allocation, showcasing the possibilities of leveraging technology for the greater good.

# Contents

Certificate . . . . .	i
Acknowledgement . . . . .	ii
Abstract . . . . .	iii
List of figures . . . . .	vii
List of tables . . . . .	viii
<b>1 Introduction</b>	<b>1</b>
1.1 General . . . . .	1
1.2 Uniqueness . . . . .	2
1.3 Significance . . . . .	2
1.4 Socioeconomic Cause . . . . .	2
<b>2 Objective</b>	<b>4</b>
2.1 Data Collection . . . . .	4
2.2 Machine Learning Model . . . . .	4
2.3 Data Visualization . . . . .	4
2.4 Web Application Development . . . . .	5
<b>3 Literture Review</b>	<b>6</b>
3.1 Ambore, Charan, et al. (2023) . . . . .	6
3.2 Mosavi, Ozturk, et al. (2018) . . . . .	6
3.3 Xiong & O'Connor (2002) . . . . .	7
3.4 Nevo, Morin et al. (2018) . . . . .	7
<b>4 Methodology</b>	<b>9</b>
4.1 Data Collection . . . . .	9
4.1.1 Web Scraping . . . . .	9

4.1.2	Weather API . . . . .	9
4.1.3	Data Augmentation . . . . .	9
4.2	Machine Learning Model . . . . .	10
4.3	Data Visualization . . . . .	10
4.4	Web Application Development . . . . .	10
4.4.1	Frontend . . . . .	10
4.4.2	Integration and Hosting . . . . .	11
<b>5</b>	<b>Results</b>	<b>12</b>
5.1	Final data required for ML training . . . . .	12
5.2	Machine Learning Model Accuracy . . . . .	13
5.3	Final data for plotting . . . . .	13
5.4	Plots . . . . .	14
5.5	Density Maps . . . . .	15
5.6	Prediction Page . . . . .	16
<b>6</b>	<b>Conclusion</b>	<b>21</b>
<b>7</b>	<b>References</b>	<b>23</b>
<b>8</b>	<b>Appendices</b>	<b>25</b>
8.1	Python code for data collection and mining . . . . .	25
8.1.1	scraper.py . . . . .	25
8.1.2	get_forecast.py . . . . .	27
8.1.3	data_augment.py . . . . .	29
8.2	Python code for Machine Learning model training . . . . .	30
8.3	Python code for data visualization and plotting . . . . .	30
8.3.1	generate_plotting_data.py . . . . .	30
8.3.2	population_gen.py . . . . .	31
8.3.3	pop_multiply_precip.py . . . . .	33
8.4	Python code prediction of flood . . . . .	33
8.5	Flask code to connect Frontend & ML Model Pickel . . . . .	35
8.6	Run code in local machine . . . . .	37

# List of Figures

1.1	Flood rescue in Assam, India, June 2022 – NDRF . . . . .	1
5.1	The plot shows ML powered prediction of where a flood is going to occur, marked by red dots. . . . .	14
5.2	The plot shows the current precipitation data across the nation. The larger bubbles indicate more precipitation. . . . .	15
5.3	The plot shows damage analysis, based on the flood risk prediction. The size of the bubbles indicate the extent of predicted damage. . . . .	16
5.4	The plot shows damage analysis, based on the flood risk prediction. The colorscale of the density map indicates the extent of predicted damage. . . . .	17
5.5	The plot shows the current precipitation data across the nation. The darker areas indicate a greater volume of precipitation. . . . .	18
5.6	The plot below shows our ML powered prediction of where a flood is most likely to occur given the current environmental factors, marked primarily by the darker spots. . . . .	19
5.7	Prediction page showing user to input any city of India. . . . .	19
5.8	Based on the Machine Learning model it is showing is city safe. . . . .	20
5.9	Screen showing meteorological data based on which it has done prediction. . .	20



# List of Tables

5.2	Final data required for ML training . . . . .	12
5.4	Final data for plotting . . . . .	13

# Chapter 1

## Introduction

### 1.1 General

Floods, as one of the most devastating natural disasters, pose a significant threat to communities worldwide, necessitating innovative approaches to prediction and mitigation. While flood forecasting models abound, the uniqueness of this report lies in its original methodology, particularly tailored for the challenging context of the Indian Subcontinent.



Figure 1.1: Flood rescue in Assam, India, June 2022 – NDRF

## **1.2 Uniqueness**

Unlike regions like the United States, where abundant flood-related data simplifies predictive modeling, the scarcity of comprehensive data in India prompted the development of an unconventional approach. Rather than relying on satellite imagery, my project pioneers the extraction of information from local news reports. This involves mining data on flood occurrences, discerning location and time frames, and subsequently collating relevant weather conditions such as temperature, wind speed, cloud cover, humidity, and precipitation. This novel approach distinguishes my project, providing a robust dataset that sets it apart from traditional flood forecasting models.

## **1.3 Significance**

Motivated by the urgent need to address the pervasive and severe impact of floods, the significance for my project is rooted in the alarming statistics surrounding flood-related disasters in India. With over 80% of the population, approximately 1.08 billion people, at risk, and recent calamities like the Kerala flood tragedy claiming lives and destroying homes, the imperative for proactive flood prediction becomes evident. Coupled with the escalating risks attributed to climate change, the frequency and intensity of floods demand innovative solutions to mitigate their devastating consequences.

In response to this imperative, my project emerges as a proactive tool aimed at predicting and visualizing potential flood scenarios before they occur. By synthesizing complex data into an interactive and comprehensible format, the application empowers both individuals and governments to make informed decisions in the face of imminent threats.

## **1.4 Socioeconomic Cause**

Beyond the immediate threat to life and property, floods exert a profound socioeconomic toll. Globally, floods contribute to an annual economic loss of nearly 40 billion USD, with India alone accounting for 15 billion USD per year. Recognizing the critical role of preventive measures in mitigating these losses, my project endeavors to bridge the resource gap. By offering a cost-effective and accessible solution to governments and citizens alike, the application aims to facilitate equitable distribution of resources, thus minimizing the impact of floods on vulnerable

communities. In essence, my project aligns with the principle that proactive intervention can be a powerful force in averting the devastating consequences of floods and fostering societal resilience.

# Chapter 2

## Objective

### 2.1 Data Collection

The primary objective is to introduce an innovative approach to data collection for flood prediction, particularly in the challenging context of the Indian Subcontinent. This involves pioneering a methodology that taps into local news reports to extract crucial information about flood occurrences. The uniqueness of this approach lies in its fusion with historical weather conditions, overcoming the scarcity of comprehensive flood-related datasets in the region. The goal is to establish a robust and contextually rich dataset that enhances the accuracy and specificity of flood predictions.

### 2.2 Machine Learning Model

The core focus is on implementing and optimizing a cutting-edge machine learning model. The objective is to achieve a high level of accuracy through an exhaustive process of model experimentation. This includes testing various machine learning models, from Logistic Regression to K-Nearest Neighbors. The aim is to contribute to the advancement of machine learning applications in the domain of natural disaster prediction.

### 2.3 Data Visualization

The objective is to employ advanced data visualization techniques to enhance the interpretability and accessibility of flood-related insights. Utilizing the Plotly chart studio, the focus is on

creating dynamic and interactive visualizations such as bubble plots and density maps. These visualizations play a pivotal role in conveying complex information in an easily understandable format, facilitating effective communication of flood predictions, damage estimates, and precipitation analyses to both experts and the general public.

