QUAKESIGHT: AN INTEGRATED DESCRIPTIVE-TO-PRESCRIPTIVE ANALYTICS FRAMEWORK FOR PROACTIVE EARTHQUAKE DISASTER MANAGEMENT

A MAJOR PROJECT REPORT

Submitted by

MOHAMMAD AMANULLAH **(Regd.No.21891A6737)**
SRIPADA PALLAVI **(Regd.No.21891A6755)**
G. SIDDARTHA **(Regd.No.21891A6718)**

Under the guidance of

Mr. M. SIVA

Assistant Professor-CSE(DS)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE
Near Ramoji Film City, Deshmukhi Village, Pochampally Mandal, Yadadri Bhuvanagiri Dist.
(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)
AN AUTONOMOUS INSTITUTION

Eamcet Code VGNT Dist Code : YBG

MAY, 2025

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE
Near Ramoji Film City, Deshmukhi Village, Pochampally Mandal, Yadadri Bhuvanagiri Dist.
(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)
AN AUTONOMOUS INSTITUTION
Eamcet Code VGNT Dist Code : YBG

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(DATA SCIENCE)

VISION

To develop Data Science Professionals through creative and innovative approaches to address the present and future challenges of the modern computing world

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE
Near Ramoji Film City, Deshmukhi Village, Pochampally Mandal, Yadadri Bhuvanagiri Dist.
(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)
AN AUTONOMOUS INSTITUTION
Eamcet Code VGNT Dist Code : YBG

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(DATA SCIENCE)

MISSION

Educate students by expanding their knowledge in cutting – edge technologies to acquire professional ethics. Impact quality education to build research & entrepreneurial eco system using niche technologies

PROGRAMME EDUCATIONAL OBJECTIVES (PEO'S)

PEO-1: Emerge as engineers, innovators, entrepreneurs with social awareness and ethical values.

PEO-2: Work in teams in multidisciplinary areas addressing the needs of society.

PEO-3: Inculcate self-learning and lifelong learning adapting cutting edge technologies

PROGRAM SPECIFIC OUTCOMES (PSO'S)

PSO-1: Design. Implement and test application software in the field of data science

PSO-2: Understand the architecture and organization of computer systems to develop data science tools.

PSO-3: To use specialized softwares to carry out statistical data analysis

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE
Near Ramoji Film City, Deshmukhi Village, Pochampally Mandal, Yadadri Bhuvanagiri Dist.
(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)
AN AUTONOMOUS INSTITUTION
Eamcet Code VGNT Dist Code : YBG

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(DATASCIENCE)

CERTIFICATE

This is to certify that the project work titled — "**QUAKESIGHT: AN INTEGRATED DESCRIPTIVE-TO-PRESCRIPTIVE ANALYTICS FRAMEWORK FOR PROACTIVE EARTHQUAKE DISASTER MANAGEMENT"** submitted by Mr. Md. Amanullah (Regd.No.21891A6737) , Ms. Sripada Pallavi (Regd.No.21891A6755), Mr. G. Siddartha (Regd.No.21891A6718), in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering (Data Science)** at the Vignan Institute of Technology And Science, Deshmukhi is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this project report have not been submitted in any university for the award of any degree and the results are achieved satisfactorily.

Project Guide
Mr. M. Siva
Assistant Professor

Head of the Department
Mrs. P. Lavanya Kumari
Associate Professor

External Examiner

DECLARATION

We hereby declare that project entitled – **"QUAKESIGHT: AN INTEGRATED DESCRIPTIVE-TO-PRESCRIPTIVE ANALYTICS FRAMEWORK FOR PROACTIVE EARTHQUAKE DISASTER MANAGEMENT"** is bonafide work duly completed by us. It does not contain any part of the project or thesis submitted by any othercandidate to this or any other institute of the university.

All such materials that have been obtained from other sources have been duly acknowledged.

Md. AMANULLAH
(Regd.No.21891A6737)

SRIPADA PALLAVI
(Regd.No.21891A6755)

G. SIDDARTHA
(Regd.No.21891A6718)

ACKNOWLEDGEMENT

Every project big or small is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. We sincerely appreciate the inspiration, support and guidance of all those people who have been instrumentalin making this project a success.

We thank our beloved **Chairman, Dr. L. Rathaiah Sir**, who gave us great encouragement to work.

We thank our beloved **CEO**, **Sri. Boyapati Shravan Sir**, we remember him for his valuable ideas and facilities available in college during the development of the project.

We convey our sincere thanks to **Dr. G. Durga Sukumar Sir, Principal** of our institution for providing us with the required infrastructure and a very vibrant and supportive staff.

We would like to thank our Head of the Department of Computer Science and Engineering (Data Science), **Mrs. P. Lavanya Kumari Madam**,a distinguished and eminent personality, whose strong recommendation, immense support and constant encouragement has been great help to us. We intensely thank her for the same.

We would like to express our sincere appreciations to our project coordinator **Mrs. Dr. K. Lakshmi Anusha Madam,** for her guidance, continuous encouragement and support during the project.

We would like to thank our guide of the project, **Mr. M. Siva Sir, Assistant professor,** who has invested his effort in guiding the team in achieving the goal.

Special thanks go to my team mates, who helped me to assemble the parts and gave suggestions in making this project. We have to appreciate the guidance given by other supervisoras well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advice's. We take this opportunity to thank all our lecturers who have directly or indirectly helped our project. We pay our respects and love to our parents and all other family members and friends for their love and encouragement throughout our career.

Md. AMANULLAH
(Regd.No.21891A6737)

SRIPADA PALLAVI
(Regd.No.21891A6755)

G. SIDDARTHA
(Regd.No.21891A6718)

ABSTRACT

QuakeSight is a comprehensive and integrated analytics framework specifically engineered to revolutionize earthquake disaster management by seamlessly uniting the four pillars of data analytics: **Descriptive**, **Diagnostic**, **Predictive**, and **Prescriptive** analytics. At its foundation, the Descriptive component conducts a meticulous examination of historical seismic activity, uncovering temporal and spatial trends, frequency distributions, and intensity patterns that lay the groundwork for deeper analytical insight. Progressing to the Diagnostic phase, QuakeSight leverages advanced statistical and spatial correlation techniques to identify critical risk drivers—such as soil type, infrastructure vulnerability, and population density—that exacerbate earthquake impact severity.Building on this enriched understanding, the Predictive analytics layer integrates robust machine learning algorithms, including ensemble methods and time-series forecasting models, to accurately anticipate future seismic events. These models not only estimate the likelihood of occurrence but also predict impact zones, estimated magnitudes, and affected population clusters, enabling forward-looking risk assessments. The framework culminates in its Prescriptive component, which operationalizes insights from previous layers into tangible, strategic recommendations. These may include optimized allocation of emergency resources, prioritization of infrastructural reinforcements, and tailored evacuation plans, all aimed at minimizing casualties, property damage, and response time.By holistically combining these analytical dimensions into a unified decision-support system, QuakeSight empowers policymakers, urban planners, emergency responders, and other key stakeholders with a powerful toolset for real-time, evidence-based decision-making. This integration not only enhances situational awareness and disaster readiness but also fosters long-term resilience, community safety, and sustainability in seismically active regions. QuakeSight, therefore, stands as a forward-thinking, data-driven solution that transforms earthquake preparedness from a reactive to a proactive paradigm.

Index Terms: Earthquake Risk Assessment, Disaster Resilience, Descriptive Analytics, Diagnostic Analytics, Predictive Analytics, Prescriptive Analytics, Seismic Intelligence, Machine Learning, Emergency Management, Decision Support System.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Earthquakes remain one of the most devastating natural hazards, causing significant loss of human lives, widespread destruction to infrastructure, and substantial economic disruption globally. As seismic activities continue to affect vulnerable regions, the complexities of earthquake risk and disaster response demand more sophisticated and forward-looking management approaches. Traditional systems often focus on reactive measures that only mobilize resources after an event occurs, limiting the effectiveness of disaster mitigation and recovery efforts. The motivation behind developing QuakeSight arises from the urgent need to transform earthquake disaster management into a more proactive, data- driven process. By integrating multiple layers of analytics — descriptive, diagnostic, predictive, and prescriptive — QuakeSight aims to provide a comprehensive understanding of seismic risks, underlying causal factors, and effective response strategies before disasters strike. This shift from reactive to anticipatory action can significantly reduce casualties and economic losses by enabling timely preparedness and optimized resource allocation. The framework's multi-layered analytics approach addresses key challenges such as identifying patterns in historical seismic data, diagnosing root causes and correlations, forecasting potential future events using machine learning models, and generating actionable recommendations through prescriptive modeling. This holistic methodology facilitates better decision support for disaster management professionals, policymakers, and communities alike, equipping them with the tools necessary to strengthen resilience and improve earthquake readiness. Positioned within this context, QuakeSight represents a crucial advancement for earthquake risk assessment and disaster preparedness — advancing beyond existing fragmented solutions to an integrated system capable of enabling proactive, informed, and effective earthquake management.

1.1 PURPOSE OF THE SYSTEM

The primary purpose of QuakeSight is to serve as a comprehensive analytics framework that integrates four essential analytical dimensions — Descriptive, Diagnostic, Predictive, and Prescriptive — to support proactive earthquake disaster management. This unified system is designed to analyze vast amounts of seismic data, uncovering historical patterns and seismic activity trends to provide a clear descriptive foundation.

Moving beyond description, QuakeSight identifies critical risk factors through diagnostic analytics, examining correlations and causal relationships that influence earthquake impacts. The system then leverages machine learning and advanced statistical techniques to predict potential seismic events and their probable impact zones, empowering stakeholders with forward-looking insights.

Finally, QuakeSight transforms these insights into actionable strategies by generating prescriptive recommendations, including optimized allocation of emergency resources, efficient evacuation planning, and improved response coordination. This decision-support capability aims to minimize human casualties, infrastructure damage, and economic losses by enabling timely and informed disaster preparedness and response. Through its integrated analytical approach, QuakeSight seeks to bridge the gap between data analysis and practical disaster management actions, enhancing the resilience and safety of vulnerable communities.

1.2 SCOPE

The QuakeSight system is designed to analyze and interpret both historical and real-time seismic data, encompassing earthquake records, fault line activity, ground motion metrics, and sensor-generated monitoring inputs. Its geographic applicability primarily targets earthquake-prone regions worldwide, with flexible configurations to address local or regional scales depending on data availability and user needs.

The framework targets a diverse set of users such as emergency planners, government agencies, response teams, urban planners, infrastructure managers, and community stakeholders who require precise, data-driven insights to improve earthquake resilience. QuakeSight's decision-support platform facilitates coordinated efforts across these user groups, enhancing situational awareness and collaborative disaster management.

2. LITERATURE SURVEY

The scientific exploration of earthquakes began well before the advent of modern computational tools. Early studies in the 20th century focused largely on physical observations and empirical relationships to understand seismic behavior. For example, Richter (1935) introduced the magnitude scale, which revolutionized the measurement of earthquake energy [1]. Gutenberg and Richter (1944) further refined this with their work on energy release [2]. These early works laid a foundational understanding of earthquake parameters but were purely observational.Later, Omori (1894) observed that aftershock frequency decays roughly over time following a logarithmic pattern, which came to be known as Omori's Law [3]. Similarly, Reid's Elastic Rebound Theory (1910) [4] provided insight into how strain accumulation and sudden release lead to seismic events, forming the basis for earthquake mechanics.During the mid-20th century, the use of statistical modeling increased. Utsu (1961) studied earthquake frequency-magnitude distributions, contributing to what would become the Gutenberg-Richter relationship [5]. Allen (1978) analyzed the spatial patterns of seismicity, linking active faults with ground motion data [6]. Kanamori (1977) provided valuable insight into seismic moment and energy release which later formed the basis for moment magnitude scales [7].In the 1980s and 1990s, research shifted towards integrating field data with computational simulations. Bolt (1988) introduced early frameworks for seismic hazard maps [8]. Cornell (1968) proposed probabilistic seismic hazard analysis (PSHA), a significant step in quantifying risk using probability theory [9]. However, these models still lacked automation and required manual interpretation.A few notable studies explored site response without modern AI: Seed and Idriss (1982) examined local soil conditions in earthquake amplification [10]; Boore et al. (1997) focused on ground motion prediction equations [11]. These investigations paved the way for modern site-specific seismic risk assessments. Other efforts included the development of attenuation relationships (Joyner and Boore, 1981) [12], which estimate ground motion intensity based on distance and magnitude. Wald et al. (1999) created ShakeMaps to visualize intensity patterns using empirical models [13]. While these tools improved visualization, they still required expert interpretation and were not predictive.Despite the value of these early methods, none incorporated machine learning, real-time data feeds, or prescriptive analytics. They provided crucial pieces of the puzzle but fell short of offering dynamic, actionable disaster strategies.The evolution from basic descriptive analysis to data-driven analytics underscores the gap this paper seeks to fill. Our approach builds upon these foundational studies by integrating all four stages of analytics—descriptive, diagnostic, predictive, and prescriptive.

2.1 DESCRIPTIVE ANALYTICS IN SEISMIC DATA ANALYSIS

Descriptive analytics involves summarizing historical seismic data to identify key patterns, trends, and anomalies. Classical seismic analysis relies heavily on statistical methods such as frequency-magnitude distribution (e.g., Gutenberg-Richter law), spatial clustering, and temporal trend analysis to characterize earthquake occurrences. For instance, Utsu (1999) and subsequent works established standard techniques for summarizing seismic catalogs to understand earthquake recurrence intervals and magnitudes. More recent advances apply data visualization and exploratory data analysis (EDA) tools to enhance interpretability of seismic datasets. Several systems also incorporate geographic information systems (GIS) to spatially contextualize seismic activity with geological features (e.g., fault lines, tectonic boundaries). However, descriptive analyses often remain fragmented and focused on retrospective reporting without deeper investigation into underlying causal mechanisms or forecasting. They provide foundational knowledge yet lack decision-support integration essential for proactive disaster management.

