Chapter 14

Earthquake Predictions using Time Series Analysis

Nikhil Raj[1*] **Sadhana Tiwari**[2]

Abstract

As of late, there have been huge headways in utilizing man-made brainpower (simulated intelligence) to foresee quakes. Specialists at The College of Texas at Austin fostered an artificial intelligence calculation that had the option to foresee 70% of tremors seven days before they happened during a preliminary in China. The man-made intelligence was prepared to identify factual examples in seismic information and effectively anticipated 14 tremors inside a 200-mile range of their assessed area with practically careful strength. The scientists intend to additionally test the computer-based intelligence in Texas and at last coordinate it with material science-based models to make a summed-up framework that can be applied anyplace on the planet.

One more way to deal with tremor expectation is utilizing the force of time series examination. An exploration paper named "Disentangling Seismic Tremor Forecast: A Period Series Examination Approach" proposes a one-of-a-kind and imaginative way to deal with quake expectations utilizing time series investigation. The review means to add to the improvement of more exact and solid tremor forecast models by uncovering stowed-away examples inside seismic time series information. The proposed philosophy includes procuring broad seismic time series datasets incorporating different locales and levels of seismic action, trailed by thorough preprocessing and highlight designing to remove significant data. Best-in-class time series ex-

[1,2] Sharda School of Business Studies, Sharda University, Gr. Noida, India
[1]Corresponding Author ✉ 2022302135.nikhil@pg.sharda.ac.in

© QTanalytics® 2023
Prabhat Mittal and Rajeev Bansal (eds), Social Development and Governance: Innovations in Education, Technology & Management (ISBN: 978-81-966500-9-4)
https://doi.org/10.48001/978-81-966500-9-4_14

amination procedures, including autoregressive models, Fourier changes, and repetitive brain organizations, are then applied to uncover fleeting conditions and patterns inside the information. Consolidating geospatial data, land information, and natural factors further improves the models' prescient power.

The fundamental goal of the exploration is to foster a prescient system fit for assessing tremor probabilities throughout various time skylines. The aftereffects of this exploration might offer important experiences for early advance notice frameworks, catastrophe readiness, and chance alleviation procedures, eventually lessening the effect of seismic occasions on living souls and foundations. The review plans to overcome any issues between information-driven science and geophysical investigation, making ready for another period in seismic tremor expectation.

Keyword Earthquake Predictions, Earthquake Predictions using Time Series, Disentangling Seismic Tremor Forecast

14.1 Introduction

A seismic tremor happens when the world's outside goes through unexpected development because of the sudden arrival of aggregated pressure along land shortcomings on the inside. This delivered energy goes through the Earth as seismic waves, which are low-recurrence sound waves, starting development. Indeed, even after the issue movement stops, seismic waves continue going through the Earth.

Investigation into seismic tremor expectation has been in progress for nearly a hundred years. In the event that effective, the capacity to foresee the time, area, and greatness of a tremor might actually save lives and forestall huge monetary misfortunes concerning lodging and foundation. Be that as it may, accomplishing fruitful tremor expectations remains incredibly uncommon.

Tremor expectations fall into two essential classifications: long haul estimates, spreading over months to years, and transient forecasts, happening inside the space of hours or days. Long-haul conjectures depend on different exploration strategies, for example, breaking down verifiable quake designs in unambiguous districts, examining shortcoming qualities like length, profundity, and division, and surveying strain aggregation. These investigations offer surmised assessments of quake sizes and the time spans between seismic occasions. A striking illustration of a drawn-out tremor estimate includes distinguishing seismic holes, which are segments of plate limits that haven't encountered a significant quake for a lengthy period. These regions are viewed as high-risk destinations for critical seismic tremors from here on out.

Momentary quake forecast stays a critical test, and no technique has been shown to be solid. Because of the complicated and eccentric nature of the quake cycle, there is a developing conviction that accomplishing exact transient forecasts might be intrinsically inconceivable. As innovation has progressed and organized frameworks have become broad, significant volumes of geographic information have been and keep on being amassed through present-day information-securing strategies like worldwide situating frameworks Global Positioning System (GPS), satellites, high-goal remote detecting, area-mindful administrations, overviews, and chipped-in geographic data open on the web. Subsequently, there is a rising interest in instruments and advances able to effectively investigate these broad logical datasets, principally pointed toward deciphering the fundamental actual peculiarities.

Time series information is consistently gathered at characterized time stretches from different frameworks, including everyday securities exchange cost changes, vacillations in gold costs, securities exchange varieties, and yearly populace development in a country. A period series comprises of a significant number of distinct perceptions organized all together in view of equivalent time or spatial spans. Information that is sporadic or happens just once doesn't meet the standards for time series information. Regularly, a noticed time series is separated into three parts:

1. **The Occasional Part:** This reflects efficient or normal developments in the information.
2. **The Pattern Part:** This shows long-haul changes in the information.
3. **The Sporadic Part:** This obliges unsystematic or transient variances in the information

Time series models track down their principal use in factual determining. Different expectation strategies, like relapse, time series, and tumultuous techniques, are accessible, each with its own arrangement of assets and shortcomings. For determining future qualities, authentic successions of information are used, with time series models anticipating what will happen without fundamentally giving a clarification as to why it works out.

To create expectations, nonlinear techniques are utilized to change time series values into stage space, and fluffy rationale is then applied to conjecture ideal qualities. Time series information is obtained from different frameworks at ordinary time spans. Regular fixed time series models incorporate Autoregressive integrated moving average (ARIMA) and Least Mean Square blunder-determining techniques. Information mining is applied to extricate significant and relevant data from broad data sets. For expectation purposes, man-made reasoning and example acknowledgement techniques are utilized.

