# **Comprehensive Approach For Extreme Weather Event Forecasting Based on Machine Learning and Data Science Techniques**

**A blue logo with white text

Description automatically generated with low confidence**

By

Tariq Aziz

In partial fulfilment of the requirement for the degree

Bachelor of Science in Computer Science

Department of Computer Science, School of Arts & Sciences

University of Central Asia, Kyrgyz Republic.

(2025)

**DECLARATION BY AUTHOR**

I certify that this work has not been accepted in substance for any degree and is not concurrently being submitted for any degree other than that of Bachelor of Science in Computer Science being studied at the Department of Computer Science, School of Arts & Sciences, University of Central Asia, Kyrgyz Republic. I/we also declare that this work is the result of my findings and investigations, except where otherwise identified by references, and that I/we have not plagiarized another’s work.

Tariq Aziz

===================================================================

**DECLARATION BY SUPERVISOR**

I, the undersigned hereby certify that I have read this project report and finally approve it with recommendation that this report may be submitted by the authors above to the final year project evaluation committee for final evaluation and presentation, in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science at the Department of Computer Science, School of Arts & Sciences, University of Central Asia, Kyrgyz Republic.

A black line drawing of a face

Description automatically generated



Dr. Muhammad Fayaz Dr. Azmat Hussain

## **Abstract**

In recent years, we have seen clear signs of climate change and its worldwide impact. The frequency and intensity of various natural disasters, such as avalanches, landslides, and droughts, have drastically increased. Traditional disaster prediction techniques are not efficient and often lack accuracy. We are predicting avalanches, landslides, and droughts using machine learning. We have also created a user-friendly web app and Telegram bot for real-time risk alerts. It’s going to help meteorologists and people living in disaster-prone areas. Our dataset consists of environmental variables and relevant meteorological features. As part of the training process, we tried different machine learning algorithms. The ultimate choice of algorithm was based on performance. Our web application also provides interactive visualizations. These visualizations are essential for in-depth analysis of risk assessment. Our project provides a comprehensive solution for disaster predictions. The use of machine learning enhances the reliability of predictions made. Our goal is to help local communities, decision makers, and emergency responders.

**Keywords:** Disaster Prediction, Drought Forecasting, Landslide, Avalanche, Machine Learning, Ensemble Methods, Interactive Dashboard, Real-Time Prediction, Telegram Bot, Natural Disaster.

## **Acknowledgement**

I extend my heartfelt gratitude to my respected professor serving at the University of Central Asia, Naryn, for helping me understand the core concepts of Computer Science. Moreover, I am thankful to my supervisor, Dr. Muhammad Fayaz, and co-supervisor Dr. Azmat Hussain for their kind support and expert supervision. Overall, the regular feedback, motivation, and support from my teachers helped me to complete this project.

I am also grateful for the love and support from my family members. Additionally, I sincerely offer my gratitude to my friends for their tremendous support and motivation to take on and complete this project.

Contents

[Comprehensive Approach For Extreme Weather Event Forecasting Based on Machine Learning and Data Science Techniques 1](#_Toc196862604)

[List of Figures 7](#_Toc196862607)

[List of Tables 7](#_Toc196862608)

[CHAPTER 1. INTRODUCTION 8](#_Toc196862609)

[1.1 Problem Statement 10](#_Toc196862610)

[1.2 Proposed Solution 10](#_Toc196862611)

[1.3 Project Objectives 11](#_Toc196862612)

[1.4 Business Benefits 11](#_Toc196862613)

[1.5 Contributions 12](#_Toc196862614)

[CHAPTER 2. LITERATURE REVIEW 13](#_Toc196862615)

[2.1 Similar applications 16](#_Toc196862616)

[CHAPTER 3. Proposed Methodology 18](#_Toc196862617)

[3.1 Data Acquisition and Integration 18](#_Toc196862618)

[3.2 Data Cleaning and Preprocessing 19](#_Toc196862619)

[3.3 Model Selection and Training 20](#_Toc196862620)

[3.3 Model evaluation 21](#_Toc196862621)

[3.4 Risk Level Classification 23](#_Toc196862622)

[CHAPTER 4. IMPLEMENTATION 24](#_Toc196862623)

[4.1 Technology Stack 24](#_Toc196862624)

[4.2 Development Approach 25](#_Toc196862625)

[4.3 System Architecture 26](#_Toc196862626)

[4.4 System Workflow 32](#_Toc196862627)

[4.5 Database Model 33](#_Toc196862628)

[4.6 Security and Performance Considerations 34](#_Toc196862629)

[4.7 Technical specification of the project 35](#_Toc196862630)

[*4.7.1 Functional requirements* 35](#_Toc196862631)

[*4.7.2 Nonfunctional requirements* 36](#_Toc196862632)

[CHAPTER 5. TESTING AND VALIDATION 37](#_Toc196862633)

[5.1 Unit Testing and Integration Testing 37](#_Toc196862634)

[5.2 Usability Testing Results 38](#_Toc196862635)

[5.3 Scalability and Response Time Analysis 38](#_Toc196862636)

[5.4 Validation and Output Verification 39](#_Toc196862637)

[CHAPTER 6. RESULTS AND LIMITATIONS 40](#_Toc196862638)

[6.1 Results 40](#_Toc196862639)

[6.1.1 End-to-End System Validation 40](#_Toc196862640)

[6.1.2 User Experience and Interface Responsiveness 40](#_Toc196862641)

[6.1.3 Visual Data Insights 40](#_Toc196862642)

[6.2 Limitations 40](#_Toc196862643)

[6.2.1 Static ML Models and Lack of Real-time Learning 41](#_Toc196862644)

[6.2.2 Geographic and Data Generalizability 41](#_Toc196862645)

[6.2.3 Satellite and Sensor Data Integration 41](#_Toc196862646)

[6.2.4 Prediction Interpretability 41](#_Toc196862647)

[6.2.5 Mobile Optimization 41](#_Toc196862648)

[CHAPTER 7. CONCLUSION AND FUTURE WORK 43](#_Toc196862649)

[7.1 Future Work 43](#_Toc196862650)

[7.1.1 Real-Time Model Retraining and Self-Learning 43](#_Toc196862651)

[7.1.2 Integration of Remote Sensing and IoT Data 43](#_Toc196862652)

[7.1.3 Disaster Onset Prediction and Severity Estimation 43](#_Toc196862653)

[7.1.4 Multi-Language and Accessibility Features 43](#_Toc196862654)

[7.1.5 Integration with Disaster Management Agencies 43](#_Toc196862655)

[7.2 Conclusion 44](#_Toc196862656)

[References 45](#_Toc196862657)

## List of Figures

[Figure 3.1 - Schematic Diagram for Machine Learning Pipeline 18](#_Toc165469016)

[Figure 3.2 - Detailed diagram of the Ensemble Model. 21](#_Toc165469017)

[Figure 3.3 - Confusion Matrix for Landslide Prediction 22](#_Toc165469019)

[Figure 3.4 - Confusion Matrix for Avalanche Prediction. 23](#_Toc165469020)

[Figure 3.5 - Confusion Matrix for Drought Prediction. 23](#_Toc165469022)

[Figure 4.1 – Homepage of our web app 26](#_Toc165469023)

[Figure 4.2 – Fetching Weather Data 27](#_Toc165469024)

[Figure 4.3 – Dynamic form rendering for chosen disaster 28](#_Toc165469025)

[Figure 4.4 – Prediction made interface 29](#_Toc165469026)

[Figure 4.5 – Graph of recent predictions on Natural Earth 30](#_Toc165469027)

[Figure 4.6 – Disaster-wise casualties map 30](#_Toc165469028)

[Figure 4.7 – Graph of country-wise death count 30](#_Toc165469029)

[Figure 4.8 – Yearly disaster trend chart 31](#_Toc165469030)

[Figure 4.9- Log of recent predictions 31](#_Toc165469031)

[Figure 4.10 – System workflow model 32](#_Toc165469033)

Figure 4.11 – ERD Diagrams for Database Models………………………………………….33

## List of Tables

[Table 2.1 - Similar Applications with Features Comparison 16](#_Toc165469243)

[Table 3.1 – Performance metrics of each model 21](#_Toc165469243)

[Table 4.1 – Python Libraries Used 24](#_Toc165469246)

[Table 4.2 – Disaster Table with Details 33](#_Toc165469249)

[Table 4.3 – PredictionRecord Table with details 34](#_Toc165469260)

[Table 5.1 - Test Case Table (for Functional Testing) 37](#_Toc165469262)

# **CHAPTER 1. INTRODUCTION**

Imagine waking up to a sudden landslide wiping out roads or a devastating avalanche burying entire villages or an unexpected drought crippling agriculture. You might unconsciously begin thinking that these are some fictional scenes from movies, but they are not. These disasters are the harsh reality of the climate crisis we face now and then. Climate change is increasing the frequency and severity of such extreme weather events. Extreme weather and climate events, such as floods, cyclones, and heat waves, represent the climate’s variability (Ebi et al., 2021). The severity of these disasters kept escalating due to the unchecked burning of fossil fuels, industrial pollution, and deforestation. According to recent reports, the number of extreme weather events associated with hydrological phenomena has increased by 60% in Europe over the last three decades (Furtak & Wolińska, 2023). These events impact the infrastructure, agriculture, and energy systems and cause worldwide social and economic disruption.

Central Asia is particularly vulnerable to landslides, droughts, floods, avalanches, mudflows, and extreme temperatures, which cause an astounding economic loss of $10 billion every year (Burunciuc, 2020). This region is home to some of the world’s largest glaciers, such as those in the Pamir region of Tajikistan and the Tien Shan mountains of Kyrgyzstan. Consequently, the region faces frequent avalanches and landslides. At the same time, the steppes of Kazakhstan and Kyrgyzstan suffer the fate of recurring droughts, which devastate crops. Despite these significant hazards, limited research, conflicting data, and dataset gaps have hindered the qualitative analysis and in-depth research within high-risk areas (Thurman, 2011).

Relying heavily on statistical models and physical simulations, traditional weather forecasting isn't up to the task of providing accurate predictions of extreme weather events (Zhang et al., 2025). Such methods draw largely upon historical data, augmented by empirical techniques, yet they don't have the wherewithal to handle the kinds of disasters we now see as occurring with greater frequency—hurricanes, tornadoes, etc. By contrast, machine learning enjoys a near state-of-the-art performance in extreme weather forecasting due to its reliability and speed of computation. Moreover, machine learning provides an elite path toward extracting useful knowledge from the complex, non-linear, and often interrelated relationships that exist among different meteorological factors (Chen et al., 2023).

