Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #47291

Earthquake Prediction: Irregular Time-series Forecasting with Deep Learning

Mrs. Neda Farahmandi, University of Massachusetts Dartmouth
Ashok Patel, University of Massachusetts Dartmouth

Earthquake Prediction: Irregular Time-series Forecasting with Deep Learning

Abstract

Earthquake prediction is an area of interest to researchers around the world as well as anyone who has experienced a major earthquake. Major earthquakes often cause loss of lives and property, as well as injuries and destruction. Large investments of time and money are required to build communities back to near where they were before disasters such as earthquakes. For decades, scientists have considered different methods for earthquake prediction. Machine learning (ML) applications have been used in seismology for at least a decade but ML applications in seismology have increasingly grown during the past few years. In this study, deep learning models will be applied to three different earthquake datasets, with the goal of predicting earthquake magnitude. A specific type of recurrent neural network, Long Short-Term Memory (LSTM), with memory cells that allow for utilizing information form recent past steps, will be applied to earthquake datasets. These datasets vary in size and are in the form of time-series where earthquake magnitudes, in Richter scale, are recorded across the time axis. Dataset II is the largest dataset, containing 50 years of seismic data from 1973/01/02 to 2023/12/31, in a large region that covers the state of California with a minimum longitude of -133, maximum longitude of -107, minimum latitude of 24 and maximum latitude of 50. Dataset I is of medium size, covering 3 years of seismic data from 1970/01/02 to 1973/01/02, in the same region. Dataset III is the smallest dataset which contains seismic data for 30 days from 2024/05/05 to 2024/06/04. Different sizes of datasets have been used to study the effect of different timescales. LSTM architectures will be proposed and tested on three different datasets that are acquired from the United States Geological Survey Website and their performance will be evaluated and compared. Since earthquakes are natural phenomena that happen at arbitrary points in time, these time-series are irregular time-series, meaning the time intervals in between consecutive observations vary in size. To address the irregularity of the time-series, interpolation will be applied to datasets. It is observed that interpolation considerably improves the model performance.

Introduction

Earthquakes are catastrophic natural disasters that are caused by sudden changes in earth's crust. According to United States Geological Survey (USGS), an earthquake happens when two blocks of earth suddenly slip past one another [2]. The Encyclopedia Britannica defines an earthquake as any sudden shaking of the ground by the passage of seismic waves through earth's rocks [3].

Earthquake Prediction Background

Earthquake prediction has been an interesting subject for researchers for over 100 years [4], but it is still not a mature field yet [5]. There are references about unusual animal behavior before a significant earthquake as early as 373 B.C. [6]. There are research papers on earthquake prediction as early as 1939. Wood & Gutenberg (1939), Macelwane (1946) and Allen (1976) are some of the early examples [4]. There exist two main approaches to earthquake prediction, precursors based, and trend based. Precursor based prediction relies on anomalous phenomena that may be a sign of a forthcoming earthquake such as [7] localized changes in magnetic and electric fields, radon gas emissions from earth [8], variations in humidity, patterns of cloud formation, soil temperature, and crustal change near the site of the epicenter of a forthcoming earthquake [9]. Trend based earthquake prediction methods try to find patterns in the seismic data that lead to occurrence of an earthquake [7]. Earthquakes are catastrophic natural disasters that are caused by sudden changes in earth's crust. According to United States Geological Survey (USGS), an earthquake happens when two blocks of earth suddenly slip past one another [2].

Machine Learning Background

Machine learning (ML) is a subfield of artificial intelligence that learns or finds patterns in the data that it is presented to and uses that knowledge to make predictions on data that it has never seen before. With machine learning, computers do not need to be explicitly programmed to solve a problem. Machine learning utilizes data and algorithms and statistical models to solve a problem using inference instead of instructions. A simplified machine learning flow is shown in Figure 1.

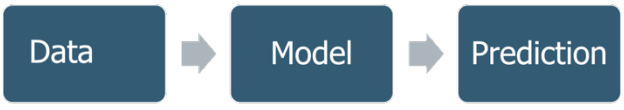

Figure 1: Machine learning flow. Algorithm is trained on training and makes predictions on test data.

Machine Learning Types

Machine learning has three main categories, supervised learning, unsupervised learning and reinforcement learning (Figure 2). Supervised machine learning is a type of machine learning that uses labeled data with known input and output to train an algorithm. The trained algorithm can then make predictions on the data it has never seen before. In unsupervised learning, model is not presented with labeled data and finds patterns in the data on its own. In reinforcement learning, an algorithm is in interaction with its environment and makes decisions with the goal of maximizing its total reward.

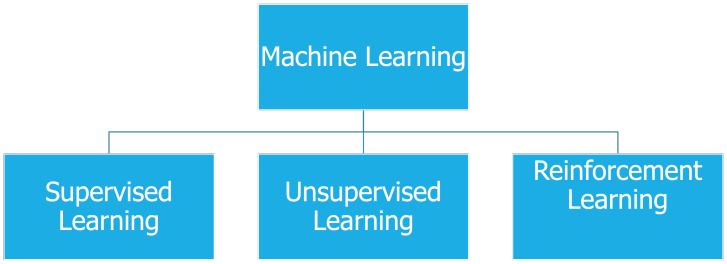

Figure 2. Three main categories of machine learning.

Deep Learning

Deep learning is an Artificial Intelligence (AI) method and a sub-category of machine learning [10]. Deep learning models are Artificial Neural Networks (ANN) that are inspired by human brain. Artificial Neural Networks have an input layer, output layer and several hidden layers in between them. Each layer consists of several Artificial Neurons (AN). Input layer is a layer of neurons that receive the input data. Hidden layers receive the data from the input layer and process it in various forms. Output layer nodes provide the output data. Neural Networks also consist of synapses, weights, biases and activation functions [1].

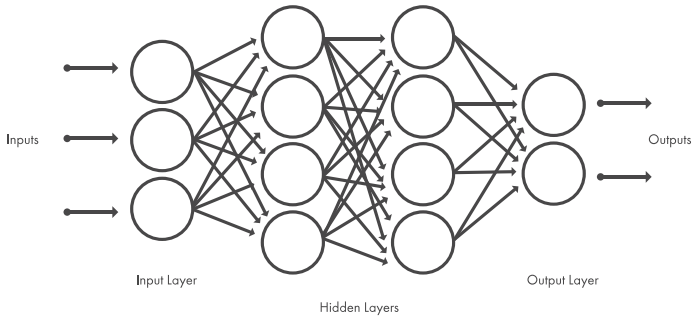

Figure 3. Typical Neural Network (NN) architecture used in Deep Learning [11].

Regression Problem

Machine learning can solve classification problems and regression problems. A classification problem is where an input needs to be mapped to one of the two (binary classification problem) or more (multi-class classification problem) defined categories. Accuracy of a classification model can be evaluated by calculating sensitivity (rate of true positives or correct predictions of positive cases) or specificity (rate of true negatives or correct predictions of negative cases). A regression problem, however, is where an input is mapped to a continuous value, such as an integer. Examples of regression problem include weather forecast or housing market forecast. One approach to evaluating a regression predictive model is to calculate and compare the Mean Squared Error (MSE) for different models. MSE is defined as the average of the squares of the errors, where error is the difference of the observed (x_o), and predicted value (x_p).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_p - x_o)^2$$

Machine Learning in Earthquake Prediction

History of machine learning applications in seismology goes back to 1990s. (e.g., Turhan Taner et al. 1988, Dowla et al. 1990), but recently, the use of ML applications in seismology has grown rapidly with promising results. Predicting earthquakes with ML ae more accurate compared to conventional prediction techniques [12]. Machine learning research in seismology focuses on using different ML models and algorithms with historical seismic data or lab created datasets to predict large magnitude earthquakes. Some researchers have employed an Artificial Neural Network for earthquake prediction. Cheraghi and Chanbari for example, utilized an Artificial Neural Network to predict the time and magnitude of earthquakes based on seismotectonic survey and faults information. The maximum error of their model was 3.5% and the average error for magnitude prediction was 0.5%. [13]. Other researchers have studied the spatio-temporal relationship of earthquakes in different locations. One of these studies was conducted by Wang et

al. in 2017. They used an LSTM model on USGS datasets of different locations for their research. This study used a dataset with earthquakes with a magnitude of larger than 4.5 in Richter scale from 1966 to 2016. The accuracy of model in this research was 63.50% for one-dimensional input and 87.59% for two-dimensional input [14]. Vardan et. Al compared a Feed-Forward Neural Network (FFNN) with 2 hidden layers to an LSTM model with 2 hidden layers with the goal of forecasting the trend of an earthquake. Their research covered a large area including Indian subcontinent region, Afghanistan, Tajikistan, Thailand, Laos, Vietnam, Malay Peninsula, and several provinces in China. They concluded that LSTM results were significantly superior compared to FFNN [7].