Time series information mining is used to figure quakes, variances in financial exchanges, weather conditions, and changes in gold costs. Fluffy rationale strategies demonstrate particularly productive in foreseeing occasions like tremors and changes in the financial exchange, weather patterns, and gold costs. Fluffy sets, distinguished in light of comparable examples and characterized inside participation capabilities, empower precise expectations of future occasions utilizing fluffy rationale. Time series information digging is also applied for bunching and anticipating regular occasions.

The essential advantage of time series examination is its ability to estimate future qualities by looking at authentic information. The examination of past groupings of authentic information offers significant experiences, helping with the forecast of future successions. Time series strategies are important for pattern examination, estimating in exchange markets, applications in money, climatology, and seismic tremor expectation.

14.2 Literature Review

Preethi & Santhi (2011) showed that in geosciences and economics, time series projections are more significant. Data mining and time series analysis are combined in time series data mining. strategies. Among the historical information gathered are the following combining the time series approach with data mining preprocessing, followed by prediction based on fuzzy logic principles. Earthquake prediction has been done by examining the strategy for utilizing past earthquake time data. Taking a step back first, preprocessing large data collections is done with data mining methods. Using this method, data projections are able. Soft computing and statistics are the main topics of this study methods for examining the seismic data.

Amei et al. (2012) reviewed that between 1896 and 2009, it is believed that earthquakes with a magnitude of 8.0 or higher on the Richter scale follow a Poisson process. To predict the occurrence of these significant earthquakes, several Autoregressive Integrated Moving Average (ARIMA) models have been introduced. These models are fine-tuned using time series data of Empirical Recurrence Rates (ERRs), and the most recent five or ten data points are employed for predictive evaluation. The most appropriate model forecasts a total of 12 major earthquakes occurring worldwide in the next 6 years. The use of ERR-based ARIMA models for long-term earthquake prediction not only acts as a link connecting point processes and traditional time series analysis but also broadens the application of statistical methods for predicting various natural disasters.

Shah et al. (2013) studied overviews within the domain of neural networks, computer scientists are increasingly focusing on studying the behaviours of social insects to address a range of intricate combinatorial and statistical challenges. A notable example of this emerging trend is the adoption of the Artificial Bee Colony (ABC) algorithm. This research delves into the practical application of the ABC algorithm, which replicates the intelligent foraging behaviour observed in honey bee swarms. The conventional process of training a Multilayer Perceptron (MLP) using the backpropagation algorithm often involves computationally demanding procedures. One significant hurdle associated with the backpropagation (BP) algorithm is its tendency to generate neural networks with less-than-optimal weight configurations, primarily due to the presence of numerous local optima in the solution space.

To address this challenge, the study employs the ABC algorithm to train the MLP in understanding the complex patterns found in earthquake time series data, as an alternative to the standard BP approach. The experimental findings underscore that MLP-ABC outperforms MLP-BP when it comes to working with time series data, highlighting its superior performance.

Otari & Kulkarni (2012) provided that catastrophic events emerge when regular dangers, similar to floods, cyclones, typhoons, volcanic ejections, quakes, heatwaves, or avalanches, lead to complex actual occasions. These occasions, including seismic tremors, avalanches, tidal waves, and volcanoes, can prompt huge monetary, natural, and human misfortunes. Foreseeing these geographical calamities is of most extreme significance, however, an intricate cycle relies upon different physical and ecological elements.

Intending for this test, mainstream researchers have explored different logical and factual techniques. Furthermore, information mining procedures have arisen as important instruments for anticipating normal perils. This paper presents an exhaustive assessment of the utilization of information mining in gauging land fiascos, exploring 16 diary articles distributed somewhere in the range of 1989 and 2011 on this point.

The essential information-digging strategies utilized for quake expectation incorporate calculated models, brain organizations, Bayesian conviction organizations, and choice trees. These techniques present promising roads for anticipating quakes, waves, avalanches, and other miniature-seismic occasions. Besides, the paper means to move further exploration in this field and closes by proposing headings for future examination attempts.

Lyubushin (1999) introduced a method for detecting synchronized signals in multidimensional time series data. It relies on estimating eigenvalues of spectral matrices and calculating canonical coherence within moving time windows. The key step involves deriving an aggregated signal, a single scalar signal designed to capture spectral components that are present concurrently across all individual scalar time series. It is acknowledged that an increase in the coordinated behaviour of components within specific systems and the widening spatial extent of parameter fluctuations can serve as a critical indicator of an impending catastrophe, such as a sudden shift in the system's parameter values. From this perspective, the identification of synchronized signals among diverse geophysical parameters recorded at multiple points within a network covering a specific region of the Earth's crust becomes an essential tool for identifying precursors to significant earthquakes. The practical application of this method to real geophysical time series data is illustrated with examples presented in the paper.

Morales et al. (2010) showed that earthquakes strike suddenly, posing a rapid and severe threat to entire cities, resulting in substantial loss of life and significant economic ramifications.

Currently, extensive efforts are in progress to develop techniques aimed at predicting these unpredictable natural disasters and implementing preventive measures. This study utilizes clustering methods to extract patterns that effectively capture the behaviour of temporal seismic data, ultimately assisting in the forecasting of medium to large earthquakes. In the initial phase, earthquakes are categorized into distinct groups, with the optimal number of these groups, initially unknown, determined through the process. Subsequently, the study identifies patterns associated with the occurrence of medium to large earthquakes. The research presents and discusses its findings based on temporal seismic data from Spain, provided by the Spanish Geographical Institute. These results are further substantiated and confirmed through non-parametric statistical tests, underscoring the remarkable performance and significance of the outcomes derived from the clustering approach.

