Duale Hochschule Baden-Württemberg Mannheim

**Advanced Machine Learning**

# Earthquake Forecasting

## Studiengang Wirtschaftsinformatik

**Studienrichtung Data Science**


| | |
|---|---|
| Verfasser: | André Anan Gilbert, Felix Noll, Marc Grün |
| Matrikelnummern: | 3465546, 9467152, 9603221 |
| Kurs: | WWI 21 DSB |
| Modul: | Advanced Machine Learning |
| Kursleiter: | Prof. Dr. Maximilian Scherer |
| Bearbeitungszeitraum: | 06.05.2024 - 24.07.2024 |

# Contents

# List of Figures

# List of Source Code

# Acronyms

| | |
|---|---|
| **ARIMA** | Autoregressive Integrated Moving Average |
| **EMA** | Exponential Moving Average |
| **FDSN** | International Federation of Digital Seismograph Networks |
| **FFT** | Fast Fourier Transformation |
| **LLM** | Large Language Model |
| **LSTM** | Long Short-Term Memory |
| **MAE** | Mean Absolute Error |
| **MAPE** | Mean Absolute Percentage Error |
| **MSE** | Mean Squared Error |
| **IOT** | Internet of Things |
| **PACF** | Partial Autocorrelation Functions |
| **RMSE** | Root Mean Squared Error |
| **RNNs** | Recurrent Neural Networks |
| **SMA** | Simple Moving Average |
| **USGS** | United States Geological Survey |

# Abstract

Earthquakes are a common, yet sometimes catastrophic, phenomenon in our world. Approximately 2,000 earthquakes with a magnitude of 5 on the Richter scale occur annually (WELT, 2011). As this threat frequently endangers many lives, we aim to improve response time and enhance public understanding of earthquakes, particularly how to act in extreme situations. To achieve this, we have developed a CatBoost model to forecast the magnitude and depth of upcoming earthquakes for the 25 most vulnerable regions over the next three days. Our model achieves a Mean Squared Error (MSE) of 2.9344, a Root Mean Squared Error (RMSE) of 10.3577, and an R2-Score of 0.9495. Additionally, we have developed a Large Language Model (LLM)-powered AI agent, enabling user interaction with the model results and providing answers to general questions about earthquakes to improve awareness and in-depth analysis of potential risks and safety measures. This project contributes to the field of forecasting research by exemplifying the use case of a categorical boosted tree model to forecast earthquakes.

# 1 Introduction

Earthquakes are a common phenomenon in our world, with approximately 2,000 earthquakes of magnitude 5 on the Richter scale occurring annually (WELT, 2011). While these earthquakes are classified as moderate, they still have the potential to be deadly and cause significant damage to infrastructure and communities. The unpredictability of earthquakes poses a considerable challenge to disaster preparedness and risk mitigation.

To mitigate the risks of injuries or death, our goal is to develop a forecasting model capable of predicting when and where an earthquake might occur. Such a model would be invaluable in providing early warnings, allowing for timely preparation and potentially saving lives. By leveraging machine learning libraries and adapting their methodology to the use case of earthquake forecasting, we aim to build a model to enhance the accuracy and reliability of earthquake forecasts.

To further expand our web application, we plan to integrate an LLM. This LLM will not only offer easy access to the calculated predictive information but will also provide comprehensive insights into earthquakes. Our goal is to strengthen public understanding of earthquakes and promote safety precautions. This additional resource will serve as a tool, offering explanations on seismic activity, historical earthquake data, and guidelines for preparedness. Thus, we aim not only to develop an approach capable of predicting earthquakes but also to ensure long-term usage by the population in endangered areas.

This technology allows regions to prepare for potential earthquakes by providing enough time for preparation or evacuation. Early warnings from our predictive model can lead to better-organized emergency responses, reducing the impact of earthquakes on human lives and property. Our goal is to empower communities with predictive data and educational resources to enhance resilience against seismic hazards.

As a result, this paper is organized as follows: Chapter 2 explains forecasting and the theoretical background behind our approach in detail. Chapter 3 describes the development and deployment of our model and web application. Additionally, we evaluate the model's performance and feature importance. Finally, we interpret the results and outline potential improvements and next steps in Chapter 4.

# 2 Time Series Forecasting

Time series forecasting is a statistical technique used to predict future values based on previously observed values. This method is particularly valuable in fields, where understanding and anticipating trends and patterns over time is crucial, such as finance (Andersen et al., 2005), medicine (Topol, 2019), environmental science (Mudelsee, 2019), and biology (Stoffer & Ombao, 2012).

At its core, time series forecasting involves analyzing data points collected or recorded at specific time intervals. These data points can represent anything from daily stock prices and monthly sales figures to hourly weather readings and Internet of Things (IOT) sensor data. The primary objective is to use this historical data to make informed predictions about future events, trends, or behaviors (Zhang, 2001, ch. 1).

One of the key characteristics of time series data is that the observations are sequentially dependent. This temporal ordering introduces a unique challenge and opportunity: the potential to identify patterns such as seasonality, trends, and cycles (Assfalg et al., 2009). For example, retail sales data often exhibit seasonal patterns, with higher sales during holidays, while economic indicators might show long-term trends reflecting economic growth or decline.

This chapter aims to give a detailed overview about important terminology and time series forecasting models.

## 2.1 Terminology

To understand time series forecasting, we first need to define some terminology. A time series is a set of observations recorded over time (Haben et al., 2023), as shown in Figure 2.1.

| Timestamp | Feature_1 |
|---|---|
| 2000-04-01 | 100 |
| 2000-04-02 | 150 |
| 2000-04-03 | 170 |

Table 2.1: A simple example for a time series.