Over the last few years, weather prediction has evolved considerably due to the introduction of artificial intelligence (AI) and machine learning (ML). The forecasting accuracy can be improved by inputting large historical data to the ML models for unknown patterns. For instance, some ML models used for predicting promising weather patterns in various regions of the world include Decision Trees, K-means clustering, Linear Regression, and Convolution Neural Networks (CNNs) (Bochenek & Ustrnul, 2022). These models enable early warning systems, allowing communities and governments critical lead time to respond to disasters before they happen.

The purpose behind this project is to come up with predictive models that can forecast landslides, avalanches, as well as droughts. Considering Central Asia's topography, these phenomena represent a serious threat to human life as well as economic stability. Landslides, along with avalanches, are common in mountain areas, whereas droughts hit Kazakhstan and Kyrgyzstan, both being semi-arid steppes.

The biggest challenge in Central Asia in predicting extreme weather phenomena is a lack of precise climate data. Central Asia lacks detailed meteorological data as well as infrastructure, making the proper training of a predictive model difficult to accomplish. It also has a diverse topography, making generalization in differing conditions even more difficult. Existing methods are capable of forecasting a single type of calamity only. An entire system capable of foreseeing multiple types of extreme phenomena in optimal terms, regardless of differences in geography, needs to be developed.

The purpose of this project is to overcome these challenges through a strong forecast model tailored to Central Asia. The predictive model uses a combination of machine-learning methods along with climate data from a range of sources. It would fill a gap in weather forecasting in the region, offering an urgently required early warning system.

The establishment of an extreme weather forecasting system will benefit disaster preparedness and risk management interventions. Governments, municipal governments, and communities can take early precautions through accurate and timely forecasts, thereby minimizing the impacts of extreme weather phenomena. Early warning infrastructure will facilitate fast response by disaster response teams, minimizing death tolls and losses to infrastructure. More advanced forecasting will also inform better agricultural planning, enabling farmers to anticipate droughts and maximize irrigation schemes. Additionally, accurate weather forecasts will help governments to strengthen weak infrastructure before disasters, minimizing economic losses in the long term.

## **Problem Statement**

Current disaster and extreme weather forecasting systems are not effective and advanced enough to address several key issues:

* **Multi-event forecasting**: The majority of functional systems are capable of forecasting a single calamity. Such systems find it difficult to forecast calamities such as drought, avalanches, and landslides. This makes users switch between various predicting systems, resulting in the late issuance of forecasts and warnings.
* **Data Integration Challenges**: Traditional systems don’t effectively integrate data from different sources. As a result, the system misses out on important meteorological features, and predictions made are not reliable.
* **Real-time disaster predictions:** Lack of timely predictions leads to a delay in the mobilization of emergency responders in case of a catastrophe. These delayed responses lead to casualties and increased damage to infrastructure.
* **Limited Public Access to Information:** The public is often unaware of potential disasters due to a lack of information. In most cases, the information available is fragmented and too complex to interpret. Consequently, the people remain unprepared and vulnerable due to a lack of disaster risk assessment.

## **Proposed Solution**

This project proposes the use of machine learning models to predict the occurrence of avalanches, landslides, and droughts. The following are the key components of the proposed solution:

* **Predictive modeling**: Machine learning models such as Random Forest, Logistic Regression (LR), CatBoost, LGBM, and XGBoost would be used to train our models. These algorithms will analyze the historical data to generate accurate predictions.
* **Real-time data integration**: The system would utilize real-time data from sources such as OpenWeatherMap for on-the-go predictions.
* **Disaster Risk Assessment**: The system would generate the risk level based on the input meteorological features and would also provide recommendations and strategies.
* **Web-based interface**: A user-friendly interface would help non-technical people navigate through different sections and understand disaster predictions in real time.
* **Telegram Bot extension**: The Telegram bot would further make it easier for users to get timely predictions.
* **Geospatial Visualization**: Interactive maps and various graphs would be used to visualize the predictions and historical data.

## **1.3 Project Objectives**

* **Develop Predictive Models**: Design and implement a machine learning pipeline to predict avalanches, landslides, and droughts based on real-time meteorological data with greater accuracy.
* **Integrate Data from Multiple Sources**: Integrate data from multiple sources, like past data and weather forecast data.
* **Web-Based Dashboard**: Create a user-friendly dashboard that displays a predictions section, level of risk, selection of geographical region, geospatial maps, and interactive charts, as well as historical trends.
* **Telegram bot extension**: To develop a user-friendly bot to get instant predictions within the palm of your hand with a simple yet robust bot.
* **Safety Recommendations**: The system would provide safety tips and recommendations based on the predictions.
* **Scalable and Reliable Deployment**: Deploy the system on scalable platforms for continuous accessibility and performance.

## **1.4 Business Benefits**

* **Improved Disaster Preparedness**: The focus of the project is to offer early warnings and hazard evaluations to assist local communities and agencies.
* **Informed Decision-Making**: The information and forecast provided by the system would help policymakers and the government make appropriate decisions in the event of any emergency. Typical policies may include resource allocation, evacuation planning, and emergency response strategies.
* **Cost Reduction**: Timely predictions can reduce the costs associated with disaster relief efforts. By better preparation, the system can prevent or minimize the impact of disasters, thus reducing the associated recovery costs.
* **Increased Public Awareness**: The user-friendly web application would allow users to analyze the risks and take preventive measures. This can significantly reduce the loss in various sectors such as agriculture, energy, and insurance.
* **Scalability and Adaptability**: The system can be expanded to handle more disasters, making it an ideal choice in various geographical regions. Additionally, the system can be extended to process real-time data from satellites and other sources.

## **1.5 Contributions**

This project can contribute extensively to the field of extreme event forecasting, risk assessment, and disaster management. It will assist students and researchers in using machine learning techniques for disaster predictions. Additionally, the project involves the common mass and makes it convenient for them to access the disaster data. Moreover, this project would be available on my GitHub and would serve as a prototype for researchers and students for further research. There are a lot of disaster forecasting apps, but very few of them utilize machine learning techniques. Furthermore, most of these apps target a specific disaster, thus lacking the multi-forecasting ability. Our app would not only allow the prediction of multiple disasters but would also leave room for the integration of other disaster data in the future. Apart from being a robust app, it will be a reliable and cost-effective solution for governments, local agencies, and emergency responder teams.

# **CHAPTER 2. LITERATURE REVIEW**

Predicting extreme weather events has long been a critical area of research. However, due to climate change, the need for a robust and efficient disaster forecasting system has increased. Camps‐Valls et al. (2025) explain that the impact of climate change is observable in the growing intensity and frequency of weather events, which are detrimental to the environment and humanity. These events include long periods of drought, severe floods, destructive typhoons, and avalanches. Methods of disaster forecasting based on statistics, including numerical weather prediction (NWP), are obsolete. They have profound inadequacies because of the intricate, ever-changing, and non-linear interrelations that exist among the components of the climate. Recently, advances in technology such as deep learning methods have shown vast potential in the prediction as well as analysis of extreme events (Camps‐Valls et al., 2025). Through the application of various algorithms, it is possible to identify and interpret multiple relationships and trends within big datasets. Unfortunately, datasets containing meteorological information are often heterogeneous, noisy, and limited, which impacts the performance of the algorithms.

A well-optimized forecasting model learns from data from various sources to capture inherent climate characteristics. Data from multiple sources, such as topographical, atmospheric, and meteorological observations, form typical data used to train a model. Heterogeneous data are consistently integrated into studies: Examples include soil moisture, hydrological variables, and atmospheric variables, in the context of drought forecasting (Nandgude et al., 2023). Such data can be obtained from IOT devices, satellites, and weather stations (Nandgude et al., 2023). Despite having a vast amount of data, a significant challenge comes in integrating data effectively. Current studies indicate that a vast amount of data is not being used to enhance accuracy in forecasting disasters (R. Chen et al., 2020). ML algorithms perform flawlessly when learned using such rich datasets. For example, studies have discovered that ML model accuracy grows by incorporating remote sensing data in landslide forecasting (Akosah et al., 2024).

Machine learning, more specifically, supervised machine learning, has widespread applications in forecasting extreme phenomena. For instance, river water level or drought index can be considered a regression problem, whereas forecasting or classification of the category would be a classification example. Mosavi et al. (2018) showcased the potential of ML algorithms like support vector machine (SVM), Decision Trees, Random Forest, and Artificial Neural Network in flood forecasting more accurately. Some other studies have shown the superior performance of ML models over classical statistical models. For instance, Nandgude et al. (2023b) used a number of ML models to forecast drought in Algeria, in which SVM performed better than other models with a coefficient of determination of 0.95. Similarly, tree-based ensemble models along with logistic regression have also produced encouraging results in flood as well as landslide predictions. Supervised ML models get an advantage over simplistic statistical models on account of being able to identify subtle relationships between predictor and outcome variables. With a limitation of needing plenty of data to train upon, these models are not always possible when rare disasters are to be forecasted. The datasets are generally unbalanced (few “extreme” instances vs a lot of normal instances). Techniques like cross-validation, feature selection, and hyperparameter tuning prove to be helpful in such a situation to ensure generalization.

Recent applications of machine learning models in disaster forecasting indicate an increased intersection of AI technology and extreme weather forecasting. Multiple trends are promoting the use of AI to forecast extreme phenomena. One of those trends includes developing “Trustworthy AI” for forecasting weather and climate phenomena. Beyond simple forecasting, Generative models such as GANs (Generative Adversarial Networks) and VAEs (Variational Autoencoders) are employed to enhance data. To demonstrate this, Camps-Valls et al. (2025b) employed VAE to enhance the prediction of heatwaves by generating various variations of patterns of heatwaves. Additionally, interdisciplinary collaboration illustrates the involvement of climate scientists, hydrologists, engineers, computer experts, as well as policymakers. All these examples show the rise of AI in extreme event forecasting.

Central Asia is subject to various disasters like drought, earthquakes, avalanches, and landslides. These countries have been using traditional methods of forecasting for a long period of time. Over recent years, this region has seen a boost in the implementation of AI in weather and climate forecasting. Sadrtdinova et al. (2024b) employed machine learning classifiers (logistic regression, random forests, gradient-boosted trees) and a deep learning model to forecast drought. Their predictive models achieved an impressive 97-99% accuracy at a 6-month lead time. Xu et al. (2024) also came up with a machine learning model that was able to forecast 33% of flash drought. With this, international agencies like the World Bank are incorporating ML into drought monitoring platforms. These developments are a giant leap towards using AI techniques for disaster forecasting.