## **2.4 Web Application Development**

A key objective is the development of an intuitive and user-friendly web application, named "Flood Predictor." Built on the Flask Python framework, the application seamlessly integrates the machine learning model, datasets, and visualizations. By allowing users to input the name of a city, the application fetches real-time weather forecast data, runs it through the machine learning model, and presents instantaneous flood predictions. The goal is to make flood-related information easily accessible to a broad audience, including government officials, researchers, and the general public, thereby promoting proactive measures and informed decision-making.

# Chapter 3

## Literature Review

### 3.1 Ambore, Charan, et al. (2023)

The authors presents an examination of diverse machine learning algorithms for predicting floods based on rainfall data in Indian districts. Previous works on flood prediction using techniques like support vector machines, neural networks, image processing, and big data analytics are reviewed. The employed machine learning algorithms in this study include XGBoost, K Nearest Neighbors, Decision Trees, Logistic Regression, and Random Forest. Performance metrics such as accuracy, precision, recall, and F1-score are used to demonstrate the effectiveness of the models. A comparison with existing methods is conducted, and the advantages and limitations of the approach are discussed.

### 3.2 Mosavi, Ozturk, et al. (2018)

The report presents the challenges and importance of flood prediction for risk reduction and water resource management. The advantages and disadvantages of different types of models, such as physically based, statistical, and data-driven models. The state of the art of machine learning methods, such as artificial neural networks, support vector machines, decision trees, and ensemble methods, for both short-term and long-term flood prediction. The performance comparison and evaluation of various machine learning models based on accuracy, robustness, generalization, and computation cost. The trends and future directions of improving machine learning models, such as hybridization, data decomposition, algorithm ensemble, and model optimization.

### **3.3 Xiong & O'Connor (2002)**

The author presents a comparison of four updating models for real-time river flow forecasting, based on the simulation errors of a rainfall-runoff model. The four models are:

1. Single autoregressive (AR) model: the most widely used model, which fits a linear regression to the error series.
2. Autoregressive-threshold (AR-TS) model: a model that divides the error series into different bands according to the flow magnitude, and fits a separate AR model for each band.
3. Fuzzy autoregressive-threshold (FU-AR-TS) model: a model that uses fuzzy logic to assign different degrees of membership to the error values for each band, and combines the outputs of the AR models using weighted averages.
4. Artificial neural network (ANN) model: a model that uses a three-layer feed-forward network to capture the nonlinear relationship in the error series.

The author applies these four models to 11 test catchments, using the soil moisture accounting and routing (SMAR) model as the substantive rainfall-runoff model. The author evaluates the performance of the models using the Nash-Sutcliffe efficiency index and other criteria. The author finds that all four models can improve the flow forecast accuracy over the nonupdating mode, but the AR-TS, FU-AR-TS, and ANN models do not show significant advantages over the simple AR model. The author recommends the use of the AR model for error-forecast updating in real-time river flow forecasting.

### **3.4 Nevo, Morin et al. (2018)**

The report talks about:

1. Flood forecasting with machine learning: Google's operational flood warning system uses machine learning models to provide accurate and timely alerts to agencies and the public in India and Bangladesh. The system consists of four subsystems: data validation, stage forecasting, inundation modeling, and alert distribution.



2. Stage forecast models: The system uses two types of models to predict future river stages at target gauges: a Linear model based on multiple linear regression, and a LSTM model based on a recurrent neural network. The LSTM model showed higher skills than the Linear model in most cases.
3. Inundation models: The system uses two types of models to map flood extent and depth from forecasted river stages: a Thresholding model that does not require a digital elevation model (DEM), and a Manifold model that does require a DEM and also computes inundation depth. The Manifold model is a novel machine-learning alternative to hydraulic models. The Thresholding and Manifold models achieved similar performance metrics for modeling inundation extent.
4. Alerts: The system disseminates flood alerts to relevant agencies and the public via different channels, such as Google Search, smartphone notifications, and Google Maps. The alerts include information about the forecasted water stage, the inundation map, and the inundation depth if available. The alerts are sent in the local language and are designed to be informative and actionable.

# Chapter 4

## Methodology

### 4.1 Data Collection

#### 4.1.1 Web Scraping

In the initial phase of data collection, the Python Beautiful Soup 4 library was employed to perform web scraping on the designated website, <http://floodlist.com/tag/india>. This process involved the systematic extraction of flood-related data encompassing both historical and current events. The extracted information included crucial details such as the dates and geographical locations of the reported floods. By leveraging this web scraping technique, a foundational dataset was established to form the basis for subsequent analyses.

#### 4.1.2 Weather API

The utilization of the Visual Crossing Weather API constituted a pivotal aspect of the data collection process. This API was instrumental in retrieving comprehensive historical weather data for areas affected by floods. The specific weather parameters obtained included precipitation levels, humidity, temperature, cloud cover, and wind speed. This dual approach of combining web scraping with data from the Weather API ensured a holistic and nuanced dataset, incorporating both flood-specific and meteorological information.

#### 4.1.3 Data Augmentation

To enhance the richness and diversity of the dataset, various data augmentation techniques were systematically applied. These techniques were employed to synthetically increase the volume

of training data without necessitating the collection of entirely new information. By incorporating augmentation strategies, the dataset's robustness and representativeness were substantially improved, contributing to the overall efficacy of the machine learning model.

## **4.2 Machine Learning Model**

My machine learning model is based on the python sci-kit learn library. A thorough exploration of multiple machine learning models, including Logistic Regression, K-Nearest Neighbors, and Random Forest Classification, was conducted. Subsequently, the trained model was preserved in a pickle file, setting the stage for seamless integration into the web application.

## **4.3 Data Visualization**

I first obtained a dataset of the major cities and towns in India (around 200 of them) along with their latitude, longitude, and population. I then obtained the numerous weather factors in each city using the weather API and ran the data into my machine learning model. Next, I plotted the data from the model on various different types of maps, using Plotly chart studio.

The development of data visualizations comprised three fundamental components: Scatter Plots, Bubble Plot and Density Maps. Plots provided insights into flood predictions, damage predictions, and heavy rainfall predictions through visually intuitive representations. Density maps, on the other hand, employed color gradients to convey the magnitude of flood-related metrics.

## **4.4 Web Application Development**

### **4.4.1 Frontend**

The web application was implemented using the Flask Python framework, leveraging its capabilities for seamless integration with machine learning models. The presentation layer incorporated HTML templates for structure, CSS for styling, and Javascript to enhance the user experience and functionality. On my predict page, the user simply enters the name of any city in India. My app then automatically fetches the weather forecast data of that city in realtime, runs this data into my Machine Learning model, and gives instantaneous results, which include

the flood prediction, the temperature, the maximum temperature, the humidity, the cloud cover, the windspeed, and the precipitation.

#### **4.4.2 Integration and Hosting**

The integration phase involved the seamless connection of the Flask back-end with the developed machine learning models and datasets. To make the web application accessible to a wider audience, AWS Elastic Beanstalk was utilized for hosting, ensuring scalability and reliability in serving the application to users. This robust integration framework laid the foundation for the successful deployment and utilization of the Flood Predictor web application.