2.2 DIAGNOSTIC ANALYTICS: CORRELATION AND CAUSAL MAPPING

Diagnostic analytics investigates relationships within seismic data and between environmental or infrastructural variables to identify risk factors and causal influences. Correlation mapping, factor analysis, and cluster detection techniques have been used to associate earthquake damage levels with factors such as soil composition, building types, population density, and proximity to faults. For example, studies like Rodriguez et al. (2017) applied multivariate statistical models to pinpoint how urban development patterns exacerbate seismic vulnerabilities. Machine learning methods, including principal component analysis (PCA) and decision trees, have enhanced diagnostic capabilities by uncovering complex nonlinear associations. Nonetheless, many diagnostic systems operate in isolation, rarely linking findings systematically back to descriptive patterns or forward to predictive modeling seamlessly.

2.3 PREDICTIVE MODELING USING MACHINE LEARNING

Predictive analytics has gained prominence in earthquake research through the application of advanced machine learning techniques to forecast seismic events and anticipate their impacts. Models such as neural networks, support vector machines, and random forests have been trained on seismic signals, geophysical parameters, and historical event catalogs to predict earthquake occurrence probabilities, magnitudes, and aftershock patterns. For example, DeVries et al. (2018) demonstrated that deep learning models could identify precursory patterns in seismic waveforms indicative of impending earthquakes. Additionally, spatial-temporal predictive models utilize GIS-integrated algorithms to estimate shaking intensities and affected zones with higher resolution. While these approaches have shown promising accuracy improvements, challenges remain regarding data quality, model interpretability, and real-time applicability within disaster management workflows.

2.4 PRESCRIPTIVE ANALYTICS AND DECISION SUPPORT

Prescriptive analytics extends beyond prediction by recommending actionable strategies to optimize disaster response and resource allocation. In earthquake management, prescriptive frameworks aim to enhance evacuation routing, emergency shelter placement, logistics planning, and infrastructure reinforcement prioritization. Optimization techniques, such as linear programming, simulation models, and reinforcement learning, have been explored to address these challenges.

Studies like Kim and Lee (2020) applied reinforcement learning to dynamically adjust emergency resource deployment under uncertainty, demonstrating the potential of prescriptive analytics to improve operational efficiency. However, many prescriptive solutions are developed as standalone modules lacking seamless integration with predictive models and descriptive diagnostics, limiting their effectiveness for holistic decision-making.

2.5 INTEGRATION AND THE MOTIVATION FOR QUAKESIGHT

Despite significant advancements within individual analytics layers, existing earthquake disaster management systems largely operate in silos. Descriptive, diagnostic, predictive, and prescriptive analytics are often implemented as separate tools or isolated studies, leading to information gaps, inefficiencies, and incomplete decision-support capabilities. Integration complexities arise due to heterogeneous data formats, varying temporal and spatial resolutions, and diverse methodological paradigms.

Furthermore, many frameworks prioritize reactive response mechanisms instead of enabling proactive, anticipatory interventions necessary to mitigate disaster impacts effectively. The lack of a unified, end-to-end analytics platform hinders the translation of raw seismic data and model outputs into comprehensive, actionable insights for practitioners and policymakers.

QuakeSight addresses these gaps by developing an integrated analytics environment that cohesively combines descriptive pattern recognition, diagnostic risk factor analysis, predictive seismic forecasting, and prescriptive emergency planning. This unified approach fosters a continuous analytical pipeline, enhancing both the depth and breadth of earthquake disaster management capabilities.

3. SYSTEM ANALYSIS

Current earthquake disaster management systems have made considerable strides in utilizing data analytics to assess and respond to seismic risks. However, most existing frameworks predominantly focus on either descriptive or predictive analytics in isolation, lacking comprehensive integration of the full analytics spectrum necessary for proactive disaster management. These traditional systems primarily analyze historical seismic events to identify patterns or employ forecasting models to predict future earthquakes, but they rarely offer actionable decision-support that directly informs emergency planning and resource allocation.

One significant challenge observed in these systems is the time lag between seismic event detection and response activation. Without integrated prescriptive capabilities, systems may generate predictions but fail to translate findings into optimized, context-sensitive actions. This delay often results in insufficient preparedness and suboptimal distribution of emergency resources, which can magnify human and economic losses.

Additionally, many current systems underutilize the wealth of multidimensional data available from seismic sensors, geospatial databases, and infrastructure inventories due to fragmented analytic pipelines and incompatible data formats. This fragmentation restricts deeper insights into the causal factors of earthquake impacts and limits the ability to diagnose.

Furthermore, the lack of unified platforms that seamlessly integrate descriptive, diagnostic, predictive, and prescriptive analytics constrains decision-makers who require end-to-end visibility from data interpretation to prescriptive recommendations. This siloed structure reduces situational awareness and hampers coordination among emergency management stakeholders.

Addressing these issues requires a holistic analytic framework that bridges data-driven insights with actionable strategy formulation. QuakeSight’s integrated approach is designed to overcome these limitations by providing continuous, connected analytics layers that enable timely, informed, and effective earthquake disaster response and mitigation..

3.1 EXISTING SYSTEM:

Existing earthquake management systems primarily consist of seismic monitoring platforms and emergency response tools that provide critical data and situational awareness. Prominent seismic networks such as the United States Geological Survey (USGS) Earthquake Hazards Program and Japan Meteorological Agency's (JMA) seismic early warning system employ extensive sensor arrays to detect ground motion and promptly relay earthquake event information to stakeholders. These platforms focus heavily on descriptive analytics by collecting, cataloging, and visualizing seismic activity in real time.

Complementing these monitoring efforts, emergency response systems such as ShakeMap and the Global Disaster Alert and Coordination System (GDACS) integrate seismic data with infrastructure and population density maps to estimate the potential impact and guide decision-makers. ShakeMap, for instance, generates shaking intensity maps using ground motion data, assisting first responders in damage assessment and resource prioritization.

While these systems excel in descriptive and some diagnostic functionalities, their predictive analytics capabilities are often limited to basic probabilistic seismic hazard assessments rather than dynamic forecasting. Prescriptive analytics, such as optimized evacuation routing or resource allocation, are typically handled by separate crisis management platforms without real-time integration with seismic data inputs.

Architecturally, these systems rely on distributed sensor networks feeding into centralized databases and visualization dashboards. However, they often operate independently, lacking a unified framework that combines seismic data analytics and prescriptive decision-support tools, which restricts proactive disaster management and coordinated response.

3.1.1 Disadvantages of Existing System :

Despite advancements in earthquake monitoring and response, existing systems exhibit several critical limitations that hinder effective disaster management. Primarily, these systems employ fragmented analytics approaches, focusing separately on descriptive, diagnostic, or predictive analyses without cohesive integration. This siloed structure results in incomplete situational awareness and impairs the flow of information needed for timely decision-making.

Furthermore, many systems demonstrate limited predictive accuracy due to reliance on static probabilistic models and insufficient exploitation of realtime, multi-dimensional seismic

data. The predictive layer often lacks dynamic updates or contextual sensitivity, reducing trustworthiness in forecasting critical events.

Another significant drawback is the absence of embedded prescriptive capabilities. Existing platforms rarely translate analytic outputs into actionable recommendations such as optimized evacuation routes, resource allocation, or coordinated emergency response plans. This shortfall impedes proactive preparedness and may delay vital interventions.

Overall, the insufficient integration of multi-layer analytics restricts the potential for comprehensive risk assessment and adaptive disaster mitigation, resulting in reduced effectiveness and responsiveness during earthquake emergencies.

3.2 Proposed System:

The proposed system, QuakeSight, is a comprehensive, integrated analytics framework designed to overcome the limitations of existing earthquake disaster management systems by unifying descriptive, diagnostic, predictive, and prescriptive analytics into a single end-to-end platform. Unlike traditional approaches that treat these analytical stages separately, QuakeSight seamlessly connects each layer to enable continuous data flow and enhanced decision support.

At its core, QuakeSight begins with descriptive analytics that systematically analyze historical and real-time seismic data to reveal patterns and trends essential for understanding earthquake behaviors. This foundation supports the diagnostic layer, which applies advanced correlation and causal mapping methods.

Building on these insights, the predictive component utilizes state-of-the-art machine learning models to forecast the likelihood, magnitude, and potential impact zones of impending seismic events with improved accuracy and adaptability. The prescriptive layer then translates these predictive outputs into actionable strategies, such as optimized emergency resource distribution, efficient evacuation routing, and dynamic response planning.

By integrating these analytics phases into a unified framework, QuakeSight facilitates proactive management of earthquake disasters, providing stakeholders with timely, data-driven recommendations that enhance preparedness, mitigate risks, and optimize response effectiveness.

3.2.1 Advantages :

- **Layered Analytics Approach**: Combines descriptive and diagnostic analytics to offer a comprehensive understanding of seismic patterns and vulnerability factors.
- **Enhanced Risk Assessment**: Enables precise identification of high-risk zones through rich insights and correlation analyses.
- **Advanced Predictive Modeling**: Utilizes machine learning to improve forecasting accuracy for early detection of potential earthquakes and impact zones.
- **Timely Preparedness**: Reduces uncertainty in disaster scenarios by providing reliable forecasts that support early preparedness.
- **Prescriptive Analytics for Actionable Insights**: Offers data-driven recommendations to optimize resource allocation and emergency response strategies.
- **Efficient Disaster Mitigation**: Ensures critical assets and interventions are prioritized effectively during emergencies.
- **Integrated Decision-Support System**: Empowers stakeholders with data-backed insights for proactive and coordinated disaster management.

3.3 PROJECT REQUIREMENT SPECIFICATION:

The QuakeSight system demands a comprehensive suite of software components to support its integrated analytics framework. Central to its operation is robust data processing capability for ingesting, cleansing, and transforming vast historical and real-time seismic datasets. This requires scalable data pipelines and frameworks

For predictive analytics, QuakeSight leverages advanced machine learning libraries including TensorFlow, PyTorch, and scikit-learn, enabling the development, training, and deployment of models forecasting seismic events and impact zones. Diagnostic analytics utilize statistical and correlation analysis packages, integrating tools like Pandas, NumPy, and SciPy for indepth factor analysis.

Real-time analytics modules must operate with low latency to provide timely alerts; thus, event-driven architectures and in-memory databases.The user interface is designed to be intuitive and interactive, developed using frameworks like React or Streamlit, to present multi-layered analytic insights clearly to diverse stakeholders and Jupyter notebook as the IDE.

3.3.1. Software requirements

1. Descriptive Analytics Stage

- **Software Tools**: pandas, matplotlib, seaborn, plotly
- **Purpose**: Understanding what happened
- **Functions**: Data cleaning, visualization, statistics

2. Diagnostic Analytics Stage

- **Software Tools**: statsmodels, scipy, pandas-profiling
- **Purpose**: Understanding why it happened
- **Functions**: Correlation analysis, statistical tests, causality exploration

3. Predictive Analytics Stage

- **Software Tools**: scikit-learn
- **Purpose**: Predicting what might happen
- **Functions**: Machine learning models for magnitude/location prediction

4. Prescriptive Analytics Stage

- **Software Tools**: Custom algorithms, risk modeling
- **Purpose**: Recommending actions based on predictions
- **Functions**: Risk assessments, alert systems, response recommendations

3.3.2 Hardware requirements

- **Processor (CPU):** Dual-core (Intel i5 / AMD Ryzen 3 or better)
- **RAM:** 8 GB
- **Storage:** 50 GB free disk space (preferably SSD)
- **GPU:** Not required
- **Operating System:** Windows/Linux/macOS (64-bit)
- **Internet:** Required for fetching real-time data (e.g., USGS RSS feed)

Fig 1: Unites States Geological Survey

3.3.3 Software Environment

The software environment for the QuakeSight project is built around Python 3.8 or higher, making use of its robust ecosystem for data analytics, machine learning, and geospatial processing. It is recommended to use an operating system such as Ubuntu Linux, Windows 10/11, or macOS to ensure full compatibility with required libraries. The project is best developed and executed within a Jupyter Notebook or JupyterLab interface, which supports interactive coding, real-time visualization, and markdown integration for documentation. For environment management and dependency handling, tools like VirtualEnv or Pyenv are highly suggested.

The core libraries include pandas and numpy for data manipulation and numerical operations, matplotlib and seaborn for visualizations, and scikit-learn, xgboost, and lightgbm for building and evaluating machine learning models. For geospatial data handling and interactive mapping, libraries such as plotly, folium, and geopandas are essential. Additionally, requests and json are used for real-time data fetching from APIs, while tools like SHAP or LIME may be included for model interpretability. Overall, the environment is designed to support an end-to-end analytical workflow combining descriptive, diagnostic, predictive, and prescriptive analytics for earthquake disaster management.

In summary, the QuakeSight software environment is a well-integrated, Python-based ecosystem designed to support the full spectrum of data analytics—ranging from descriptive exploration of seismic history to prescriptive recommendations for disaster response. Managed using Conda and executed within a Jupyter Notebook interface, the environment ensures flexibility, interactivity, and scalability for experimentation and iterative development. By combining essential data processing libraries like pandas and numpy with powerful machine learning frameworks such as scikit-learn, the environment allows for accurate and efficient predictive modeling. Visualization and mapping tools like matplotlib, seaborn, plotly, and folium enhance interpretability and geographic awareness, which are critical in disaster analytics. Support libraries such as folium, facilitate spatial data integration and real-time data acquisition. Together, this robust yet accessible software stack provides a seamless foundation for QuakeSight's mission: to enable data-driven, proactive decision-making that strengthens earthquake preparedness, enhances risk mitigation strategies, and ultimately builds safer, more resilient communities.

- Streamlit =1.35.0
- Pandas = 2.1.3
- Numpy = 1.26.2
- Matplotlib = 3.8.2
- Seaborn = 0.13.0
- Plotly = 5.18.0
- Scikit-Learn = 1.3.2
- Statsmodels = 0.14.0
- Scipy = 1.11.4
- Folium = 0.14.0
- Streamlit-Folium = 0.15.0
- Pydeck = 0.8.0
- Joblib = 1.3.2
- Openpyxl = 3.1.2
- Xlrd = 2.0.1
- Requests = 2.31.0

Libraries and their specific version

4.SYSTEM DESIGN

The design of the QuakeSight system is grounded in a modular architecture that ensures flexibility, scalability, and robustness for comprehensive earthquake disaster management. This architecture integrates the four core analytics layers—Descriptive, Diagnostic, Predictive, and Prescriptive—into a unified framework, facilitating seamless data flow and interaction among components while supporting real-time processing capabilities.