The magnificent Tien Shan, as well as the Pamir mountain range, are found in Central Asia. These mountain areas are prone to snow avalanches. Scientists are attempting to find these high-risk areas and forecast avalanches well in advance. As an example, Rahmati et al. (2019) employed ML models to identify the likelihood of avalanche activity in two mountain ranges (the Iranian Alborz and a section of the European Alps). Similarly, Wen et al. (2022) were able to identify avalanche hazard areas by using ML techniques in the high-altitude Tibetan Plateau region. In a similar work, Bian et al. (2022) utilized an ensemble classifier to enhance avalanche hazard mapping accuracy. Based on earlier work, Choubin et al. (2020) employed four algorithms: generalized additive model (GAM), multivariate adaptive regression splines (MARS), boosted regression trees, and support vector machines (SVM) to determine avalanche predictive features. Among these, their models were highly accurate to a level of more than 88%. It reflects the potential of ML models to identify these terrain, snowfall, and climate interactions that contribute to avalanches effectively.

The geographic nature of Central Asian, particularly Tajikistan and Kyrgyzstan, makes them prone to landslides. Landslide susceptibility assessment in these areas by traditional methods fails because of the vast extent of geographic areas involved, along with remote sites. Machine learning models are filling this gap by using, as well as learning from, various data (e.g., rain, geology, topography, earthquakes) (Rosi et al., 2023). In a pioneering work, Rosi et al. (2023) produced the first detailed landslide susceptibility map covering the entire region. They employed a dataset of 13,000 historic landslide reports to train a random forest model that could forecast areas of landslide susceptibility. Several other authors have also contributed a lot to this field of work. For instance, X. Chen et al. (2024) developed a “Tien Shan–Kunlun Mountains Landslide Susceptibility Model (TKLSM)” to estimate the risk of landslides in data-scarce areas. This model estimated an annual casualty rate of 3–4 in Kyrgyzstan and Tajikistan, much lower than in remote regions (X. Chen et al., 2024b). Overall, Random Forest and other ML models have shown great results in landslide susceptibility mapping​ in the region. More work needs to be done on the quality of data and data integration from other sources for a robust and reliable landslide forecasting model in Central Asia.

Despite the increase in ML use for disaster predictions, several research gaps remain. For instance, most countries still rely on experts for avalanche prediction. This indicates the hesitation in applying ML and data-driven methods for hazard forecasting. Secondly, most of the existing ML-integrated systems hardly evaluate the performance of different algorithms. Researchers have noted that “only a few studies” compare different ML models for drought predictions (Oyounalsoud et al., 2024). Another major challenge is the class imbalance in hazard datasets. Avalanches and landslides rarely occur compared to non-hazardous instances. This skewness highly impacts the learning of the model as it gets accustomed to the non-hazardous instances. Furthermore, the lack of models trained on Central Asian data implies that models developed elsewhere are used in the Central Asian context. Acharya et al. (2023) state that Central Asian mountains have “so far remained inadequately studied” in avalanche forecasting. The models used in these contexts are trained in Europe or North America, so they don’t capture the unique geographical and climate conditions. Central Asia is a “data-scarce region,” thus making it even harder to train a model locally for drought or any other disaster prediction (Pyarali et al., 2022).

In conclusion, we have seen great advancement in the use of Machine Learning for droughts, avalanches, and landslides forecasting in Central Asia. Different models, such as Support Vector Machine (SVM) and Random Forest, have shown promising results in extreme weather forecasting. Government and organizations are switching to ML for better hazard predictions, early warnings, and risk assessments. This contributes to the body of knowledge and finds practical implications in risk reduction strategies. As climate change increases the frequency and intensity of natural hazards, ML-integrated systems play a pivotal role in safeguarding communities and infrastructure.

## **2.1 Similar applications**

The table below shows the comparison of our website with similar applications. For a brief overview, we have mentioned 3 applications: NASA Earth Exchange, UN-SPIDER Risk Tool, and GFDRR GeoNode. Many of these applications focus on a single type of disaster, and they don’t use machine learning for predictions, making them less reliable. Similarly, they lack location-based prediction. Our web app uses ML and weather APIs to predict extreme events in real-time. Furthermore, it offers interactive maps and visualizations to locate the risk zones and analyze past disasters. Given below are the links to similar applications.

* <https://www.nasa.gov/nasa-earth-exchange-nex/>
* <https://www.un-spider.org/risks-and-disasters>
* <https://www.gfdrr.org/en/onlinetools>

Table 2.1 - Similar Applications with Features Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Features** | **NASA Earth Exchange** | **UN-SPIDER Risk Tool** | **GFDRR GeoNode** | **My App** |
| Predicts multiple disasters | Drought, Flood | Flood, Landslide | Earthquake, Flood | Drought, Avalanche, Landslide |
| Uses Machine Learning | Deep Learning | Rule-based | GIS-driven | RF, LR, Stacking |
| Disaster-specific tips for end users |  |  |  | Contextual safety tips |
| Real-time weather integration |  |  |  | OpenWeatherMap |
| Interactive web dashboard |  | Limited UI | Map UI | Full dashboard |
| Risk prediction using local weather input |  | From satellite |  | Form-based manual inputs |
| Free and open-source | Yes | Yes | Yes | Yes |

# 

# **CHAPTER 3. Proposed Methodology**

This project uses weather and topographical data to predict avalanches, droughts, and landslides. The proposed methodology consists of several stages: data collection, preprocessing, feature engineering, model training, validation, and deployment. Each stage addresses a specific challenge in hazard forecasting, like data sparsity, regional heterogeneity, and model reliability. Several machine learning algorithms are used to train on the disaster datasets. The process starts with data acquisition from different sources, then the processing stage uses different Python libraries, and ultimately, the clean data is fed into the machine learning model. The training phase is a trial-and-error approach phase where the ML models are tuned for greater precision and accuracy. Figure 3.1 shows the proposed methodology for model development.

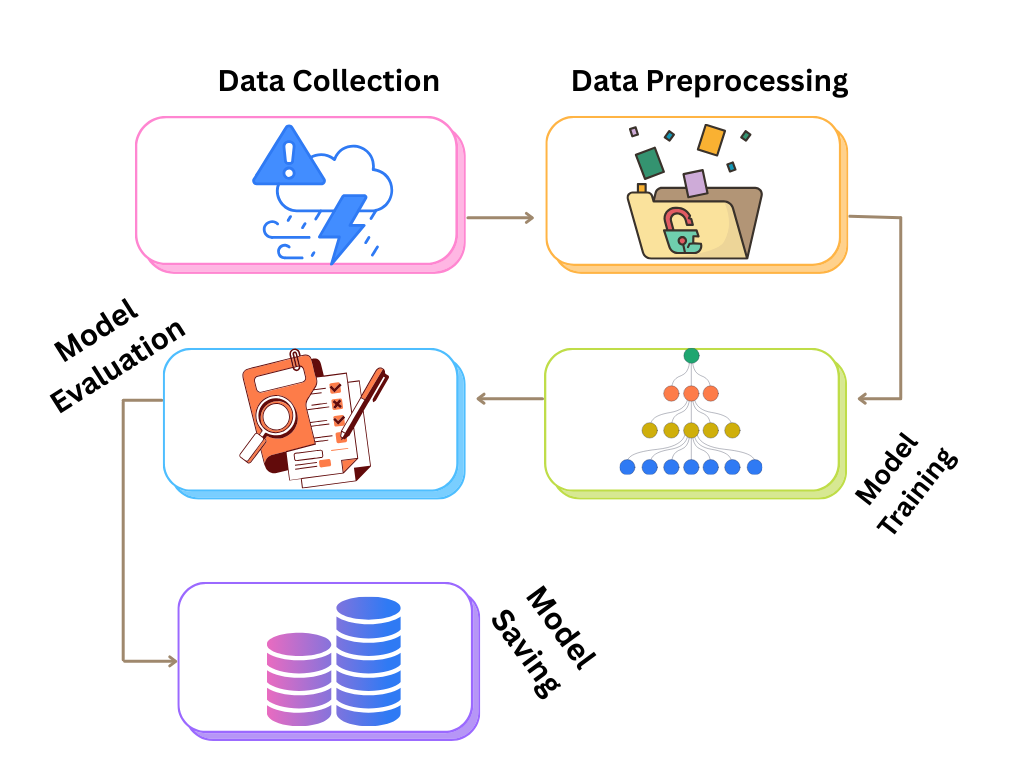


Figure 3.1 - Schematic Diagram for Machine Learning Pipeline

## **3.1 Data Acquisition and Integration**

In the first step, we gather relevant data from different credible sources. We used NASA’s POWER Project, NOAA, and Kaggle’s datasets to obtain climate variables, including temperature, precipitation, wind speed, and humidity. The obtained data is converted into tabular form for further processing. Additionally, we obtained data from EM-DAT for historical records of avalanches, landslides, and droughts. We have a dataset of 99984 records and 19 columns for our drought model. The following are the columns in our drought dataset: PRECTOT (Total Precipitation), PS (Surface Pressure), QV2M (Specific Humidity at 2 Meters), T2M (Temperature at 2 Meters), T2MDEW (Dew/Frost Point Temperature at 2 Meters), T2MWET (Wet Bulb Temperature at 2 Meters), T2M\_MAX (Maximum Temperature at 2 Meters), T2M\_MIN (Minimum Temperature at 2 Meters), T2M\_RANGE (Temperature Range at 2 Meters), TS (Earth Surface Temperature), WS10M (Wind Speed at 10 Meters), WS10M\_MAX (Maximum Wind Speed at 10 Meters), WS10M\_MIN (Minimum Wind Speed at 10 Meters), WS10M\_RANGE (Wind Speed Range at 10 Meters), WS50M (Wind Speed at 50 Meters), WS50M\_MAX (Maximum Wind Speed at 50 Meters), WS50M\_MIN (Minimum Wind Speed at 50 Meters), WS50M\_RANGE (Wind Speed Range at 50 Meters) and Score. Likewise, we have a dataset of 5000 records with 6 columns for our landslide model. The following are the columns of our landslide dataset: Temperature, Humidity, Precipitation, Soil moisture, Elevation, and Landslide Risk Prediction. Finally, we have a dataset of 881934 records with 6 columns for our avalanche model. The following are columns for our avalanche dataset: Elevation, Temperature, Wind Speed, Humidity, Snow Depth, and Avalanche Activity.