Related Work

Machine learning applications in seismology include event discrimination, earthquake signal detection, seismic phase picking, Polarity determination, phase association, earthquake source parameterization, seismogram simulation, ground motion characterization, direct investigation of seismic waveforms, and earthquake forecasting [1]. Making prediction about magnitude, time or location of an earthquake is a challenging task. Earthquakes are natural phenomena that follow no apparent well understood pattern. During recent years, machine learning techniques have been used in variety of studies to discover possible patterns within seismic data. Al Banna et al. (2020) categorized earthquake prediction methods to rule-based approaches, shallow machine learning algorithms and deep learning (DL). They mention Fuzzy Logic and Fuzzy Neural Network as examples of rule-based approaches. Support Vector Machine (SVM), Support Vector Regression (SVR), K-Nearest Neighbor (KNN) algorithm, Random Forest (RF) algorithm and K-Means clustering are amongst the shallow machine learning methods discussed by Al Banna et al. (2020) Deep Learning has been used in earthquake characteristics studies and earthquake prediction [15]. An example of earthquake characteristic studies is Seismic Electric Signals (SES) anomaly prediction. In a 2017 study, Karabachos et.al. proposed a hybrid algorithm for predicting anomaly in SES time series data [15] [16]. There are numerous examples of deep learning applications in earthquake studies. Supervised learning methods have shown to be very effective for event discrimination [1] In 2019, Mousavi et al. employed an unsupervised deep learning method for feature learning and dimensionality reduction to distinguish waveforms [17]. Other applications of deep learning in earthquake seismology include signal detection, seismic phase picking, polarity determination, phase association, ground motion characterization and forecasting earthquakes [1].

Time-Series

A time-series dataset is a collection of observations sequenced in time [18]. Time-series data is usually a sequence taken at regular time intervals. Time-series have applications in finance, weather forecasting, earthquake prediction and biological sciences, where patterns in data are used to predict future values. Examples of time-series data include maximum daily temperature readings, stock prices or heart rate readings of a monitoring device. Time-series forecasting is

concerned with predicting future values based on previously observed datapoints. Time-series forecasting can be formulated as a supervised machine learning problem. Large datasets are essential to ensure successful training and testing of a machine learning model to forecast time-series. Due to availability of compressive earthquake catalogues, seismic time-series are a suitable candidate for time-series forecasting.

Irregular Time Series

Time-series are often a sequence of datapoints taken at equally spaced intervals. A sequence of datapoints recorded at unevenly spaced time intervals is an irregular time-series dataset. Examples of irregular time-series include natural disasters such as earthquakes, where data is received unpredictably. Earthquakes naturally occur at arbitrary times, therefore the intervals between consecutive occurrences are irregular. One way of approaching irregular time-series is interpolation. Interpolation is a method of mathematical estimation, where new data points are created based on the known data points. Utilizing interpolation was indirectly mentioned in [19], however the interpolation sampling interval is one month. In this study, a much more refined resampling interval will be used to achieve more accuracy.

Deep Neural Networks

Deep Learning is a sub-category of machine learning that is based on utilizing neural networks. Deep Neural Networks (DNN) have input layer, output layer and multiple layers in between, hence the name “deep”. Each layer consists of artificial neurons that are interconnected. There are variants of neural networks, but regardless of the type, all NNs consist of neurons, synapses (connections between neurons), weights, activation functions and biases. Based on the direction of the flow of data in model, DNNs can be categorized into two main types of Feedforward Neural Networks and Recurrent Neural Networks.

Feedforward Networks

In a Feedforward deep network, data flows only in one direction. Data flow starts at input layer, moving forward to possible hidden layers and ends at output layer. In feedforward networks, data flow does not loop back.

Recurrent Neural Networks (RNN)

As humans, our thoughts have persistence. That means we process new information based on previous information that we have [21]. Traditional neural networks do not have this ability and perceive new information without the knowledge of what they have previously seen. Recurrent Neural Networks (RNN) however, are different in that their input at any time includes both new input and output from the previous step. In LSTM, previous steps impact how future outputs are calculated, therefore, they can be said to have a memory. At every step of training RNNs with gradient descent, the gradient will diminish and eventually disappear, therefore it faces the vanishing gradient issue. As a result, RNNs are not as successful if they must go back to more than a couple of steps to gather the information they need to calculate the output for the current step

[21]. Data flow is bi-directional in Recurrent Neural Networks. As it is depicted in figure 4, block A of neural networks receives input x_t, and outputs value h_t. This loop allows information to flow forward and loop back and flow backward, allowing output of a node affect the input of the same node.

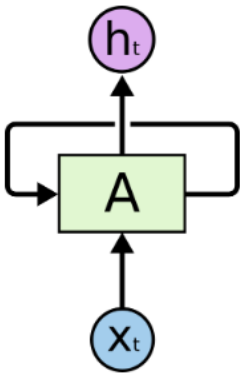

Figure 4. Loop in a Recurrent Neural Network

Some areas that have benefited from RNN applications include speech recognition, image captioning and language modeling. RNNs are very useful where the gap between previous information and output is not large. When the model needs to look further back for information to generate the output, RNNs become less efficient. Long Short-Term Memory Networks are a solution to these long-term dependency problems [22].

Long Short-Term Memory (LSTM) Model

Long Short-Term Memory networks are RNNs that can learn long-term dependencies. The goal of LSTM is to add short-term memory to a recurrent neural network, that is capable of going back in time-steps at least a couple of thousand steps [22]. LSTM is designed to avoid the traditional RNNs vanishing gradient problem. LSTM consists of a component that can learn when to remember and when to forget the information based on its decision on whether that information may or may not be needed [23]. A review of literature was done to understand the current applications of LSTM in seismology. Wang et al. (2017) [14] used LSTM to predict earthquakes by learning the spatio-temporal relationship. Wang et al. reported accuracy of % 87.59 for two-dimensional input [9]. Vardaan et al. (2019) compared LSTM with FFNN to study the trend of earthquakes. This study used R^2 to evaluate the model performance [7]. Cao et al. (2021) modelled a LSTM network to create a catalogue for earthquakes with a reported MSE of 0.08 [24]. Bhargava and Pasari proposed an LSTM architecture to predict earthquakes and reported an MSE of 0.2048 [25]. AL Banna et all. (2021) created an LSTM network to predict location of earthquake epicenter with MSE of 1.5579 [12]. In 2021, Kavianpour et al. reported MSE of 0.1824, when they used LSTM to predict the mean magnitude of a forthcoming earthquake in one month timeframe [19].

Cao et al. modelled an LSTM network to make predictions on synthetic data and reported 0.08 for MSE [26].

Method

Here we discuss the process of data collection and data preparation, as well as the machine learning models considered and used for this work.

Data

Earthquake datasets have a natural temporal ordering. This characteristic is what makes earthquake datasets time-series datasets. Time-series is defined as sequence of relevant values that are ordered by time. Often, time-series data points are recorded at equally spaced points in time [24]. There is an abundance of time-series data in the field seismology. First earthquake recorded happened in 1831 B.C. in China. According to USGS, several million earthquakes occur in the world annually. Many of these shakes are not detected due to occurring in remote areas or having a very small magnitude [27]. Currently, USGS records about 50 earthquakes each day, 20,000 per year. There are many organizations around the world that monitor seismic activity and provide datasets for researchers. USGS is a reliable source that monitors and reports accurate earthquake catalogues.