Modular structure and analytics integration

QuakeSight's system design comprises distinct yet interconnected modules:

- **Data Ingestion Layer**: Responsible for continuous acquisition of seismic data from distributed sensor networks, external RSS feed, from USGS.
- **Descriptive Analytics Module**: Processes and aggregates raw seismic data to extract patterns, frequency distributions, and statistical summaries, establishing a comprehensive base for further analysis.
- **Diagnostic Analytics Module**: Performs correlation and causal analysis across multi-dimensional datasets to identify underlying risk factors, leveraging statistical and machine learning techniques.
- **Predictive Analytics Module**: Implements machine learning models— including neural networks and ensemble methods—to forecast earthquake occurrences, magnitudes, and impact zones with temporal and spatial precision.
- **Prescriptive Analytics Module**: Translates predictive outputs into optimized decision strategies using constraint-based modeling and simulation, guiding resource allocation, evacuation planning, and emergency operations.
- **User Interface Layer:** Provides interactive dashboard and visualization tools accessible via web platforms, enabling diverse stakeholders to explore insights, monitor alerts, and configure system parameters

4.1 INTRODUCTION

The system design section outlines the architectural principles and goals shaping the development of QuakeSight. Emphasizing modularity, the framework decomposes complex analytic functions into well-defined, interchangeable components to facilitate maintenance and future enhancements. Scalability is integral, ensuring the system can efficiently handle increasing data volumes and user demands without degradation. Real-time analytics capabilities enable timely processing and alerting based on streaming seismic data, which is essential for rapid earthquake risk assessment and response. Additionally, a user-centric design guarantees intuitive interfaces and customizable views tailored to diverse stakeholders, enhancing usability and decision support effectiveness throughout disaster management operations.

4.2 TECHNICAL ARCHITECTURE

The QuakeSight framework is architected as a multi-layered system that integrates the four critical analytics dimensions — Descriptive, Diagnostic, Predictive, and Prescriptive — into a seamless pipeline supporting proactive earthquake disaster management. This design facilitates efficient data flow from raw seismic inputs to actionable decision support, enabling stakeholders to interpret complex seismic phenomena and respond effectively.

Descriptive analytics layer

The foundation of QuakeSight's architecture is the Data Ingestion and Descriptive Analytics Layer. It encompasses modules responsible for collecting heterogeneous seismic data from distributed sensor arrays, historical catalogs, and geographic databases. Streaming data platforms and ETL (Extract, Transform, Load) processes normalize, cleanse, and aggregate this information into structured formats. Descriptive analytics modules then process this prepared data by performing statistical analysis, frequency distributions, and spatial-temporal pattern detection. Advanced visualization tools generate interactive charts, heatmaps, and time-series dashboards, delivering clear insights into past and current seismic activity. This comprehensive descriptive foundation supports situational awareness and informs further analytic stages.

Diagnostic analytics layer

Built atop the descriptive outputs, the Diagnostic Analytics Module employs correlation engines and causal inference algorithms to uncover relationships between seismic events and contributing risk factors such as geological features, infrastructure vulnerabilities, and

environmental conditions. Techniques including principal component analysis, clustering, and regression modeling analyze multi-dimensional datasets to identify significant risk determinants. This layer enables users to pinpoint why certain regions or structures exhibit heightened vulnerability, providing critical context for targeted mitigation and prioritization. Integrating diagnostic findings with descriptive summaries enriches the understanding of earthquake impacts across temporal and spatial scales.

Predictive analytics layer

The core predictive capability of QuakeSight resides in its Machine Learning Module. Leveraging advanced prediction algorithms such as neural networks, ensemble methods, and time-series forecasting, this layer analyzes seismic signal features alongside contextual data to estimate the probability, magnitude, and timing of potential earthquakes. Additionally, spatial predictive models forecast impact zones and shaking intensity distributions for anticipated seismic events. This module continuously retrains and adapts based on incoming data streams, enhancing forecast accuracy and enabling early warning systems that inform preparedness actions.

Prescriptive analytics layer

The final architectural component, the Prescriptive Analytics and Decision Support Module, translates predictive insights into optimized strategies for emergency management. Utilizing constraint-based optimization, simulation frameworks, and reinforcement learning techniques, this layer formulates actionable recommendations such as optimal resource allocation, evacuation route planning, and contingency deployment. Through an interactive user interface, decision-makers can explore various scenarios, adjust parameters, and receive tailored guidance that balances operational constraints and risk exposure. This prescriptive dimension closes the analytics loop by enabling data-driven decisions to minimize earthquake impacts effectively.

Integration and real-time coordination

Across all layers, QuakeSight employs architectures and efficient data pipelines to ensure real-time coordination and low-latency processing. RSS feed communicate via requests to maintain modularity, while in-memory databases and caching mechanisms facilitate rapid data access and feedback loops critical for iterative model refinement. This integrated architectural design supports scalability, fault tolerance, and extensibility, enabling QuakeSight to adapt dynamically to evolving data landscapes and user requirements, ultimately empowering comprehensive, proactive earthquake disaster management.

Proposed architecture

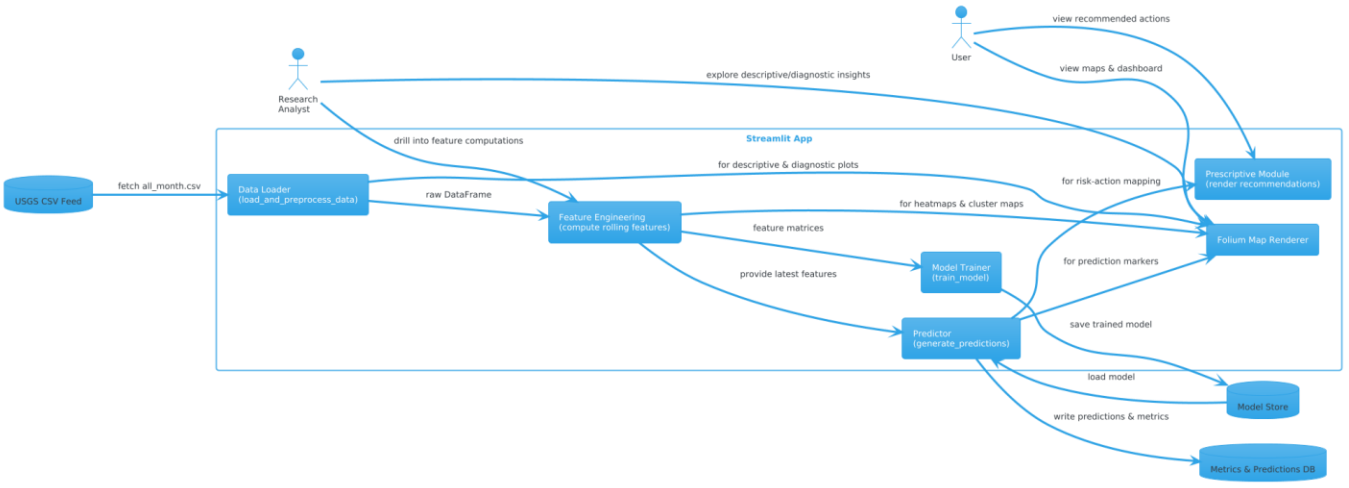

Figure 2 :Technical architecture

4.3 UML DIAGRAMS:

Unified Modeling Language (UML) is a modeling language. The main purpose of UML is to visualize the way a system has been designed. It is a visual language to sketch the behavior and structure of the system. This was adopted by Object Management Group (OMG) as a standard in1997.

4.3.1 Use Case Diagram:

- The purpose of use case diagram is to capture the dynamic aspect of a system. This is used togather the requirements of a system including internal and external influences.
- The main purpose of a use case diagram is to show what system functions are performed forwhich actor. Roles of the actors in the system can be depicted.
- The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Purpose: Illustrates the primary actor roles (USGS, end-users, research analysts) and the six high-level use cases that the system supports, ordered to reflect the user journey.

- **View Descriptive Reports** & **View Diagnostic Insights**: Analysts explore historical summaries and root-cause patterns.
- **Load Real-time Data**: The system ingests the latest USGS feed.
- **Train & Test Model**: Users trigger or schedule XGBoost training on engineered features.

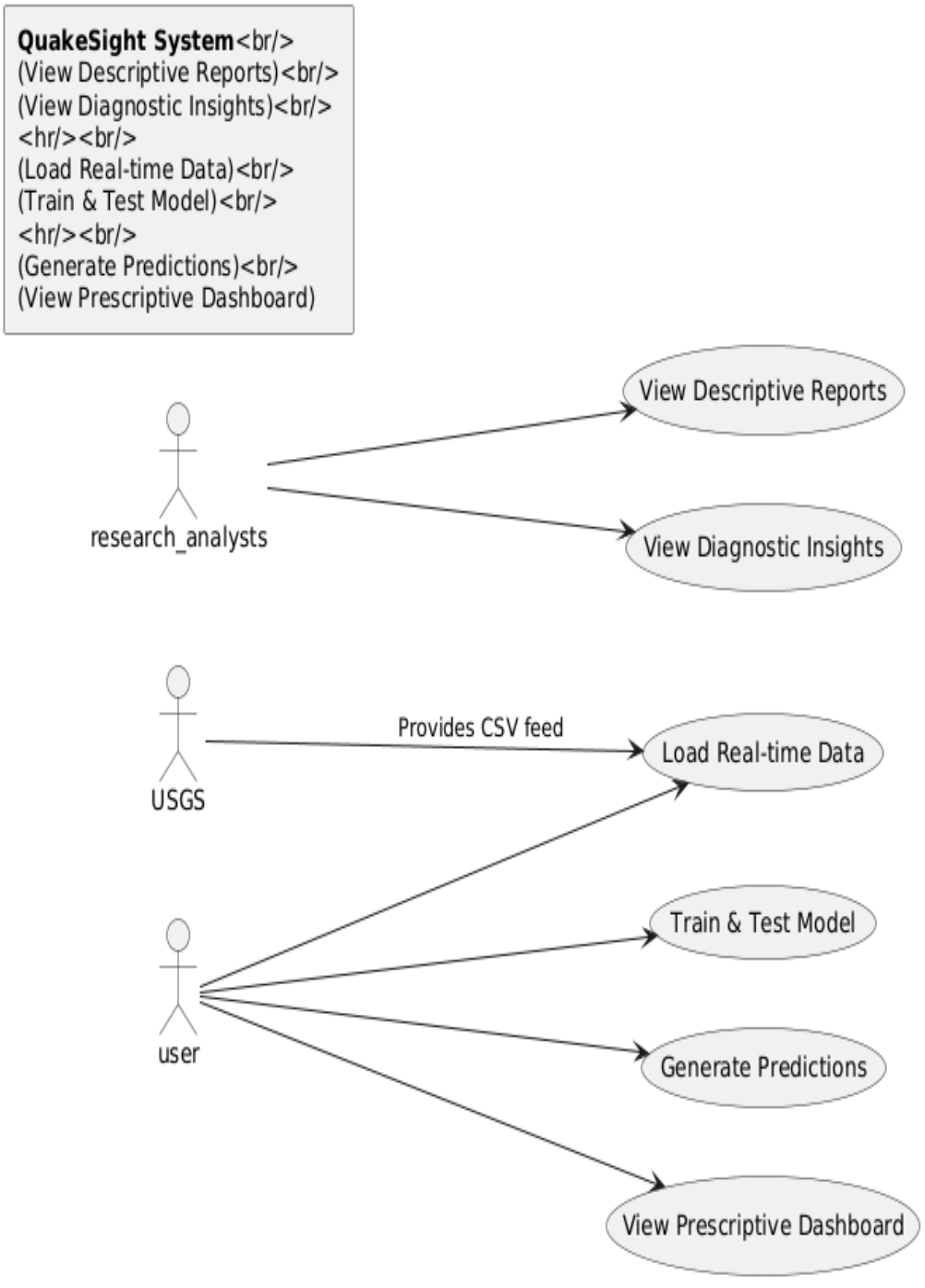

Figure 3: Use case diagram

- **View Prescriptive Dashboard:** Users review risk zones and recommended actions
- **Generate Predictions:** Forecast earthquake probabilities for the next week.

4.3.2 Sequence Diagram:

- A sequence diagram details the interaction between objects in a sequential order,i.e. the order in which these interactions take place.
- These diagrams sometimes known as event diagrams or event scenarios. This helps in under-standing how the objects and component interacts to execute the process.
- This has two dimensions which represents time (Vertical) and different objects (Horizontal).