## **3.2 Data Cleaning and Preprocessing**

The next stage involves cleaning and preprocessing of raw datasets. Data pre-processing plays a huge role in the performance of ML algorithms. According to an estimate, the cleaning and preprocessing stage takes around 80% of the total time in building a model (Maharana et al., 2022). We started our preprocessing step by handling the missing values. We deleted the records with sparsely missing values. We also used median imputation and interpolation techniques to fill in missing values in different columns. In the second step, we standardized our features using the StandardScalar function. The third step included the label encoding process. Our data had disaster intensity classes such as Low, Medium, and High, so we mapped them into categorical classes: Low (0), Medium (1), and High (2). In case of the drought dataset, we had a continuous range of values for our score (target variable), so we reclassified the data into three distinct categories: 0 for Low, 1 for Medium, and 2 for High. This technique was useful in making the output data more interpretable and actionable for decision-making. It is important to mention that we had the drought observations weekly, so we aggregated the data to make it simpler to work with. In the fourth step, we performed the data augmentation using SMOTE (Synthetic Minority Oversampling Technique). Natural disasters have severe class imbalance, as was exemplified by our avalanche dataset (0: 699720, 1: 182214). The imbalanced dataset makes the model more biased towards the majority class. To address this issue, techniques like SMOTE are used to either create extra samples for the minority class or to remove the samples from the majority class (Matharaarachchi et al., 2024). In our case, we generated the synthetic examples of the minority class using the over\_sampling method of SMOTE. Additionally, we had quite a lot of outliers in our datasets. We applied the interquartile range (IQR) method to detect the outliers and eliminate them from our datasets.

## **3.3 Model Selection and Training**

We trained three different models for each type of disaster.

* Avalanche: We used a Random Forest Classifier for the avalanche dataset. Random forest is an ensemble [machine learning technique](https://www.sciencedirect.com/topics/engineering/machine-learning-technique) for regression analysis and classification (Salman et al., 2024). Random forest works by constructing multiple decision trees and taking their average (in case of regression) or mode (in case of classification) for the output. During the training process, we tuned the hyperparameters to enhance the performance of our model. Our Random Forest Model had 100 decision trees (n\_estimators=100) and a random\_states of 42 to ensure reproducibility. The optimal split within each decision tree was determined using the Gini Index method.
* Landslide: We used a Logistic Regression model for our landslide dataset. Logistic regression is a model that estimates the association of one or more independent variables with a dependent variable (Schober & Vetter, 2021). Logistic regression is simple to implement, computationally effective, and capable of managing multiple feature types. Our logistic regression model contained a class\_weight='balanced' to address the skewed class distribution. In addition, random\_state=42 makes it reproducible, and max\_iter=1000 defines the optimal number of iterations.
* Drought: We employed a stacking ensemble model comprising XGBoost, CatBoost, LightGBM, and Random Forest for training. Stacking ensemble refers to a technique where more than one machine learning model is combined using a meta-learner in an effort to enhance overall performance. Khoshkroodi et al. (2024) found that the ensemble model predicts more accurately than individual models. The base models in our stacking ensemble model are CatBoost, XGBoost, LightGBM, and Random Forest, whereas Logistic Regression serves as our meta-model. We have set each model’s hyperparameters using a grid search technique in a bid to grasp the underlying pattern in the dataset. Let’s discuss every model of our stacking model.
  + CatBoost: It performs superbly using the gradient boost strategy on categorical features. It was reported by Hancock and Khoshgoftaar (2020) that boosting makes CatBoost a perfect fit where data has sparse or rare-occurring target variables. Especially true in disaster data, where rare hazardous instances are present in comparison to non-hazardous instances. Regarding parameters, we have iterations=400 to regulate the number of trees, learning\_rate=0.01 for slow learning, and depth=10 to manage model complexity.
  + XGBoost: It employs the regularized gradient boosting approach using parallel processing. It achieves its performance optimization through tree pruning along with hardware optimization (Ali et al., 2023). For this model, we utilize n\_estimators=500 for controlling the number of trees, learning\_rate = 0.05 to avoid overfitting, and max\_depth=10 to track the model’s complexity.
  + LightGBM: Another algorithm that follows the gradient Boosting Decision Tree approach to make accurate predictions. This model uses leaf-wise tree growth instead of the common level-wise tree growth to enhance the training process. For LGBM, our n\_estimators=500, learning\_rate=0.05, and num\_leaves=64 (to control the model’s complexity).
  + Random Forest: The last of our base models that works on ensemble decision trees using bagging. For random forest, our n\_estimators=500 and max\_depth=15.
  + Logistic Regression: The meta-model of our stacking model. The Logistic Regression (LR) trains on the output from all the base models and makes the final prediction. The class\_weight='balanced' parameter of the LR model handles the imbalanced dataset.

Figure 3.2 shows the process of making predictions through an ensemble model.

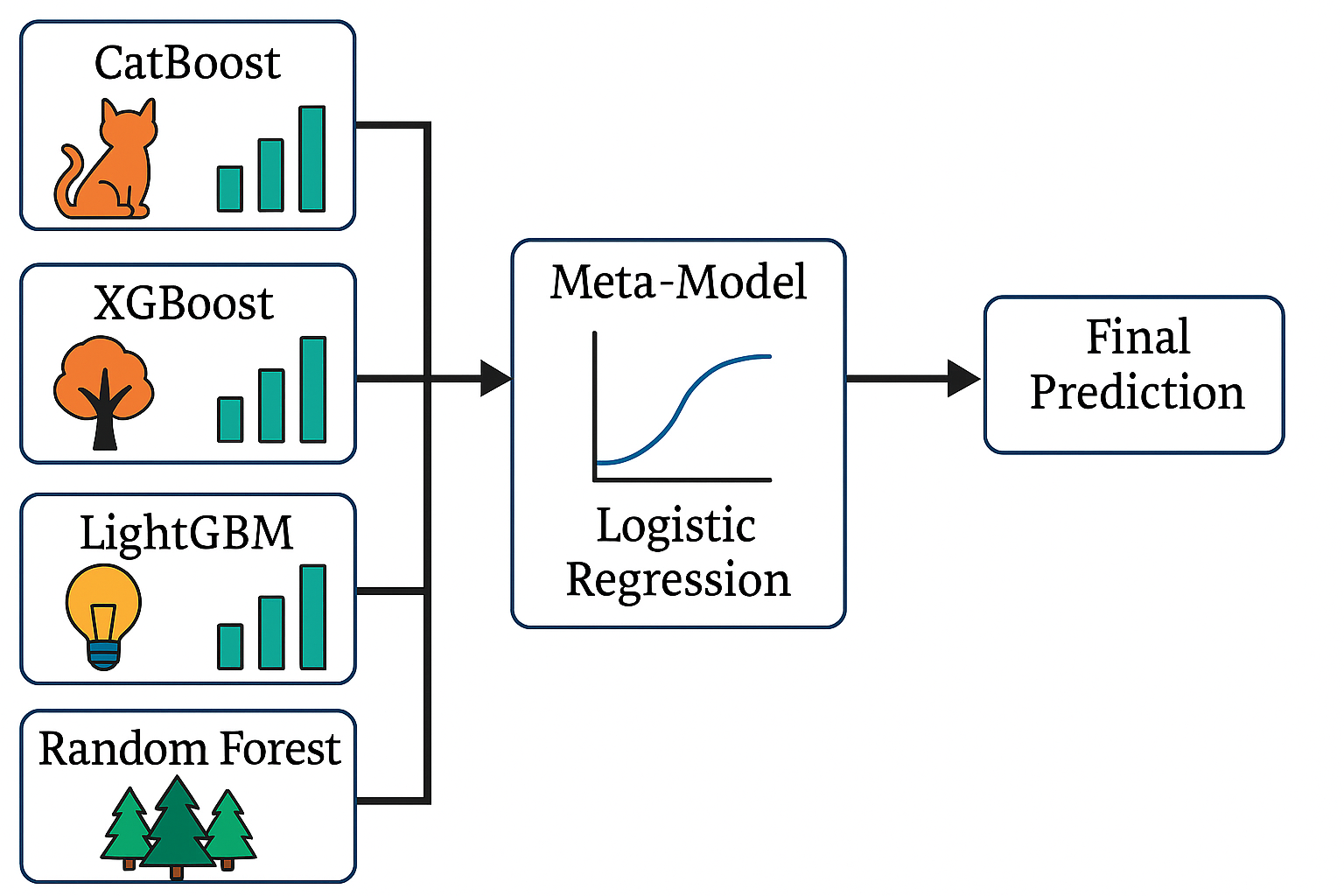


Figure 3.2 – Ensemble Model

## **3.3 Model evaluation**

We evaluated our models on various metrics such as accuracy, F1-score, and confusion matrix. All three datasets were split into 80% for training and 20% for testing. Table 3.1 shows the values for the performance metrics for each model.

Table 3.1 – Performance metrics of each model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Disaster | Model | Train Accuracy | Test Accuracy | Cross-Validation Score ) |
| Landslide | Logistic Regression | 89.7% | 90.0% | 89.6% ± 1.1% |
| Landslide | Decision Tree | 78.0% | 76.5% | 75.9% ± 2.0% |
| Landslide | SVM (Linear Kernel) | 82.5% | 81.0% | 80.8% ± 1.5% |
| Avalanche | Random Forest | 88.0% | 86.0% | 85.8% ± 0.1% |
| Avalanche | K-Nearest Neighbors (k=5) | 75.0% | 72.5% | 73.2% ± 2.4% |
| Avalanche | AdaBoost Classifier | 81.5% | 80.0% | 79.8% ± 1.3% |
| Drought | Stacking Ensemble | 84.5% | 82.0% | 81.5% (±1.7%) |
| Drought | Random Forest | 78% | 76.2% | 75.8% (±1.5%) |
| Drought | XGBoost (alone) | 79.5% | 75.5% | |  | | --- | |  |  |  | | --- | | 77.9% (±1.2%) | |

Our logistic regression model for landslide prediction has a training accuracy of 89.7% and a test accuracy of 90.0%. We also checked our model through the cross-validation approach. Our cross-validation revealed a uniform range of scores, ranging from 88.2% to 91.1%. The mean cross-validation score of 89.6% ±1.1% illustrates our model's stability across various subsets of data. The balanced accuracy, precision, recall, as well as f1-score of 0.90 demonstrate our model's reliability in landslide predictions. Figure 3.3 illustrates our logistic regression model's

Confusion matrix.