Ali et al. (2017) talked about estimating time series information is a typical test crossing different logical disciplines, with a long history of exploration around here. Various strategies are accessible for getting ready time series information, and one prominently viable methodology is the utilization of wavelet procedures. These strategies have exhibited their capacity to resolve issues connected with information-lopsided characteristics emerging from anomalies and clamour. In this review, an original model called the Wavelet Multilayer Perceptron (W-MLP) is presented. This model integrates wavelet strategies to preprocess time series information prior to taking care of it into the Multilayer Perceptron (MLP). The model has gone through preparing and testing for the expectation of tremor information in California. The reenactment results relating to seismic tremor time series estimating highlight the W-MLP's better exhibition when looked at than the conventional Multi-facet Perceptron (MLP) regarding expectation precision.

Barkat et.al. (2018) showed that the utilization of time series investigation on soil radon information has been recently proposed for seismic tremor peril expectation, despite the fact that it has not acquired all-inclusive acknowledgement for this reason. In this ongoing review, we play out a period series examination of soil radon information gathered along a functioning issue zone in North Pakistan, meaning to investigate pre-seismic tremor peculiarities during the period from July 24, 2014, to April 30, 2015. Our approach includes geochemical investigation of soil, deterministic investigation utilizing the Hurst type (H), measurement of meteorological impacts, and assessment of soil radon abnormalities with regards to quake anticipating. In particular, in the examination of radon information peculiarities, we apply lingering signal-handling strategies to moderate the customary impacts of meteorological variables. Moreover, we utilize a measurable rule (x ± 2σ) at a 95% certainty stretch. The aftereffects of geochemical examination recommend that any strange expansion in soil radon fixation isn't related to the presence of key radionuclides, for example, 226Ra, 232Th, and 40K. Deterministic examination of radon and meteorological boundaries uncovers that the Hurst example (H) falls inside the scope of 0.5 H 1, showing a persevering pattern with negligible rarity and inconsistency. Essentially, the impact of meteorological boundaries on soil radon is measured through relationship coefficients, recommending a minor effect.

Besides, the fleeting changes in leftover radon around the hour of quake movement demonstrate six critical atypical pinnacles that surpass the measurable basis during the analyzed period. The absence of strange lingering radon conduct for specific quake occasions in the review period can be made sense of by their low extent and high RE/RD esteem. To summarize, our outcomes avow past examination and backing the use of radon as a mark of seismic movement.

14.3 Objective

14.3.1 Ensemble Learning for Robust Predictions

Random Forest Regressor, as an ensemble learning technique, combines the predictive power of multiple decision trees. This approach enhances the model's robustness by mitigating overfitting and capturing complex relationships within earthquake data. The ensemble nature of Random Forest ensures more reliable predictions, making it a suitable choice for earthquake prediction where diverse and intricate patterns may exist.

14.3.2 Handling Multidimensional Data

Earthquake prediction involves analyzing a multitude of factors, such as geological, seismological, and environmental variables. Random Forest Regressor is well-suited for handling multidimensional datasets and can effectively process a large number of features. This capability allows researchers to incorporate various input parameters, improving the model's ability to discern subtle patterns and correlations contributing to earthquake occurrences.

14.3.3 Non-linear Relationship Detection

Earthquake prediction often involves understanding non-linear relationships between input features and the target variable. Random Forest Regressor excels in capturing complex, non-linear patterns present in earthquake data. Unlike linear regression models, which may struggle to represent intricate relationships, the Random Forest's ability to model non-linearities enhances the accuracy of earthquake predictions.

14.3.4 Resilience to Outliers and Noise

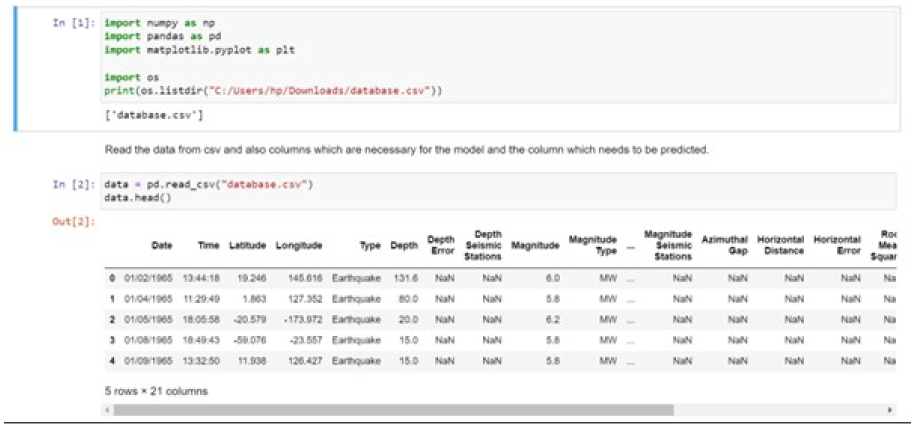
Earthquake datasets may contain outliers or noisy observations that can adversely affect the performance of predictive models. Random Forest Regressor exhibits resilience to outliers and noise due to its aggregation of multiple decision trees. By averaging predictions across the ensemble, the impact of individual outliers is minimized, resulting in a more robust and reliable earthquake prediction model.

14.3.5 Feature Importance and Interpretability

Understanding the contributing factors to earthquake occurrence is crucial for effective risk mitigation and disaster preparedness. Random Forest Regressor provides a measure of feature importance, allowing researchers to identify which variables have the most significant impact on predictions. This interpretability aids in gaining insights into the underlying mechanisms of earthquake generation and can inform targeted interventions and monitoring strategies.