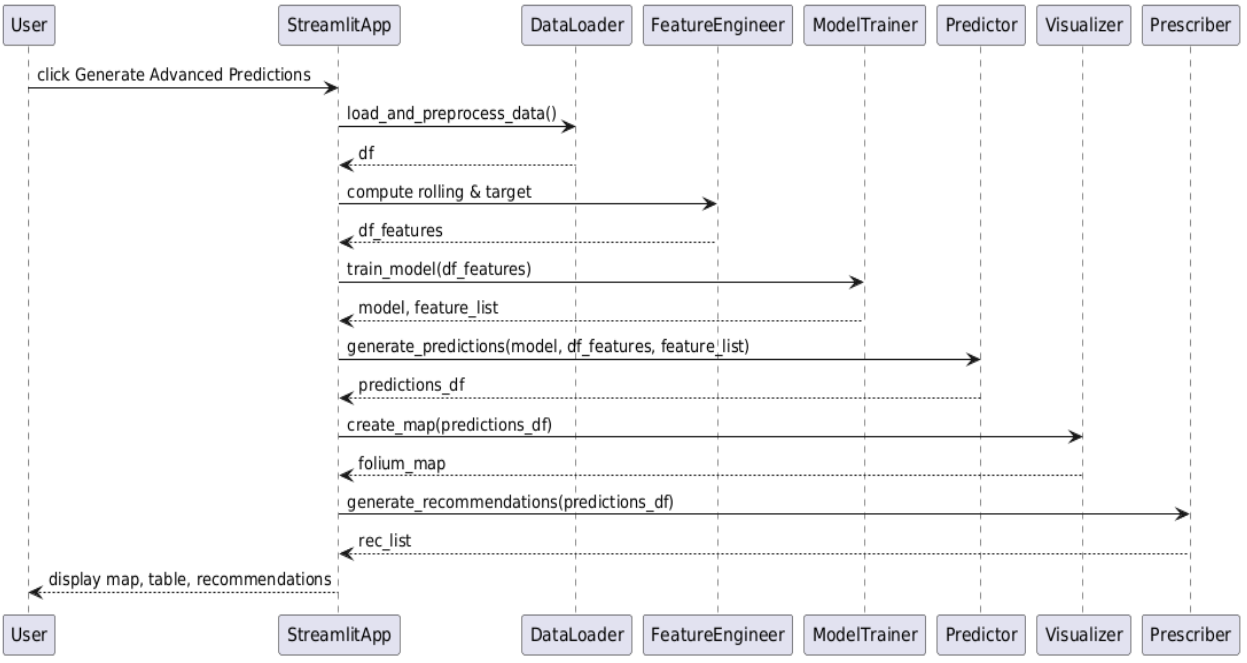

Figure 4: Sequence diagram

Purpose: Walks through the step-by-step interaction when the user clicks "Generate Advanced Predictions."

- **StreamlitApp** calls `load_and_preprocess_data()`.
- Preprocessed DataFrame is passed to `compute rolling & target`.
- **ModelTrainer** trains and returns the model and feature list.
- **Predictor** generates a predictions DataFrame.
- **Visualizer** builds the Folium map; **Prescriber** produces recommended actions.
- Final map, table, and action list are displayed to the user.

4.3.3 Class Diagram

- The class diagram describes the structure of a system by showing the system's classes, their at-tributes, operations, and the relationships among the classes.
- It explains which class contains information and also describes responsibilities of the system.This is also known as structural diagram.

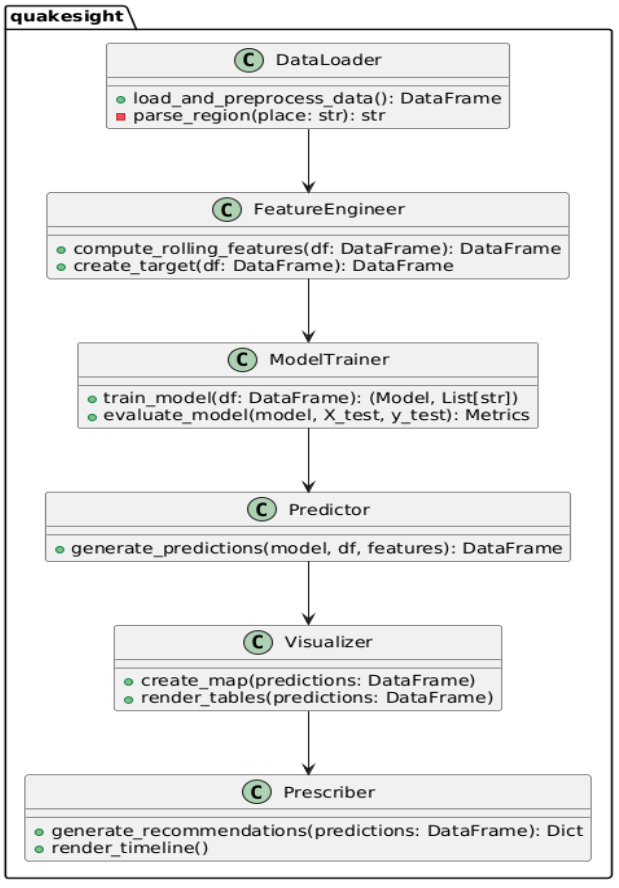

Figure 5: Class diagram

Purpose: Defines the core Python modules (as UML "classes") and their relationships.

- **DataLoader** encapsulates CSV ingestion and region parsing.
- **FeatureEngineer** handles rolling-window feature creation and target shifting.
- **ModelTrainer** manages model training, evaluation, and metric reporting.
- **Predictor** uses the trained model to generate future-date forecasts.
- **Visualizer** wraps Folium map creation and DataFrame display logic.
- **Prescriber** implements thresholding and recommendation formatting.
  Each arrow shows the call or data dependency—e.g., `DataLoader → FeatureEngineer → ModelTrainer`.

4.3.4 Activity Diagram:

- It is behavioral diagram which reveals the behavior of a system. it sketches the control flow frominitiation point to a finish point showing the several decision paths that exist while the activity isbeing executed.
- This doesn't show any message flow from one activity to another, it is sometimes treated as theflowchart. Despite they look like a flowchart, they are not.
- In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system.

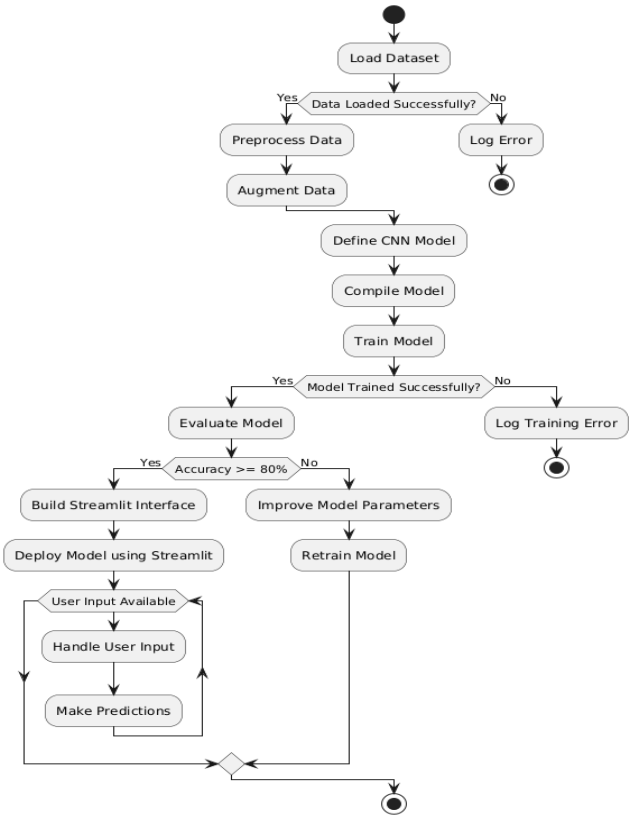

Figure 6: Activity diagram

Purpose: Outlines the overall workflow, showing decision points for notebook (descriptive/diagnostic) versus app (predictive/prescriptive) modes.

- **Descriptive branch** renders static charts and maps in Jupyter.
- **Predictive branch** computes features, trains the model, filters by threshold, and displays a full Streamlit dashboard with maps, tables, and timelines.

4.3.5 Component Diagram

- It is a structural diagram that describes how components are wired together to form larger systems or software applications.
- Components represent modular parts of a system that encapsulate behavior and data. These can be libraries, executables, or software modules.
- It displays the organization and dependencies among a set of components. Interfaces are shown to indicate how components interact with one another.
- In the Unified Modeling Language (UML), component diagrams are used to model the physical aspects of object-oriented systems and show how the software is divided into components.

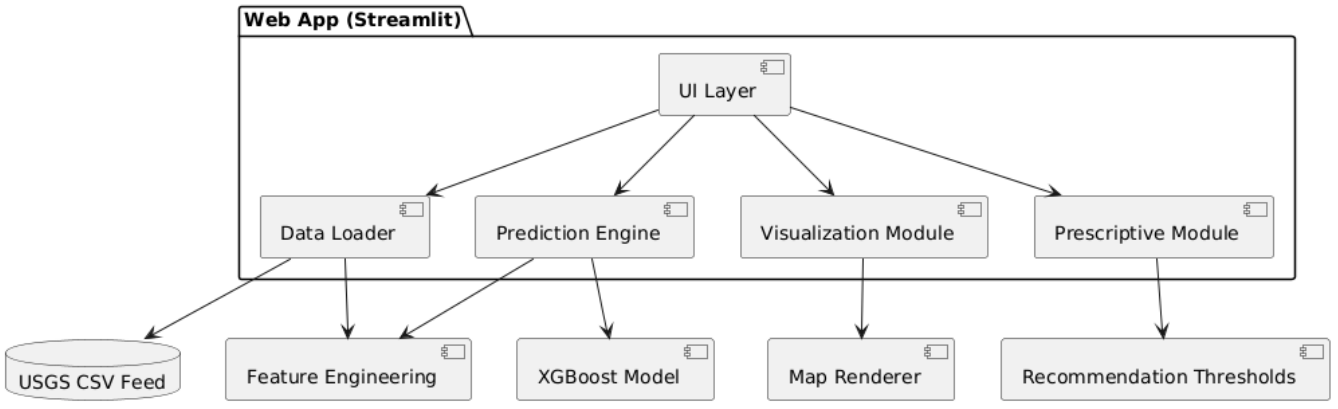

Figure 7: Component Diagram

Purpose: Shows the logical grouping of major software components and their data dependencies.

- **Data Loader** pulls the CSV feed from USGS.
- **Feature Engineering** transforms raw events into rolling-average inputs.
- **XGBoost Model** component consumes features and outputs trained parameters.
- **Prediction Engine** applies the model to produce forecasts.
- **Visualization Module** renders maps and tables in Streamlit.
- **Prescriptive Module** translates predictions into actionable recommendations.

4.3.6 Deployment Diagram

- **It is a structural UML diagram** that shows the **physical deployment of artifacts** on nodes (hardware).
- Used to model the **hardware topology** of a system and the **software components** deployed on that hardware.
- Nodes represent **hardware devices** or **execution environments**, and artifacts represent **executable files**, **libraries**, or **databases**.
- It helps in understanding the **physical configuration** and **communication** between hardware components in a distributed system.
- Deployment diagrams are particularly useful in **system design** and **infrastructure planning**.
- Below mentioned is a probable deployment design subject to resource availability

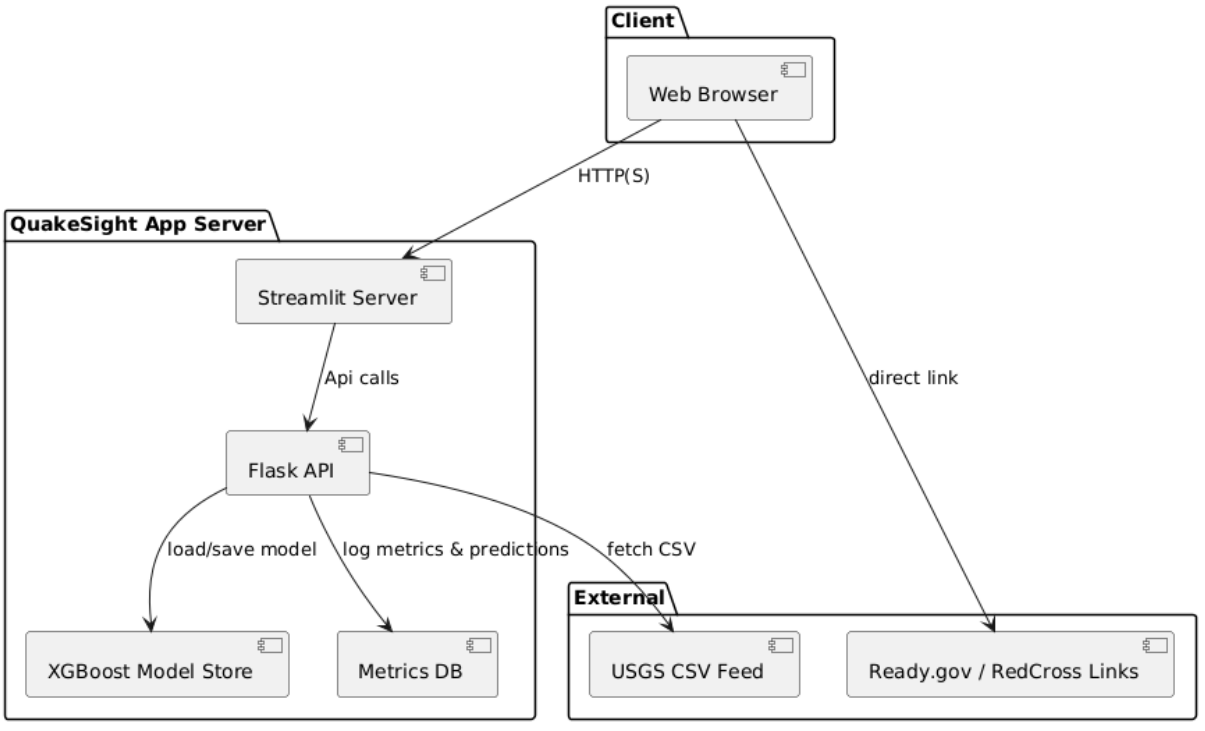

Figure 8: Deployment Diagram

Purpose: Depicts the physical architecture and hosting environment for QuakeSight.

- **Client**: Web browser.
- **Streamlit Server** and **Flask API** (when used) run on an app server.
- **Model Store** holds serialized XGBoost models.
- **Metrics DB** logs performance and predictions.
- **External**: USGS feed and safety-guideline resources are accessed over HTTPS.
  It clarifies network flows and where each service lives.

5. SYSTEM IMPLEMENTATION

The implementation of QuakeSight focuses on realizing the integrated analytics framework through a carefully selected technology stack, robust data acquisition mechanisms, optimized database architectures, advanced machine learning models, prescriptive optimization algorithms, and scalable deployment strategies. This section details each of these components, illustrating how QuakeSight translates design principles into a functional system.