A blue squares with white text

AI-generated content may be incorrect.

Figure 3.3 – Confusion Matrix for Landslide Prediction

Our model of random forest for avalanche prediction achieved a training accuracy of 88% and a test accuracy of 86.0%. Cross-validation ranged from 85.7% to 85.9%. Cross-validation average was 85.8% ±0.1%, reflecting its high stability level. The model achieved a 90% recall score in ensuring correct identification of non-avalanche instances and 83% in avalanche instances. The F1-scores of 0.90 show the model’s reliability in avalanche risk assessment. Figure 3.4 shows the confusion matrix for the random forest model for avalanche forecasting.

A blue squares with black numbers

AI-generated content may be incorrect.

Figure 3.4 – Confusion Matrix for Avalanche Prediction

Our stacking ensemble model for drought prediction achieves an accuracy of 82%. The model was validated using the K-fold cross-validation technique to create unbiased predictions. The model performed exceptionally well in the "Low" drought category (precision: 0.81, recall: 0.83) and the "Extreme" category (precision: 0.84, recall: 0.79); however, it achieved a moderate performance (precision: 0.67, recall: 0.56) for the "Medium" drought category. Figure 3.5 shows the confusion matrix for the stacking ensemble model for drought forecasting.

A blue squares with numbers and black text

AI-generated content may be incorrect.

Figure 3.5 – Confusion Matrix for Drought Prediction

## **3.4 Risk Level Classification**

We convert the categorical outputs of the model into different risk levels (Low, Medium, High/Extreme). The user gets the output, which aligns with the historical impact data. The risk level classification provides early information to emergency responders and the severity of the threat to the public.

# **CHAPTER 4. IMPLEMENTATION**

We have created an interactive web application capable of predicting the risk levels for drought, landslides, and avalanches. The whole process starts with training our models. We use Python’s built-in Pandas and Numpy libraries to read the datasets and perform numerical operations. Typical operations include describing the datasets and segregating the feature and target columns. We then use the model\_selection method from the Sklearn library to split the data into training and testing portions. Additionally, we use the preprocessing and imputation methods from the Sklearn library for data cleaning and processing. Matplotlib and Seaborn libraries are used for visualization and finding the correlation between different features. Our disaster models are then trained using the Scikit-learn, XGBoost, CatBoost, and LightGBM libraries. Finally, we use the Joblib library to save our models. We then integrated these trained models into a Django-based web application and a Telegram Bot capable of real-time risk assessments.

## **4.1 Technology Stack**

I used my HP laptop (Intel® Core i5) with 32 GB RAM for the project. Additionally, Python 3.9.2 was used for the project implementation. I used VSCode and Jupyter Notebook as the development environment. Below is the detailed information about the technologies used for the implementation.

* **Frontend:** HTML5, CSS3, Bootstrap 5, JavaScript
* **Backend:** Python 3.11, Django 4.x
* **ML Frameworks:** Pre-trained models using Scikit-learn, XGBoost, LightGBM, CatBoost
* **Libraries:** Pandas, Numpy, Joblib
* **Data Storage:** SQLite3 (Django ORM)
* **Visualization:** Matplotlib, Seaborn, and Plotly Express for interactive graphs
* **External APIs:** OpenWeatherMap API for current weather data
* **Hosting Environment:** A dedicated server to host our web app

Table 4.1 – Python Libraries Used

|  |  |  |
| --- | --- | --- |
| **Name of Library** | **Version** | **Functionality** |
| Django | 4.1 | Python’s framework for web development |
| numpy | 1.21.2 | It is a basic package for dealing with large datasets and matrices. (NumPy, n.d.) |
| pandas | 1.3.3 | Python’s package to analyze and manipulate data. |
| Joblib | 1.1.0 | A library to save and load machine learning models. |
| Plotly | |  | | --- | |  |  |  | | --- | | 5.4.0 | | An interactive library for creating dynamic plots. |
| Matplotlib | 3.4.3 | A 2D plotting library for creating static, animated, and interactive visualizations in Python. |
| Seaborn | 0.11.2 | A data visualization library based on Matplotlib |

## **4.2 Development Approach**

We have used the Agile methodology to develop the disaster risk prediction system. The iterative approach allowed us to incorporate constant feedback and adapt to new insights as we progressed. Agile focuses more on adaptation than process optimization (Omonije, 2024), which is particularly suitable for this project because we were dealing with a machine-learning project. We divided the project into several sprints, each lasting 3-4 weeks. This allowed the modular development of our project, from data gathering and training to the frontend and backend.

At the start of the project, a project backlog was set up, specifying tasks and deadlines. Tasks were assigned for a duration based on difficulty level. Meetings were conducted with the supervisor to track the project’s progress. The following are the sprints for our project.

Sprint 1: Data collection and preprocessing.

Sprint 2: Training of ML models and evaluation.

Sprint 3: UI/UX design for our web application and telegram bot.

Sprint 4: Development and deployment of the Telegram bot.

Sprint 5: Database model and Django project setup.

Sprint 6: ML models and API integration into the Django project.

Sprint 7: Visualization dashboard and user interface enhancements.

We conducted testing and reviews at the end of each sprint. A working demo was presented to the supervisors and peers for usability and performance testing. It was essential in refining the subsequent sprint’s plans. The agile approach was very beneficial in the context of our project. For instance, models were changed and updated based on the feedback from the supervisor for improved accuracy and performance. Additionally, it allowed the independent development of modules like ML models and the frontend. Moreover, core issues were identified earlier due to rigorous testing and evaluation.

## **4.3 System Architecture**

The modular and scalable system architecture ensures real-time responsiveness and extensibility. Our disaster prediction system integrates machine learning models and weather APIs into a Django-based web application. The system's architecture can be divided into six key components: User Interface (UI), Weather Fetch Module, Dynamic Form Renderer, Model Prediction Pipeline, Visualization Engine, and Database Logger.

* **User Interface (UI)**: Our web application has an interactive and visually appealing user interface. It is implemented using Django templates. Bootstrap 5 and AJAX (via jQuery) also work with the basic templates for responsive layout and dynamic behavior. Some key features include dropdowns for disaster selection (Drought, Avalanche, Landslide) and input fields for meteorological data. Figure 4.1 shows the homepage of our application. The asynchronous JavaScript calls (AJAX) work in the background, so the user doesn’t have to refresh the page. Furthermore, our interface also offers interactive and real-time charts, such as choropleth maps and bar plots.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.1 – Homepage

* **Weather Fetch Module**: Django’s RESTful view named fetch\_weather sends an API request to the OpenWeatherMap API. It returns the current weather data in JSON format, which can be used for hazard prediction. The user inputs the name of any particular city, and the system retrieves the important weather parameters such as Temperature, Humidity, Atmospheric Pressure, and Wind Speed. Figure 4.2 shows a section of the homepage for fetching weather data. These parameters are automatically filled in the corresponding fields. Some fields, like the elevation, might not automatically fill because the OpenWeatherMap API doesn’t return such a parameter. So, in that case, the user manually chooses the value. This module ensures the real-time integration of weather data into our system.

A screenshot of a weather report

AI-generated content may be incorrect.

Figure 4.2 – Fetching Weather Data

* **Dynamic Form Renderer:** This component ensures that only the fields related to the chosen disaster are displayed to the user. When a disaster is selected from the drop-down menu, the frontend sends a request to the get\_fields view. The Django view returns a list of features depending on the type of disaster. For instance, it would return snow depth for Avalanche and soil moisture for Landslide. A typical returned response for a response would have a display name, feature name, and acceptable value range (minimum and maximum). Figure 4.3 shows the dynamic form after selecting a disaster on our homepage.

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 4.3 – Dynamic Form Rendering

* **Model Prediction Pipeline:** As discussed earlier, we have trained three different models for each type of disaster. Furthermore, each model is accompanied by a preprocessing scaler (e.g., StandardScaler) to normalize the inputs before feeding them into the algorithm. When the user submits the values for different fields, they are converted into floats. A function checks the value against the realistic range for environment variables. The values are then converted into a pandas DataFrame for further processing. As mentioned earlier, the scalar is then used to standardize the values. Depending on the type of chosen disaster, the model makes a prediction and returns the class label. This label is then mapped to one of the three risk levels (Low, Medium, and extreme/high) and displayed to the end user.
* **Risk Feedback and Safety Recommendations:** Our disaster prediction system generates a risk assessment, consisting of three key elements: the predicted risk level, disaster label, and a set of safety recommendations specific to the disaster type. The recommendations are generated based on the disaster type. For instance, in the case of a high risk for avalanche prediction, the recommendation would be “avoid high elevations during snowstorms”. Figure 4.4 shows the risk level and a safety message. This human-centered design communicates the risk level and provides safety precautions in hazardous conditions.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.4 – Prediction

* **Visualization Engine:** We used Plotly Express to display interactive and aesthetically pleasing visualizations on our dashboard. Firstly, we have a Natural Earth map which automatically updates based on recent predictions.Suppose we chose ‘Naryn’ as our location and call our weather API to get the weather data, then make a prediction. The prediction for Naryn will appear on the map, with a color specifying the risk level. The green color on the map would show a low risk, while red would depict a high probability of hazard occurrence. Figure 4.5 displays our recent predictions on the world map. Additionally, we have a choropleth map on our dashboard page. The choropleth map displays country-level and disaster-wise distribution of casualties using projections like Robinson and Natural Earth. We have used different color shades to map the country-wise casualties on the map. A dark blue region specifies a region with the highest death toll, while a brown region means countries with fewer casualties. Furthermore, we have bar charts on our visualization section. These charts display the frequency of disasters and the regional total deaths. These charts and maps are interactive, responding to hover actions and allowing zooming or filtering. Figures 4.6 and 4.7 show the casualties caused by each disaster and the country-wise deaths, respectively.

A map of the world

AI-generated content may be incorrect.

Figure 4.5 – Recent Predictions on Natural Earth

A map of the world

AI-generated content may be incorrect.

Figure 4.6 – Disaster type casualties

A map of the world

AI-generated content may be incorrect.

Figure 4.7 – Country-wise Total Death

A graph of a graph

AI-generated content may be incorrect.