14.4 Data Ananlysis

- **Date:** The date when the seismic event occurred.



```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        import os
        print(os.listdir("C:/Users/hp/Downloads/database.csv"))

        ['database.csv']
```

Read the data from csv and also columns which are necessary for the model and the column which needs to be predicted.

```
In [2]: data = pd.read_csv("database.csv")
        data.head()
Out[2]:
```

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	...	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Error	Roo Mea Squar
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	...	NaN	NaN	NaN	NaN	Na
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	NaN	Na
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	...	NaN	NaN	NaN	NaN	Na
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	NaN	Na
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	NaN	Na

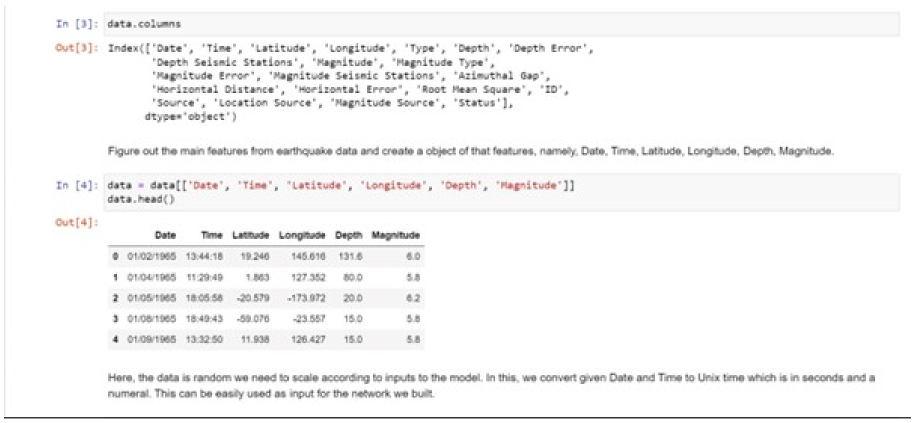
5 rows × 21 columns

- **Time:** The time of day when the seismic event occurred.
- **Latitude:** The geographic coordinate specifying the north-south position of the earthquake epicentre.
- **Longitude:** The geographic coordinate specifying the east-west position of the earthquake epicentre.
- **Type:** The type of seismic event (e.g., earthquake).
- **Depth:** The depth below the Earth's surface at which the seismic event occurred.
- **Depth Error:** The margin of error associated with the depth measurement.
- **Depth Seismic Stations:** The number of seismic stations that contributed to the depth measurement.
- **Magnitude:** The magnitude of the seismic event, a measure of its size or energy release.
- **Magnitude Type:** The method or scale used to determine the magnitude (e.g., Richter scale, Moment Magnitude Scale).
- **Magnitude Error:** The margin of error associated with the magnitude measurement.

- Magnitude Seismic Stations: The number of seismic stations that contributed to the magnitude measurement.
- **Azimuthal Gap:** The angular gap between seismic stations that are used to locate an earthquake, providing information about the reliability of the location.
- **Horizontal Distance:** The horizontal distance from the epicenter to a location on the Earth's surface.
- Horizontal Error: The margin of error associated with the horizontal distance measurement.
- **Root Mean Square (RMS):** A measure of the consistency of observed and predicted values.
- **ID:** An identifier for the seismic event.
- Source: The organization or network responsible for reporting the seismic event.
- **Location Source:** The source that provided the location information for the seismic event.
- **Magnitude Source:** The source that provided the magnitude information for the seismic event.
- **Status:** The status of the seismic event (e.g., reviewed, automatic).

data=data [['Data', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']] By using this code, we are selecting specific columns so that we can get precise results and work on the data that is useful. Another method to do this is PCA (Principal Component Analysis)

As our data was so huge there were so many dates of earthquakes that were not in according to our desired format, so we manipulated the interpreter in converting different formats into similar ones.



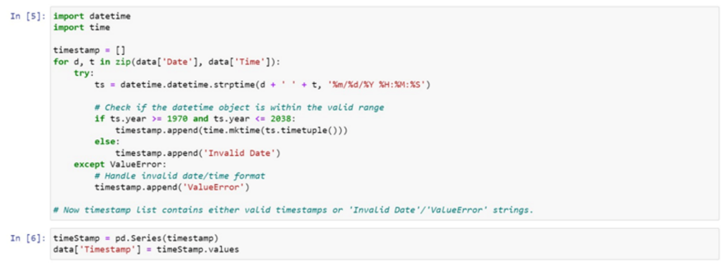
```
In [3]: data.columns

Out[3]: Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
               'Depth Seismic Stations', 'Magnitude', 'Magnitude Type',
               'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
               'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
               'Source', 'Location Source', 'Magnitude Source', 'Status'],
              dtype='object')
```

Figure out the main features from earthquake data and create a object of that features, namely, Date, Time, Latitude, Longitude, Depth, Magnitude.

```
In [4]: data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]
        data.head()

Out[4]:
```

	Date	Time	Latitude	Longitude	Depth	Magnitude
0	01/02/1965	13:44:18	19.246	145.616	131.6	6.0
1	01/04/1965	11:29:49	1.863	127.352	80.0	5.8
2	01/05/1965	18:05:58	-20.579	-173.972	20.0	6.2
3	01/08/1965	18:49:43	-59.076	-23.557	15.0	5.8
4	01/09/1965	13:32:50	11.938	126.427	15.0	5.8

Here, the data is random we need to scale according to inputs to the model. In this, we convert given Date and Time to Unix time which is in seconds and a numeral. This can be easily used as input for the network we built.

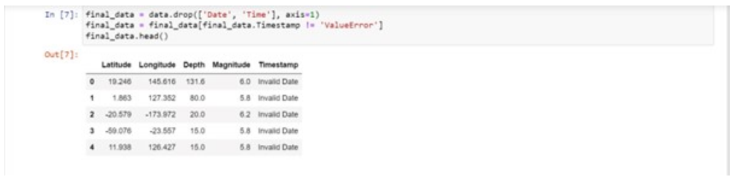


```
In [5]: import datetime
        import time

        timestamp = []
        for d, t in zip(data['Date'], data['Time']):
            try:
                ts = datetime.datetime.strptime(d + ' ' + t, '%m/%d/%Y %H:%M:%S')

                # Check if the datetime object is within the valid range
                if ts.year >= 1970 and ts.year <= 2038:
                    timestamp.append(time.mktime(ts.timetuple()))
                else:
                    timestamp.append('Invalid Date')
            except ValueError:
                # Handle invalid date/time format
                timestamp.append('ValueError')

        # Now timestamp list contains either valid timestamps or 'Invalid Date'/'ValueError' strings.

In [6]: timeStamp = pd.Series(timestamp)
        data['Timestamp'] = timeStamp.values
```

This is a plot of all the earthquakes that took place over a period of 50 years across the world map.