There are two unique features in time series analysis: time-step features and lag features (Haben et al., 2023). Time-step features indicate the interval between two timestamps, while lag features represent the difference in a feature metric compared to the previous timestamp (Table 2.2).

| Timestamp | Feature_1 | Time-step | Lag |
|---|---|---|---|
| 2000-04-01 | 100 | 0 | NaN |
| 2000-04-02 | 150 | 1 | 100 |
| 2000-04-03 | 170 | 2 | 150 |

Table 2.2: A simple time series extended with the time-step and lag feature.

Time step features allow for modeling time dependence. For example, if the time-step feature counts the days within the month, this can help identifying seasonal dependencies of *Feature_1* within the month (Table 2.2). On the other hand, lag features help identify serial dependence, meaning the correlation of current feature values with past feature values. Therefore, serial dependence can test the hypothesis whether past feature values influence current feature values.

Box et al. (2015) defined the term autocorrelation as a measure of correlation between the values of a time series at different time lags. A positive autocorrelation coefficient at a certain lag indicates that high values tend to follow high values and low values tend to follow low values, while a negative coefficient suggests that high values follow low values and vice versa (Box et al., 2015, ch. 2).

### 2.1.1 Trend

Another commonly used term in time series forecasting is the "trend". A trend describes a long-term change in the mean of the time series (Haben et al., 2023, ch. 5), as shown in Figure 2.1.

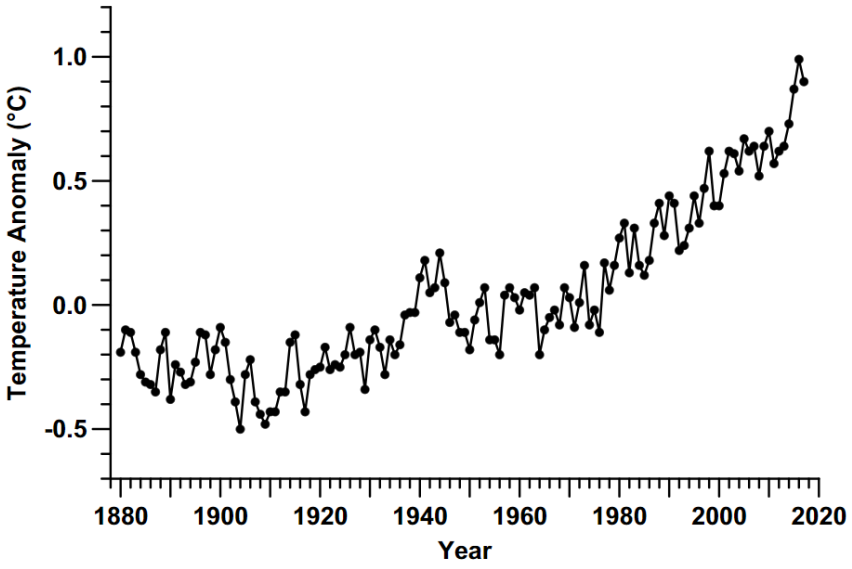


Figure 2.1: The time series of the temperature anomaly shows a clear upwards trend (Mudelsee, 2019, p. 311).

The trend is the slowest moving part of a time series, but it is essential to understanding movement patterns that influence the time series. A trend can be calculated in different ways. One common approach is the Simple Moving Average (SMA), which calculates the mean of values within a sliding window (Klinker, 2011). The larger the width of the sliding window, the slower the moving average adapts to (temporal) changes within the time series. Trends can be linear or quadratic for instance. A more sophisticated moving average is the Exponential Moving Average (EMA) which places greater significance on the most recent data points (Hansun, 2013). Unlike SMA, which assigns equal weight to all observations in the period, EMA applies a weighting factor that decreases exponentially (Klinker, 2011). This makes EMA more responsive to recent price changes, which is particularly useful in financial markets for analyzing trends and making trading decisions (Dzikevičius & Šaranda, 2010).

Besides computing the trend through a moving average, we can also try to identify it using a machine learning model, such as a linear regression. Using this trend, we can then forecast future feature values by extending the learned function representing the underlying trend within the data over future timestamps.

### 2.1.2 Seasonality

A recurring, periodic trend is often referred to as seasonality. A seasonality is said to occur when the mean of the series changes repeatedly in the same point in time within a given time range, e.g., year or month (Haben et al., 2023). Seasonality can be caused by cycles of nature or conventions of social behaviour surrounding dates or times. Simple examples for seasonality are the temperature within the year peaking in summer or the sales for toys peaking around Christmas (Haben et al., 2023).

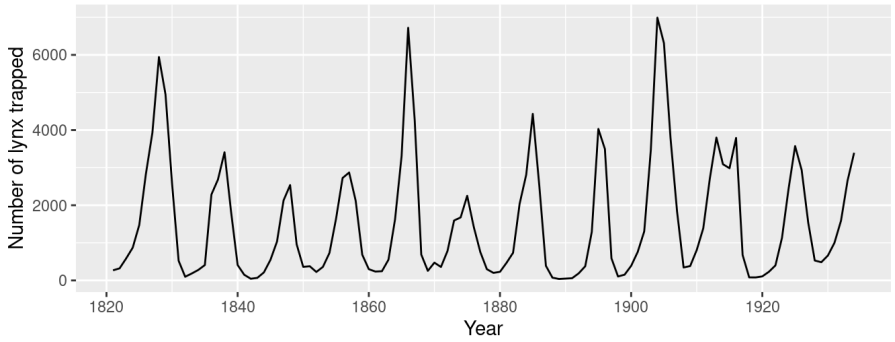


Figure 2.2: A time series showing a clear example of seasonality (Hyndman, 2011).