Data Collection

USGS provides an Application Programming Interface (API) for the Federation of Digital Seismograph Networks (FDSN) that allows custom searches for earthquake information using a variety of parameters [28]. This data is freely available and highly accurate and reliable because almost all recordings are reviewed by a human. Query method was used to collect earthquake data for this research. 3 different datasets were acquired for this paper. These datasets have extensive information about each recorded earthquake, also referred to as event in this text. Earthquake dataset I targets earthquakes with a magnitude larger than 2 in Richter scale that occurred in the 3-year period from 1970 to 1973. This dataset is in Comma-separated Values (CSV) format and has 2387 rows and 17 columns. CSV is a text file format which uses commas to separate values and newlines for separating records. Each line of CSV file represents one data record [29]. Each row associates with a recorded event/ earthquake and each column represents a feature of that event such as magnitude, longitude, latitude, depth and time of occurrence. The method used for data extraction and time-series dataset creation is based on the work in [30]. Earthquake dataset I, II and III were directly extracted from the USGS API for the same location. Dataset II is capturing the same region during the next 50 years, from 1973 to 2023. This dataset targets earthquakes with a magnitude of greater than 2 in Richter scale. Python [31] with Anaconda-Navigator [30] was utilized for pulling earthquake data directly from USGS API. The libraries used for data collection include Pandas [33], NumPy [34] and Matplotlib [35]. The USGS API web services limits the queries to 20000, and any that exceed this limit will generate a HTTP response code “400 Bad

Request" [36]. Therefore, each year was divided to four quarters and the raw data was saved individually for each quarter of each year. A snippet of the code for data collection is shown in Data is extracted for earthquakes with magnitude between 2-10 in Richter scale. This data is limited to 20,000 rows and ordered by time. Data is collected from a large region that covers the state of California with a minimum longitude of -133, maximum longitude of -107, minimum latitude of 24 and maximum latitude of 50.
Dataset II is in CSV format, has 330,812 recorded earthquakes/ rows of data and 17 attributes/ data columns that describe features of each recorded event. The most notable features are magnitude, longitude, latitude, depth and time of occurrence. Earthquake dataset III is directly extracted from USGS Earthquake Hazards Program website [37]. This web service allows for searching earthquake catalogs. Earthquakes in Dataset III have a magnitude of larger than 1 in Richter scale and for an area smaller and within the region that is covered by dataset I and dataset II. Dataset III covers a 30-day period from 2024/05/05 to 2024/06/04. Dataset III is in CSV format, has 309 records of earthquake events/ rows and 22 features/ columns including magnitude, longitude, latitude, depth and time of occurrence. This dataset is much smaller compared to dataset I and dataset II and was downloaded in batch. Having a smaller size dataset with a lower threshold for magnitude allows for much more accelerated testing of different models.

Data Preparation

The USGS API limits datasets to 20,000 records per download. For this reason, data for dataset II was downloaded for each quarter of each year from 1973 to 2024. These files were concatenated and created a combined earthquake time-series data frame. Same method was used for Dataset I and Dataset III.
Google Colab [38] was used for cleaning the dataset and used TensorFlow [39], Pandas and NumPy libraries. Using Pandas.DataFrame.info () method on this data frame provides detailed information on this dataset.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 330811 entries, 0 to 330810
Data columns (total 17 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Unnamed: 0  330811 non-null  int64
 1   type        330811 non-null  object
 2   time        330811 non-null  object
 3   mag         330811 non-null  float64
 4   place       330811 non-null  object
 5   status      330811 non-null  object
 6   tsunami     330811 non-null  int64
 7   sig         330811 non-null  int64
 8   net         330811 non-null  object
 9   nst         318950 non-null  float64
 10  dmin        242946 non-null  float64
 11  rms         326505 non-null  float64
 12  gap         321707 non-null  float64
 13  magType     330808 non-null  object
 14  longitude   330811 non-null  float64
 15  latitude    330811 non-null  float64
 16  depth       330807 non-null  float64
dtypes: float64(8), int64(3), object(6)
memory usage: 42.9+ MB
```

Figure 5. Dataset II information reveals the count and data type for each column.

Model Selection

Most traditional machine learning and deep learning methods are compromised when applied to irregular time series and fail to properly model the temporal irregularity of time-series where datasets have unequal intervals in between observations and possible missing data. Gated Recurrent Neural Networks (RNN) such as Long Short-Term Memory (LSTM) have shown great success in modeling sequential data [40]. Based on literature review, LSTM was selected for earthquake time-series prediction in this study. Different LSTM model architectures were proposed and tested with different activation functions, optimizers and learning rates to find the best model for dataset I, II and III.

Interpolation

Machine learning tasks require suitable datasets to perform well but datasets created from observational data are far from perfect. Collecting data from natural phenomena that occur randomly is even more problematic. In the case of seismic data, the randomness of occurrence, causes irregularity in the dataset. Irregularity and missing data points can reduce the efficiency of machine learning models. Specifically, LSTM has become a promising tool in modeling irregular time-series [40]. One solution to address the irregular time-series is interpolation. With interpolation, dataset is transformed into a new dataset with fixed intervals in between the datapoints. Due to randomness of the earthquake occurrence, there will be missing values in some areas. These missing datapoints can be filled with interpolation. Current literature that addresses this issue, have utilized interpolation [41] with larger intervals such as monthly [19] or daily intervals. In this study, much smaller apertures will be tested to observe the performance of the model and test the hypothesis.

Implementation

The programming language used for this is Python 3.10. For data collection and data preparation for datasets, Python was used within Jupyter Notebook through Anaconda-Navigator [42]. The remaining of the scripts for this work is implemented using Google Colab. Google Colab is a hosted Jupyter Notebook service with access to accelerated hardware such as Graphics Processing Unit (GPU) [43] and Tensor Processing Units (TPU) [44]. This service is provided at no cost for a basic user.

Long Short-Term memory (LSTM)

In this section, different LSTM architectures will be proposed, trained and tested. These different architectures will be tested with different activation functions, look-back windows, learning rates and optimizers to achieve the best result. Desired goal is to achieve the smallest Mean Squared Error for the unseen data. LSTM expects input to be a 3-D tensor in the form of [batch size, look-back window size, dimension of output feature]. Batch size determines the number of readings available which is exactly the number of the rows in the dataset. Look-back

window is the number of steps, or values that the model will consider when computing the next output value. Dimension of output feature demonstrates the shape of the value we are considering against time. We are trying to predict earthquake magnitude against time, so in this case our output is “mag”, and the dimension of the output feature is 1.

64-unit LSTM layer - RELU Activation Function - 50 step Look-Back Window

The snippets of code from the initial attempt with LSTM is shown in this section. Dataset II was used for this implementation. As a first step, initial libraries are imported.
At the next step, dataset II is read into pandas data frame to make reading and further inspecting the data frame possible.
Index column has been replaced with column time to make data visualization more convenient.
In order to create the time series representing earthquake magnitude over time, Column mag, which represents the earthquake magnitude in this data frame, is read into a separate data frame. The result is a new data frame with first column as time and second column as magnitude. There are 330802 data readings, and the object type is float 64.
A plot of magnitude distribution over time is shown in Figure 6. It is seen that magnitude of earthquakes range from 2 in Richter scale to greater than 7.

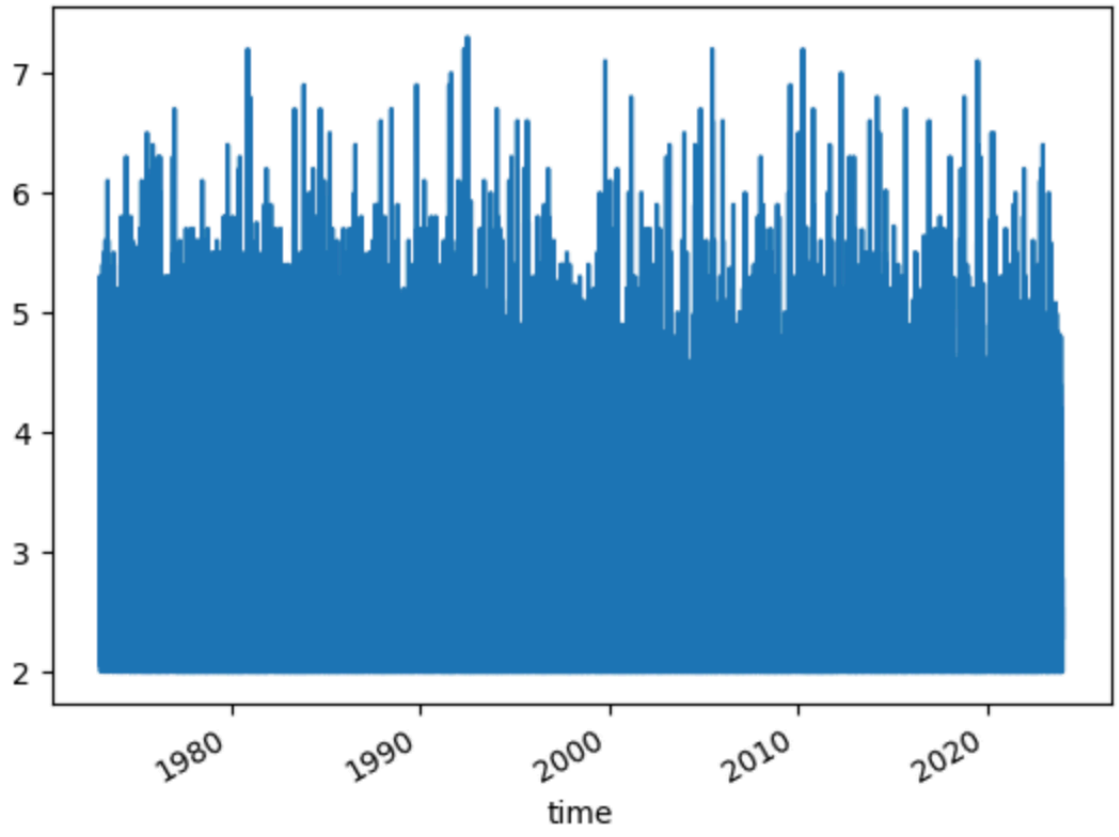

Figure 6. Distribution of magnitude from 1973 to 2024.

Next step is to define the input and output data and determine the size of the look back window. This number represents the number of steps that the LSTM model will go back and consider in computing the output at current step. In this case, a window size of 50 is selected, meaning that magnitude of the last 50 earthquakes will be considered when computing the output value for the next earthquake in the time series. Dimensions of the input and output arrays can be demonstrated by using the .shape command. The input tensor, X, is of size (330752, 50, 1) and output array, y, dimensions are (330752, 1).
A closer look at the input and output arrays reveals a more detailed picture of what these data elements actually look like. As it is seen, X is a tensor comprised of 330752 elements, each having mags for the previous 50 earthquakes. y is the output, representing the magnitude computed for the next earthquake. Next, input and output data are divided into training, validation and test datasets. Training data set comprises 70 percent of the dataset and is used to train this model. 20 percent of the data is specified to validation data and the remaining 10 percent is the testing data. At the final stage, the trained and validated model will be tested on the test data, which it has never seen before, to evaluate the performance of the model.
Next step is to design the architecture of the LSTM model. Here it can be seen in Figure 7. That a sequential model is chosen with an LSTM layer with 64.