# Chapter 5

## Results

### 5.1 Final data required for ML training

Temp	Max_Temp	Wind_Speed	Cloudcover	Precip	Humidity	Class
29.90	37.00	23.00	58.60	146.00	83.86	1
93.04	82.41	63.44	102.42	192.07	129.30	1
87.13	98.82	72.29	122.12	191.80	142.25	1
92.12	72.39	73.84	96.50	196.53	150.42	1
92.64	76.03	67.13	92.61	191.76	126.61	1
...						
61.43	77.45	80.24	77.38	81.32	124.24	0
72.26	78.98	47.48	73.92	67.72	113.09	0
76.40	82.69	79.23	58.95	65.84	122.76	0
61.03	74.17	70.44	61.39	64.09	106.96	0
85.19	64.06	73.84	56.96	71.45	109.65	0

*[5040 rows x 7 columns]*

Table 5.2: Final data required for ML training

The Python code to generate this data can be found in Section 8.1

A binary classification system is utilized in the "Class" column, where the label "1" indicates the presence of a flood, while "0" signifies the absence of a flood.

## 5.2 Machine Learning Model Accuracy

The parameters employed in the machine learning model include temperature, max temperature, wind speed, cloud cover, humidity, and precipitation. Following extensive experimentation with various models, the Random Forest Classifier yielded the highest accuracy of 98.71% on the test set. Subsequently, we saved the model in a pickle file.

## 5.3 Final data for plotting

City	Lat	Lon	Precip	Class	Population	Damage
Mumbai	18.987807	72.836447	1.57	0	12691836	1992.62
Delhi	28.651952	77.231495	0.00	0	10927986	0.00
Kolkata	22.562627	88.363044	0.00	0	4631392	0.00
Chennai	13.084622	80.248357	1.63	0	4681087	763.02
Bengaluru	12.977063	77.587106	0.15	0	8443675	126.66
...						
Calicut	11.248016	75.780402	3.07	0	0	0.00
Kagaznagar	19.331589	79.466051	0.44	0	0	0.00
Jaipur	26.913312	75.787872	0.00	0	2711758	0.00
Ghandinagar	23.216667	72.683333	1.34	0	0	0.00
Panchkula	30.691512	76.853736	0.00	0	200000	0.00

*[212 rows x 7 columns]*

Table 5.4: Final data for plotting

The Python code to generate this data can be found in Section 8.3

Quantifying the extent of damage resulting from floods poses a considerable challenge due to the complex and multifaceted nature of such events. In an effort to assess the impact more comprehensively, I have undertaken the approach of multiplying the precipitation factor by the population. This methodology aims to provide a reference point for estimating the potential damage caused by floods, taking into account the interplay between meteorological factors and the density of the affected population.

## 5.4 Plots

The 3 visualizations plots display flood predictions, damage predictions, and heavy rainfall predictions across India, taking in factors such as precipitation, wind speed, humidity, temperature, cloud cover, as well as previous data history.

1. The first plot is my flood prediction plot, which shows my ML-powered prediction of where a flood is going to occur, marked by red dots. As shown in Figure 5.1
2. The second plot is a precipitation plot, showing the forecasted precipitation data across the nation, with larger bubbles indicating more precipitation. As shown in Figure 5.2
3. Lastly, the third plot is a damage analysis plot, which shows the estimated damage for various places in India, based on the flood risk prediction and population size. The size of the bubbles indicates the extent of predicted damage. As shown in Figure 5.3

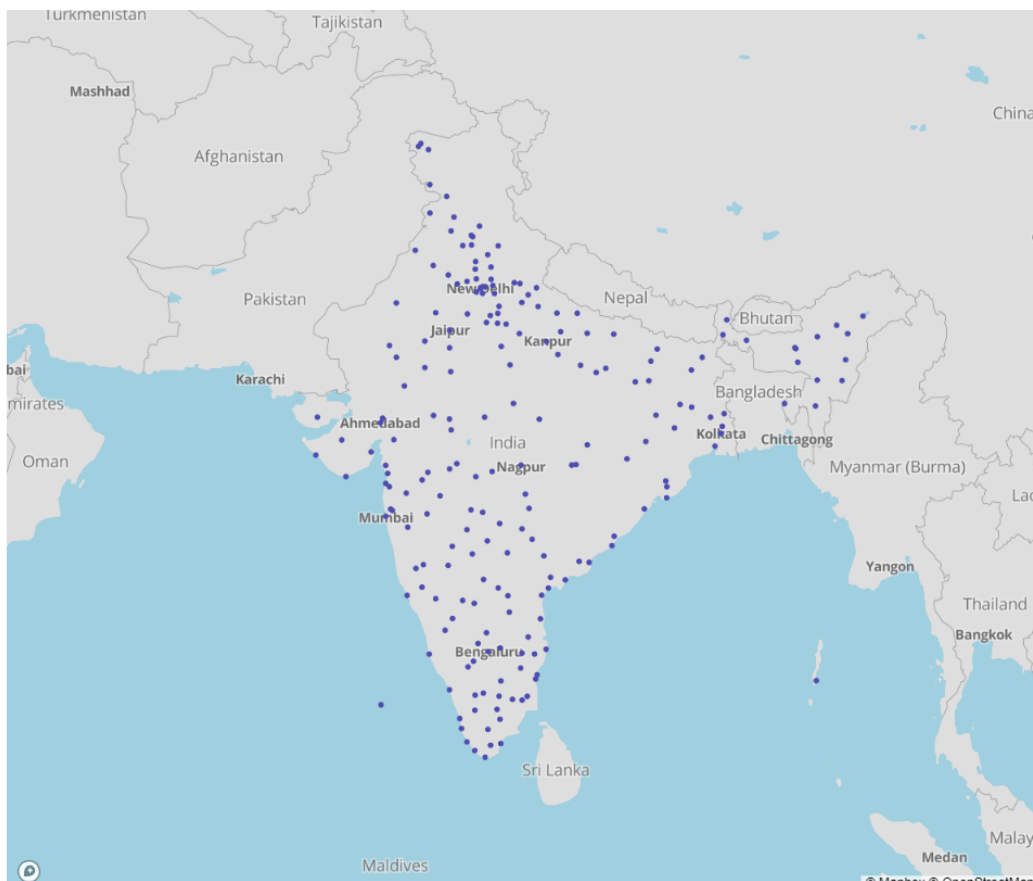


Figure 5.1: The plot shows ML powered prediction of where a flood is going to occur, marked by red dots.