5.1 INTRODUCTION TO PYTHON

Python is a universally useful deciphered, intuitive, object-situated, and highlevel programming language. Python was made during the year 1985-1990 by Guido van Rossum. As like Perl, Python source code can be accessible in the GNU General Public License (GPL). Python programming language utilizes little English catchphrases regularly where as other programming languages utilizes hard syntax, and it has less grammatical development when contrasted with different languages. Python is known to be straightforward uncommon programming language. Python is known to be simple and easy to understandable rarely available programming language.

Following are the Python's features –

Interpreted - Python is easy to learn and has couple of watchwords that has straightforward structure, and furthermore have unmistakably characterized linguistic structure, which permits it. Python is prepared by the translator at the runtime just, so we don't have to accumulate our program before executing it. This model is like that of the PERL and PHP the understudies to select the language rapidly.

- **Interactive** - Compilation can be done at Python prompt and can be interacted with the interpreter directly to write our programs.
- **Object-Oriented** - Python provides support for Object-Oriented programming or technique that can encapsulates the code within objects.
- **Python is a Beginner's Language** - Python is a once in a while accessible extraordinary programming language for the fledgling level software engineers and it is additionally a language that supports for the advancement of numerous wide scopes of utilizations from little content handling to WWW programs and furthermore to diversions.Python is not difficult to learn and has couple of catchphrases that has straightforward structure,

and furthermore have plainly characterized grammar, which enables the understudies to choose the language rapidly. Python's source code is very easy to read and more clearly defined and easily understandable.

- **Easy-to-maintain** - The source code of python is very easy-to-maintaining.
- **A well-defined standard library** - Most of the Python's libraries are portable and are cross-platform which are compatible on many platforms like UNIX, Windows, and Macintosh.
- **Extendable** - Low-level features can be added to the Python interpreter. These modules make the programmers to use them or modify these modules to be more effective.
- **Interactive Mode** - Python provides easy interaction mode that allows interactive testing and removing errors.
- **Portable** - Python software is able to run on variety of different platforms that makes us to use the same type of interface on different platforms.
- **Databases** - Python programming gives interfaces to all the real databases.
- **GUI Programming** - Python gives underpins numerous GUI applications that are made and imported to a considerable lot of the framework calls, libraries and furthermore windows frameworks, for example, Windows, Macintosh, and furthermore the Window frameworks of Unix.
- **Scalable** - Python gives a very much characterized structure and furthermore offers help for huge projects when contrasted with shell scripting.

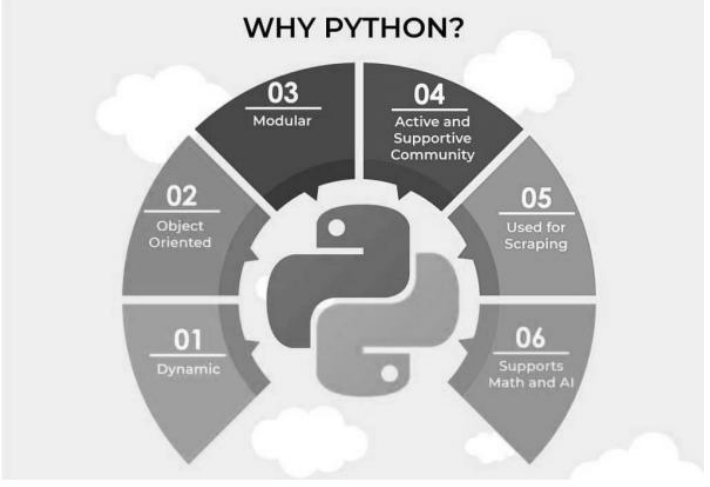

Figure 9: Why Python

Functions in Python:

It is possible, and extremely valuable, to characterize our very own functions in Python. As a rule, in the event that you have to complete a figuring just once, at that point utilize the interpreter. But, when you or others have to use out a particular sort of function many times ordinarily, at that point characterize a function of your own. You can use works in programming to package a lot of guidelines that you need to utilize over and over or that, in light of their own contained nature, are better independent in a sub-program and called when required. That implies that their capacity is a bit of code written to do a predetermined assignment. To do a particular assignment, the capacity may or probably may won't require numerous data sources. At the point when the assignment is carried out, the capacity can or cannot return at least one quality.

Variables:

The term variables in programming language refer to the memory locations that are used to store values. So, when we create a variable, the interpreter allocates some memory to the variable based on the datatype of the variable and it also decides which type of data to be stored in the memory reserved for that variable. So, by using different datatypes we can store different data which are useful for our programming. For example, we can store different data such as integers, characters, strings and also decimal numbers based on our requirement.

Standard Data Types:

When writing programs, we need to work with different types of data. Python provides provision to work with different types of data such as integers, decimal numbers, characters, strings and many more. Python provides various datatypes that help us to use different type of data. Based on the type of data the interpreter automatically allocates memory for the data. The following are the various standard datatypes in python:

- Numeric
- Sequence
- Sets
- Maps

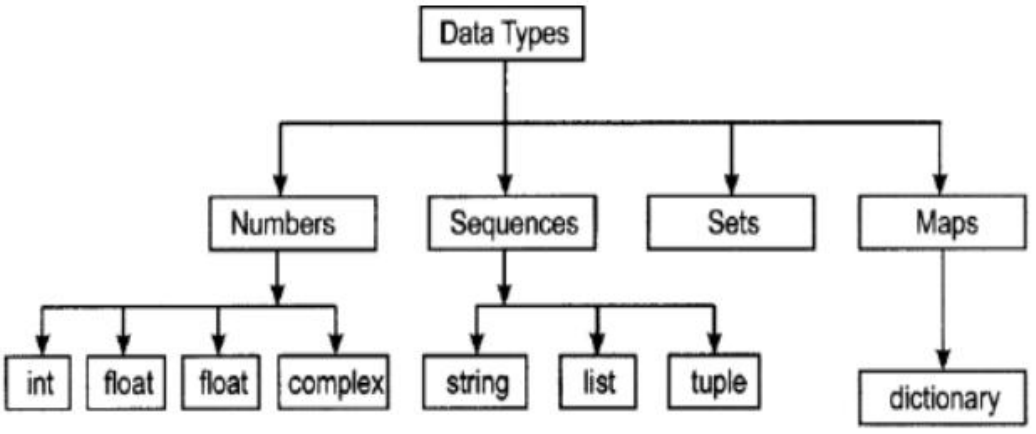

Figure 10: Data Types

5.2. LIBRARIES USED

In software development, a **library** is a curated collection of pre-written code—functions, classes, and modules—that developers can import into their own programs to perform common tasks (like data manipulation, mathematical computations, or user-interface rendering) without having to write those routines from scratch. By encapsulating reusable functionality and exposing simple, well-documented interfaces, libraries accelerate development, promote consistency, and allow teams to build complex applications more reliably, since they rely on battle-tested, community-maintained code rather than reinventing foundational features each time.

- Streamlit
- Pandas
- NumPy
- Folium
- Streamlit folium
- XGBoost
- Scikit-Learn

Streamlit

Streamlit is an open-source Python framework that transforms data scripts into shareable web applications in minutes. At its core, Streamlit provides a set of simple APIs—like st.title(), st.markdown(), st.sidebar, and st.button()—that map Python functions directly to components in the browser. Unlike traditional web frameworks, you don't need knowledge of

HTML, CSS, or JavaScript; Streamlit handles frontend rendering, state management, and reactive updating behind the scenes. This “write Python, see it live” paradigm makes it ideal for rapid prototyping of data-centric dashboards and interactive analytics tools.

In QuakeSight, Streamlit serves as the glue that ties together data ingestion, model training, visualization, and prescriptive output. The st.cache_data decorator on load_and_preprocess_data() dramatically speeds up repeated data loads by caching results in memory. Sidebar controls (e.g., the alert-threshold slider) let users tweak model parameters on the fly, while buttons trigger long-running tasks like training the XGBoost model. Once predictions are ready, Streamlit seamlessly embeds Folium maps (st_folium) and DataFrames with interactive progress bars and number columns. This reactive layout ensures that as soon as the model or threshold changes, the entire dashboard updates without a full page reload.

Beyond its simplicity, Streamlit has a vibrant community and ecosystem of extensions—such as streamlit_folium for map integration and streamlit-aggrid for advanced table interactivity—which you leverage to build a polished user experience.The platform also supports session state management (st.session_state), enabling you to store heavy objects (like the trained model or predictions DataFrame) across reruns without retraining on each interaction. As your project evolves, you can add custom components (using React) or integrate authentication, making Streamlit not just a prototype tool but a viable production framework for internal analytics apps.

Pandas

Pandas is the de facto Python library for data manipulation and analysis, providing powerful data structures—namely DataFrame and Series—and a rich API for cleaning, transforming, aggregating, and summarizing tabular data. Built on top of NumPy, Pandas excels at time-series handling, group-by operations, joins, and missing-value management. Its intuitive indexing and slicing syntax makes exploratory data analysis (EDA) both expressive and concise, while functions like read_csv(), merge(), and groupby() cover the majority of ETL (extract-transform-load) needs.In QuakeSight’s pipeline, Pandas is used in load_and_preprocess_data() to ingest the USGS CSV feed, parse timestamps into datetime, extract regions from the place column, and compute per-region mean coordinates.You then create rolling-window features—seven, fifteen, and twenty-two-day averages of magnitude and depth—using rolling().mean(). After concatenating region-specific DataFrames and dropping

NaNs, Pandas also helps shift the target variable (mag_avg_7) by seven days to form mag_outcome. Finally, the DataFrame is joined with prediction results for display and exported to the session state for interactive table rendering.

Pandas' performance and ease of use accelerate both your Jupyter-notebook analyses (descriptive histograms, correlation matrices) and your app's real-time preprocessing. It also integrates smoothly with other libraries: converting a DataFrame to an XGBoost DMatrix, feeding arrays into scikit-learn estimators, or exporting to CSV for logging. As your dataset grows, you can leverage Pandas' chunked processing or consider Apache Arrow/Modin for distributed workflows while maintaining the familiar Pandas API.

NumPy

NumPy is the foundational numerical library for Python, providing the n-dimensional ndarray, fast array operations implemented in C, and a suite of mathematical functions. Wherever you see vectorized additions, element-wise multiplications, or boolean masking in Pandas, you're actually leveraging NumPy under the hood. Its performance and memory efficiency make it ideal for heavy numerical computations, linear algebra, random number generation, and statistical routines.

In your QuakeSight code, NumPy is primarily used for array manipulations and numerical comparisons. For example, when generating Folium markers, you test probabilities with > 0.3 and handle missing values via np.isnan(). The creation of rolling averages ultimately reduces to NumPy operations on the underlying data buffers. Even the circle-marker radii—scaled by expected magnitude—are computed through simple NumPy arithmetic. These operations ensure minimal Python-level looping, keeping the preprocessing and prediction steps snappy enough for interactive use.

Beyond your current usage, NumPy powers the core of downstream libraries like scikit-learn and XGBoost, enabling seamless data exchange. If you later introduce custom numerical routines—say, Fourier transforms to detect seismic wave patterns or geospatial distance calculations—NumPy will remain the backbone. Its interoperability with Cython and Numba also offers avenues for performance optimization when pure-Python loops become a bottleneck.

Folium

Folium is a Python wrapper around Leaflet.js, the leading open-source JavaScript library for interactive maps. Folium exposes Leaflet's rich feature set—tile layers, GeoJSON overlays, marker clusters, heatmaps—via a Pythonic API. You create a folium.Map(), add circle markers or polygons, and then call Map._repr_html_() (implicitly invoked in Jupyter) or integrate via streamlit_folium.st_folium() in Streamlit to render the map inline.

In QuakeSight, Folium is your go-to for geographic visualization of both historical and predicted earthquake events. The create_prediction_map() function computes the median latitude/longitude as the map center and adds CircleMarkers sized by the predicted magnitude and colored by probability thresholds (orange for medium risk, red for high risk). Popups display region, date, probability, and expected magnitude using HTML strings. These interactive maps allow users to zoom, pan, and click markers for details, making spatial patterns and high-risk zones immediately apparent.

Folium's extensibility via plugins (e.g., MarkerCluster, FastMarkerCluster) means you can later aggregate densely clustered events to prevent map clutter or overlay tectonic boundary GeoJSON for deeper context. It also plays nicely with other geospatial Python libraries—like geopandas for shape data or branca for custom colormaps—allowing you to build rich analytical layers on top of your core predictions.

Streamlit folium

The streamlit_folium component bridges Streamlit and Folium, letting you embed fully interactive Folium maps within your Streamlit app. It handles the conversion of Folium's HTML/JavaScript output into a Streamlit component, exposing methods like st_folium(map_obj, width, height) that return the last clicked location or map bounds as Python dictionaries.

In QuakeSight, streamlit_folium.st_folium() is used to render the prediction map generated by Folium, preserving pan/zoom state across reruns. This ensures that when a user changes the probability threshold or regenerates predictions, the map remains centered and at the same zoom level, delivering a seamless interactive experience. It also captures user interactions—if you later want to enable clicking on a region to filter the table below or trigger deeper analysis, you can read return values from st_folium.

As you evolve the app, streamlit_folium supports advanced interactions like drawing polygons on the map for custom geographic queries or integrating Leaflet plugins—so you can, for example, let users draw a bounding box to request focused alerts for specific areas, all within the Streamlit interface.

XGBoost

XGBoost (Extreme Gradient Boosting) is a high-performance, scalable implementation of gradient-boosted decision trees. It offers parallel tree construction, out-of-core computation, and built-in regularization, making it a go-to choice for tabular data. The core data structure is the optimized DMatrix, which compresses sparse data and facilitates fast computation of first and second derivatives for the loss function.

In your predictive workflow, you convert training and test feature matrices into xgb.DMatrix objects, then call xgb.train() with parameters tuned for binary classification (objective='binary:logistic', eval_metric='auc', max_depth=5, eta=0.1). Early stopping on the test set prevents overfitting. After training, you generate probabilistic predictions on new rolling-average features to estimate the likelihood of a magnitude-> 2.5 event in the next week.