```
model1 = Sequential()
model1.add(InputLayer((50,1)))
model1.add(LSTM(64))
model1.add(Dense(8, 'relu'))
model1.add(Dense(1, 'linear'))

model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 dense (Dense)               (None, 8)                 520

 dense_1 (Dense)             (None, 1)                 9

=================================================================
Total params: 17425 (68.07 KB)
Trainable params: 17425 (68.07 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 7. LSTM model summary.

Next, loss is defined as Mean Squared Error, and “Adam” is chosen as the optimizer. Learning rate is set to 0.0001 and a check point is added to save the best model. This model is trained for 10 epochs, and the Mean Squared Error (MSE) is 0.2379 for training. Next, the results of applying best model to training, validation and test datasets ae shown. Actual values can be seen against predicted values at each level, and data is visualized at each step for better understanding of the

model performance. Actual values for earthquake magnitude and predicted magnitudes are provided in a data frame below.

```
train_predictions = model1.predict(X_train).flatten()
train_results = pd.DataFrame(data={'Train Predictions':train_predictions, 'Actuals' :y_train})
train_results
```

```
7235/7235 ━━━━━━━━━━━━━━━━━━━━ 65s 9ms/step
```

	Train Predictions	Actuals
0	2.658201	2.80
1	2.667721	2.00
2	2.613438	3.16
3	2.671713	2.93
4	2.672209	2.35
...	...	...
231495	2.472145	2.80
231496	2.480958	2.04
231497	2.436869	2.84
231498	2.476216	2.35
231499	2.453130	2.71

231500 rows × 2 columns

Figure 8. Predicted mag values against actual mag values for training data.

This plot is showing 1000 readings for better visualization.

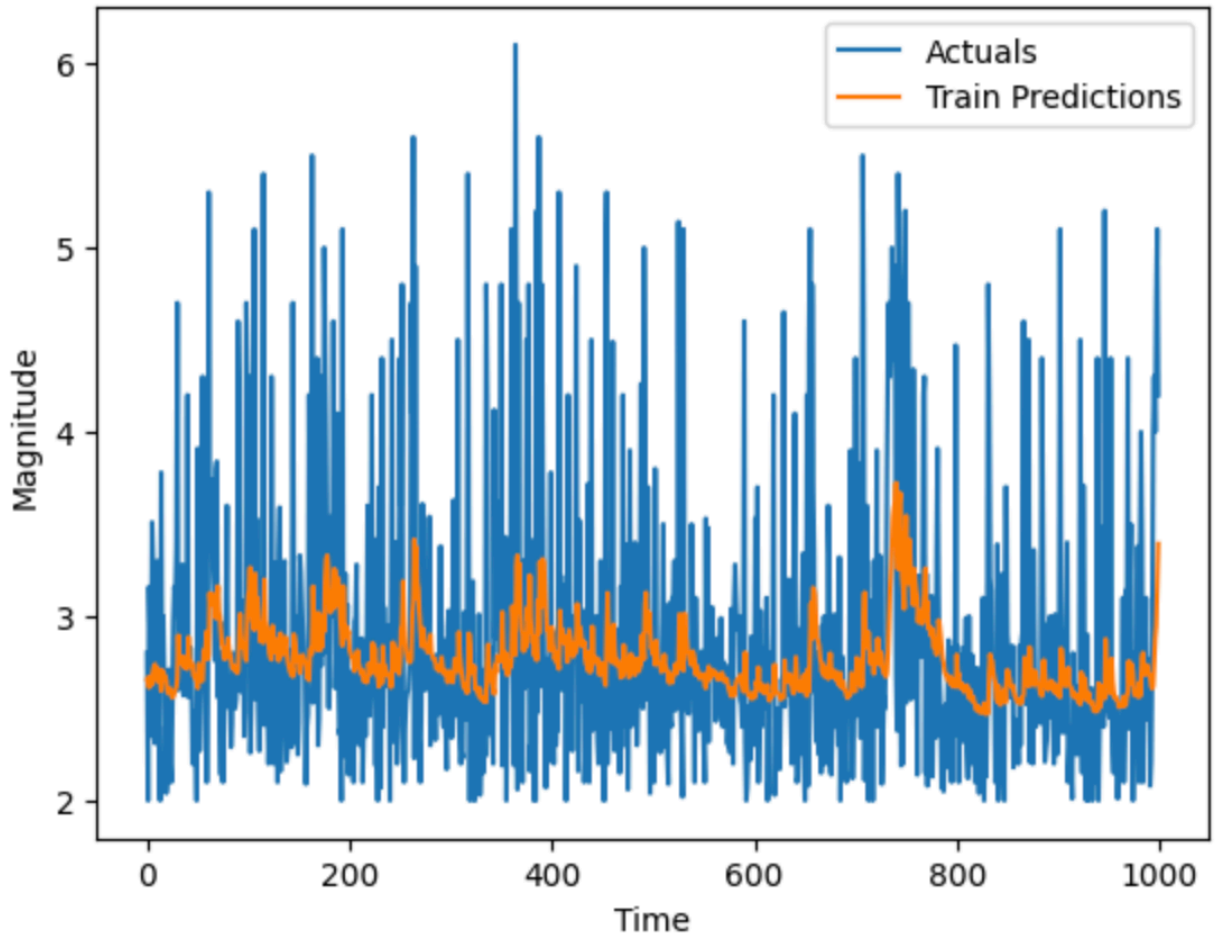

Figure 9. Earthquake magnitude prediction vs actual magnitude values for training dataset.

Same process is done for the validation dataset. Predicted values for earthquake magnitude are compared with actual magnitude values. Mean Squared Error (MSE) obtained for validation is 0.2617.