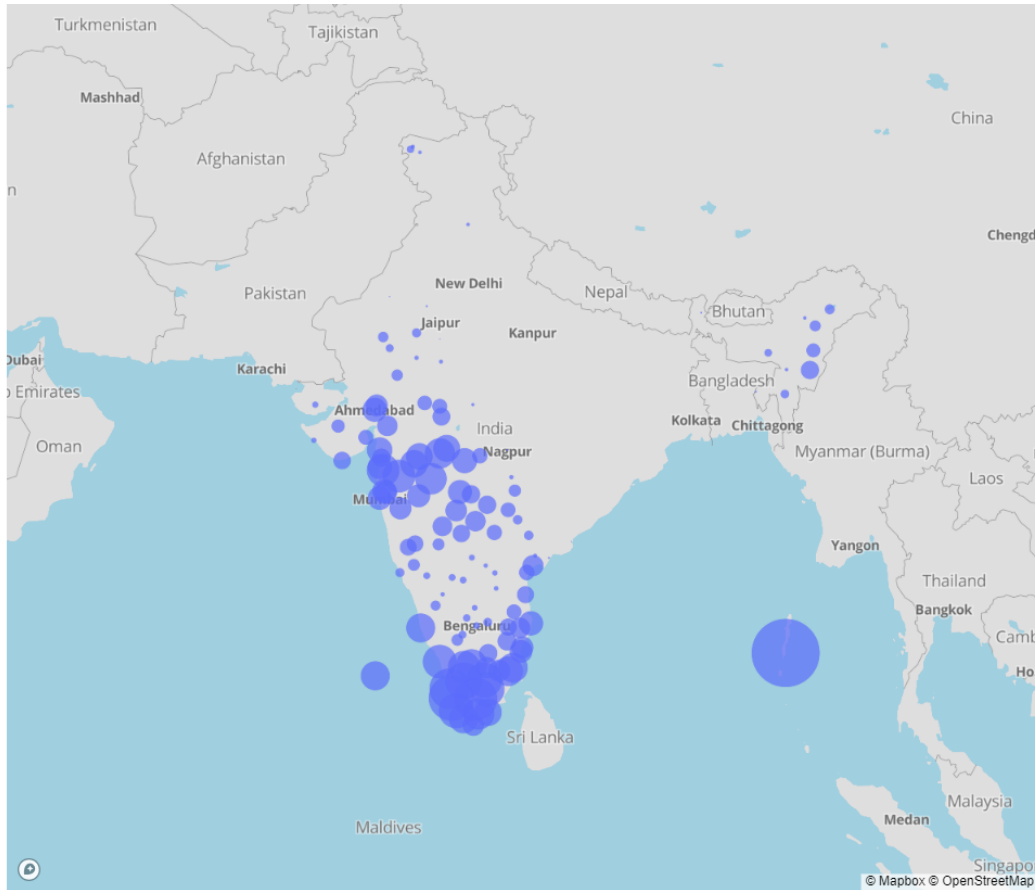


Figure 5.2: The plot shows the current precipitation data across the nation. The larger bubbles indicate more precipitation.

## 5.5 Density Maps

The 3 density maps show flood predictions, damage predictions, and heavy rainfall predictions across India, taking in factors such as precipitation, wind speed, humidity, temperature, cloud cover, as well as previous data history.

1. The first plot is my damage analysis plot, which shows a damage analysis, with the colorscale of the density indicating the extent of predicted damage. As shown in Figure 5.4
2. The second plot is a precipitation plot, showing the forecasted precipitation data across the nation, with the darker areas indicating a greater volume of precipitation.
3. Lastly, the third plot is a flood prediction plot, which shows my ML-powered prediction of where a flood is most likely to occur given the forecasted environment factors, marked primarily by the darker red spots on a continuous colorbar.



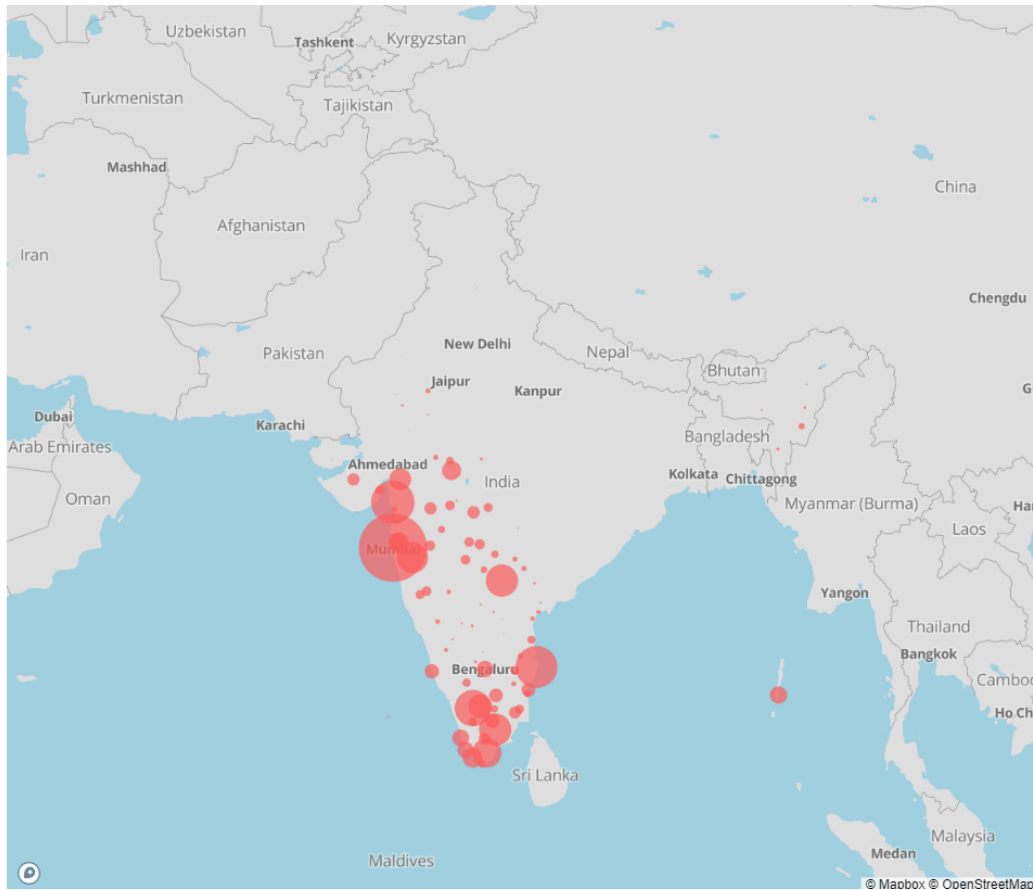


Figure 5.3: The plot shows damage analysis, based on the flood risk prediction. The size of the bubbles indicate the extent of predicted damage.

## 5.6 Prediction Page

On predict page, the user simply enters the name of any city in the India. My app then automatically fetches the weather forecast data of that city in realtime, runs this data into my Machine Learning model, and gives instantaneous results, which include the flood prediction, the temperature, the maximum temperature, the humidity, the cloud cover, the windspeed, and the precipitation. As shown in Figure 5.7, 5.8 and 5.9