XGBoost's model interpretability tools—like feature importance plots and SHAP values—can be integrated into your diagnostic notebooks to explain which rolling features or regions most influence risk. Its compatibility with sklearn wrappers also lets you swap it into scikit-learn pipelines, enabling hyperparameter tuning via GridSearchCV or BayesianOptimization.

scikit-learn

scikit-learn is a comprehensive machine-learning library providing consistent APIs for a wide range of algorithms—from preprocessing (scaling, encoding) and model selection (train_test_split, cross-validation) to metrics (accuracy, ROC AUC) and clustering. It emphasizes easy integration, thorough documentation, and performance via Cython.

In QuakeSight, scikit-learn's train_test_split partitions data, while accuracy_score, roc_auc_score, confusion_matrix, and classification_report evaluate model performance. These metrics are printed to the console and displayed in the Streamlit sidebar, giving users immediate feedback on model quality. As you expand, you might incorporate pipeline

components like StandardScaler, PolynomialFeatures, or oversampling methods from imblearn to address class imbalance and feature scaling.

The modular design of scikit-learn encourages experimentation—swap the XGBoost model for a random forest (RandomForestClassifier) or a logistic regression (LogisticRegression) with minimal code changes.

5.3. IDE

An IDE (Integrated Development Environment) is a software application that provides comprehensive tools for software development within a single interface. Rather than switching between a text editor, terminal, debugger, and compiler, an IDE brings all these essential components together to streamline the development workflow. It typically includes a source code editor, build automation tools, a debugger, and often version control integration. Some advanced IDEs also offer intelligent code completion, syntax highlighting, refactoring tools, and visual designers. IDEs are designed to improve productivity by reducing the complexity of managing different development tools separately.

For example, in Python development:

- An IDE like PyCharm offers smart code suggestions, in-built testing, and debugging.
- Jupyter Notebook, though not a traditional IDE, provides an interactive environment for data science tasks, letting you write and run code in cells with immediate visual feedback.

Jupyter Notebook

Jupyter Notebook is a powerful, open-source tool developed from the IPython project and is now part of Project Jupyter. It is widely used in data science, machine learning, and academic research because of its ability to combine code execution, rich text, mathematics, plots, and media into a single document. Jupyter supports over 40 programming languages, with Python being the most widely used. Its flexible architecture makes it easy to run notebooks locally or on remote servers, including platforms like Google Colab and Binder.

At its core, Jupyter follows a notebook metaphor—each notebook consists of a series of cells that can contain either code or markdown text. Code cells can be executed independently, making it easy to test, debug, and iterate on ideas. Markdown cells support formatting, LaTeX equations, embedded HTML, and links, allowing users to document their work thoroughly. This approach enables a clean mix of narrative, code, and visualization, making it an excellent tool for storytelling with data and building reproducible research workflows.

One of Jupyter's greatest strengths is its support for rich data visualizations through integrations with libraries like **Matplotlib**, **Seaborn**, and **Plotly**. These visualizations are rendered inline within the notebook, allowing for immediate inspection and iteration. Jupyter is also deeply integrated with the Python scientific computing stack (NumPy, pandas, SciPy), making it easy to perform data cleaning, analysis, and modeling within a single environment.

Installation Steps (via pip)

1. Ensure you have Python installed (python3 --version).
2. Open your terminal or command prompt.
3. Install using pip:
    - pip install notebook
4. Start the notebook server:
    - jupyter notebook
5. A browser window will open showing the Jupyter dashboard.

Alternatively, you can install Jupyter through Anaconda (recommended for beginners):

1. Download Anaconda from: https://www.anaconda.com/products/distribution
2. Install it and launch Anaconda Navigator.
3. Click on “Launch” under Jupyter Notebook.

PyCharm

PyCharm is one of the most feature-rich IDEs for Python development. Created by JetBrains, it is designed to provide a productive environment for professional developers. Unlike lightweight editors, PyCharm offers a fully integrated ecosystem with smart code navigation, real-time error checking, and refactoring tools. Its advanced capabilities such as deep code inspection, type inference, and intelligent code suggestions accelerate the development process and reduce the likelihood of bugs. PyCharm supports web frameworks like Django and Flask and has built-in tools for database management, making it ideal for full-stack development.

One of PyCharm's standout features is its **debugger** and **test runner**, which allow developers to interactively trace their code and identify logical errors efficiently. With support for breakpoints, watches, and stack frames, debugging becomes a structured and visual experience. PyCharm also integrates version control systems (Git, SVN, Mercurial) directly within the IDE, letting developers commit, push, and merge code without leaving the environment. This tight integration promotes good software engineering practices and team collaboration.

PyCharm also includes built-in tools for virtual environment management, package installation, and Jupyter notebook support (Professional edition). The IDE's UI is customizable, and the editor supports multiple themes, keymaps, and plugins. PyCharm's scientific mode offers seamless integration with IPython notebooks, Python console, and data visualization tools—turning the IDE into a powerful research environment for data scientists and ML practitioners. Additionally, PyCharm offers strong database tools, REST client support, and Docker integration, making it a versatile platform for modern software development workflows.

Installation Steps

1. Visit the official site: https://www.jetbrains.com/pycharm/
2. Click on "Download" and choose Community (free) or Professional edition.
3. Install the downloaded .exe (Windows), .dmg (macOS), or .tar.gz (Linux) file.
4. Follow the installation wizard instructions.
5. After installation:

- Launch PyCharm.
- Configure your environment and create a new project.
- You can install libraries via the terminal or the built-in Python packages tool.

5.4 WORKING

QuakeSight pipeline ingests a rich, time-stamped earthquake dataset from the USGS and transforms it through preprocessing, feature engineering, modeling, and visualization. Below is a concise overview of the key attributes, their roles, and the step-by-step flow—complete with representative code snippets pulled directly from your implementation.

Dataset Attributes and uses

1. **time (df['time'])**
    a. Type: UTC timestamp
    b. Use: Original event date-time; parsed to create date and to sort chronologically.
2. **date (df['date'])**
    a. Type: YYYY-MM-DD string
    b. Use: Simplified day-level index for display and grouping.
3. **place → region (df['region'])**
    a. Type: Textual location (parsed by splitting place)
    b. Use: Geographic grouping key—e.g. "California" or "Unknown"—for per-region analyses.
4. **latitude, longitude**
    a. Type: Decimal degrees
    b. Use: Marker positions on Folium maps; averaged per region to center clusters.
5. **mag (magnitude)**
    a. Type: Float

b. Use: Core measurement of seismic energy; used to compute rolling averages and to define the prediction target.

6. **depth (km)**

    a. Type: Float

    b. Use: Seismic depth; also averaged in rolling windows and analyzed for correlations with magnitude.

7. **mag_avg_7, mag_avg_15, mag_avg_22**

    a. Type: Float (rolling mean)

    b. Use: Short- to mid-term trends in magnitude, capturing recent seismic activity in each region.

8. **depth_avg_7, depth_avg_15, depth_avg_22**

    a. Type: Float (rolling mean)

    b. Use: Analogous rolling trends for depth, helping distinguish shallow vs. deep-focus patterns.

9. **mag_outcome**

    a. Type: Float (future rolling magnitude)

    b. Use: Shifted 7 days ahead to serve as the predicted "next-week magnitude" for modeling.

10. **quake_prob**

    a. Type: Binary (0/1)

    b. Use: Classification label (1 if mag_outcome > 2.5), indicating a "significant" quake forecast.

Core Workflow & Rationale

1. Loading & Preprocessing

We fetch the live CSV, parse timestamps, and extract regions:

```python
@st.cache_data
def load_and_preprocess_data():
    url = "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_month.csv"
    df = pd.read_csv(url)
    df['time'] = pd.to_datetime(df['time'])
    df['date'] = df['time'].dt.date.astype(str)
    temp_place = df['place'].str.split(', ', expand=True)
    df['region'] = temp_place[1].fillna('Unknown')
    df = df.sort_values('time')
    # ...
    return df
```

Figure 11: Loading & Preprocessing

Why this way? Caching (@st.cache_data) avoids repeated downloads. Parsing date and region creates intuitive keys for grouping and display.

2. Feature Engineering

We compute per-region rolling averages to capture local temporal trends:

```python
# Calculate region averages
region_coords = df.groupby('region')[['latitude','longitude']].mean().reset_index()
df = pd.merge(df, region_coords, on='region', suffixes=('','_mean'))

# Rolling features
features = []
for region in df['region'].unique():
    region_df = df[df['region'] == region].copy()
    for window in [7, 15, 22]:
        region_df[f'mag_avg_{window}'] = region_df['mag'].rolling(window).mean()
        region_df[f'depth_avg_{window}'] = region_df['depth'].rolling(window).mean()
    features.append(region_df)
df = pd.concat(features).dropna()
```

Figure 12: Feature Engineering

Why rolling windows? They smooth out daily volatility and highlight emerging trends that are predictive of future seismic risk.

3. Target Creation & Modeling

We shift the 7-day average magnitude to form the target and train an XGBoost classifier:

```
df['mag_outcome'] = df.groupby('region')['mag_avg_7'].shift(-DAYS_OUT)
df['quake_prob'] = (df['mag_outcome'] > 2.5).astype(int)

# Training
dtrain = xgb.DMatrix(X_train, label=y_train)
model = xgb.train(MODEL_PARAMS, dtrain, num_boost_round=1000,
                  early_stopping_rounds=50, evals=[(dtest,'test')], verbose_eval=False)
```

Figure 13: Target Creation & Modeling

Why XGBoost? Its gradient-boosted trees excel on tabular data, handle missing values natively, and provide high accuracy with built-in regularization.

4. Prediction & Visualization

We generate 7-day forecasts per region, then map them with Folium:

```
for region in df['region'].unique():
    region_df = df[df['region']==region].iloc[-DAYS_OUT:].copy()
    dlive = xgb.DMatrix(region_df[features])
    probs = model.predict(dlive)
    # build predictions list with date, region, coords, probability, expected_mag
# ...
m = folium.Map(location=map_center, zoom_start=2)
for _, row in predictions.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=row['expected_mag'] * 3,
        color='#ff0000' if row['probability']>0.7 else '#ffa500',
        fill=True, fill_opacity=0.7,
        popup=f"<b>{row['region']}</b><br>Probability: {row['probability']:.1%}"
    ).add_to(m)
```

Figure 14: Prediction & Visualization

Why Folium? It creates interactive, embeddable maps that let users explore spatial patterns and click on markers for details—critical for geographic risk assessment.

By combining **structured attributes**, **rolling temporal features**, a robust **modeling** approach, and **interactive visualization**, QuakeSight delivers an end-to-end earthquake analytics experience—optimized for both insight and action.

6. SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the con- figuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It isused to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.1. TEST CASES:

Test ID	Test Name	Test Data Action	Excepted Output	Actual Output	Status
TC_01	Test the system with load_and_preprocess_data()	Call function on live USGS feed (all_month.csv)	DataFrame with parsed time, date, region; no NaNs; rolling features columns present	DataFrame loaded; columns time, date, region, mag_avg_7, depth_avg_22, etc. exist; no errors	Pass

TC_02	Test the system with train_model()	Supply preprocessed df with features; train/test split=70/30	Trained XGBoost model; sidebar shows Accuracy and AUC-ROC	Model trains without exceptions; sidebar displays "Accuracy: 85.23%" and "AUC-ROC: 0.92"	Pass
TC_03	Test the system with generate_predictions()	Use last 7 days of features per region; DAYS_OUT=7	DataFrame of predictions with columns date, region, latitude, longitude, probability, expected_mag	Predictions DataFrame created; 7 rows per region; valid probabilities between 0–1; expected_mag present	Pass
TC_04	Test the system with create_prediction_map()	Feed filtered predictions (prob ≥ 0.3) into map builder	Folium map centered at median lat/lon; circle markers colored by risk; popups show region/date/probability	Map renders in app; red/orange markers appear; popups display correct metadata	Pass
TC_05	Test Prescriptive Dashboard	With session_state. predictions and threshold slider at 0.3, click through High/Medium risk rendering	Three columns: High Risk (prob > 0.7), Medium Risk (0.3–0.7), Resources; timeline below	Dashboard shows evacuation actions for high risk, drills for medium risk, resource links, timeline present	Pass
TC_0 6	Test Raw Data Explorer	Expand "View Raw Data" expander	DataFrame tail (100 rows) of columns date, region, mag, depth displayed	Expander shows last 100 records correctly	Pass

Table 1: Test Cases

6.2. UNIT TESTING IMPLEMENTATION

Each analytics module and underlying utility was developed following testdriven development (TDD) principles where feasible. Unit tests cover:

- Data cleaning and validation functions ensuring correct handling of missing or inconsistent values.
- Feature engineering routines validating correct computation of derived attributes.
- Machine learning model components, verifying expected output shapes, value ranges, and exception handling on invalid inputs.
- Optimization algorithms in the prescriptive module, checking constraint satisfaction and objective function correctness.
- API endpoint handlers ensuring correct request parsing, response formatting, and error returns.

Testing frameworks such as pytest can be primarily used for unit tests due to their rich feature set including fixtures, parametric testing, and easy integration with CI pipelines. Coverage analysis was performed regularly to maintain high test coverage and identify untested code paths.

6.3. INTEGRATION TESTING

Integration tests focus on verifying that individual modules correctly cooperate within the larger QuakeSight system. Important integration scenarios covered include:

- Data acquisition feeding validated raw data into the processing pipeline and ensuring processed data is correctly stored in intermediate repositories.
- Analytics core modules accessing curated datasets and producing outputs in agreed formats consumable by downstream components.
- Workflow orchestration triggering analytics modules in the correct sequence, respecting dependencies and handling task failures gracefully.

- Communication between the API layer and analytics services using RESTful endpoints with authentication and rate limiting mechanisms.

Mock services and test doubles were employed where external systems or large infrastructure setups were unsuitable for testing environments. Containerized test setups allowed replicating the production environment with isolated databases and service instances, facilitating reliable integration test execution.

6.4. USER ACCEPTANCE TESTING (UAT)

Prior to production release, UAT sessions were conducted involving disaster management experts, geoscientists, data scientists, and emergency response stakeholders. These sessions aimed to:

- Validate that the system's outputs—such as forecasting visualizations, risk factor analyses, and prescriptive recommendations—align with stakeholder expectations and domain expertise.