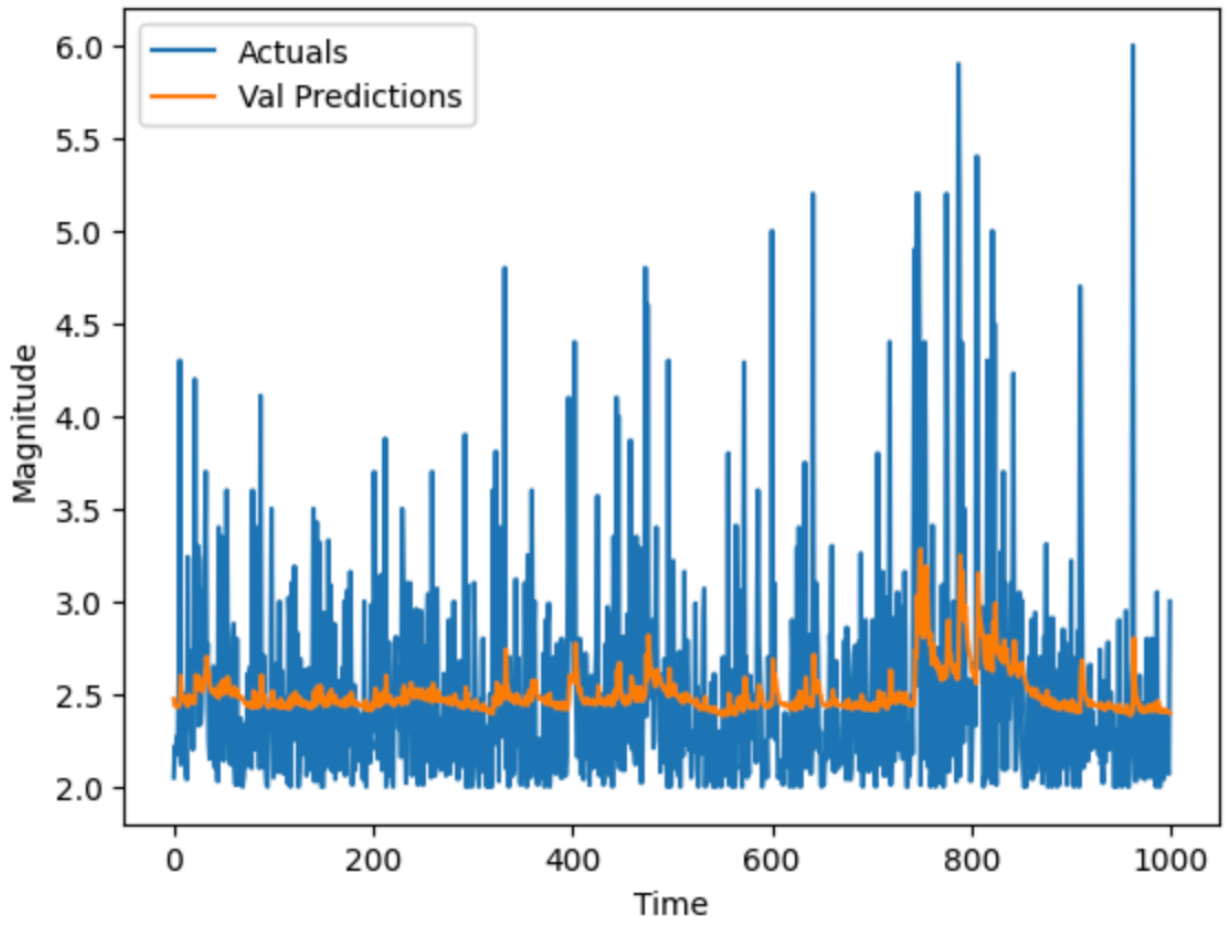

Figure 10. Earthquake magnitude prediction vs actual magnitude values for validation dataset.

Finally, the predicted values for earthquake magnitude are compared with actual magnitude values for the test data. Next, the performance of the saved model is shown on test data in a plot in Figure 11. As it is seen in the plot, the model has successfully predicted the trend of the earthquake time series and peaks are detected. MSE for the test set is equal to 0.2297.

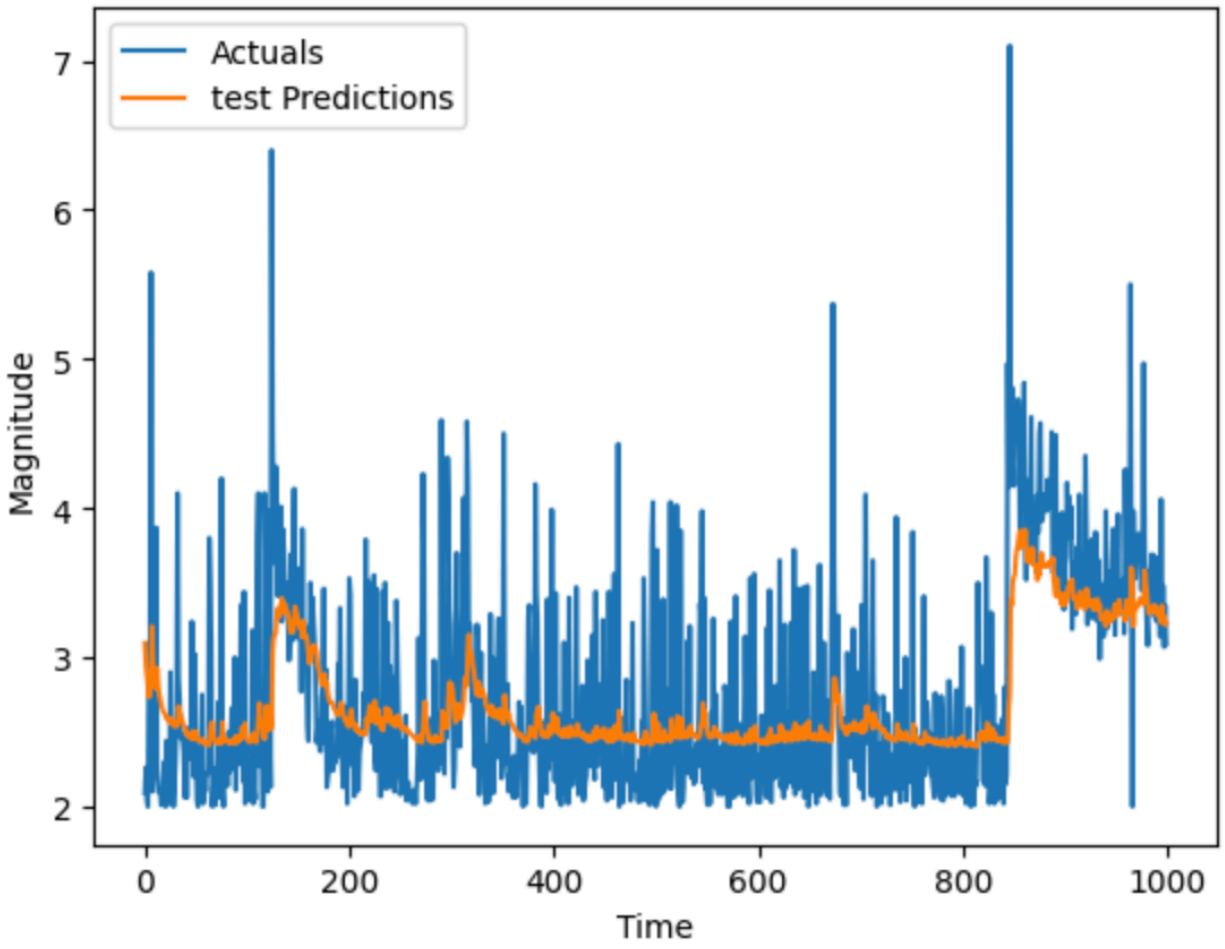

Figure 11. Earthquake magnitude prediction vs actual magnitude values for test dataset.

Now, this trained model will be tested on the other datasets to evaluate the performance of the model. First, this initial model is tested on the entirety of dataset I. Test results for dataset I are shown in Figure 12. As it is seen in the plot, the model has achieved success in testing with dataset I and predicting the trend and peaks for dataset I. In this case, MSE for the Dataset I is equal to 0.3875.