To identify seasonality, seasonal plots and indicators can be used. A seasonal plot displays segments of a time series against a consistent period, with the period representing the "season" you wish to examine. For instance, the values of a feature can be plotted with the weekday on the x-axis if a weekly seasonality is expected. Seasonal indicators can help a model distinguish

means within a seasonal period. To stick with the example of an expected seasonality within the week, the seasonal indicators can be the weekdays in a one-hot-encoded format (Table 2.3).

| Timestamp | Feature_1 | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|---|
| 2000-04-01 | 100 | 1 | 0 | 0 | 0 | 0 |
| 2000-04-02 | 150 | 0 | 1 | 0 | 0 | 0 |
| 2000-04-03 | 170 | 0 | 0 | 1 | 0 | 0 |
| 2000-04-04 | 180 | 0 | 0 | 0 | 1 | 0 |
| 2000-04-05 | 90 | 0 | 0 | 0 | 0 | 1 |

Table 2.3: A simple time series with seasonal indicators for the weekdays.

Note, that we can also model seasonality using sine and cosine curves to capture multiple seasonalities within a time series. The technique which leverages these curves to transform a time series into its constituent frequencies is called Fourier Transformation (Bloomfield, 2004, ch. 1). By decomposing a time series into a sum of sine and cosine functions, it enables the identification and characterization of various cyclical patterns within the data. Fast Fourier Transformation (FFT) is an efficient algorithm that computes the Fourier Transformation quickly, making it feasible to apply this method to large datasets in real-time applications (Bloomfield, 2004, ch. 5). This approach allows for a more nuanced understanding and forecasting of time series data by isolating and analyzing recurring patterns across different temporal scales.

### Cyclic Time Series

Another notable mention are cyclic time series, a type of data sequence characterized by periodic fluctuations that occur over irregular intervals, often spanning several years. Unlike seasonal variations, which follow a regular, predictable pattern within a year, cyclic patterns are influenced by broader economic, social, or environmental factors and do not have a fixed period (Gharehbaghi & Lindén, 2017, p.4102). Possible reasons for those cycles are long-term influences such as business cycles, economic expansions and recessions.

## 2.2 Forecasting

Time series forecasting is a statistical method used to predict future values based on previously observed data points (Box et al., 2015, ch. 5). While there are methods such as Autoregressive Integrated Moving Average (ARIMA) or exponential smoothing (Box et al., 2015, ch. 5), one can also use machine learning algorithms to examine the dependencies and relationships between different time points, in order to learn from the historical data to predict future values.

Generally, these machine learning algorithms optimize their model parameters to minimize forecasting errors (Box et al., 2015, ch. 5). Which parameters they optimize is dependent on the underlying model. Popular models will be covered in Chapter 2.4.

Typical examples for those forecasting erros are: the Mean Absolute Error (MAE), which calculates the average absolute difference between the predicted values and the actual values; the MSE, which computes the average of the squared differences between the predicted and actual values (squaring the errors penalizes larger errors more significantly, making MSE sensitive to outliers); the RMSE which is the square root of the MSE, providing an error measure in the same units as the original data and the Mean Absolute Percentage Error (MAPE) which expresses the error as a percentage of the actual values, which can be useful for comparing forecast accuracy across different datasets or scales. Especially for regressions, a common metric is the R-squared (R2) Score, which indicates how well the model's predictions approximate the actual data compared to a simple mean prediction. It ranges from 0 to 1, where 1 indicates a perfect fit. Moreover, the R2 Score can also be adjusted to account for the number of predictors in the model, providing a more realistic evaluation of model fit. This metric is then referred to as the adjusted R2 Score.

## 2.3 Feature Engineering

Feature engineering is crucial in timeseries forecasting, involving techniques such as partial autocorrelation and autocorrelation analysis to identify relevant lags. Autocorrelation measures how current values in the series relate to past values, while partial autocorrelation isolates the relationship of specific lags, which helps in the selection of relevant features for the model (Box et al., 2015).

Further, one can also create new features such as lagged variables, which are created by shifting the original timeseries data by one or more periods. The lags can be created heuristically e.g. 1, 2 and 12 months. Consequently, Partial Autocorrelation Functions (PACF) can be used to identify significant lags that should be included as features (Hyndman & Athanasopoulos, 2018, ch. 6). These lagged variables can be combined with rolling windows which calculate statistics (e.g. mean, standard deviation, sum, etc.) over a specific timeframe. After creating lagged variables, the rolling windows can be used to calculate rolling statistics on the features and capture more complex patterns (Hyndman & Athanasopoulos, 2018, ch. 6).

Additionally, it might be useful to include binary or numerical features representing the occurrence of significant external events, e.g., holidays or economic shifts (Hyndman & Athanasopoulos, 2018, ch. 3). Furthermore, live data potentially correlated with the predicted value can be incorporated as features.

## 2.4 Forecasting Models

Regarding time series forecasting models, one can distinguish between global and local models which differ fundamentally in their scope and application (Montero-Manso & Hyndman, 2021). Local models focus on individual time series, creating separate models for each dataset. This approach allows for highly customized forecasts tailored to the specific patterns and characteristics of each series, often yielding high accuracy for single series predictions. However, local models can be resource-intensive, requiring significant computational power and expertise to develop and maintain multiple models. In contrast, global models aggregate multiple time series into a single model, leveraging shared patterns and trends across datasets. This approach can be more efficient, as it reduces the need for multiple models and can improve scalability. Global models are particularly effective when individual series exhibit similar behaviors, allowing the model to learn from a broader context. However, they might not capture the unique nuances of each series as effectively as local models. Ultimately, the choice between local and global time series forecasting depends on the specific requirements of the application, including the number of series to forecast, computational resources, and the need for individualized predictions (Montero-Manso & Hyndman, 2021).

| **Model type** | **Can be used for** |
|---|---|
| linear regression | local |
| ARIMA | local |
| ETS | local |
| tree | local/global |
| RNN | local/global |
| LSTM | local/global |
| transformer based | global |

Table 2.4: Timeseries Forecasting Models

There are various algorithms with which a time series forecasting problem can be solved (Salinas et al., 2020). Regression algorithms, such as linear and polynomial regression, are foundational techniques that model the relationship between independent variables and the target variable, providing interpretable and straightforward forecasts. Tree-based methods, including decision trees, random forests, and gradient boosting machines, are powerful for handling complex, non-linear relationships and interactions among features, often delivering robust performance with less need for extensive data preprocessing. Recurrent Neural Networks (RNNs) excel in time series forecasting due to their ability to maintain memory of previous inputs, making them suitable for capturing temporal dependencies. Long Short-Term Memory (LSTM) networks, an advanced form of RNNs, further enhance forecasting capabilities by addressing the vanishing gradient problem, enabling the model to learn and retain

long-term dependencies more effectively. Each of these algorithms can be tailored to specific forecasting challenges, with selection depending on factors such as data characteristics, desired interpretability, and computational resources.