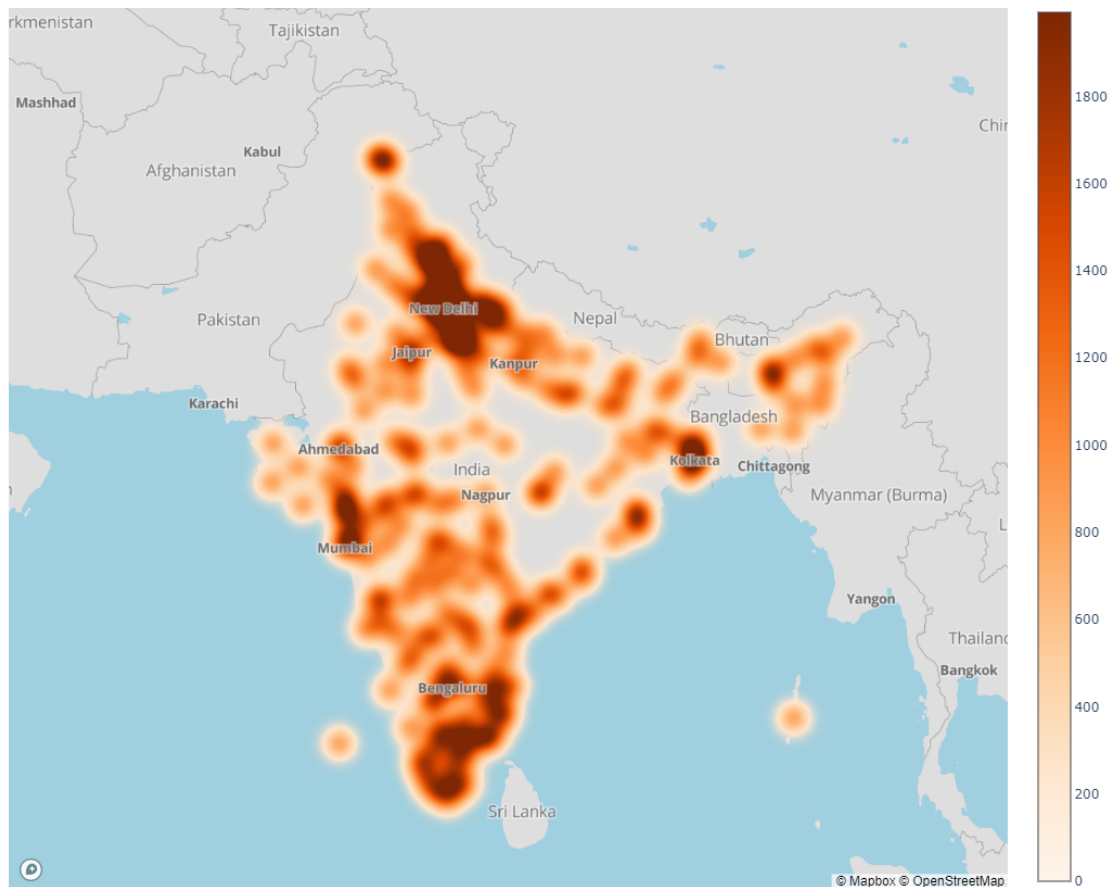


Figure 5.4: The plot shows damage analysis, based on the flood risk prediction. The colorscale of the density map indicates the extent of predicted damage.

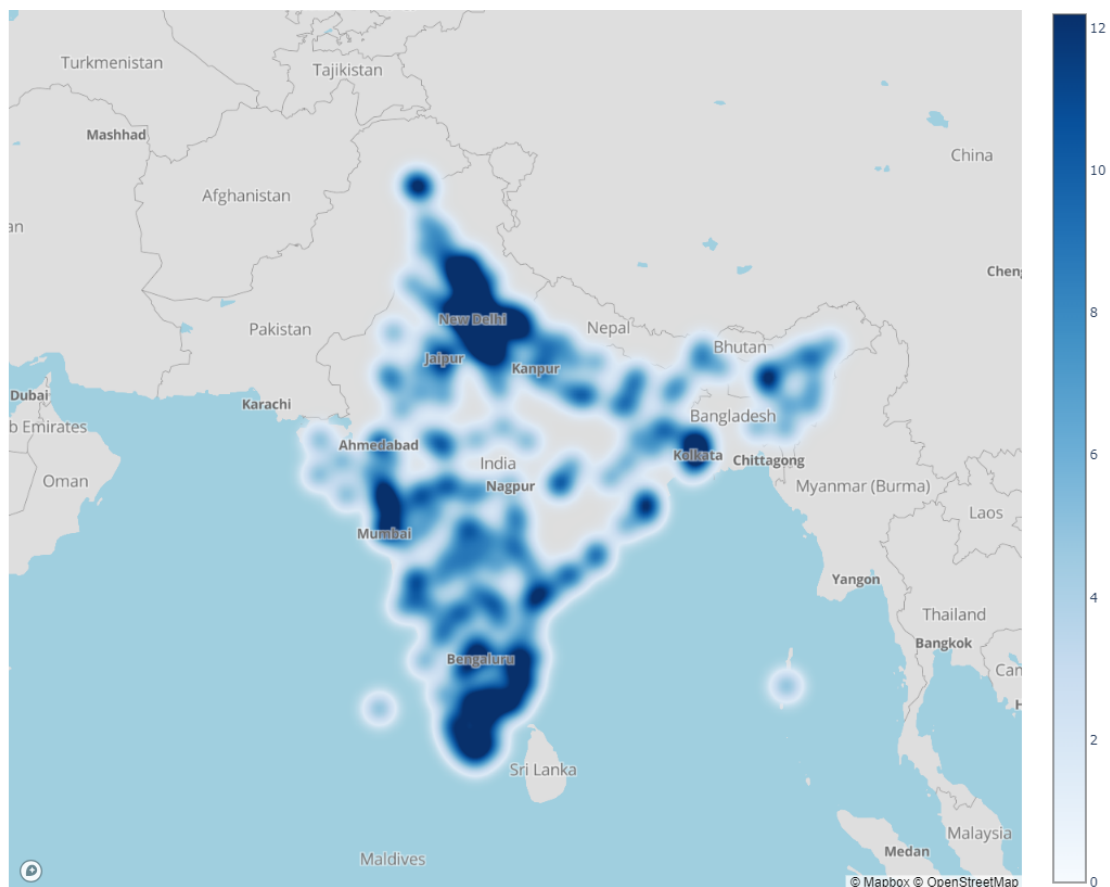


Figure 5.5: The plot shows the current precipitation data across the nation. The darker areas indicate a greater volume of precipitation.

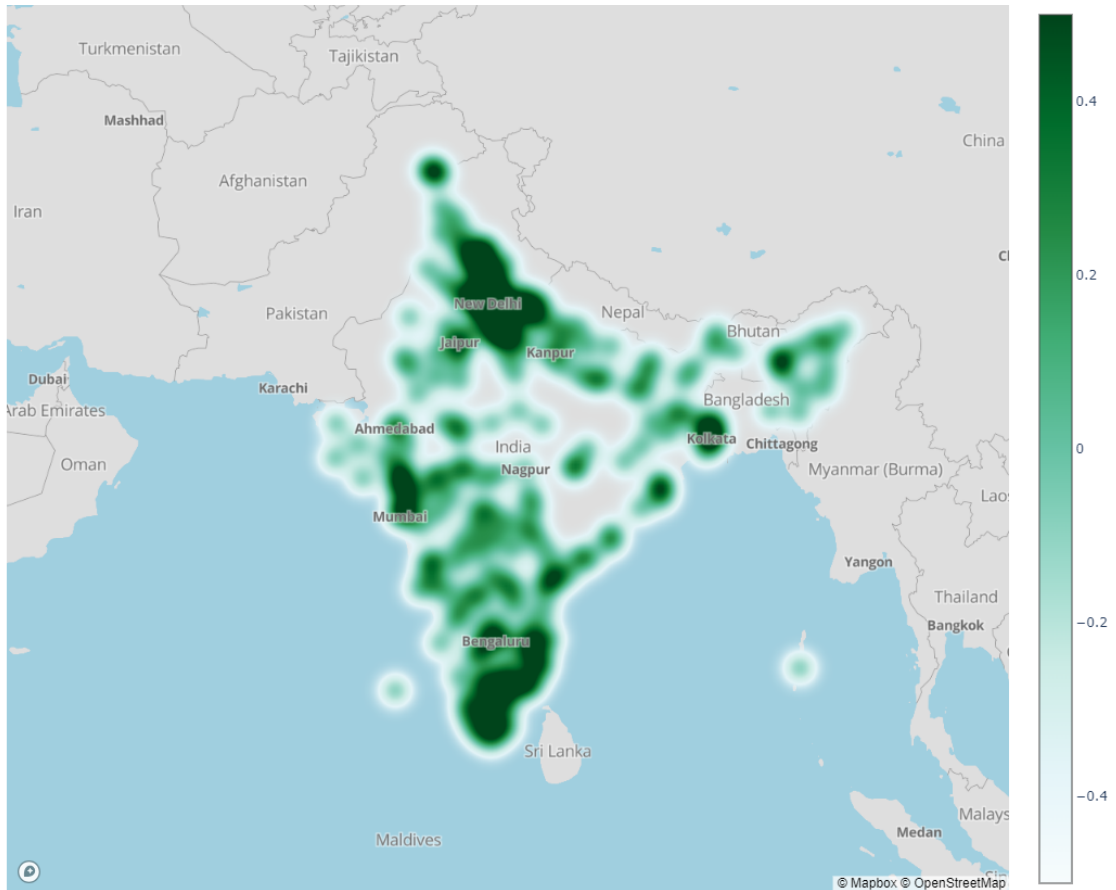




Figure 5.6: The plot below shows our ML powered prediction of where a flood is most likely to occur given the current environmental factors, marked primarily by the darker spots.