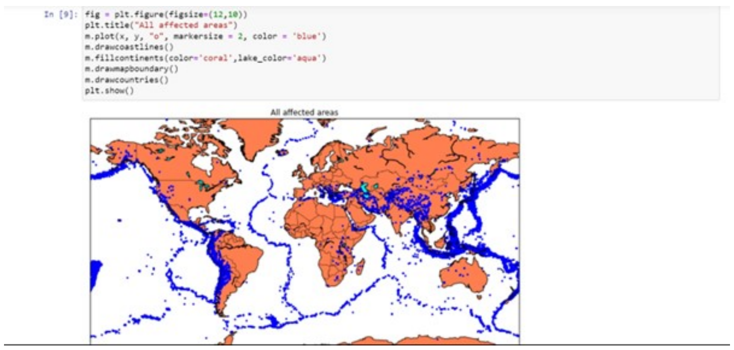
Total sizes 23413 and we have split the data in 80% training set and 20% testing dataset which means that we will be training our model on 18713 entries and then verify the performance or accuracy of the model on the remaining approx. 4500 entries.

A Random Forest Regressor from scikit-learn is utilized to develop a prescient model for tremor qualities, explicitly Size and Profundity, in view of elements like Timestamp, Scope, and Longitude. The dataset is preprocessed by sifting through lines with invalid timestamps, changing the ’Timestamp’ section completely to numeric qualities, and dropping columns with NaN values after transformation. The dataset is then divided into preparing and testing sets, with 80% utilized for preparing and 20% for testing. The Random Forest Regressor is launched, prepared on the preparation information, and in this manner used to foresee seismic tremor attributes on the testing set. The subsequent forecasts are put away in the ’expectations’ variable, which



```
In [7]: final_data = data.drop(['Date', 'Time'], axis=1)
        final_data = final_data[final_data.Timestamp != 'ValueError']
        final_data.head()
Out[7]:
```

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	Invalid Date
1	1.863	127.352	80.0	5.8	Invalid Date
2	-20.579	-173.972	20.0	6.2	Invalid Date
3	-59.076	-23.557	15.0	5.8	Invalid Date
4	11.938	126.427	15.0	5.8	Invalid Date



```
In [9]: fig = plt.figure(figsize=(12,10))
        plt.title("All affected areas")
        m.plot(x, y, "o", markersize = 2, color = 'blue')
        m.drawcoastlines()
        m.fillcontinents(color='coral',lake_color='aqua')
        m.drawmapboundary()
        m.drawcountries()
        plt.show()
```

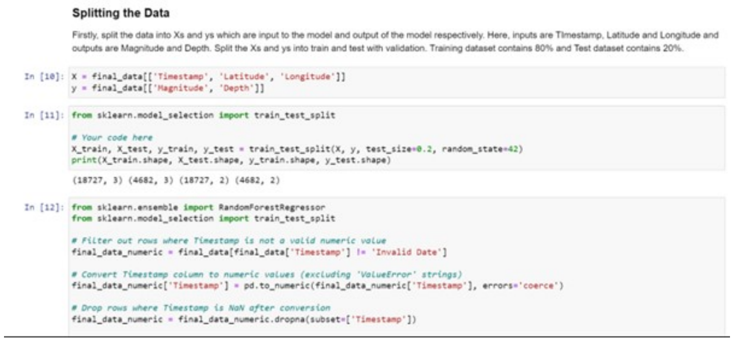
can be additionally dissected or considered in contrast to the genuine qualities to evaluate the model's exhibition. Scikit-learn, ordinarily truncated as sklearn, is an open-source AI library for the Python programming language. It gives straightforward and effective devices for information examination and demonstration, including a wide exhibit of AI calculations for errands like characterization, relapse, grouping, and dimensionality decrease, and the sky is the limit from there. Scikit-learn is based on NumPy, SciPy, and Matplotlib, and it is intended to work consistently with these libraries. It is broadly utilized in both the scholarly world and industry for undertakings connected with AI and information examination because of its convenience, broad documentation, and enormous, dynamic local area of clients and donors.

"reg.score" is used to return the coefficient of determination of the prediction.

Keras and TensorFlow are two well-known open-source AI structures, with TensorFlow filling in as the hidden library for Keras.

TensorFlow:

TensorFlow is an open-source AI structure created by the Google Mind group. It is intended for building and preparing profound learning models. TensorFlow gives an extensive arrangement of devices and libraries for mathematical calculation and AI undertakings, including brain network structures.