Additionally, deep learning frameworks have significantly advanced time series forecasting by offering sophisticated methods for modeling complex patterns and dependencies. Frameworks like DeepAR (Salinas et al., 2020) and DeepVAR (Cheng et al., 2020) are notable examples. DeepAR utilizes autoregressive recurrent neural networks to provide probabilistic forecasts, excelling in scenarios with large datasets by learning from the historical data of all time series in the dataset, thereby capturing shared patterns and improving forecast accuracy. DeepVAR extends this approach by incorporating multivariate time series data, allowing the model to understand and leverage the interdependencies between multiple series for more nuanced predictions. Recently, zero-shot time series forecasting with LLMs such as Chronos (Ansari et al., 2024) has emerged as a cutting-edge approach. Chronos leverages pre-trained LLMs to forecast time series without the need for task-specific training, utilizing the model's ability to generalize from vast amounts of data. This approach can significantly reduce the time and computational resources required for model development, making it an attractive option for rapid deployment in diverse forecasting scenarios.

## Hybrid Models

Hybrid models are advanced Models consisting of at least two different models. By combing those two different models the strengths of each model are leveraged and the weaknesses are mitigated. This works by training the first model on the original series and training the second model on the residual of the first Model. At the end both predictions have to be added in order to get the final prediction. Key beeing that the different models focus on different features. If the first model learns to predict the trend, the second model does not need a trend feature. Some models consists out of four different Models, where the models learn to predict the trend, seasonality (on a yearly basis), seasonality (on a weekly basis) and cycles (Zunic et al., 2020). Model ensembles that add up results from different underlying models are also called additive models.

A popular example of such an additive model is Facebook Prophet by Taylor and Letham (2018). The ensemble consists out of multiple models where each one predicts something different. The first model predicts the trend, which could be piecewise linear (allowing for multiple changepoints, in which the growth rate can change) or logistic. The second model accounts for the seasonality component (yearly, weekly or daily) using the Fourier series (see chapter 2.1.2). Thirdly, one of the models accounts for holiday effects, as holidays can cause significant deviations from the usual patterns. The user can input a list of holidays and the model will include their effects. Moreover, Prophet can include additional regressor effects if

the user provides external factors that are believed to affect the time series. This results in the equation:

$$y(t) = g(t) + s(t) + h(t) + \epsilon(t)$$

Where $y(t)$ is the observed value at time $t$, $g(t)$ is the trend component, $s(t)$ is the seasonality component, $h(t)$ is the holiday effect and $\epsilon(t)$ is the error term, capturing noise and other unexplained variability (Taylor & Letham, 2018).

# 3 Application

In this chapter, we describe how we leveraged the time series forecasting techniques described in the previous chapter to predict earthquakes. We wrap our forecasting model in a web application and enrich the dynamic dashboard with an AI assistant to enable human-like interactions with the forecasting results.

Together, the dashboard and AI assistant aim at improving our ability to forecast earthquakes and prepare for their consequences. This chapter explores the functionalities, underlying technologies, and real-world applications of these tools, highlighting their potential to revolutionize earthquake forecasting and enhance public safety.

Within the web app there are two main workflows, the first one being an overview and analysis of recent earthquakes and the second being a conversation with a copilot to support insights and decision making.

## 3.1 Modeling Problem

As earthquakes are not equally distributed over the surface of the earth, we decided to forecast earthquakes for the 25 most endangered regions on the planet. This facilitates the forecasting process while only marginally reducing our use case. To fulfill our goal of warning people about potential dangers we effectively need to forecast the magnitude, depth and location of an upcoming earthquake. A reliable prediction of these three features would enable users to evaluate their situation and take preliminary action if needed.

However, we decided to only forecast magnitude and depth, while leveraging historical median values for the location. This is due to most of the earthquakes occurring at the edges of tectonic plates, which leads to similar location values for earthquakes within a given region. By removing the location values as a forecasting target the modeling problem becomes computationally more efficient.

### 3.1.1 Data

To forecast, we utilized data from the United States Geological Survey (USGS) Earthquake Catalog, accessed via the International Federation of Digital Seismograph Networks (FDSN) Event Web Service. This service provides comprehensive information on seismic events using various parameters such as time, location, depth, and magnitude. We queried the catalog for earthquakes from the past 100 years and due to the limited data prior to 1970 we settled on

using the past 30 years worth of data for training. Given our focus on building an end-to-end machine learning application, we opted to use this free API, which allows us to forecast future earthquakes based on live data. This approach closely resembles a production system, enhancing the realism and practicality of our forecasting model. The dataset, available in formats such as GeoJSON, XML, and CSV, was instrumental in analyzing seismic activity trends and developing the forecasting model.

The dataset contained detailed information about the time, location (latitude and longitude), depth and magnitude of an occurring earthquake. Additionally, there are other features available on the API. A detailed list can be found on the USGS website.[1]

### Preprocessing

Effective preprocessing is critical for handling the unique characteristics of time series data, particularly in earthquake forecasting. In our study, we began by addressing the unevenly spaced timestamps in the dataset. To standardize the intervals, we opted for daily aggregation. While we also experimented with hourly aggregation, it resulted in prohibitively long model training times - ranging from 12 hours to a full day - making daily aggregation the more practical choice.