HOME PLOTS DENSITY MAPS PREDICT

### Select City!

Select a city name to get information about it!

### Information about Nagpur



Flood Prediction

Figure 5.7: Prediction page showing user to input any city of India.

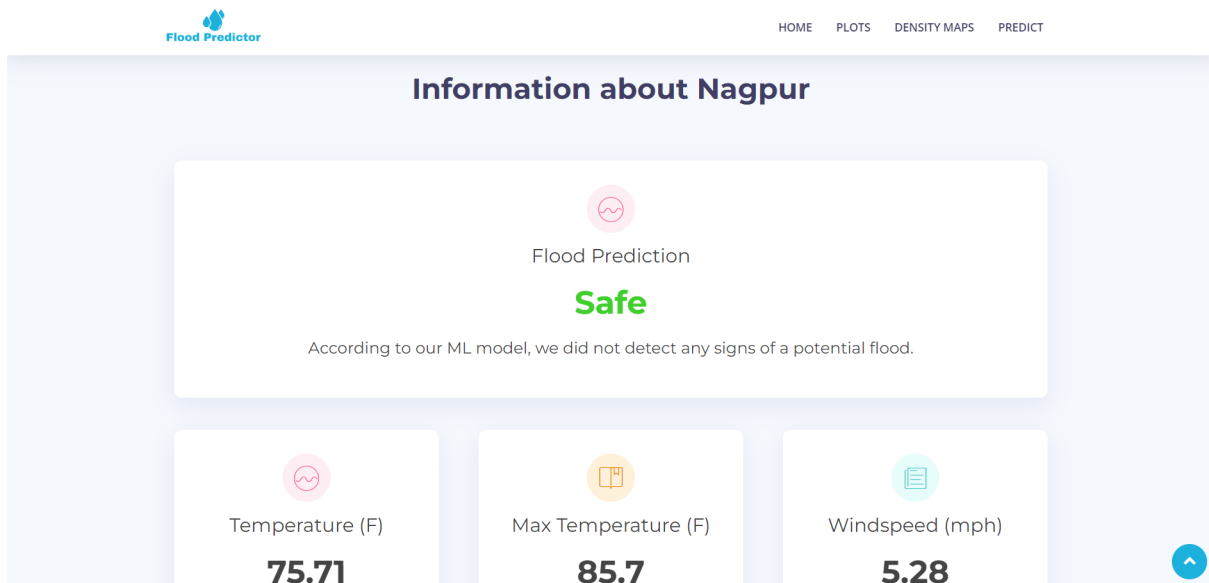


Figure 5.8: Based on the Machine Learning model it is showing is city safe.

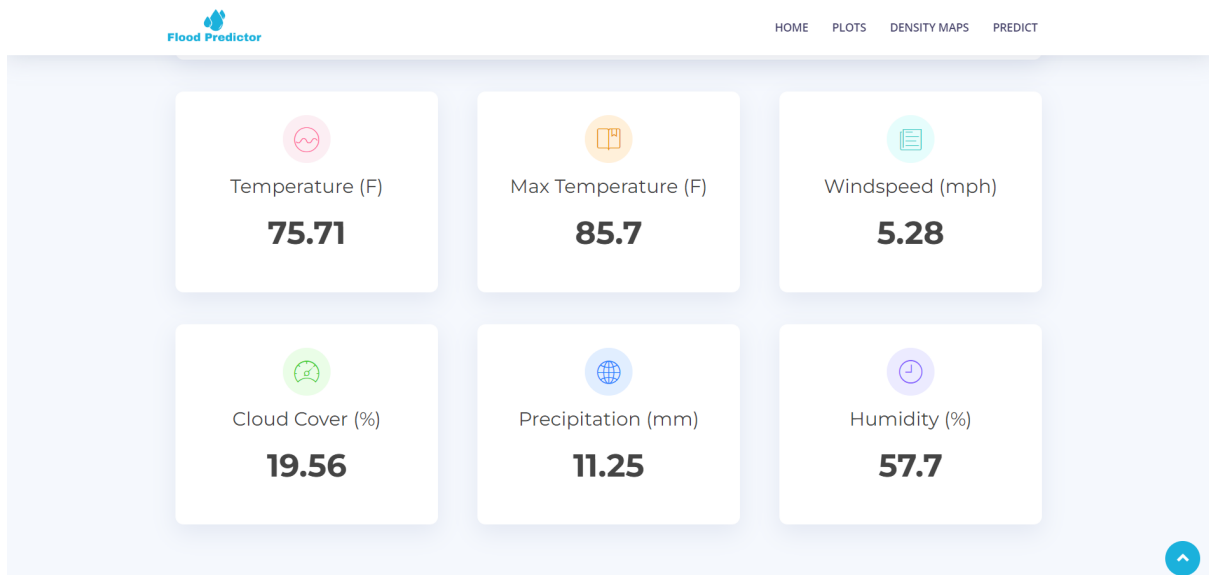


Figure 5.9: Screen showing meteorological data based on which it has done prediction.

# Chapter 6

## Conclusion

In conclusion, the development of my project represents a significant stride in addressing the critical issue of floods in India. Through the creation of a web application, equipped with advanced machine learning algorithms, I have successfully implemented a system that predicts future floods based on comprehensive weather forecast data. This endeavor is not just a technological feat; it is a proactive approach to providing users with the tools to better understand and prepare for potential flood risks.

The core components of my project, including plots and heatmaps, offer users a detailed and insightful view of flood predictions, damage estimates, and heavy rainfall forecasts across India. The integration of a user-friendly predict page further enhances accessibility, allowing users to obtain real-time flood predictions for any city worldwide.

The journey to build my project involved overcoming various challenges, most notably the scarcity of available data related to floods in India. Through innovative approaches such as mining flood reports from news sources and utilizing weather APIs, I successfully compiled a diverse dataset. The machine learning model, based on the Random Forest Classifier, demonstrated impressive accuracy, reaching 98.71% on the test set.

The data visualization aspect of my project, implemented using Plotly chart studio, showcases my commitment to creating detailed, sophisticated, yet comprehensible plots. These visualizations, coupled with the integration of various data augmentation and manipulation techniques, contribute to the effectiveness of my project in conveying critical information.

Throughout this project, I have acquired valuable skills in web scraping, data mining, and data augmentation. Additionally, my foray into Plotly and the exploration of different machine learning models, particularly the Random Forest Classification model, has expanded my

technical expertise.