Figure 4.8 – Disaster Trends Chart

* **Database Logger:** We use Django’s built-in Object-Relational Mapper (ORM) as a logging system for our prediction records. When a user makes a successful prediction, an instance of PredictionRecord is created in the ORM. The instance consists of various fields such as disaster\_type, risk\_level, city, latitude/longitude, and prediction\_time. Disaster type may be one of the three disasters (landslide, avalanche, or drought), and risk level may be either low, medium, or high. The city will be a user-provided location, and the latitude/longitude will be automatically fetched from Geo-coordinates. Finally, the prediction time contains the timestamps when the prediction was made. This helps create a recent predictions table on the dashboard page, as shown in Figure 4.9, and users can keep track of risk levels.

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 4.9 – Recent Predictions

## **4.4 System Workflow**

The implementation flow follows the user from landing on the homepage to final predictions and visualizations. When users access our web app, they are directed to the homepage. The homepage contains a navbar, a sidebar, and the main body for rendering disaster predictions. The user can choose the city name and fetch weather data for it. The fetched data is automatically filled into the appropriate input fields. Once the user retrieves the weather data, he/she can choose the disaster from the dropdown menu. The frontend automatically generates the relevant form with input fields upon selecting the option. Each disaster form has input fields with labels and min/max limits. Furthermore, placeholder values and out-of-range validation warnings also help users to have a responsive and interactive experience. Once all fields are filled and the form is submitted, the system validates the values and converts them into floats. Upon successful validation, the pretrained models in the backend make a numeric prediction. This numeric prediction is returned and mapped to risk levels. The end user gets the risk level and associated cautionary message on the homepage.

In the meantime, the prediction history is saved in the PredictionRecord model. The users can now switch to the dashboard page and check out their recent predictions in a table. The same prediction also updates on a natural Earth map. You can hover over the map and see the precise location of your chosen city. Additionally, the dashboard page contains choropleth maps and other charts. These show the statistics of past disasters, including the death toll and geographical location. This system workflow offers a personalized journey to the user, from a dynamic form to a comprehensive analysis of disasters. Figure 4.10 shows the system's workflow for our application.

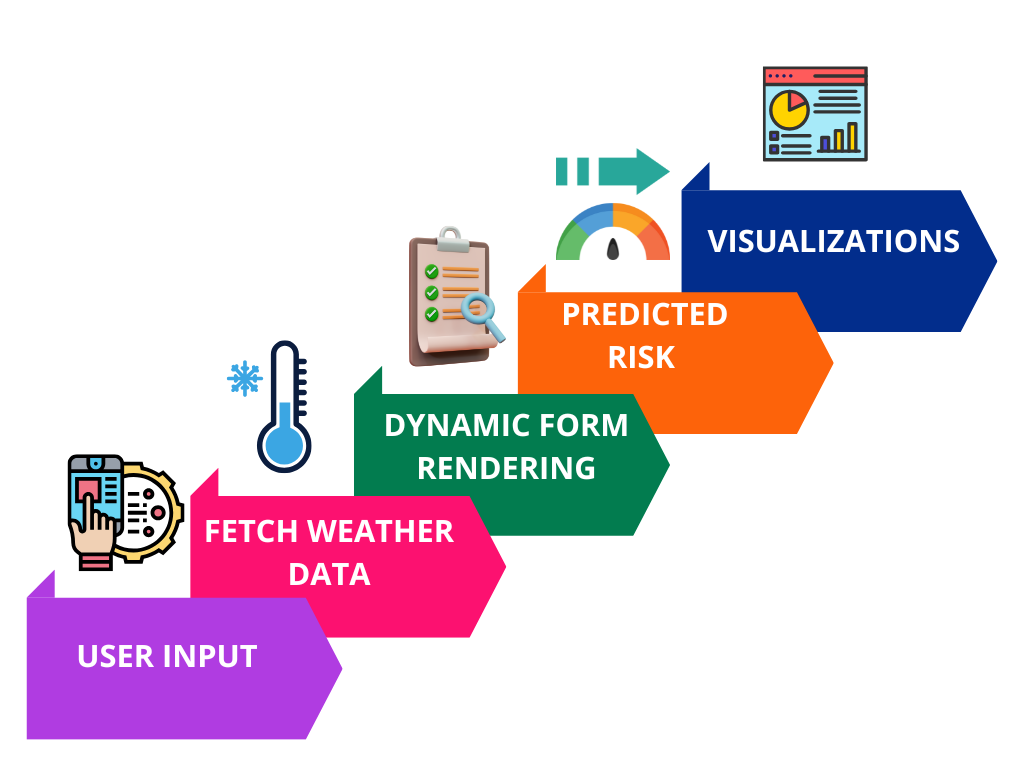


Figure 4.10 – System Workflow

## **4.5 Database Model**

We have used Django’s default SQLite as the database model for our project. We have two models: Disaster and PredictionRecord Model. Figure 4.11 shows the ERD diagrams for our models. We have saved the historical data in the disaster model for analytical purposes. The column for this model includes year, disaster type, country, location, total deaths, total affected, and CPI. This model helps to create sophisticated visualizations, including choropleth maps and bar charts showing disaster distribution. The PredictionRecord model stores the output of predictions made by users. Each time a prediction is made, it serves as a logging mechanism. The fields for this model include disaster type, risk level, city, latitude/longitude, and prediction time. This model can serve many important functions, such as logs of recent predictions and risk maps of predicted hazards. These models offer a strong base for assessing disaster risks. By examining past patterns and current predictions, users can gain a clearer understanding and evaluate risks thoroughly.

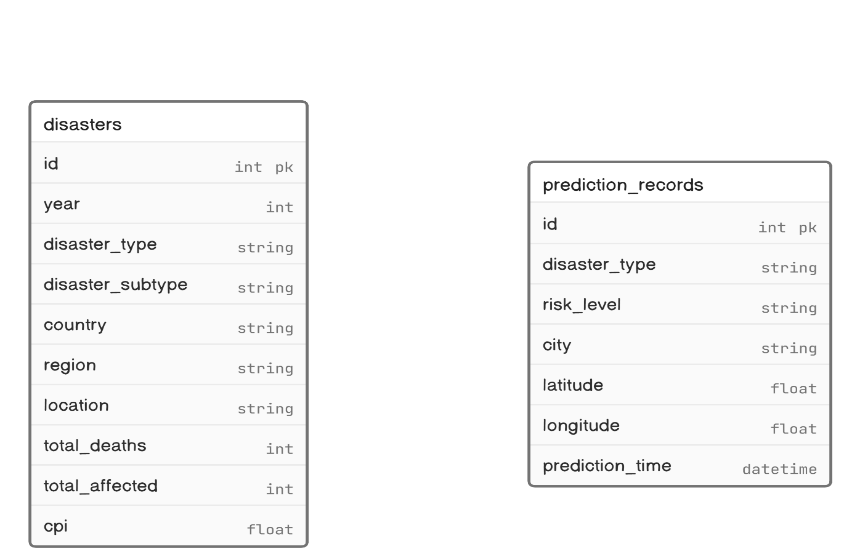


Figure 4.11 – ERD Diagrams

Table 4.2 shows the disaster table in more detail.

Table 4.2 - Disaster table with details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraints** | **Introduction** |
| id | AutoField | N/A | Primary Key | Unique ID for every disaster record. |
| year | INT | NA | Not Null | The year when the disaster occurred. |
| disaster\_type | Char | 50 | Not Null | Type of disaster (e.g., Drought, Avalanche, Landslide). |
| disaster\_subtype | Char | 50 | Nullable | Sub-type or category |
| country | Char | 50 | Not Null | The country where the disaster occurred. |
| region | Char | 100 | Not Null | Broader region classification (e.g., Central Asia, Eastern Asia). |
| location | Char | 100 | Not Null | Specific location description (e.g., city). |
| total\_deaths | INT | N/A | Nullable | Total number of deaths reported |
| total\_affected | INT | N/A | Nullable | Total number of people affected |
| cpi | Float | N/A | Nullable | Community Preparedness Index |

Table 4.3 shows the PredictionRecord table in more detail.

Table 4.3 - PredictionRecord table with details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraints** | **Introduction** |
| id | AutoField | N/A | Primary Key | Unique ID for each prediction record. |
| disaster\_type | Char | 50 | Not Null | Type of disaster being predicted |
| risk\_level | Char | 50 | Not Null | Predicted risk severity level. |
| city | Char | 50 | Not Null | City name for prediction. |
| latitude | Float | N/A | Not Null | Latitude coordinate for the location. |
| longitude | Float | N/A | Not Null | Longitude coordinate for the location. |
| prediction\_time | DateTime | N/A | Not Null | Timestamp when the prediction was made. |

## **4.6 Security and Performance Considerations**

Our website system provides security and optimization of performance in various ways. We utilize Django's built-in CSRF protection to help avoid malicious exploitation of user sessions, ensuring that the origination of legitimate user requests. Additionally, server-side data validations are employed to ensure data integrity. Additionally, we utilize caching mechanisms to improve the performance of our system. We cache prediction data to cut down on database queries. This approach is efficient in enhancing response time when performing visualizations. Also, our application has a strong mechanism of error-handling. When an error occurs, the user interface displays a proper message. Combined, these steps produce a secure and responsive application.

## **4.7 Technical specification of the project**

## ***4.7.1 Functional requirements***

FR1: City Weather Fetch

Description: When we enter a city, the system should get real-time weather data (temperature, humidity, wind speed, etc.) for it.

Acceptance Criteria: When a user enters a city's name, it fetches the weather data within 2 seconds.

FR2: Disaster Type Selection

Description: The user should have the option to select from three disasters.

Acceptance Criteria: The drop-down menu shows all three disasters.

FR3: Dynamic Form Rendering

Description: The system should dynamically display the environmental features upon selecting a disaster.

Acceptance Criteria: The correct form appears with appropriate fields.

FR4: Risk Prediction

Description: The system should use the pretrained models to make predictions.

Acceptance Criteria: The system returns the prediction with a safety tip within 3 seconds.

FR5: Data Logging

Description: The system should save the prediction data, such as city, coordinates, disaster type, and risk level, in the database.

Acceptance Criteria: Correct entries appear in the admin panel.

FR6: Visualization Dashboards

Description: The dashboard should display multiple data visualizations

Acceptance Criteria: All the maps and charts render without any error.

FR7: Responsive Interface

Description: The system should support both desktop and mobile versions.

Acceptance Criteria: Elements resizing on devices with a width < 768px.

## ***4.7.2 Nonfunctional requirements***

NFR1: Performance

Description: The application should be responsive and return the output within an acceptable latency.

Acceptance Criteria: API response time < 3s, ML prediction time < 2s.