Our analysis revealed a strong time dependency for most values, extending up to 40 lags and potentially more in certain regions. This suggests that extensive historical data would be beneficial for predictions. However, incorporating such long historical windows significantly increases dimensionality, making the model complex and computationally expensive. We question the practicality of using nearly every recorded value in a region for forecasting future earthquake occurrences, especially given that some regions have lags spanning multiple months or even years.

To address the dimensionality challenge, we capped the historical lags and the exponential moving average at 7 days into the past. This compromise ensured that we incorporated sufficient historical information while keeping the model computationally feasible. Balancing the need for historical data without overwhelming the model with too much information was crucial for our approach.

Our analysis of partial autocorrelation and autocorrelation functions revealed that only the first few lags were significantly correlated in some regions. This finding indicated that adding more lags beyond this point would not improve predictive accuracy but would instead introduce unnecessary complexity. Thus, capping historical lags at 7 days further validated our approach, focusing model training on the most relevant temporal dependencies for effective earthquake prediction in those specific regions.

[1] https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php

Given that we are conducting global time series forecasting, these measures are sufficient and appropriate. By capping the historical lags at 7 days, we successfully balanced the need for historical information with the necessity of maintaining a computationally feasible model. This strategy, backed by our autocorrelation analysis, ensures that our model remains both efficient and accurate in predicting earthquake occurrences on a global scale.

Furthermore, our analysis of depth and magnitude periodograms indicated that earthquakes exhibit a high degree of randomness, with some seasonality detected in only a minority of regions. Due to this predominant randomness, using Fast Fourier Transform (FFT) to model seasonality proved impractical. Consequently, our preprocessing did not rely on FFT-based seasonal adjustments.

For model training, we implemented an 80-20 train-test split for each region. This split was time-based, ensuring that past data was used to predict future values. This method aligns with the temporal nature of the data and the forecasting goal, allowing the model to learn from historical trends and apply this knowledge to future predictions.

### 3.1.2 Model

To forecast earthquakes for 25 regions accurately, we employ a global forecasting model. Among the various approaches available, tree-based models and RNNs, specifically LSTM networks, are considered due to their ability to handle time-series data effectively. For this particular application, we use CatBoost regression (Prokhorenkova et al., 2018), an efficient gradient boosting framework.

We chose CatBoost regression for forecasting earthquakes across 25 regions due to its handling of categorical features, which simplifies preprocessing and ensures robust feature encoding. CatBoost also offers better interpretability, allowing us to understand the impact of different variables on predictions. Its training efficiency and built-in mechanisms to prevent overfitting make it a practical choice, particularly with the volatile nature of earthquake data. Additionally, CatBoost can capture complex feature interactions without extensive feature engineering, streamlining the modeling process and enhancing overall accuracy.

While tree-based models often struggle with forecasting trends, CatBoost regression is particularly well-suited for our task for several reasons. The magnitude of earthquakes is capped within a range of [-1, 10], and their depth falls within [-100, 1000] meters (US Geopolitical Survey, 2024). In regression trees, predictions are made by traversing a series of if-else conditions to reach a leaf node, where the average of the values in that leaf are used for the prediction. Due to the nature of these trees, it is not possible to predict values outside of those observed in the training set. This makes CatBoost, with its advanced handling of

categorical features and robust performance on unseen data, an excellent choice for this type of regression problem.

CatBoost transforms categorical features into numerical ones by employing several methods, including random permutation of input objects and label value conversion to integers, tailored to the type of machine learning problem (regression, classification, or multi-classification). It calculates the transformation using various statistics, such as counting occurrences within specific buckets or calculating mean target values. The primary formula for this transformation is:

$$\text{ctr} = \frac{\text{countInClass} + \text{prior}}{\text{totalCount} + 1}$$

where *countInClass* is the relevant label count, *prior* is a predefined constant, and *totalCount* is the cumulative count of objects with the same categorical value. Additionally, CatBoost supports one-hot encoding for categorical features with limited unique values, controlled by the *one_hot_max_size* parameter (CatBoost, 2024).

To forecast earthquakes, we predict magnitudes and depths, while for location (latitude and longitude), we compute the historic median to reduce model complexity and enhance stability. The forecasting process is performed recursively or iteratively: initially, we forecast the data for one day, using the predicted values to create features for the subsequent day. This process is repeated, building on each previous prediction, until forecasts for three days are generated. This recursive approach allows the model to adapt dynamically to the changing conditions and trends observed in the time series data, providing more accurate and reliable forecasts.

### Training

The model training process for earthquake forecasting requires a structured approach, beginning with a careful feature selection. The chosen features include temporal attributes such as the day, day of the week, and day of the year, which capture the time-dependent aspects of earthquake occurrences. Additionally, exponential moving averages for both magnitude and depth are included to reflect the trends over recent periods. The model also incorporates lagged values for magnitude and depth over a specified range, allowing it to consider past events' influence on current conditions. The inclusion of the categorical feature region ensures that the model can account for regional differences in earthquake behavior.

The chosen model for this task is the CatBoost Regressor, selected for its ability to handle categorical features natively and its strong performance with complex datasets. The model is configured with early stopping rounds to mitigate overfitting and is optimized using the MultiRMSE loss function, which targets minimizing errors in both magnitude and depth predictions.

The training dataset comprises the selected features and the categorical region feature, while the target variables are the earthquake magnitude and depth.