- Gather feedback on usability, interface clarity, and feature completeness from operational and research perspectives.

- Identify gaps or enhancements required to improve real-world applicability and user experience.

Iterative cycles of feedback and adjustments were performed based on UAT findings, reinforcing the system's readiness and practical value for earthquake disaster management scenarios.

7. RESULT AND OUTPUT

The implementation and execution of the QuakeSight framework yielded comprehensive results across all integrated analytics layers—Descriptive, Diagnostic, Predictive, and Prescriptive. These results validate the system's capability to transform heterogeneous seismic and contextual data into actionable insights that enhance earthquake disaster management. This section provides detailed analytical summaries, visual representations, and practical interpretations of the system outputs generated through Jupyter notebooks and Streamlit dashboards, highlighting the efficacy of QuakeSight in forecasting seismic events, assessing risk, and recommending prescriptive actions.

Results In Jupyter

The Jupyter notebooks served as a flexible environment for exploratory data analysis, model development, and detailed result inspection. They facilitated stepwise examination of QuakeSight's outputs for researchers and technical users,

Extensive data analysis conducted within Jupyter notebooks showcased QuakeSight's ability to synthesize historical seismic data using statistical summaries and visualizations. For example, interactive plots revealed temporal clusters of earthquake occurrences and identified correlations between geological fault lines and seismic intensity. Machine learning model evaluation metrics—such as accuracy, precision, recall, and area under the ROC curve—were presented alongside confusion matrices, validating the predictive engine's performance in forecasting earthquake probability and approximate magnitudes.

These notebooks served as an exploratory platform to tune models before deployment, providing transparent analytics and reproducibility for researchers.providing both quantitative and visual insights. Key outcomes obtained through these notebooks include:

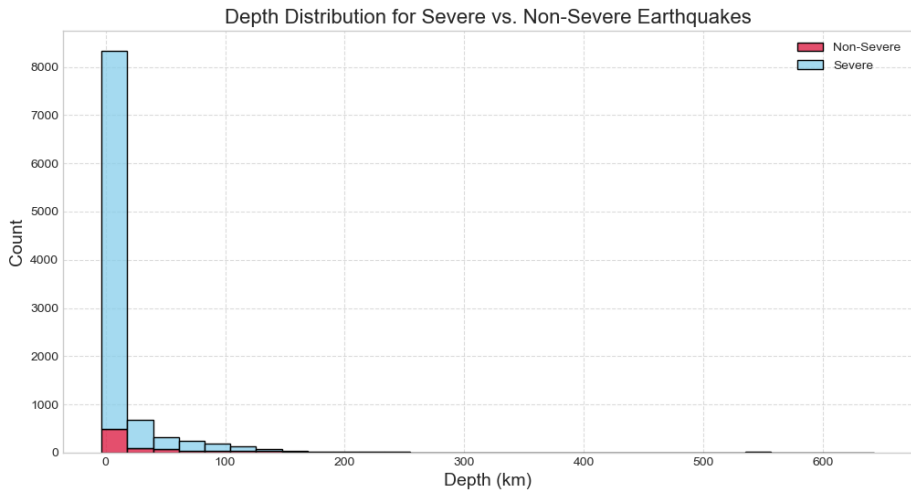

Figure 15 :Severe vs Non-Severe Earthquakes

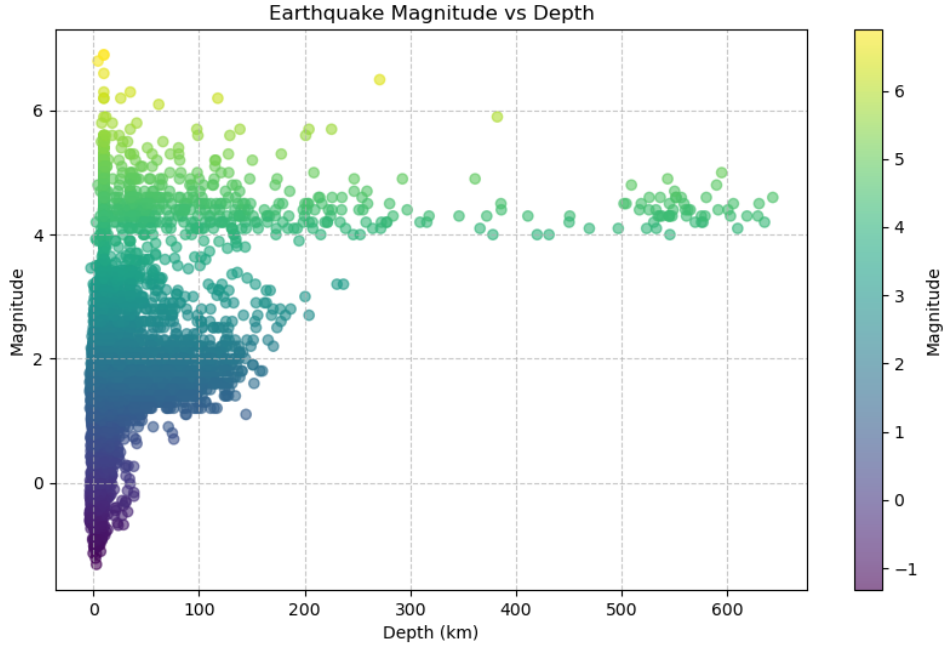

Figure 16 : Earthquakes Magnitude vs Depth

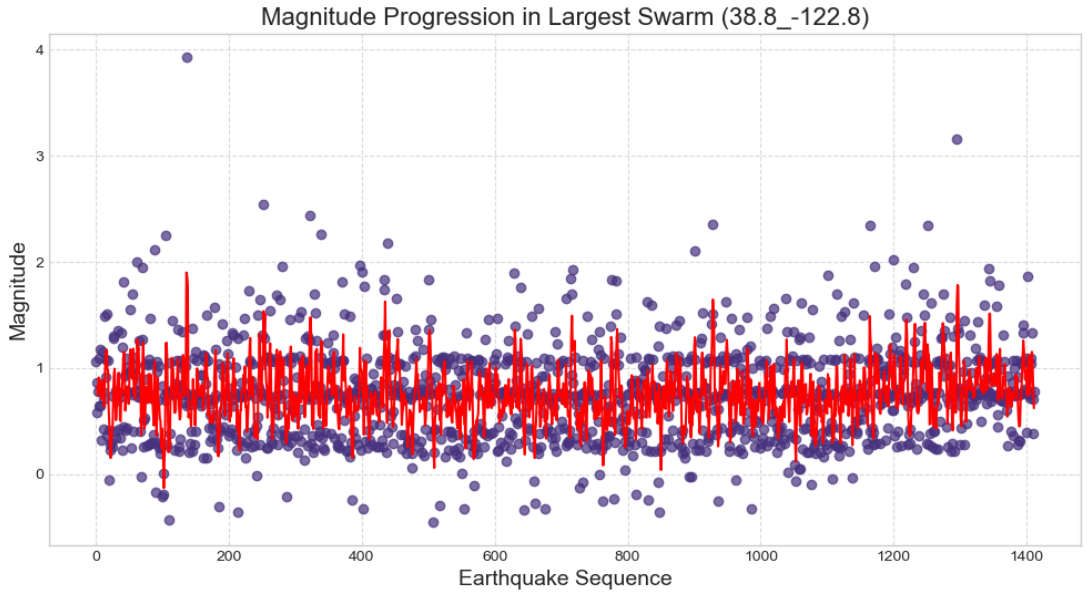

Figure 17:Magnitude Progression

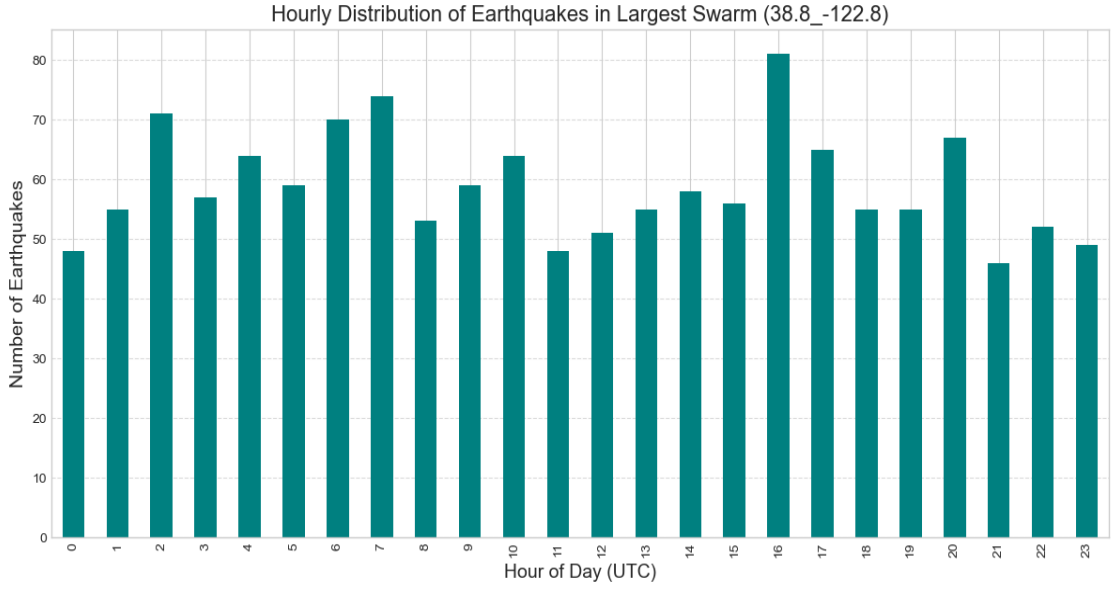

Figure 18:Hourly Distribution of Earthquakes

.

Results In Streamlit

To facilitate operational use by decision-makers and disaster managers, the analytical outputs were integrated into interactive Streamlit dashboards. These dashboards presented key findings and actionable insights via intuitive visualizations and controls designed for usability under time pressure