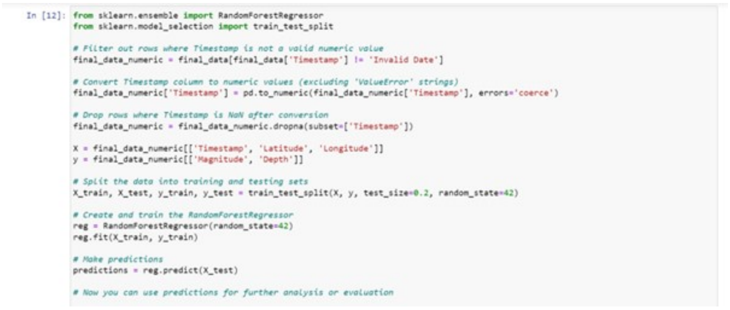
Splitting the Data

Firstly, split the data into Xs and ys which are input to the model and output of the model respectively. Here, inputs are Timestamp, Latitude and Longitude and outputs are Magnitude and Depth. Split the Xs and ys into train and test with validation. Training dataset contains 80% and Test dataset contains 20%.

```
In [10]: X = final_data[['Timestamp', 'Latitude', 'Longitude']]
         y = final_data[['Magnitude', 'Depth']]
```

```
In [11]: from sklearn.model_selection import train_test_split

         # Your code here
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
         print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

         (18727, 3) (4682, 3) (18727, 2) (4682, 2)
```

```
In [12]: from sklearn.ensemble import RandomForestRegressor
         from sklearn.model_selection import train_test_split

         # Filter out rows where Timestamp is not a valid numeric value
         final_data_numeric = final_data[final_data['Timestamp'] != 'Invalid Date']

         # Convert Timestamp column to numeric values (excluding 'ValueError' strings)
         final_data_numeric['Timestamp'] = pd.to_numeric(final_data_numeric['Timestamp'], errors='coerce')

         # Drop rows where Timestamp is NaN after conversion
         final_data_numeric = final_data_numeric.dropna(subset=['Timestamp'])
```



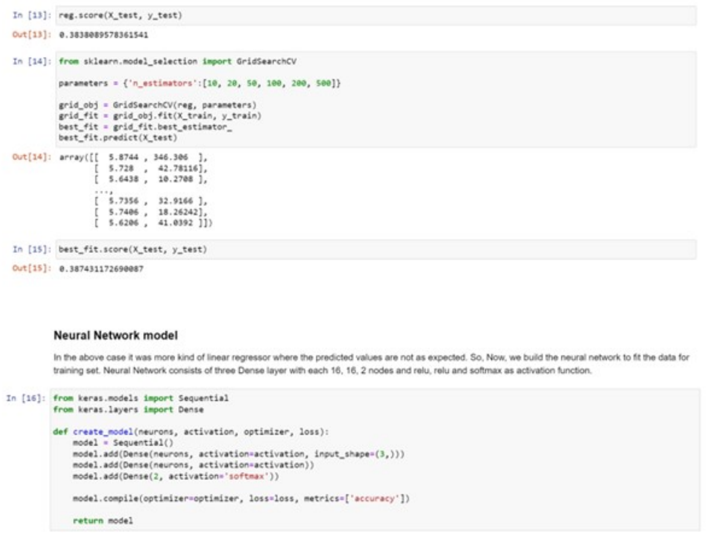
```
In [12]: from sklearn.ensemble import RandomForestRegressor
         from sklearn.model_selection import train_test_split

         # Filter out rows where Timestamp is not a valid numeric value
         final_data_numeric = final_data[final_data['Timestamp'] != 'Invalid Date']

         # Convert Timestamp column to numeric values (excluding 'ValueError' strings)
         final_data_numeric['Timestamp'] = pd.to_numeric(final_data_numeric['Timestamp'], errors='coerce')

         # Drop rows where Timestamp is NaN after conversion
         final_data_numeric = final_data_numeric.dropna(subset=['Timestamp'])

         X = final_data_numeric[['Timestamp', 'Latitude', 'Longitude']]
         y = final_data_numeric[['Magnitude', 'Depth']]

         # Split the data into training and testing sets
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

         # Create and train the RandomForestRegressor
         reg = RandomForestRegressor(random_state=42)
         reg.fit(X_train, y_train)

         # Make predictions
         predictions = reg.predict(X_test)

         # Now you can use predictions for further analysis or evaluation
```

TensorFlow backs both undeniable level APIs for speedy model prototyping, (for example, Keras, which was incorporated into TensorFlow) and lower-level APIs for all the more fine-grained command over model engineering and preparing. It can run on computer processors, GPUs, or particular equipment like TPUs (Tensor Handling Units).

Keras:

Keras is an open-source significant-level brain network Programming interface written in Python. Initially created as a free library, Keras was subsequently coordinated into TensorFlow to give an undeniable level connection point to building and preparing brain organizations. Keras is intended to be easy to understand, secluded, and extensible, making it open for the two novices and experienced analysts.

With the mix into TensorFlow, Keras clients can exploit TensorFlow's versatility and execution while profiting from the effortlessness and convenience of the Keras Programming interface. Keras likewise upholds other backends other than TensorFlow, albeit TensorFlow is the most widely recognized backend utilized.



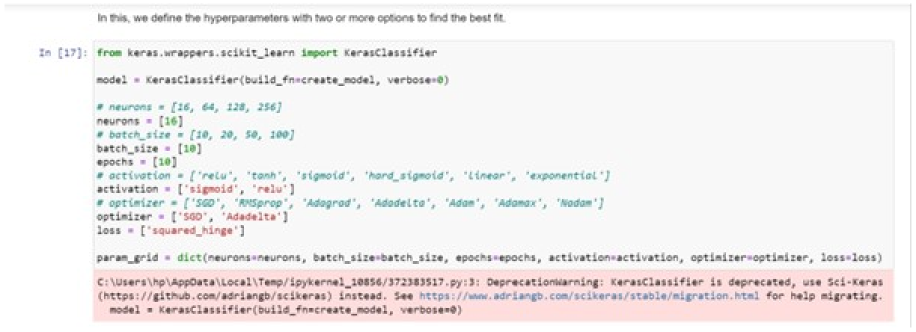
```
In [13]: reg.score(X_test, y_test)
Out[13]: 0.3838089578361541

In [14]: from sklearn.model_selection import GridSearchCV

         parameters = {'n_estimators':[10, 20, 50, 100, 200, 500]}

         grid_obj = GridSearchCV(reg, parameters)
         grid_fit = grid_obj.fit(X_train, y_train)
         best_fit = grid_fit.best_estimator_
         best_fit.predict(X_test)
Out[14]: array([[  5.8744 , 346.306  ],
                [  5.728  ,  42.78116],
                [  5.6438 ,  10.2708 ],
                ...,
                [  5.7356 ,  32.9166 ],
                [  5.7406 ,  18.26242],
                [  5.6206 ,  41.0392 ]])

In [15]: best_fit.score(X_test, y_test)
Out[15]: 0.387431172690087
```