To enhance the model's performance, hyperparameter tuning is performed using grid search. This involves testing various combinations of hyperparameters such as learning rate, model depth, and L2 leaf regularization. The grid search evaluates these combinations to identify the optimal parameters that yield the best predictive performance. This systematic approach to feature selection, data preparation, and hyperparameter tuning aims to develop a robust and accurate model capable of forecasting earthquake magnitude and depth across different regions.

| **Hyperparameter** | **Values** |
|---|---|
| Learning Rate | 0.01, 0.03, **0.1** |
| Depth | 4, 6, **10** |
| L2 Leaf Regularization | 1, 3, **5**, 7, 9 |

Table 3.1: Hyperparameter values explored during grid search.

### Evaluation

After training the CatBoost Regressor model for earthquake forecasting, it is crucial to evaluate its performance using several metrics and visualization techniques. The performance of the time series forecasting model is assessed using a variety of metrics that provide a comprehensive view of its accuracy and reliability. The selected metrics are the MAE, the RMSE, the R2-Score and the adjusted R2 Score. These metrics collectively offer insights into the error magnitude and the goodness-of-fit of the model. The rationale behind these metrics is explained in 2.2.

| **Metric** | **Score** |
|---|---|
| MAE | 2.9344 |
| RMSE | 10.3577 |
| R2 | 0.9495 |
| Adjusted R2 | 0.9495 |

Table 3.2: Model metrics.

For our model, the MAE is 2.9344, indicating that on average, the predictions are approximately 2.9344 units away from the actual values. Lower MAE values suggest better predictive accuracy, and the obtained MAE value demonstrates that the model has a reasonably low level of error.

The RMSE for our model is 10.3577, which reflects the model's error magnitude. While RMSE values are generally higher than MAE due to the squaring of errors, they provide an important perspective on how significant the larger errors are in the context of the model's predictions. A lower RMSE indicates a better fit, and our RMSE value suggests a reasonable performance with some sensitivity to larger errors.

For our model, the R2 is 0.9495, signifying that approximately 94.95% of the variance in the data is explained by the model. This high R2 value suggests that the model fits the data very well and is capable of capturing the underlying patterns in the time series.

The Adjusted R2 for our model is 0.9495, which is very close to the R2 value. This indicates that the inclusion of predictors in the model is justified and that the model does not overfit the data. The slight difference between R2 and Adjusted R2 confirms that the model maintains a high level of explanatory power while accounting for the number of predictors.

The evaluation of the model using these metrics provides a well-rounded understanding of its performance. The low MAE and RMSE values indicate that the model has a low level of error, while the high R2 and Adjusted R2 values demonstrate a strong fit to the data. Together, these metrics suggest that the model is both accurate and reliable for forecasting the time series data, making it a valuable tool for predicting future values.

In Figure **??**, the magnitude forecasts demonstrate varying degrees of accuracy across the different regions. For regions like Alaska and Nevada, the model's predictions closely follow the actual magnitudes, indicating a high level of accuracy. However, in regions such as Indonesia and Chile, there are noticeable deviations between the forecasted and actual values, suggesting areas where the model could be improved. Overall, while the model captures the general trends and fluctuations in earthquake magnitudes reasonably well, the precision varies significantly by region.

Figure **??** focuses on the depth forecasts. Similar to the magnitude forecasts, the depth predictions show regional variability in accuracy. In regions such as Alaska, California, and Nevada, the forecasted depths align closely with the actual depths, indicating strong predictive performance. Conversely, in regions like Indonesia and Papua New Guinea, the forecasted depths show more significant discrepancies from the actual values, highlighting potential areas for model enhancement.

Combining the insights from both sets of forecasts, it is evident that the model's performance is influenced by regional characteristics. The regions with higher accuracy in both magnitude and depth forecasts, such as Alaska and Nevada, suggest that the model effectively captures the underlying patterns in these areas. In contrast, regions with lower accuracy, such as Indonesia and Chile, may require additional data or model adjustments to improve predictive performance.

While the model's evaluation metrics, including MAE, RMSE, R2, and Adjusted R2, indicate excellent performance in terms of accuracy and goodness-of-fit, it is important to acknowledge a significant limitation: the inherent unpredictability of earthquakes. Earthquakes are fundamentally random events, much like predicting stock market values, making them exceptionally challenging to forecast with high precision. Despite the model's robust statistical indicators, the chaotic and complex nature of seismic activities means that accurate prediction remains elusive. Thus, while the model shows promising results according to the chosen metrics, its practical applicability in earthquake forecasting is constrained by the unpredictable and stochastic nature of the phenomenon itself.

The feature importance plot displayed in Figure **??** highlights the significance of various input features in a forecast model that predicts the magnitude and depth of earthquakes. Notably, the feature *depth_ewma* (exponential moving average of depth) stands out as the most critical feature by a considerable margin, indicating its paramount influence on the model's predictions. Following *depth_ewma*, features such as *depth_lag_1*, *depth_lag_2*, and *depth_lag_3* also demonstrate significant importance, suggesting that recent depth values have a substantial impact on the forecast. In contrast, features related to the earthquake magnitude, such as *mag_ewma* (exponential moving average of magnitude) and various magnitude lags, show much lower importance, indicating a lesser role in the prediction model. Additionally, temporal features like *dayofyear* and *dayofweek* have minimal impact. This analysis underscores that the recent history of earthquake depth, especially its exponential moving average, is the most influential factor in forecasting future earthquake magnitudes and depths.

## 3.2 LLM-powered AI Agent

Advancements in AI have given rise to AI agents as powerful tools for enhancing productivity and communication (Xi et al., 2023). Equipped with LLMs, these agents can perform a wide array of tasks, such as information processing, creative text generation, language translation, and providing insightful responses to inquiries.