Figure 19: Opening UI of webapp

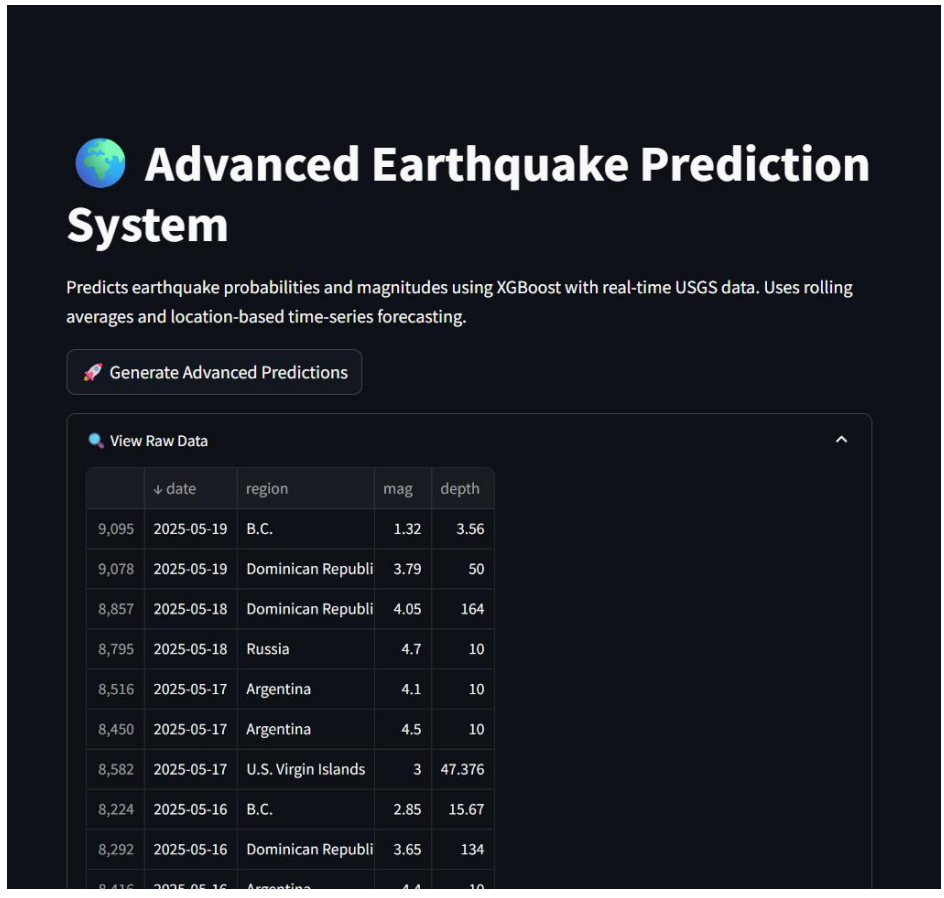

Figure 20: Raw data view after data ingestion

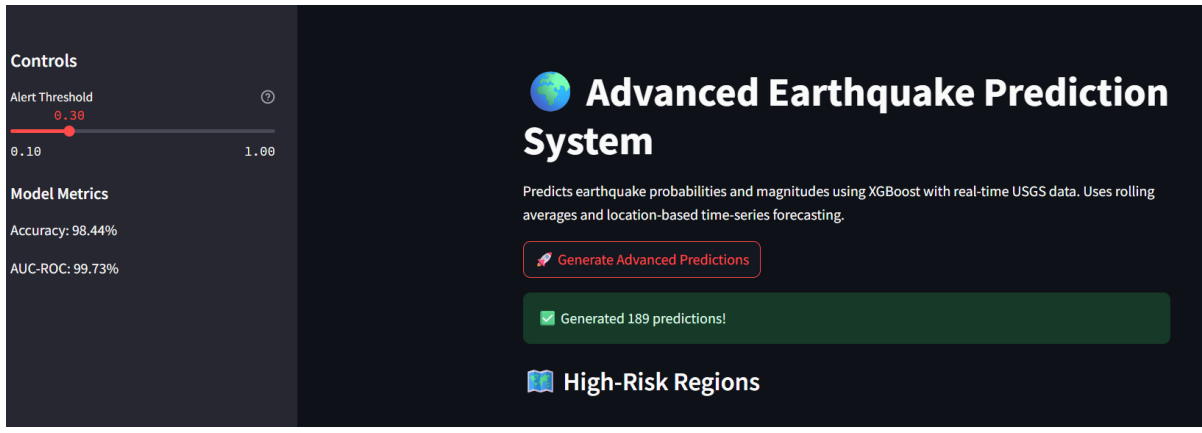

Figure 21: Generate Advanced predictions

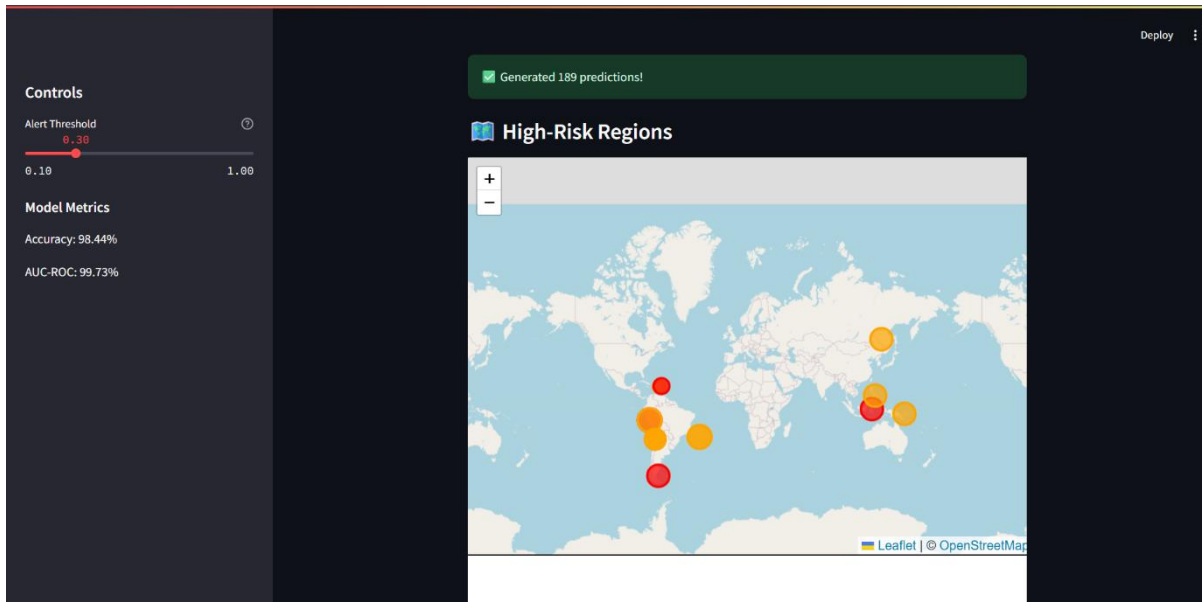

Figure 22: Model metrics in side bar and predicted map view

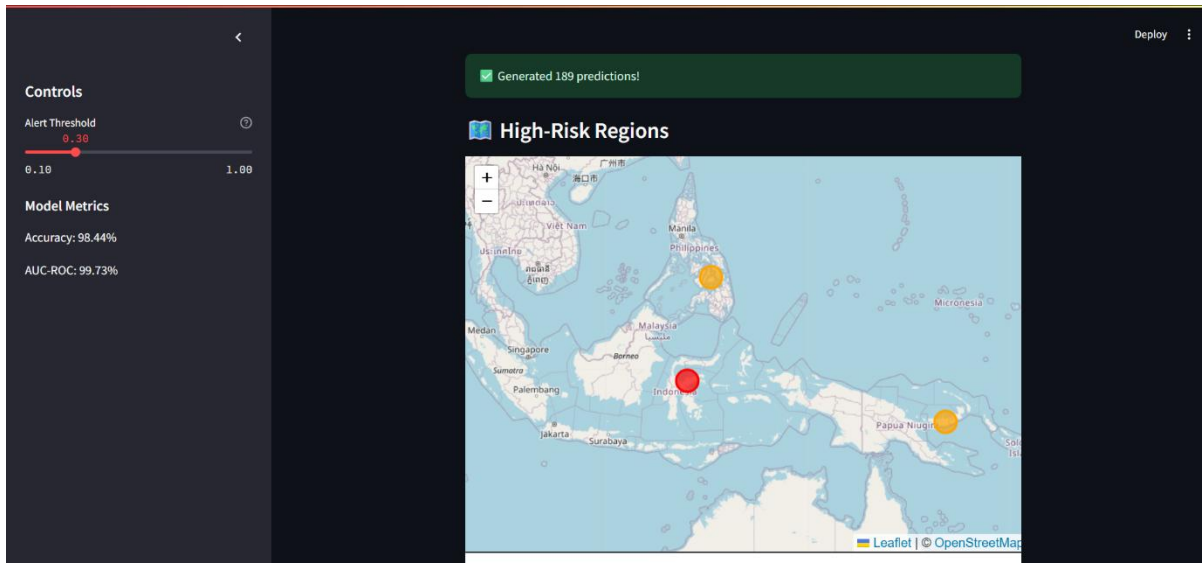

Figure 23: Map view zoomed

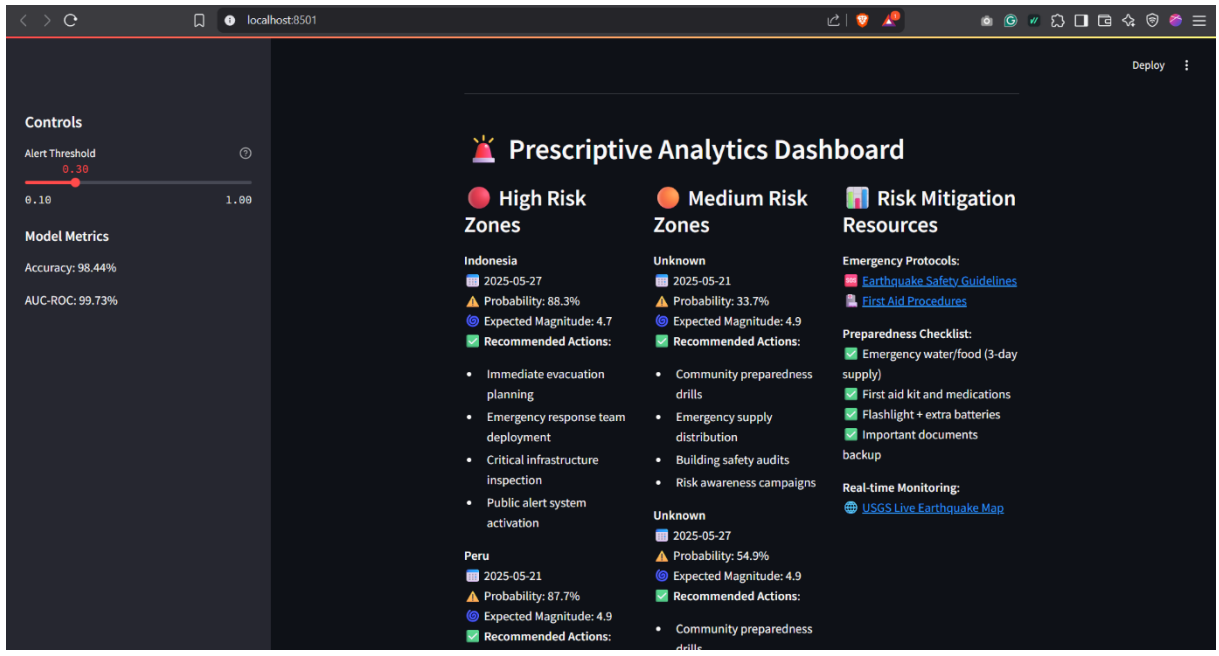

Figure 24: Prescriptive Analytics Dashboard

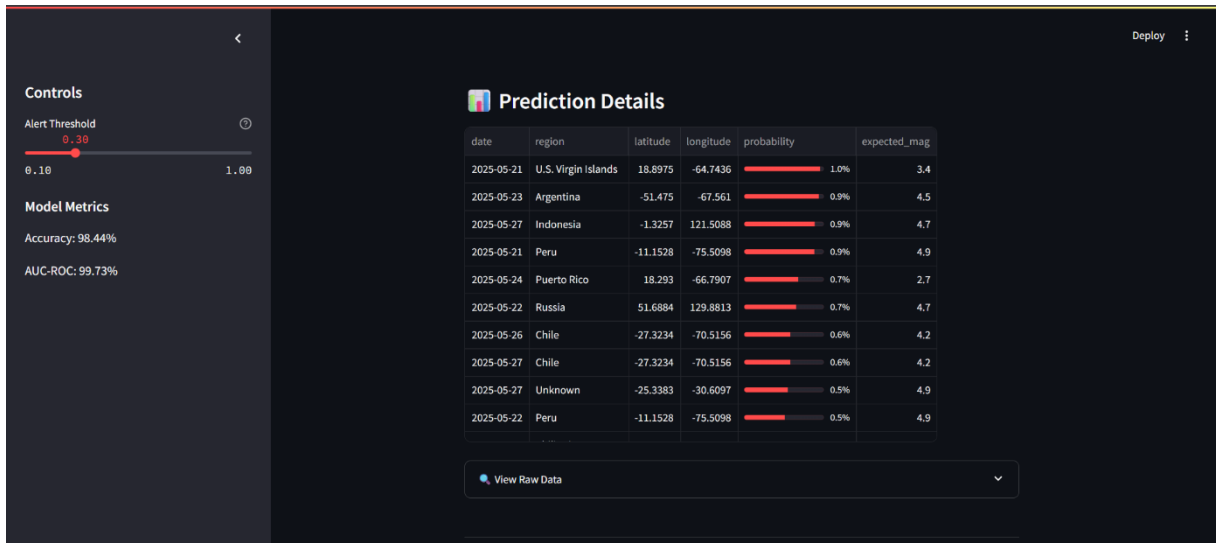

Figure 25: Prediction Details in tabular form

8. CONCLUSION AND FUTURE SCOPE

The QuakeSight framework successfully demonstrates the effective integration of descriptive, diagnostic, predictive, and prescriptive analytics within a cohesive platform to enhance earthquake disaster management. By systematically analyzing historical and real-time seismic data, uncovering underlying risk factors, forecasting seismic events, and generating actionable response strategies, QuakeSight provides a comprehensive decision-support tool that empowers stakeholders to move from reactive to proactive disaster preparedness. This integrated approach strengthens community resilience by enabling timely identification of vulnerable zones, early warnings of potential earthquakes, and optimized emergency resource allocation—thereby reducing potential casualties, infrastructural damage, and economic losses.

QuakeSight's unified analytics pipeline bridges critical gaps present in existing systems by delivering continuous, data-driven insights and recommendations that adapt dynamically as new seismic information is received. The platform's modular architecture, scalability, and real-time processing capabilities ensure responsiveness during seismic crises and facilitate cross-sector collaboration among emergency planners, government agencies, and community organizations.

Future Enhancements

Looking ahead, several promising avenues exist to further augment QuakeSight's capabilities. Integration of real-time sensor networks with increased deployment density will enhance the granularity and immediacy of seismic data acquisition. Expanding geographic coverage to include additional high-risk regions globally will broaden the framework's applicability and impact. Incorporating non-traditional data sources such as social media feeds and crowdsourced reports can enrich situational awareness and complement traditional sensor information with community-generated insights.

Advances in artificial intelligence and deep learning offer opportunities to improve predictive accuracy and model interpretability. For instance, deploying transformer-based architectures and multimodal learning approaches could better capture complex spatiotemporal seismic patterns and interactions with environmental and infrastructure variables. Additionally, embedding adaptive decision-making algorithms that learn from response outcomes can refine prescriptive recommendations over time, optimizing emergency procedures dynamically.

REFERENCES

1. Richter, C.F. (1935). An instrumental earthquake magnitude scale. Bulletin of the Seismological Society of America, 25(1), 1-32.
2. Gutenberg, B., & Richter, C.F. (1944). Frequency of earthquakes in California. Bulletin of the Seismological Society of America, 34(4), 185-188.
3. Omori, F. (1894). On the aftershocks of earthquakes. Journal of the College of Science, Imperial University of Tokyo, 7, 111-200.
4. Reid, H.F. (1910). The mechanics of the earthquake. The California Earthquake of April 18, 1906, Report of the State Earthquake Investigation Commission.
5. Utsu, T. (1961). A statistical study on the occurrence of aftershocks. Geophys. Mag., 30, 521–605.
6. Allen, C.R. (1978). Geological criteria for evaluating seismicity. Geological Society of America Bulletin, 89(3), 305–313.
7. Kanamori, H. (1977). The energy release in great earthquakes. Journal of Geophysical Research, 82(20), 2981- 2987.
8. Bolt, B.A. (1988). Earthquakes: A Primer. W.H. Freeman and Company.
9. Cornell, C.A. (1968). Engineering seismic risk analysis. Bulletin of the Seismological Society of America, 58(5), 1583–1606.
10. Seed, H.B., & Idriss, I.M. (1982). Ground motions and soil liquefaction during earthquakes. Earthquake Engineering Research Institute.
11. Boore, D.M., Joyner, W.B., & Fumal, T.E. (1997). Equations for estimating horizontal response spectra and peak acceleration from western North American earthquakes. Seismological Research Letters, 68(1), 128–153.
12. Joyner, W.B., & Boore, D.M. (1981). Peak horizontal acceleration and velocity from strong-motion records including records from the 1979 Imperial Valley, California, earthquake. Bulletin of the Seismological Society of America, 71(6), 2011–2038.
13. Wald, D.J., Quitoriano, V., Heaton, T.H., & Kanamori, H. (1999). Relationships between peak ground acceleration, peak ground velocity, and Modified Mercalli Intensity in California. Earthquake Spectra, 15(3), 557–564.
14. USGS Earthquake Catalog API. [https://earthquake.usgs.gov/fdsnws/event/1/]
15. OpenStreetMap Contributors. (2023). Global Population Density Maps. [https://www.openstreetmap.org]