Neural Network model

In the above case it was more kind of linear regressor where the predicted values are not as expected. So, Now, we build the neural network to fit the data for training set. Neural Network consists of three Dense layer with each 16, 16, 2 nodes and relu, relu and softmax as activation function.

```
In [16]: from keras.models import Sequential
         from keras.layers import Dense

         def create_model(neurons, activation, optimizer, loss):
             model = Sequential()
             model.add(Dense(neurons, activation=activation, input_shape=(3,)))
             model.add(Dense(neurons, activation=activation))
             model.add(Dense(2, activation='softmax'))

             model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

             return model
```

Neural networks in earthquake prediction analyze seismic data through interconnected layers, learning patterns to forecast events like magnitude and depth. Trained on historical data, these models use complex architectures to make predictions, though uncertainties persist, making ongoing research crucial for improving accuracy and reliability in earthquake forecasting.

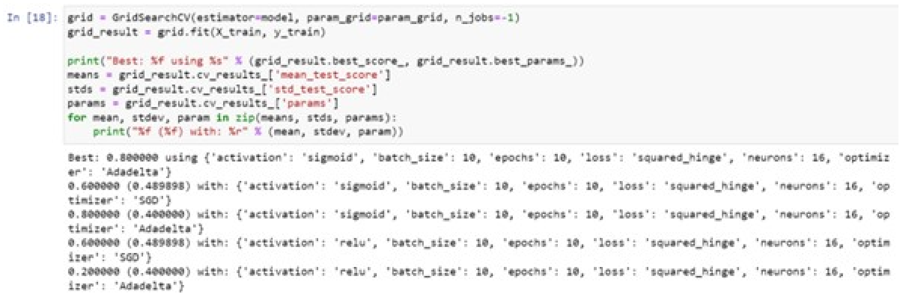


In this, we define the hyperparameters with two or more options to find the best fit.

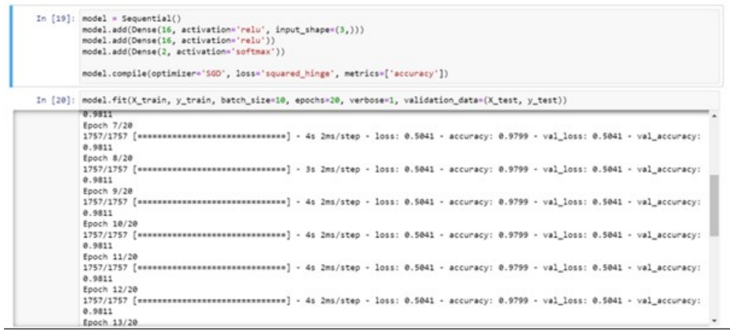
```
In [17]: from keras.wrappers.scikit_learn import KerasClassifier

         model = KerasClassifier(build_fn=create_model, verbose=0)

         # neurons = [16, 64, 128, 256]
         neurons = [16]
         # batch_size = [10, 20, 50, 100]
         batch_size = [10]
         epochs = [10]
         # activation = ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear', 'exponential']
         activation = ['sigmoid', 'relu']
         # optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
         optimizer = ['SGD', 'Adadelta']
         loss = ['squared_hinge']

         param_grid = dict(neurons=neurons, batch_size=batch_size, epochs=epochs, activation=activation, optimizer=optimizer, loss=loss)

         C:\Users\hp\AppData\Local\Temp/ipykernel_10856/372383517.py:3: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras
         (https://github.com/adriangb/scikeras) instead. See https://www.adriangb.com/scikeras/stable/migration.html for help migrating.
           model = KerasClassifier(build_fn=create_model, verbose=0)
```

Now after using different models, we reached the conclusion of using a random forest regressor for prediction in the above code we are fitting our testing data in our model so that we can find the accuracy of our model.

We have set the epoch limit of 20 iterations which has given us an average accuracy of