Unlike conventional virtual assistants and chatbots, AI agents operate based on learned patterns and data, enabling them to offer more nuanced understanding and communication. For instance, an AI agent trained on extensive datasets of text and code can retrieve pertinent articles, distill key findings into summaries, or craft tailored content based on the preferences of specific target audiences. The potential applications of LLM-powered AI agents are vast and diverse, ranging from automating repetitive tasks to supporting complex decision-making processes and creative endeavors. However, it is crucial to recognize that while AI agents excel in language manipulation and generation, they lack true sentience or consciousness, necessitating responsible usage to mitigate the risk of biased or misleading content generation.

### 3.2.1 Instructions

The AI agent uses a combination of tools to deliver accurate and timely earthquake forecasts. The agent is built using the ReAct prompting (Yao et al., 2023) method as shown in Source Code 3.1, which enhances its interaction capabilities and allows it to provide expert-level responses regarding earthquakes.

```
SYSTEM_PROMPT = """You are an United States Geological Survey
expert who can answer questions regarding earthquakes and can
run forecasts.

Before you use the Forecast Earthquakes tool, always check
which regions are available using Find Regions first.
Respond to the user as helpfully and accurately as possible.

You have access to the following tools:
{tools}

Please ALWAYS use the following JSON format:
{{
  "thought": "Explain your thought. Consider previous and
    subsequent steps",
  "tool": "The tool to use. Must be on of {tool_names}",
  "tool_input": "Valid keyword arguments (e.g. {{"key": value}})",
}}

Observation: tool result
... (this Thought/Tool/Tool input/Observation can repeat N times)

When you know the answer, you MUST use the following JSON format:
{{
  "thought": "Explain the reason of your final answer
    when you know what to respond",
  "tool": "Final Answer",
  "tool_input": "Valid keyword arguments (e.g. {{"key": value}})",
}}"""
```

Source Code 3.1: system_prompt.py

ReAct prompting is a structured approach used to guide AI agents in providing accurate and helpful responses in multi-step problems (Yao et al., 2023). The agent is programmed with a specific system prompt that positions it as a United States Geological Survey (USGS) expert. This prompt includes a format for the agent's thoughts, tool usage, and observations, ensuring

that the agent follows a clear and logical process in its interactions.

Even though we define a specific output format, the LLM still predicts the most likely token that should come next, which can lead to errors in the output format, such as responding with text instead of JSON. This process can be compared to a trial-and-error learning process. We then enter a loop of refinement. If the LLM's output can't be parsed into the correct format, it's reminded of the requirement and prompted to adjust. Conversely, a successful parse leads to the LLM's output being used to call a relevant tool. If the tool call fails, the LLM is asked to refine it further. This cycle continues until a satisfactory answer is produced, either through a parsable output or the LLM exhausting its attempts. This iterative process ensures continuous feedback and correction, ultimately aiming to achieve the desired outcome: a correct response in the specified format.

### 3.2.2 Tools

The AI agent designed for earthquake forecasting integrates several tools to provide accurate and comprehensive information. These tools, leveraging forecasting models and data processing techniques, enhance the agent's capability to predict and analyze seismic events. Here's a detailed look at the tools used by the AI agent:

1. **Current Date:** This tool provides the current local date and time. It is essential for timestamping queries and ensuring that all data processing is aligned with the date the user asked about.
2. **Query Earthquakes:** This tool searches for recent earthquakes based on various parameters such as start time, end time, depth, magnitude, and alert level. It utilizes the US Geological Survey (USGS) API to retrieve data in the geojson format.
3. **Count Earthquakes:** This tool counts and aggregates recent earthquake events based on specified criteria. It uses the same parameters as the Query Earthquakes tool to filter the events.
4. **Find Regions:** This tool retrieves the available regions for which earthquake forecasts can be made. It ensures that the LLM is aware of the regions supported by the forecasting model before making predictions.
5. **Forecast Earthquakes:** This tool predicts future earthquakes in a specified region. It processes the forecast data to return predictions including the date, magnitude, and depth of potential earthquakes.

## 3.3 Deployment and Monitoring

The first step in deploying our ML application involved containerizing the model using Docker.[2] Docker allowed us to package the model along with its dependencies into a container, ensuring consistency across different deployment environments. After saving the model, we created the Dockerfile and the Docker Image to install dependencies and run the code.

For deploying the Flight Base App on Render.com, the team optimally utilized the platform's versatile features. Render.com uses Amazon Web Services (AWS) under the hood, which provided the team with a solid infrastructure and worldwide availability through the AWS data center in Frankfurt. The app deployment is automatic as soon as changes are pushed to the team's GitHub repository and the deployment branch is synchronized.

To handle the model's computational demands, especially for real-time predictions, deploying the Docker container on a cluster with GPU support would be ideal. Although we did not have access to a GPU-supported cluster for this project, it is a practical approach when high performance is needed. The deployment would typically be managed through a fully orchestrated workflow. In practice, the training and deployment workflow are fully automated, which can be implemented using Argo Workflows.[3] However, this was out of scope for this project.

Additionally, when deploying models in practice, using a platform like Kubeflow[4] is beneficial. Kubeflow leverages Kubernetes for machine learning and allows you to write templates defining how to deploy models, specifying resources such as RAM, CPU, and GPU, as well as managing canary rollouts and other deployment strategies.

The management of environment variables is handled directly through Render.com. These are automatically inserted during the build of the Docker container and are available to the application. Through this approach, the team ensured that the entire deployment configuration remains clear and efficient, while also ensuring a smooth live deployment of the application on Render.com. This method minimizes the risk of misconfigurations and simplifies the management of sensitive data such as API keys.