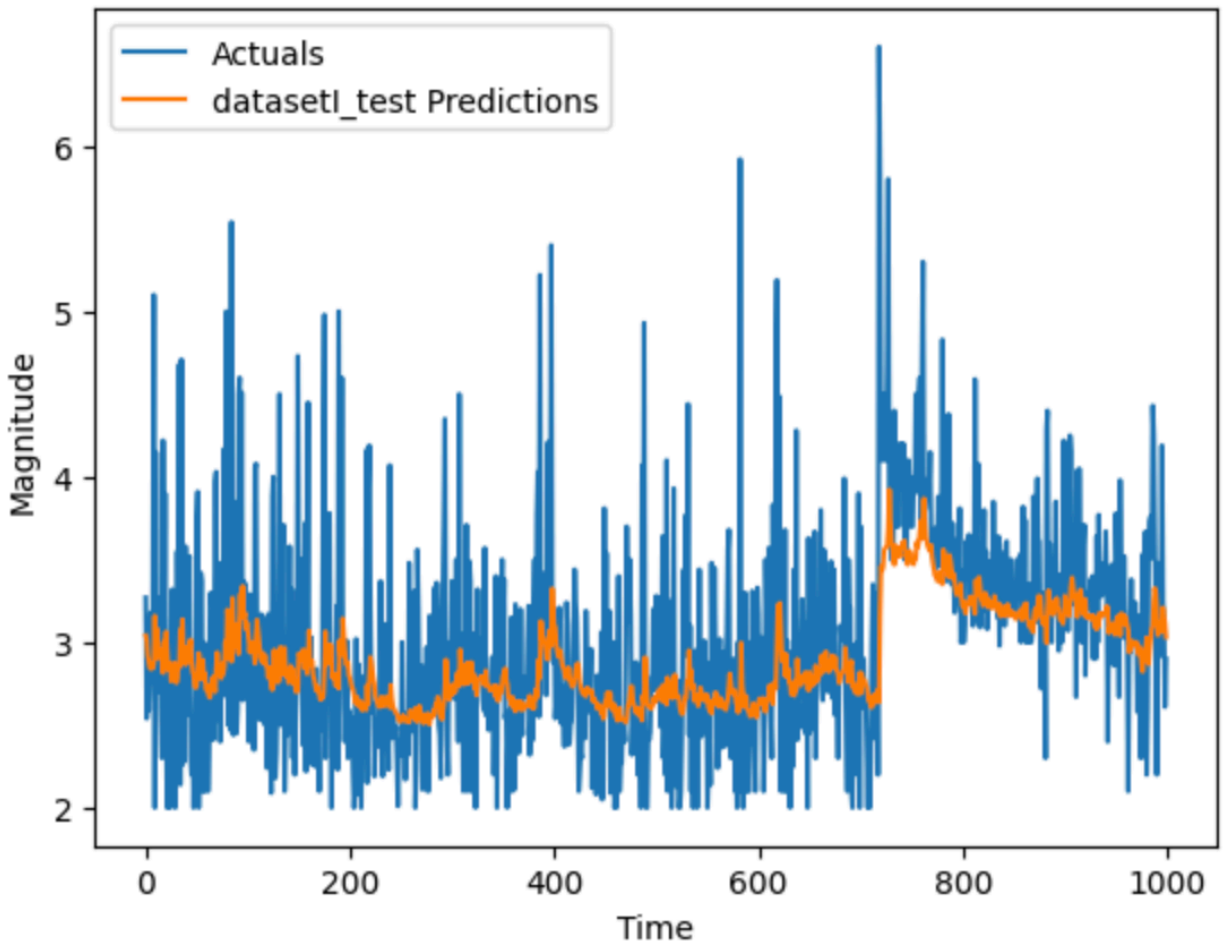

Figure 12. LSTM initial attempt with Dataset I.

Model is tested with dataset III, which is a smaller dataset, covering only 1 month, and successfully recognized the data peaks. MSE for this case was 0.7120.

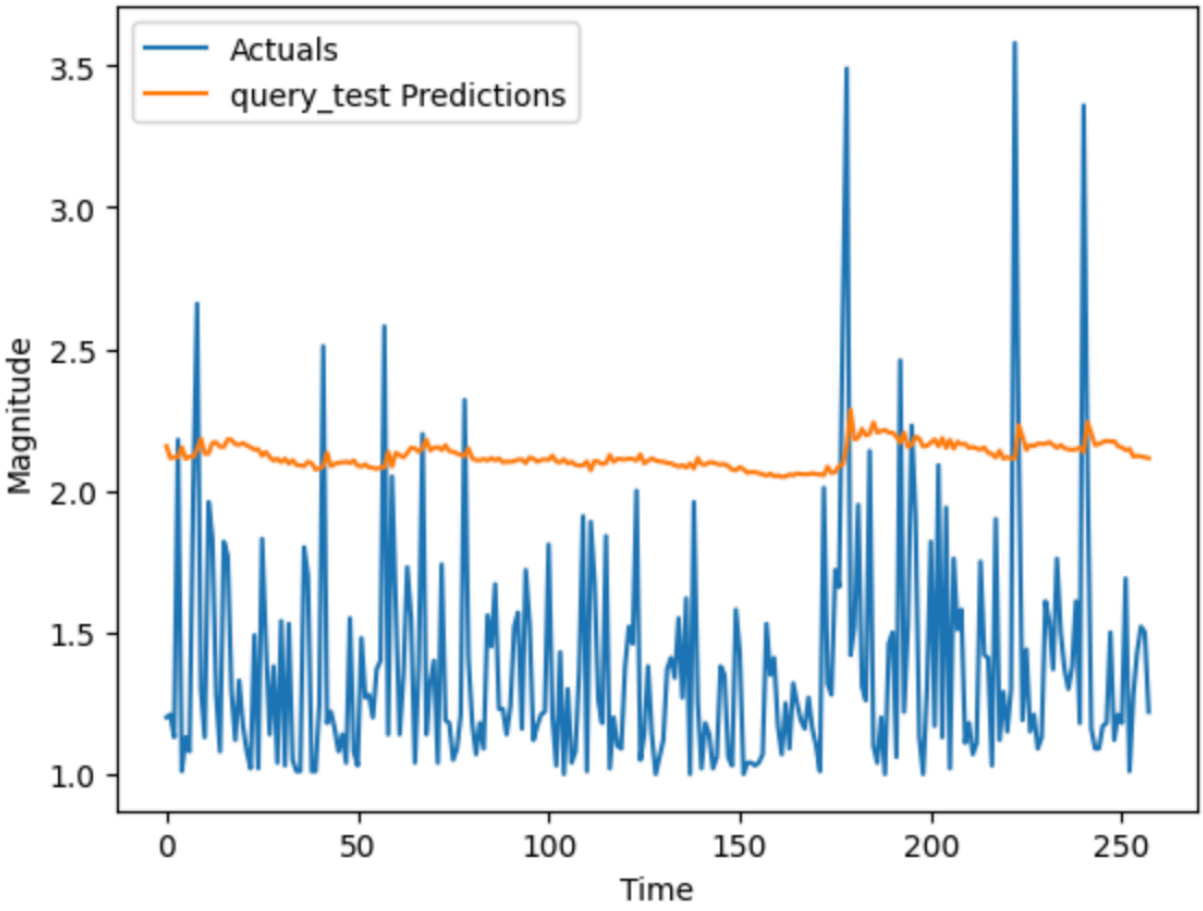

Figure 13. LSTM initial attempt with Dataset III

64-unit LSTM layer - RELU Activation Function - 200 step Look-Back Window

Next, the look back window size was increased from 50 to 200 to study the effect of a larger window on different datasets. This change means each output magnitude is computed with the consideration of the last 200 earthquakes. The LSTM model was trained on Dataset II and tested on Dataset I, II and III to observe the changes from previous results. This change decreased the MSE for Dataset I and II, and slightly increased MSE for Dataset III.

The model architecture is same as the architecture in previous section. Model was trained on the training set of dataset II, and the training MSE for best model was equal to 0.2374. Best model is then loaded to inspect and visualize performance on training data and validation data and to make predictions on test data.

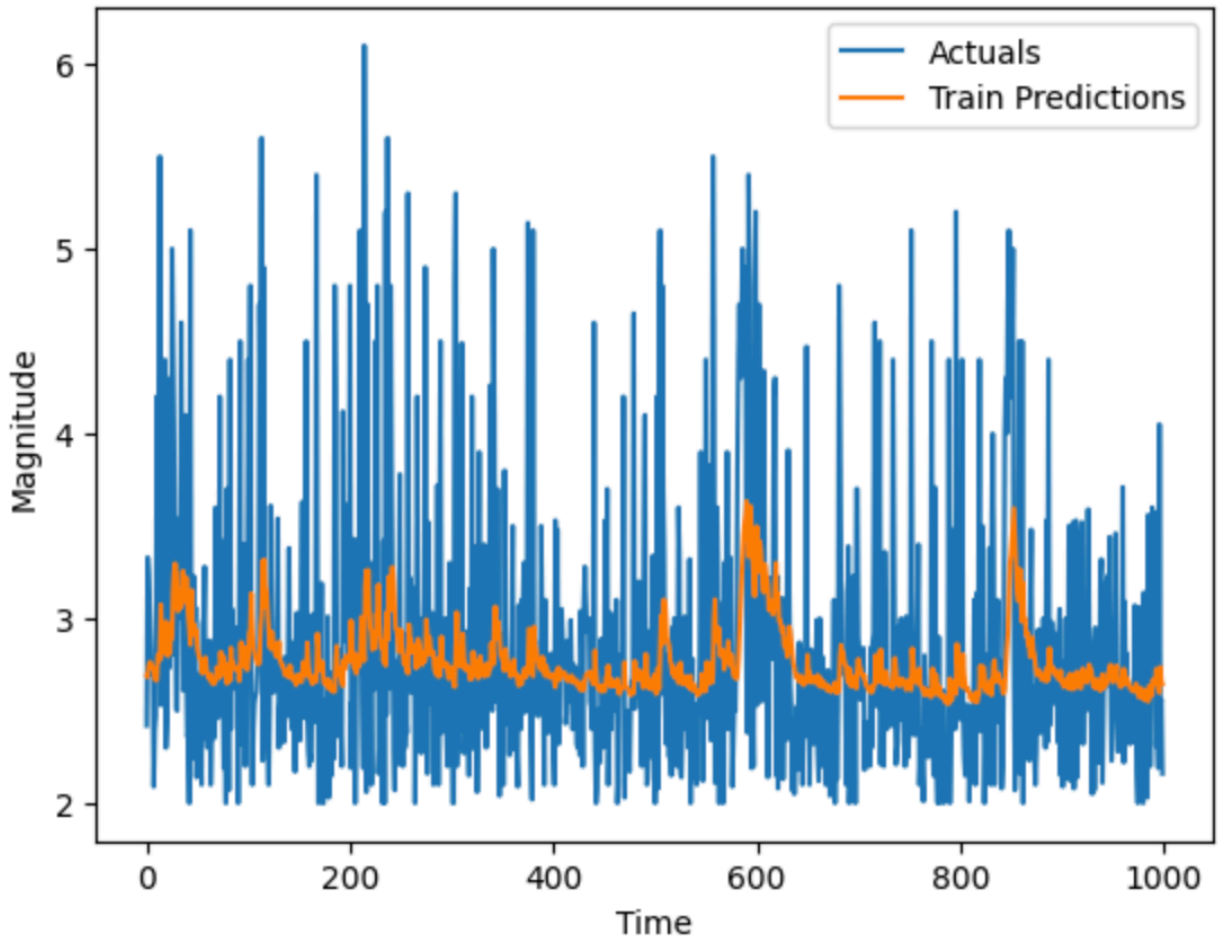

Figure 14. Comparison in between the predicted and actual values for earthquake magnitudes over time.