NFR2: Scalability

Description: The system should be able to support more disasters in the future.

Acceptance Criteria: New models can be added without much change.

NFR3: Usability

Description: The system should be user-friendly.

Acceptance Criteria: Testing shows more than 90% user satisfaction.

NFR4: Reliability

Description: The system should be up 99% of the time.

Acceptance Criteria: Downtime results in alerts and cached content.

NFR5: Cross-Browser Compatibility

Description: The system should work identically on all types of browsers.

Acceptance Criteria: Identical layout and performance on Chrome, Firefox, and Safari.

NFR6: Accessibility

Description: Mild visual impairments should be able to use the system.

Acceptance Criteria: Font sizes >14px, color contrast ratio >4.5:1, keyboard navigability ensured.

NFR10: Responsiveness to Invalid Input

Description: Invalid or incomplete forms should not be processed further.

Acceptance Criteria: Warnings and errors on missing fields.

# **CHAPTER 5. TESTING AND VALIDATION**

We have performed rigorous testing to ensure reliability, accuracy, and usability. We conducted functional testing to validate the performance of all user interface components. This included dynamic form rendering and min/max constraints. Furthermore, a backend test was carried out to check the loading mechanism of our saved models. We also tested the database model for logging all the predicted instances. Additionally, we chose several geographical locations to check the functionality of the OpenWeatherMap API. We also checked the system’s ability to handle any incorrect city. We also reviewed clarity, responsiveness, and visual aspects of maps and charts as part of visual validation. In addition, cross-platform testing was performed to check the compatibility of our system on different browsers (Chrome, Firefox, Edge) and devices (laptop, tablet, and mobile). Lastly, we evaluated our application to handle various cases, such as empty forms and non-numeric values.

## **5.1 Unit Testing and Integration Testing**

Initially, we tested every component individually using Django’s built-in testing framework. We also tested several other functionalities, like data fetching from the API and JSON responses. Some unit testing includes testing for invalid city input, API errors, empty responses, missing form values, and out-of-range numeric values. After unit testing, we tested the functionality of interconnected modules. For instance, we tested the whole process of entering the city’s name to the rendering of the map on the dashboard page. One more such example is from entering the values in different fields to the database logging. If we talk about the results, all the individual and connected modules passed their tests, demonstrating a comprehensive and working web application. Table 5.1 shows the functional test cases in detail.

Table 5.1 - Test Case Table (for Functional Testing)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S No.** | **Test Scenario** | **Test Steps** | **Expected Result** | **Actual Result** | **Test Browser** | **Status** |
| 1 | Weather data fetching | Enter a valid city name and fetch data | Weather data returned (temp, humidity, etc.) | Weather data was correctly fetched | Chrome | Passed |
| 2 | Disaster type field loading | Select disaster type | Form fields appear dynamically | Correct fields loaded | Chrome | Passed |
| 3 | Prediction API | Submit the form with valid inputs | The risk level is predicted and shown | Risk level received | Chrome | Passed |
| 4 | Prediction storage | Check the database after the database | PredictionRecord created with correct fields | Correct database entry | Chrome | Passed |
| 5 | Dashboard visualization | Load the dashboard page | Load the dashboard page | All visualizations visible | Chrome | Passed |

## **5.2 Usability Testing Results**

We conducted the usability test to get feedback on the user-friendliness of our system. Our participants were students and faculty members from UCA. Participants were asked to navigate our web app and predict different disaster types. Later, we asked them several questions about the visual aspect of our system. Our findings are as follows: 100% of our participants were able to make successful predictions of all the disasters. 95% of our participants found our website easy to use and accessible. 90% of our participants commended the responsiveness and user-friendliness of our website. Additionally, 80% found the visualization self-explanatory and easy to understand. Furthermore, 70% of our participants also appreciated the various emojis on our website. Overall, we received positive feedback regarding the clarity, usability, and visual appeal from our participants.

## **5.3 Scalability and Response Time Analysis**

Scalability is an important requirement when working with disaster prediction systems. We can deploy the Django backend on any cloud environment and then scale it further accordingly. We have the pretrained model working in the backend, which we can add more to scale up our project. Furthermore, these models are lightweight and allow fast predictions. Additionally, most of the visualizations are powered by Plotly, which reduces the server-side load.

We are utilizing an external service API to fetch weather data. This may hinder the overall performance as the external API may suffer from network and availability issues. Apart from that, the prediction cycle mostly utilizes lightweight and pretrained models, which allows quick loading and execution.

Overall, execution time greatly depends on the network conditions, server load, and the client’s device. The system architecture relies less on server-side execution to reduce the latency. Although we have robust and lightweight models, the system may face delays for other reasons.

## **5.4 Validation and Output Verification**

We also validated the prediction outputs via thorough testing. The validation process starts with the system checking for a realistic range of climate data. This step is crucial in eliminating data that may cause erroneous predictions. For instance, we don’t want to feed temperature data that is not real. In the case of the model validation, we tested the predicted outcomes of our model against the known outcomes. This step ensures that our model produces consistent results when put into production. Moreover, it can help to keep track of the model’s performance over the years. When a prediction is made, it is usually accompanied by an associated safety tip message. This verification process checks the relevance of the message to the disaster. For instance, when the predicted outcome is high avalanche risk, it should display ‘avoid hilly areas in case of snow’ or any similar message.

# **CHAPTER 6. RESULTS AND LIMITATIONS**

## **6.1 Results**

Our disaster prediction system incorporates machine learning models with a Django web app. It successfully makes real-time predictions and visualizations of droughts, avalanches, and landslides. The pre-trained models serve as the core of our project. The implementation demonstrates strong performance, smooth integration, and potential scalability. We will go through a few of the key outcomes of this project.

## **6.1.1 End-to-End System Validation**

All the components are working as expected after the deployment of the project. Let’s go through the whole cycle to have an overview of what users can do. Users can enter the city name and get weather data for it. They can also switch between the three disaster types and submit the form for any chosen disaster. This part also offers a range of realistic values, thus helping them avoid any unrealistic values. Moreover, our system processes the input form values and returns the accurate risk level prediction to the user. Additionally, the users also receive safety tips alongside the predicted risk. Lastly, users can check the visualization section for disaster trends, recent predictions, and other maps for detailed statistics.

## **6.1.2 User Experience and Interface Responsiveness**

Our system has a highly responsive site capable of holding several requests simultaneously. The AJAX-based prediction avoids the need to reload the page, thus creating a seamless user experience. Furthermore, real-time feedback while interacting with the disaster form makes it easier for the users, especially non-technical users, to use. Additionally, emojis and appealing color design further enhance the user interface.

## **6.1.3 Visual Data Insights**

We deal with two kinds of data in this project. The first is generated due to predictions, and the other is the historical data stored in the backend. The system leverages this historical data to create modern maps and charts to showcase trends and disaster hotspots. We can check out the charts and graphs to get insights about the past disasters.

## **6.2 Limitations**

Although our system works fine with the existing functionalities, we need to consider the limitations of our project, too. These limitations may constrain our project in terms of scalability and broader reach.

## **6.2.1 Static ML Models and Lack of Real-time Learning**

Our system works on pre-trained models, so no feedback loop for the models to adapt to changing climate patterns. If we want to update our models, we manually train them. The whole process of training the models again may interrupt the availability of our system. Our models are trained on historical data, and climate change may alter the weather patterns. So, we might encounter situations different from the past hazards, and our models may lose their prediction accuracy.

## **6.2.2 Geographic and Data Generalizability**

Although our system supports input from any location on the planet, the data was gathered from a specific region. The distinctive weather and topographical conditions may vary from our desired location. As a result, our model may produce inconsistent results and lose its reliability. Additionally, even if the climate and topography are similar, the presence of vegetation impacts the occurrence of hazards like avalanches and landslides. So, it significantly varies among regions, and our model doesn’t capture such features.

## **6.2.3 Satellite and Sensor Data Integration**

Our model currently works only with meteorological and topographical parameters. Satellite data is an emerging trend in hazard forecasting, and our system doesn’t incorporate it. For instance, in the case of landslides and drought predictions, we can use the vegetation index feature for more comprehensive forecasting. Additionally, our system doesn’t rely on ground sensors. Ground sensors are crucial in accurately providing measurements such as soil moisture and precipitation amount. Additionally, our models are trained on simple meteorological features and lack complex features such as atmospheric pressure gradients. These complex features are highly relevant when forecasting an extreme event.

## **6.2.4 Prediction Interpretability**

Our model makes a prediction, but users remain unaware of the underlying process. In other words, it may appear like a black box and lead users to mistrust the application. For instance, if the output for avalanche is “high”, we don’t show how a decision was made and which feature led to the specific output. The current system doesn’t provide SHAP (SHapley Additive exPlanations), which is important for understanding the risk assessment process.

## **6.2.5 Mobile Optimization**

Although our disaster prediction app is mobile responsive, it works best with a desktop or tablet. Some maps, such as a choropleth map, are powered by Plotly and may not perform optimally on mobile devices. Additionally, the comprehensive form is ideally suited for large screens, and users may face usability issues. Furthermore, our system doesn’t offer offline functionality, which would be ideal for communities living in remote and disaster-prone areas. Another major challenge is the compatibility with mobile browsers. Some browsers may not render the Plotly maps and graphs due to processing constraints on mobile devices.

# **CHAPTER 7. CONCLUSION AND FUTURE WORK**

## **7.1 Future Work**

The current system works fine with our pre-trained models, but we can make several improvements to improve the reliability and capacity of the project. Additionally, more research needs to be done to implement disaster forecasting models in Central Asia.

## **7.1.1 Real-Time Model Retraining and Self-Learning**

Our disaster models are static and don’t update with the ingestion of new climate data. We need a pipeline that could support the retraining of models over time and with new data inflow. This may include training models on the cloud. This strategy may help our disaster models adapt to changing climate conditions and improve accuracy.

## **7.1.2 Integration of Remote Sensing and IoT Data**

Satellite and sensor data are a great source for improving the accuracy of our disaster forecasting model. Future work may include using ground sensors to detect soil moisture and other climate variables. We can also have sensors measuring the humidity and real-time slope.

## **7.1.3 Disaster Onset Prediction and Severity Estimation**

Our current models are limited to predicting drought, avalanche, and landslide risk levels. We can extend the project to incorporate a stream of time series data. Additionally, we can also train a model to predict the severity and magnitude of the hazard. Such a system would help the emergency responders and government agencies to prepare in advance.