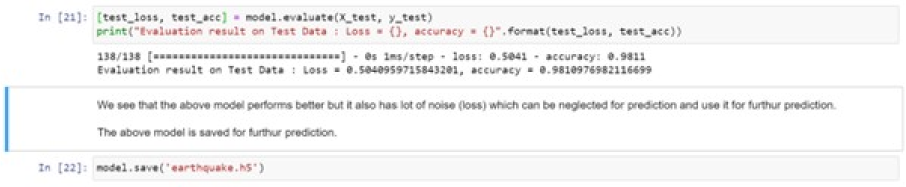


```
In [18]: grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
         grid_result = grid.fit(X_train, y_train)

         print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
         means = grid_result.cv_results_['mean_test_score']
         stds = grid_result.cv_results_['std_test_score']
         params = grid_result.cv_results_['params']
         for mean, stdev, param in zip(means, stds, params):
             print("%f (%f) with: %r" % (mean, stdev, param))

         Best: 0.800000 using {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimiz
         er': 'Adadelta'}
         0.600000 (0.489898) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'op
         timizer': 'SGD'}
         0.800000 (0.400000) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'op
         timizer': 'Adadelta'}
         0.600000 (0.489898) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optim
         izer': 'SGD'}
         0.200000 (0.400000) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optim
         izer': 'Adadelta'}
```



```
In [19]: model = Sequential()
         model.add(Dense(16, activation='relu', input_shape=(3,)))
         model.add(Dense(16, activation='relu'))
         model.add(Dense(2, activation='softmax'))

         model.compile(optimizer='SGD', loss='squared_hinge', metrics=['accuracy'])

In [20]: model.fit(X_train, y_train, batch_size=10, epochs=20, verbose=1, validation_data=(X_test, y_test))
         0.9811
         Epoch 7/20
         1757/1757 [==============================] - 4s 2ms/step - loss: 0.5041 - accuracy: 0.9799 - val_loss: 0.5041 - val_accuracy:
         0.9811
         Epoch 8/20
         1757/1757 [==============================] - 3s 2ms/step - loss: 0.5041 - accuracy: 0.9799 - val_loss: 0.5041 - val_accuracy:
         0.9811
         Epoch 9/20
         1757/1757 [==============================] - 4s 2ms/step - loss: 0.5041 - accuracy: 0.9799 - val_loss: 0.5041 - val_accuracy:
         0.9811
         Epoch 10/20
         1757/1757 [==============================] - 4s 2ms/step - loss: 0.5041 - accuracy: 0.9799 - val_loss: 0.5041 - val_accuracy:
         0.9811
         Epoch 11/20
         1757/1757 [==============================] - 4s 2ms/step - loss: 0.5041 - accuracy: 0.9799 - val_loss: 0.5041 - val_accuracy:
         0.9811
         Epoch 12/20
         1757/1757 [==============================] - 4s 2ms/step - loss: 0.5041 - accuracy: 0.9799 - val_loss: 0.5041 - val_accuracy:
         0.9811
         Epoch 13/20
```

0.9810.



```
In [21]: [test_loss, test_acc] = model.evaluate(X_test, y_test)
         print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))

         138/138 [==============================] - 0s 1ms/step - loss: 0.5041 - accuracy: 0.9811
         Evaluation result on Test Data : Loss = 0.5040959715843201, accuracy = 0.9810976982116699
```

We see that the above model performs better but it also has lot of noise (loss) which can be neglected for prediction and use it for furthur prediction.

The above model is saved for furthur prediction.

```
In [22]: model.save('earthquake.h5')
```

14.5 Findings Discussion

In this exploration paper, quake expectations are investigated from the perspective of time series examination, utilizing the Irregular Woods Regressor. The review dives into the complicated examples and fleeting conditions inside seismic information, using the vigorous prescient abilities of the Arbitrary Backwoods model. The paper talks about the preprocessing steps, including separating and changing over timestamps, as well as the preparation and assessment process.

By utilizing the qualities of the Arbitrary Backwoods Regressor, the examination adds to propelling comprehension we might interpret tremor expectation in view of testing and preparing information, offering bits of knowledge into the worldly elements intrinsic in seismic occasions.

14.6 Conclusion

In the ongoing review, an efficient checking of soil radon along with key meteorological boundaries is led inside the setting of quake estimating. Specifically, prior to corresponding vacillations of soil radon with looming quakes, we have performed: (1) a geochemical examination of soil around the checking site, and (2) measurable

In hurst examination of soil radon and meteorological boundaries, our outcomes recommend a reasonable connection between the fluctuation of soil radon and seismic occasions. The primary aftereffects of this study are summed up as:

- Geochemical examinations uncover that the substance of radionuclides in the soil is similar to standard qualities. This proposes that any improvement in radon isn't connected with the event of normal radionuclides as well as other soil properties like soil dampness content, pH, conductivity and so forth.

- The deterministic investigation of radon and meteorological boundaries uncovers an emphatically auto-connected steady lengthy memory pattern demonstrating that the previous pattern of chosen information is bound to proceed than to be upset from here on out.

- The low relationship coefficients among radon and meteorological boundaries unveil that any odd improvement of radon is not credited to a meteorological change.

- The factual examination (lingering signal handling strategy; $x \pm 2\sigma$) of soil radon changeability uncovers the presence of three periods (Zones A, B and C) of huge radon improvements.

  In each period, outstanding pinnacles of lingering radon were related to looming seismic tremor movement. It is worth focusing on that the current factual methodology neglects to catch any strange change related to the seismic tremors of low greatness and high RE/RD esteem. Comprehensively, the presence of such a critical relationship between the outcomes noticed for various observing stations embraces the utilization of a thick organization of radon observing for seismic tremor expectation studies.

Reference

Preethi, G., and Santhi, B. (2011). Study on techniques of earthquake prediction. International Journal of computer applications, 29(4), 55-58

Amei, A., Fu, W., and Ho, C. H. (2012). Time series analysis for predicting the occurrences of large-scale earthquakes. International Journal of Applied Science and Technology, 2(7).

Shah, H., Ghazali, R., and Nawi, N. M. (2011). Using artificial bee colony algorithm for MLP training on earthquake time series data prediction. arXiv preprint arXiv:1112.4628.

Otari, G. V., and Kulkarni, R. V. (2012). A review of application of data mining in earthquake prediction. International Journal of Computer Science and Information Technologies, 3(2), 3570-3574.

Lyubushin, A. A. (1999). Analysis of multidimensional geophysical monitoring time series for earthquake prediction.

Morales-Esteban, A., Martínez-Álvarez, F., Troncoso, A., Justo, J. L., and Rubio-Escudero, C. (2010). Pattern recognition to forecast seismic time series. Expert systems with applications, 37(12), 8333-8342.

Ali, A., Ghazali, R., and Deris, M. M. (2011, December). The wavelet multilayer perceptron for the prediction of earthquake time series data. In Proceedings of the 13th International Conference on Information Integration and Web-based applications and services (pp. 138-143).

Barkat, A., Ali, A., Hayat, U., Crowley, Q. G., Rehman, K., Siddique, N., ... and Iqbal, T. (2018). Time series analysis of soil radon in Northern Pakistan: Implications for earthquake forecasting. Applied Geochemistry, 97, 197-208.