Next step is to read the actual values of earthquake magnitude from the validation set within dataset II and the values that the model predicted into pandas data frame for comparison. These time series are then plotted next to each other to compare the performance of the model on validation data. MSE for validation was equal to 0.2627.

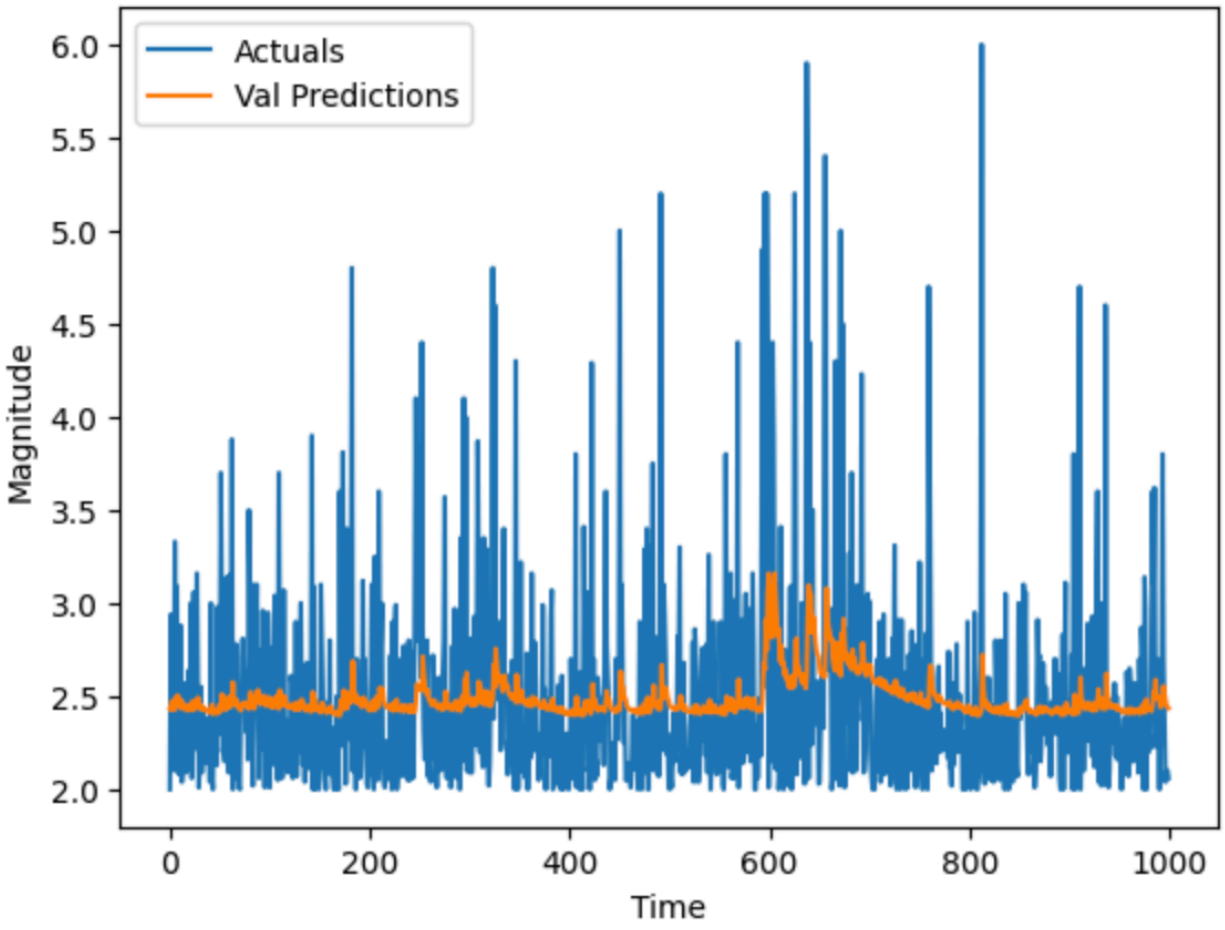

Figure 15. Comparison of actual vs predicted magnitude values within Validation set of Dataset II.

Actual values of earthquake magnitude from the test set within dataset II and the values that the model predicted are read into pandas data frame for comparison. MSE for the test set was equal to 0.2287.

Chart below shows the actual magnitude values versus model prediction on test set of dataset II.

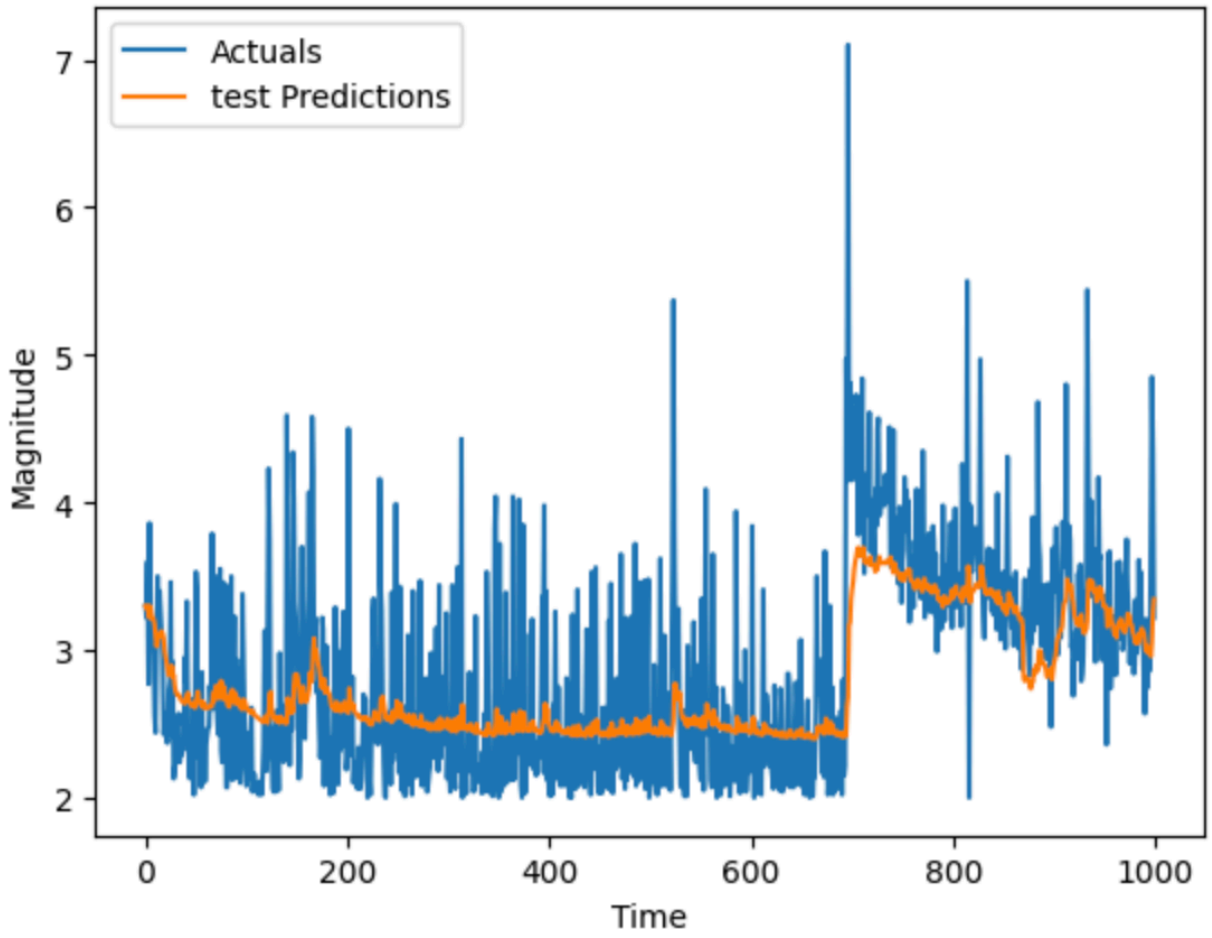

Figure 16. Comparison of actual vs predicted magnitude values within the test set of Dataset II.