## **7.1.4 Multi-Language and Accessibility Features**

Central Asia is a diverse region with linguistically and demographically diverse populations. So, our system should support multiple language options to make it convenient for common users. The multilingual capability would bring more inclusivity and make it accessible to all linguistically diverse groups in the region.

## **7.1.5 Integration with Disaster Management Agencies**

The system should be integrated into disaster management agencies, so when a risk assessment is made, the emergency responders should act on time to disseminate the information and take vital steps to reduce the impact. Additionally, it would assist agencies in allocating resources and planning their course of action accordingly.

## **7.2 Conclusion**

In conclusion, climate change is inevitable, and you may have witnessed the threats it has already posed to humans. For instance, you may have heard or observed the rise of natural disasters such as landslides and avalanches. These disasters need to be predicted in advance to minimize the damage. Regions like Central Asia are prone to landslides, avalanches, and droughts, and require a robust disaster prediction system. This project proposes a Django-based web application integrated with machine learning models to predict these extreme weather events. The key contribution it makes is the use of a machine learning approach for multi-event forecasting. The models are trained on datasets from different sources and then integrated into a Django application for a user-friendly dashboard. Some of the key features include real-time weather integration, user-friendly forms, robust and scalable backend, and comprehensive visualizations. We completed all the key requirements and rigorously tested the platform for functionality issues.

Although the system works well with meteorological and topological data, it has some limitations. Some key limitations include the reliance on static models and a limited geographical focus. These limitations are great options for us to scale and improve our project. Overall, the project performs well with real-time weather data and is proof of an intelligent disaster forecasting system. This project not only serves as a disaster forecasting system but also contributes to the body of research. It also finds its application in environmental science and public safety, particularly in regions susceptible to natural hazards.

# **References**

1. Ebi, K. L., Vanos, J., Baldwin, J. W., Bell, J. E., Hondula, D. M., Errett, N. A., Hayes, K., Reid, C. E., Saha, S., Spector, J., & Berry, P. (2021). Extreme Weather and Climate Change: population health and health system implications. *Annual Review of Public Health*, *42*(1), 293–315. <https://doi.org/10.1146/annurev-publhealth-012420-105026>
2. Furtak, K., & Wolińska, A. (2023). The impact of extreme weather events as a consequence of climate change on the soil moisture and on the quality of the soil environment and agriculture – A review. *CATENA*.
3. Burunciuc, L. (2020, November 5). *Natural disasters cost Central Asia $10 billion a year – Are we doing enough to prevent them?* Retrieved from worldbank.org: <https://blogs.worldbank.org/en/europeandcentralasia/natural-disasters-cost-central-asia-10-billion-year-are-we-doing-enough>
4. Thurman, M. (2011). *Natural Disaster Risks in Central Asia: A Synthesis.* UNDP/BCPR, Regional Disaster Risk Reduction Advisor, Europe and CIS.
5. Zhang, H., Liu, Y., Zhang, C., & Li, N. (2025). Machine Learning Methods for Weather Forecasting: A survey. *Atmosphere*, *16*(1), 82. <https://doi.org/10.3390/atmos16010082>
6. Chen, L., Zhong, X., Zhang, F., Cheng, Y., Xu, Y., Qi, Y., & Li, H. (2023). FuXi: a cascade machine learning forecasting system for 15-day global weather forecast. *Npj Climate and Atmospheric Science*, *6*(1). <https://doi.org/10.1038/s41612-023-00512-1>
7. Bochenek, B., & Ustrnul, Z. (2022). Machine Learning in Weather Prediction and Climate Analyses—Applications and Perspectives. *Atmosphere*, *13*(2), 180. <https://doi.org/10.3390/atmos13020180>
8. Camps-Valls, G., Fernández-Torres, M., Cohrs, K., Höhl, A., Castelletti, A., Pacal, A., Robin, C., Martinuzzi, F., Papoutsis, I., Prapas, I., Pérez-Aracil, J., Weigel, K., Gonzalez-Calabuig, M., Reichstein, M., Rabel, M., Giuliani, M., Mahecha, M. D., Popescu, O., Pellicer-Valero, O. J., . . . Williams, T. (2025). Artificial intelligence for modeling and understanding extreme weather and climate events. *Nature Communications*, *16*(1). <https://doi.org/10.1038/s41467-025-56573-8>
9. Nandgude, N., Singh, T. P., Nandgude, S., & Tiwari, M. (2023). Drought prediction: A comprehensive review of different drought prediction models and adopted technologies. *Sustainability*, *15*(15), 11684. <https://doi.org/10.3390/su151511684>
10. Chen, R., Zhang, W., & Wang, X. (2020). Machine Learning in Tropical Cyclone Forecast Modeling: A review. *Atmosphere*, *11*(7), 676. <https://doi.org/10.3390/atmos11070676>
11. Akosah, S., Gratchev, I., Kim, D., & Ohn, S. (2024). Application of Artificial intelligence and remote sensing for landslide detection and prediction: Systematic review. *Remote Sensing*, *16*(16), 2947. <https://doi.org/10.3390/rs16162947>
12. Mosavi, A., Ozturk, P., & Chau, K. (2018). Flood Prediction Using Machine Learning Models: Literature review. *Water*, *10*(11), 1536. <https://doi.org/10.3390/w10111536>
13. Sadrtdinova, R., Perez, G. a. C., & Solomatine, D. P. (2024b). Improved drought forecasting in Kazakhstan using machine and deep learning: a non-contiguous drought analysis approach. *Hydrology Research*, *55*(2), 237–261. <https://doi.org/10.2166/nh.2024.154>
14. Xu, L., Zhang, X., Wu, T., Yu, H., Du, W., Zhang, C., & Chen, N. (2024). Global prediction of flash drought using machine Learning. *Geophysical Research Letters*, *51*(21). <https://doi.org/10.1029/2024gl111134>
15. Rahmati, O., Ghorbanzadeh, O., Teimurian, T., Mohammadi, F., Tiefenbacher, J. P., Falah, F., Pirasteh, S., Ngo, P. T., & Bui, D. T. (2019). Spatial modeling of snow avalanche using machine learning models and Geo-Environmental factors: comparison of effectiveness in two mountain regions. *Remote Sensing*, *11*(24), 2995. <https://doi.org/10.3390/rs11242995>
16. Wen, H., Wu, X., Liao, X., Wang, D., Huang, K., & Wünnemann, B. (2022). Application of machine learning methods for snow avalanche susceptibility mapping in the Parlung Tsangpo catchment, southeastern Qinghai-Tibet Plateau. *Cold Regions Science and Technology*, *198*, 103535. <https://doi.org/10.1016/j.coldregions.2022.103535>
17. Bian, R., Huang, K., Liao, X., Ling, S., Wen, H., & Wu, X. (2022). Snow avalanche susceptibility assessment based on ensemble machine learning model in the central Shaluli Mountain. *Frontiers in Earth Science*, *10*. <https://doi.org/10.3389/feart.2022.880711>
18. Choubin, B., Borji, M., Hosseini, F. S., Mosavi, A., & Dineva, A. A. (2020). Mass wasting susceptibility assessment of snow avalanches using machine learning models. *Scientific Reports*, *10*(1). <https://doi.org/10.1038/s41598-020-75476-w>
19. Rosi, A., Frodella, W., Nocentini, N., Caleca, F., Havenith, H. B., Strom, A., Saidov, M., Bimurzaev, G. A., & Tofani, V. (2023). Comprehensive landslide susceptibility map of Central Asia. *Natural Hazards and Earth System Sciences*, *23*(6), 2229–2250. <https://doi.org/10.5194/nhess-23-2229-2023>
20. Chen, X., Wang, Y., Wang, X., Li, Y., Qi, J., & Lin, Q. (2024). Risk Assessment of Landslide Casualty under Incomplete Information——Tienshan and Kunlun Mountainous Regions of Central Asia. *International Journal of Disaster Risk Reduction*, 105057. <https://doi.org/10.1016/j.ijdrr.2024.105057>
21. Oyounalsoud, M. S., Yilmaz, A. G., Abdallah, M., & Abdeljaber, A. (2024). Drought prediction using artificial intelligence models based on climate data and soil moisture. *Scientific Reports*, *14*(1). <https://doi.org/10.1038/s41598-024-70406-6>
22. Pyarali, K., Peng, J., Disse, M., & Tuo, Y. (2022). Development and application of high resolution SPEI drought dataset for Central Asia. *Scientific Data*, *9*(1). <https://doi.org/10.1038/s41597-022-01279-5>
23. Maharana, K., Mondal, S., & Nemade, B. (2022). A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, *3*(1), 91–99. <https://doi.org/10.1016/j.gltp.2022.04.020>
24. Matharaarachchi, S., Domaratzki, M., & Muthukumarana, S. (2024). Enhancing SMOTE for imbalanced data with abnormal minority instances. *Machine Learning With Applications*, 100597. <https://doi.org/10.1016/j.mlwa.2024.100597>
25. Salman, H. A., Kalakech, A., & Steiti, A. (2024). Random Forest algorithm Overview. *Deleted Journal*, *2024*, 69–79. <https://doi.org/10.58496/bjml/2024/007>
26. Schober, P., & Vetter, T. R. (2021). Logistic regression in medical research. *Anesthesia & Analgesia*, *132*(2), 365–366. <https://doi.org/10.1213/ane.0000000000005247>
27. Khoshkroodi, A., Sani, H. P., & Aajami, M. (2024). Stacking Ensemble-Based Machine Learning model for predicting deterioration components of steel W-Section beams. *Buildings*, *14*(1), 240. <https://doi.org/10.3390/buildings14010240>
28. Hancock, J. T., & Khoshgoftaar, T. M. (2020). CatBoost for big data: an interdisciplinary review. *Journal of Big Data*, *7*(1). <https://doi.org/10.1186/s40537-020-00369-8>
29. Ali, Z. A., Abduljabbar, Z. H., Tahir, H. A., Sallow, A. B., & Almufti, S. M. (2023). eXtreme Gradient Boosting Algorithm with Machine Learning: a Review. *Academic Journal of Nawroz University*, *12*(2), 320–334. <https://doi.org/10.25007/ajnu.v12n2a1612>
30. Omonije, A. (2024). Agile Methodology: A comprehensive impact on modern business operations. *International Journal of Science and Research (IJSR)*, *13*(2), 132–138. <https://doi.org/10.21275/sr24130104148>
31. <https://github.com/tariqaxix/Final_Year_Project>