Furthermore, effective monitoring of the deployed model is crucial to ensure that it continues to perform as expected. Our monitoring process includes checking for model drift, both data drift and concept drift. Model drift occurs when the statistical properties of the target variable (concept drift) or the input features (data drift) change over time, leading to a decline in model performance. An example for a concept drift in earthquake forecasting would be a change in the relationship between seismic precursor signals (like foreshocks, ground deformation, or gas emissions) and the occurrence of an earthquake due to new geological activities or shifts

[2] https://www.docker.com/
[3] https://argoproj.github.io/workflows/
[4] https://www.kubeflow.org/

in tectonic plate interactions. Data drifts would include a change in sensor calibration, which would lead to a shift in the data they collect. Additionally, the performance metrics of the model on live data are monitored. Consequently, the model is retrained if the performance metrics decline below a certain threshhold.

In summary, while this project did not employ advanced orchestration tools or GPU-supported clusters due to constraints, these are critical considerations for deploying robust and scalable machine learning models in a production environment.

# 4 Conclusion

To conclude, the project was successfully completed and its goals were achieved. We preprocessed the data and trained a robust model, which was then used to build a comprehensive application. This application features a dashboard displaying recent earthquakes and an interactive map showing future earthquakes for the next three days worldwide. Given the extensive volume of the presented data, which might seem overwhelming, we developed an AI Agent to increase accessibility. This agent allows users to run personalized forecasts tailored to their specific needs. Since most users are interested in one or two regions rather than all 25 regions which are covered by the forecast, this feature is particularly useful. Additionally, the AI Agent acts as a USGS expert, providing users with in-depth analysis on potential risks and safety precautions for individual earthquakes.

Recognizing that many people are not tech-savvy and aiming to spread information about earthquakes to a broad audience, we minimized technical barriers. By integrating an interactive AI Agent, we made it easier for users to access and understand crucial earthquake information.

To enhance the predictive accuracy of earthquake forecasting models, several potential improvements can be considered. Integrating live data such as magma flow, plate tectonics movements, and real-time seismic activity could provide a more comprehensive and dynamic understanding of the underlying processes that precede earthquakes. Additionally, exploring advanced deep learning models, such as RNNs and LSTMs, which are well-suited for time series data, might offer improved prediction capabilities by capturing complex temporal dependencies. Furthermore, incorporating multi-source data fusion techniques, which combine various types of geological and environmental data, could enhance the robustness of the model. Continuous updates and retraining of the model with the latest data, along with cross-disciplinary collaboration between seismologists and data scientists, would also contribute to refining the forecasting accuracy. These improvements could collectively help in better anticipating seismic events, despite the inherent challenges in predicting such random and chaotic phenomena.

# A Benchmarks Prompt Examples

# Bibliography


Andersen, T. G., Bollerslev, T., Christoffersen, P., & Diebold, F. X. (2005). Volatility forecasting.

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., et al. (2024). Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*.

Assfalg, J., Bernecker, T., Kriegel, H.-P., Kröger, P., & Renz, M. (2009). Periodic pattern analysis in time series databases. *Database Systems for Advanced Applications: 14th International Conference, DASFAA 2009, Brisbane, Australia, April 21-23, 2009. Proceedings 14*, 354–368.

Bloomfield, P. (2004). *Fourier analysis of time series: An introduction*. John Wiley & Sons.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control*. John Wiley & Sons.

CatBoost. (2024). *Transforming categorical features to numerical features* [Accessed: 15-06-2024]. https://catboost.ai/en/docs/concepts/algorithm-main-stages_cat-to-numberic

Cheng, C., Tan, F., & Wei, Z. (2020). Deepvar: An end-to-end deep learning approach for genomic variant recognition in biomedical literature. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(01), 598–605.

Dzikevičius, A., & Šaranda, S. (2010). Ema versus sma usage to forecast stock markets: The case of s&p 500 and omx baltic benchmark. *Business: Theory and Practice*, *11*(3), 248–255.

Gharehbaghi, A., & Lindén, M. (2017). A deep machine learning method for classifying cyclic time series of biological signals using time-growing neural network. *IEEE transactions on neural networks and learning systems*, *29*(9), 4102–4115.

Haben, S., Voss, M., & Holderbaum, W. (2023). Time series forecasting: Core concepts and definitions. In *Core concepts and methods in load forecasting: With applications in distribution networks* (pp. 55–66). Springer.

Hansun, S. (2013). A new approach of moving average method in time series analysis. *2013 conference on new media studies (CoNMedia)*, 1–4.

Hyndman, R. J. (2011). Cyclic and seasonal time series. https://robjhyndman.com/hyndsight/cyclicts/

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.

Klinker, F. (2011). Exponential moving average versus moving exponential average. *Mathematische Semesterberichte*, *58*, 97–107.

Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, *37*(4), 1632–1653.

Mudelsee, M. (2019). Trend analysis of climate time series: A review of methods. *Earth-science reviews*, *190*, 310–322.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. *Advances in neural information processing systems*, *31*.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, *36*(3), 1181–1191.

Stoffer, D. S., & Ombao, H. (2012). Special issue on time series analysis in the biological sciences.

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, *72*(1), 37–45.

Topol, E. J. (2019). High-performance medicine: The convergence of human and artificial intelligence. *Nature medicine*, *25*(1), 44–56.

US Geopolitical Survey. (2024). *Api documentation - earthquake catalog* [Accessed: 15-06-2024]. https://earthquake.usgs.gov/fdsnws/event/1/

WELT. (2011). Naturkatastrophe: Schwere erdbeben sind nicht häufiger als früher. *WELT*. https://www.welt.de/wissenschaft/article12811266/Schwere-Erdbeben-sind-nicht-haeufiger-als-frueher.html

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., et al. (2023). The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). React: Synergizing reasoning and acting in language models. https://arxiv.org/abs/2210.03629

Zhang, G. P. (2001). An investigation of neural networks for linear time-series forecasting. *Computers & Operations Research*, *28*(12), 1183–1202.

Zunic, E., Korjenic, K., Hodzic, K., & Donko, D. (2020). Application of facebook's prophet algorithm for successful sales forecasting based on real-world data. *arXiv preprint arXiv:2005.07575*.