Looking ahead, the vision for my project extends beyond its current scope. I aspire to expand the application globally, covering cities in various countries, and make flood predictions and visualizations accessible worldwide. Furthermore, I recognize the potential of incorporating satellite data for enhanced flood detection and plan to develop an image classification model for this purpose.

In essence, my project is not just a technological achievement but a tool with the potential to make a significant impact on flood preparedness and resource allocation. By providing accurate predictions and valuable insights, this web application stands as a testament to the possibilities of leveraging technology for the greater good, potentially saving lives and safeguarding livelihoods. Visit the live application on <https://github.com/SaurabhTiwari4093/Flood-Forecasting> or directly on the <https://flood-predictor.iccntsd.in> to explore its capabilities and contribute to the ongoing effort to mitigate the impact of floods worldwide.

# Chapter 7

## References

1. Amazon Web Service: <https://aws.amazon.com/>
2. Anil Kumar Ambore, T. Sri Sai Charan, U. Rohit Reddy, T. Samara Simha Reddy, Tarun. G (2023), Flood Prediction using Machine Learning, International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 11 Issue V May 2023
3. Beautiful soup 4: <https://pypi.org/project/beautifulsoup4/>
4. Flask: <https://flask.palletsprojects.com/en/3.0.x/>
5. FloodList: <https://floodlist.com/>
6. LIHUA XIONG & KIERAN M. O'CONNOR (2002) Comparison of four updating models for real-time river flow forecasting, Hydrological Sciences Journal, 47:4, 621-639, DOI: 10.1080/02626660209492964
7. Mosavi, A., Ozturk, P., & Chau, K.-w. (2018). Flood prediction using machine learning models: Literature review. Department of Computer Science (IDI), Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway; Department of Civil and Environmental Engineering, Hong Kong Polytechnic University, Hong Kong, China; dr.kwok-wing.chau@polyu.edu.hk. Correspondence: amir.mosavi@ntnu.no (A.M.); pinar@ntnu.no (P.O.). Received: 1 September 2018; Accepted: 17 October 2018; Published: 27 October 2018.



8. Nevo, S., Morin, E., Rosenthal, A. G., Metzger, A., Barshai, C., Weitzner, D., ... Matias, Y. (2023). Flood forecasting with machine learning models in an operational framework. Google Research, Yigal Alon 96, Tel-Aviv 6789141, Israel; Hebrew University of Jerusalem, Safra Campus, Jerusalem 91904, Israel.
9. Plotly chart studio: <https://chart-studio.plotly.com/>
10. Python: <https://www.python.org/>
11. Population Data: <https://public.opendatasoft.com/>
12. Weather Data & API Global Forecast & History Data: <https://www.visualcrossing.com/>

# Chapter 8

## Appendices

### 8.1 Python code for data collection and mining

#### 8.1.1 scraper.py

```
1 import requests
2 import datetime
3
4 def get_data(date, month, year, days, location):
5     a = datetime.date(year, month, date)
6     b = a - datetime.timedelta(days)
7
8     k =
9         "https://weather.visualcrossing.com/VisualCrossingWebServices/
10         rest/services/weatherdata/history?&aggregateHours=" + str(24 *
11             days) + "&startDateTime=" + str(b) +
12             "T00:00:00&endDateTime=" + str(a) +
13             "T00:00:00&unitGroup=uk&contentType=json&dayStartTime=0:0:00&
14             dayEndTime=0:0:00&location=" + location +
15             ",India&key=XHWV4W6GK29MQK6J9BBM2GNKC"
16     x = requests.get(k).json()['locations']
17
18     for i in x:
19         y = x[i]
```

```

15
16     y = y['values'][0]
17     final = [y['temp'], y['maxt'], y['wspd'], y['cloudcover'],
18              y['precip'], y['humidity'], y['precipcover']]
19
20     return final
21
22 states = ['Karnataka', 'Gujarat', 'Rajasthan', 'Maharashtra',
23          'Madhya Pradesh']
24
25 import csv
26 import random
27
28 f = open('data1.csv', mode='w', newline = '')
29 writer = csv.writer(f, delimiter=',')
30
31 for i in states:
32     for j in range(1, 31):
33         a = random.randint(1, 28)
34         b = random.randint(1, 12)
35         c = random.randint(2013, 2019)
36
37         k = get_data(a, b, c, 15, i)
38
39         if k[4] != None:
40             if k[4] < 20:
41                 print(k)
42                 print(j)
43                 writer.writerow(k + [0])
44
45 def extract_date(x):
46     k = x.split(" ")
47
48     a = int(k[0])

```

```

46     d = {'january':1, 'february':2, 'march':3, 'april':4, 'may':5,
         'june':6, 'july':7, 'august':8, 'september':9, 'october':10,
         'november':11, 'december':12}
47     b = d[k[1][0:len(k[1]) - 1].lower()]
48
49     c = int(k[2])
50
51     return [a, b, c]
52
53 def process(k):
54     x = extract_date(k[1])
55
56     return get_data(x[0], x[1], x[2], 15, k[0])
57
58 f = open('data.csv', mode='w', newline = '')
59 writer = csv.writer(f, delimiter=',')
60
61 with open('mined.csv', mode='r') as csv_file:
62     csv_reader = csv.reader(csv_file)
63
64     for row in csv_reader:
65         print(row)
66
67         writer.writerow(process(row) + [1])

```

### 8.1.2 get\_forecast.py

```

1 import csv
2 import requests
3
4 def get_data(lat, lon):
5
6     k =
       "https://weather.visualcrossing.com/VisualCrossingWebServices/

```

```

7     rest/services/weatherdata/forecast?locations=" + \
8         str(lat) + "%2C%20" + str(lon) + \
9         "&aggregateHours=24&unitGroup=us&shortColumnNames=false&
10        contentType=json&key=XHWV4W6GK29MQK6J9BBM2GNKC"
11    x = requests.get(k).json()['locations']
12    for i in x:
13        y = x[i]['values']
14
15    final = [0, 0, 0, 0, 0, 0]
16
17    for j in y:
18        final[0] += j['temp']
19        if j['maxt'] > final[1]:
20            final[1] = j['maxt']
21        final[2] += j['wspd']
22        final[3] += j['cloudcover']
23        final[4] += j['precip']
24        final[5] += j['humidity']
25    final[0] /= 15
26    final[2] /= 15
27    final[3] /= 15
28    final[5] /= 15
29
30    return final
31
32    def testConnection():
33        return "yo"
34
35    f = open('cities.csv', 'r', encoding='UTF-8')
36    ff = open('plotting.csv', mode='w', newline='')
37    writer = csv.writer(ff, delimiter=',')
38
39    r = csv.reader(f)
40

```

```

41 for row in r:
42     print(row)
43     writer.writerow(get_data(row[1], row[2]))

```

### 8.1.3 data\_augment.py

```

1 import csv
2 import random
3
4 def augment(k):
5     final = []
6     for i in range(0, 7):
7         final.append(round(random.uniform(33.33, 66.66), 2) +
8                        float(k[i]))
9
10    return final + [int(k[7])]
11
12 data1 = open('data.csv', mode='r')
13 data0 = open('data1.csv', mode='r')
14
15 reader1 = csv.reader(data1)
16
17 reader0 = csv.reader(data0)
18
19 f = open('final_data.csv', mode='w', newline = '')
20 writer = csv.writer(f, delimiter=',')
21
22 for row1 in reader1:
23     writer.writerow(row1)
24     for i in range(0, 19):
25         writer.writerow(augment(row1))
26
27 for row0 in reader0:
28     writer.writerow(row0)

```

```
28     for i in range(0, 19):
29         writer.writerow(augment(row0))
```

## 8.2 Python code for Machine Learning model training

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.metrics import accuracy_score
4 from sklearn.ensemble import RandomForestClassifier
5 import pickle
6
7 data = pd.read_csv("final_data.csv")
8 y = data['class']
9 X = data.drop('class', axis=1)
10 X_train, X_test, y_train, y_test = train_test_split(X,
11     y, test_size=0.2)
12
13 classifier = RandomForestClassifier(n_estimators = 50, criterion =
14     'entropy', random_state = 0)
15
16 classifier.fit(X_train, y_train)
17
18 pred = classifier.predict(X_test)
19 accuracy = accuracy_score(y_test, pred)
20
21 print("accuracy: " + str(accuracy * 100) + "%")
22
23 pickle.dump(classifier, open('model.pickle', 'wb'))
```

## 8.3 Python code for data visualization and plotting

### 8.3.1 generate\_plotting\_data.py

```

1 import csv
2 import pickle
3
4 model = pickle.load(open('model.pickle', 'rb'))
5 c = open('cities.csv', 'r', encoding='UTF-8')
6
7 cr = csv.reader(c)
8
9 cities = []
10
11 for c_row in cr:
12     cities.append(c_row)
13
14 d = open('plotting.csv', 'r', encoding='UTF-8')
15
16 dr = csv.reader(d)
17
18 forecast = []
19 for d_row in dr:
20     k = list(map(float, d_row))
21     forecast.append([k[4], model.predict([k])[0]])
22
23 ff = open('final_plot.csv', mode='w', newline = '',
24           encoding="UTF-8")
25 writer = csv.writer(ff, delimiter=',')
26
27 for i in range(0, len(forecast)):
28     writer.writerow(cities[i] + forecast[i])
29 print(forecast)

```

### 8.3.2 population\_gen.py

```

1 import requests

```



```

2 import json
3 import csv
4
5 population_data = []
6
7 def get_city_opendata(city, country):
8     tmp = 'https://public.opendatasoft.com/api/explore/v2.1/catalog/
9     datasets/geonames-all-cities-with-a-population-500/
10    records?refine=country:%s&refine=ascii_name:%s'
11    cmd = tmp % (country, city)
12    res = requests.get(cmd)
13    dct = json.loads(res.content)
14    out = dct['results'][0]['population']
15    return out
16
17 csv_file = "finalfinal.csv"
18
19 with open(csv_file, 'r') as f:
20     reader = csv.reader(f)
21     header = next(reader)
22     header.append("population") # Add "pop" to the existing header
23
24     for row in reader:
25         if row[0] != "city":
26             try:
27                 pop = get_city_opendata(row[0], 'India')
28                 row.append(pop)
29                 population_data.append(row)
30             except:
31                 print("error in", row[0])
32                 row.append(0)
33                 population_data.append(row)
34
35 with open('finalfinal.csv', 'w', newline='') as f:

```

```

36 writer = csv.writer(f)
37 writer.writerow(header) # Write the updated header
38 for row in population_data:
39     writer.writerow(row)
40
41 # Also corrected some cities spelling and manually added population

```

### 8.3.3 pop\_multiply\_precip.py

```

1 import csv
2
3 csv_file = "finalfinal.csv"
4
5 rows = []
6
7 with open(csv_file, 'r') as f:
8     reader = csv.reader(f)
9     header = next(reader) # Read the header row
10    header.append("damage") # Add a new column header
11
12    for row in reader:
13        if row[0] != "city":
14            dam = round(float(row[3]) * float(row[5])/10000, 2)
15            row.append(dam)
16            rows.append(row)
17
18 with open('finalfinal.csv', 'w', newline='') as f:
19     writer = csv.writer(f)
20     writer.writerow(header) # Write the modified header
21     writer.writerows(rows) # Write the modified rows

```

## 8.4 Python code prediction of flood

```

1 import requests
2
3 def get_data(lat, lon):
4
5     k =
6         "https://weather.visualcrossing.com/VisualCrossingWebServices/
7         \rest/services/weatherdata/forecast?locations=" + str(lat) +
8         "%2C%20" + str(lon)+"&aggregateHours=24&unitGroup=us&
9         shortColumnNames=false&contentType=json&
10        key=XHWV4W6GK29MQK6J9BBM2GNKC"
11    x = requests.get(k).json()['locations']
12
13    for i in x:
14        y = x[i]['values']
15
16    final = [0, 0, 0, 0, 0, 0]
17
18    for j in y:
19        final[0] += j['temp']
20        if j['maxt'] > final[1]:
21            final[1] = j['maxt']
22        final[2] += j['wspd']
23        final[3] += j['cloudcover']
24        final[4] += j['precip']
25        final[5] += j['humidity']
26
27    final[0] /= 15
28    final[2] /= 15
29    final[3] /= 15
30    final[5] /= 15
31
32    return final
33
34 def testConnection():
35     return "yo"

```

---

## 8.5 Flask code to connect Frontend & ML Model Pickel

```
1 import flask
2 from flask import render_template, request
3
4 import pickle
5 from training import prediction
6 import requests
7 application = flask.Flask(__name__)
8
9 model = pickle.load(open("model.pickle", 'rb'))
10
11 @application.route("/")
12 @application.route('/index.html')
13 def index() -> str:
14     """Base page."""
15     return flask.render_template("index.html")
16
17 @application.route('/plots.html')
18 def plots():
19     return render_template('plots.html')
20
21 @application.route('/densityMaps.html')
22 def heatmaps():
23     return render_template('densityMaps.html')
24
25 @application.route('/predicts.html')
26 def predicts():
27     return render_template('predicts.html', cityname="Information
28         about the city")
29
30 @application.route('/predicts.html', methods=["GET", "POST"])
```

```

30 def get_predicts():
31     try:
32         cityname = request.form["city"]
33         # print(cityname)
34         URL = "https://geocode.search.hereapi.com/v1/geocode"
35         location = cityname
36         # Acquire from developer.here.com
37         api_key = 'ftKvGdH2ItSBc6N3Jbh4Tzph6F57uHHqw4Us4Uoj0HM'
38         PARAMS = {'apikey': api_key, 'q': location}
39         # sending get request and saving the response as response
40         # object
41         r = requests.get(url=URL, params=PARAMS)
42         data = r.json()
43         latitude = data['items'][0]['position']['lat']
44         longitude = data['items'][0]['position']['lng']
45         final = prediction.get_data(latitude, longitude)
46
47         final[4] *= 15
48         if str(model.predict([final])[0]) == "0":
49             pred = "Safe"
50         else:
51             pred = "Unsafe"
52
53         return render_template('predicts.html',
54                                cityname="Information about " + cityname,
55                                temp=round(final[0], 2), maxt=round(final[1], 2),
56                                wspd=round(final[2], 2), cloudcover=round(final[3], 2),
57                                percip=round(final[4], 2), humidity=round(final[5], 2),
58                                pred=pred)
59     except:
60         return render_template('predicts.html', cityname="Oops, we
61                                weren't able to retrieve data for that city.")
62
63 if __name__ == "__main__":

```

```
application.run(debug=True)
```

## 8.6 Run code in local machine

1. Clone the project from <https://github.com/SaurabhTiwari4093/Flood-Forecasting>
2. Activate virtual environment
3. run `pip install -r requirements.txt`
4. run `python application.py`
5. You're Done!