Next model was tested on the entirety of dataset I to compare the performance with the other datasets in this section and previous results. MSE for testing on Dataset I is equal to 0.3735.

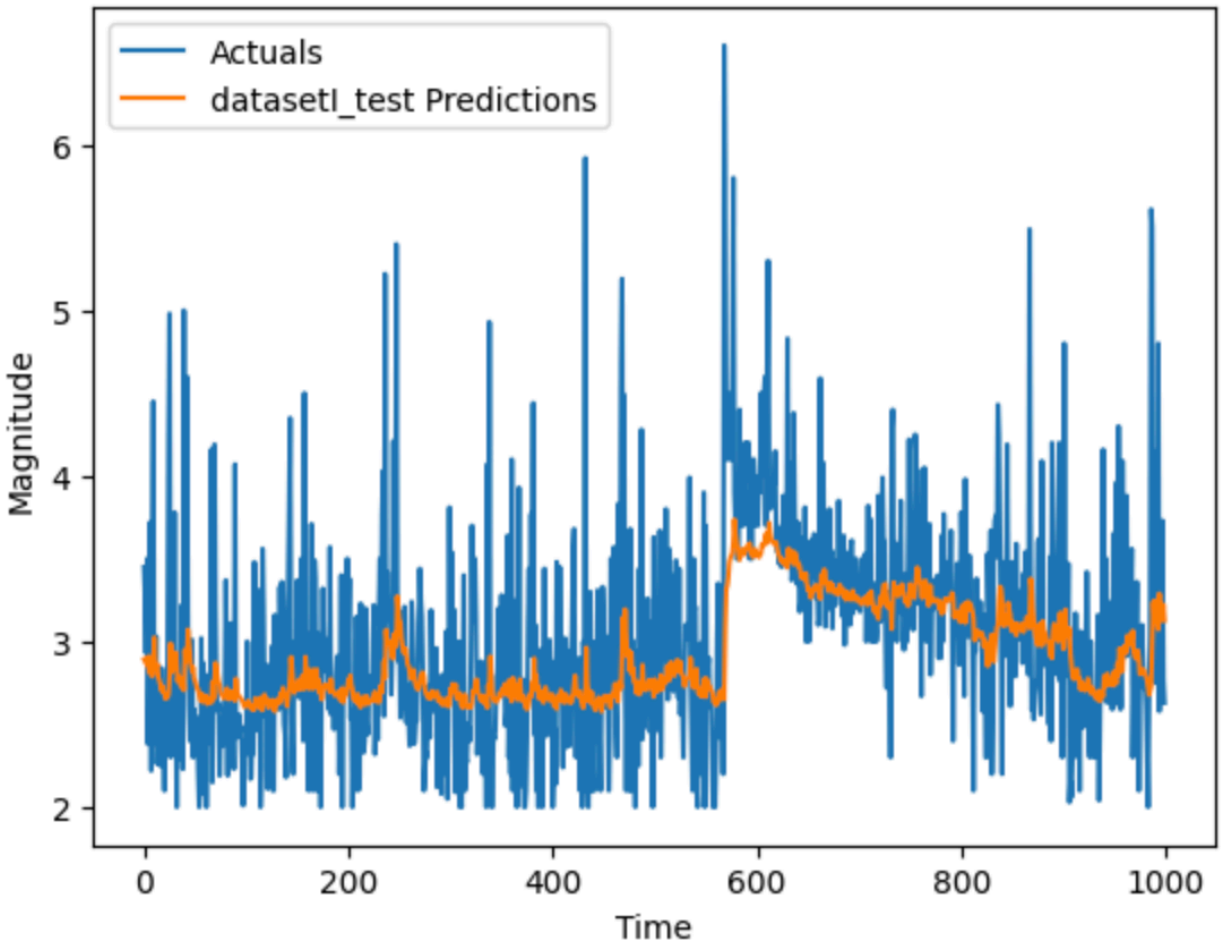

Figure 17. Actual versus predicted magnitude values for dataset I.

Next, the trained model is tested on the entire dataset III to compare the actual and model predicted earthquake magnitude values over time. Below, the actual values from dataset III are shown besides the predicted values by trained model. MSE for this case was 1.0762.

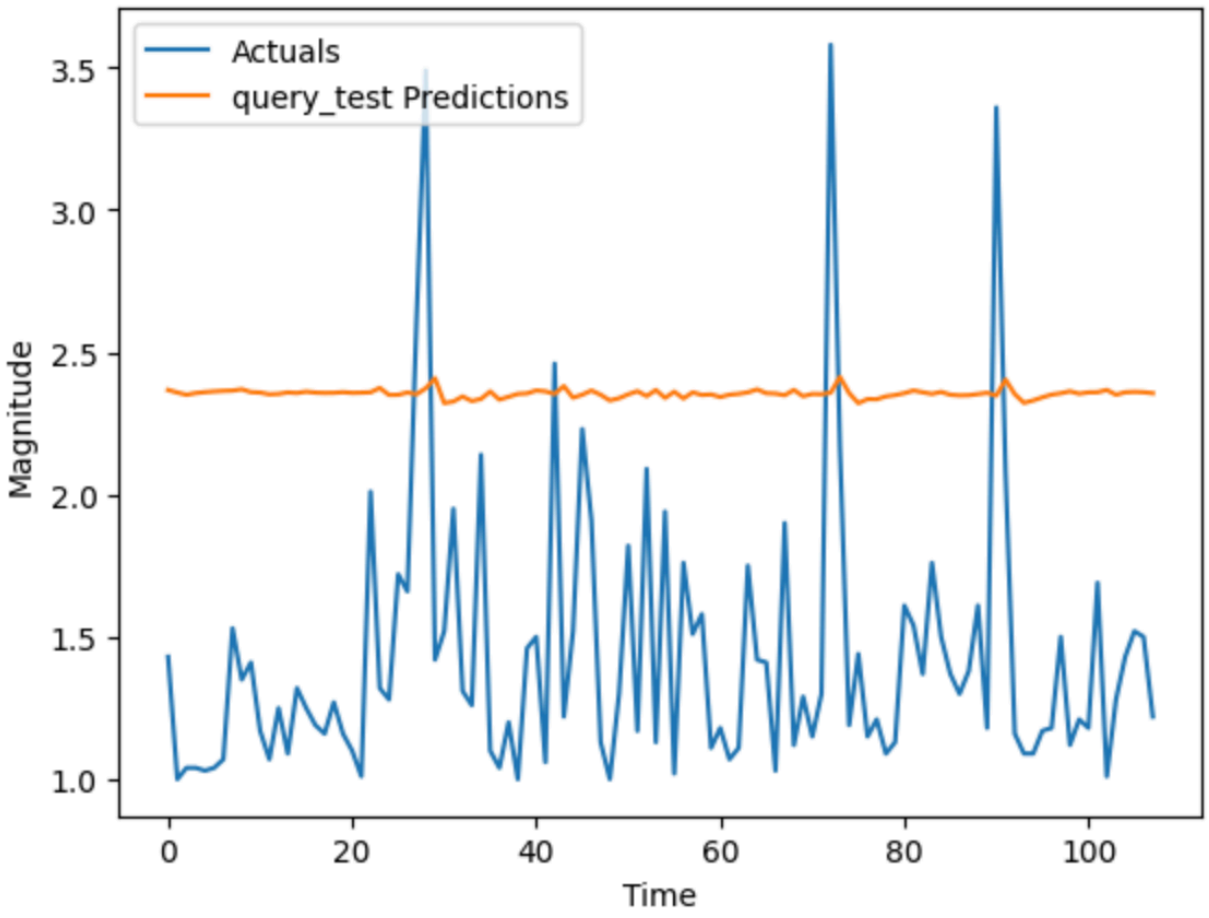

Figure 18. Actual versus predicted magnitude values for dataset III.

128-unit LSTM layer - RELU Activation Function - 200 step Look-Back Window

In this section, LSTM units were increased from 64 to 128 to study the effects of the architecture on the results. Look-back window size was set to 200. This change slightly increased the MSE for Dataset I, but results improved slightly for Dataset II. The results significantly improved for Dataset III.

The LSTM layer was updated to include 128 neurons and RELU was used as the activation function, which is similar to the previous tests. Mean Squared Error is used as loss function.
This architecture was trained on Dataset II, and tested on Dataset I, II and III. MSE for the best model with the new architecture for training was 0.2372 which is slightly better than the previous cases.

The plot below shows the predicted and actual values for magnitude over time in training set of Dataset II.

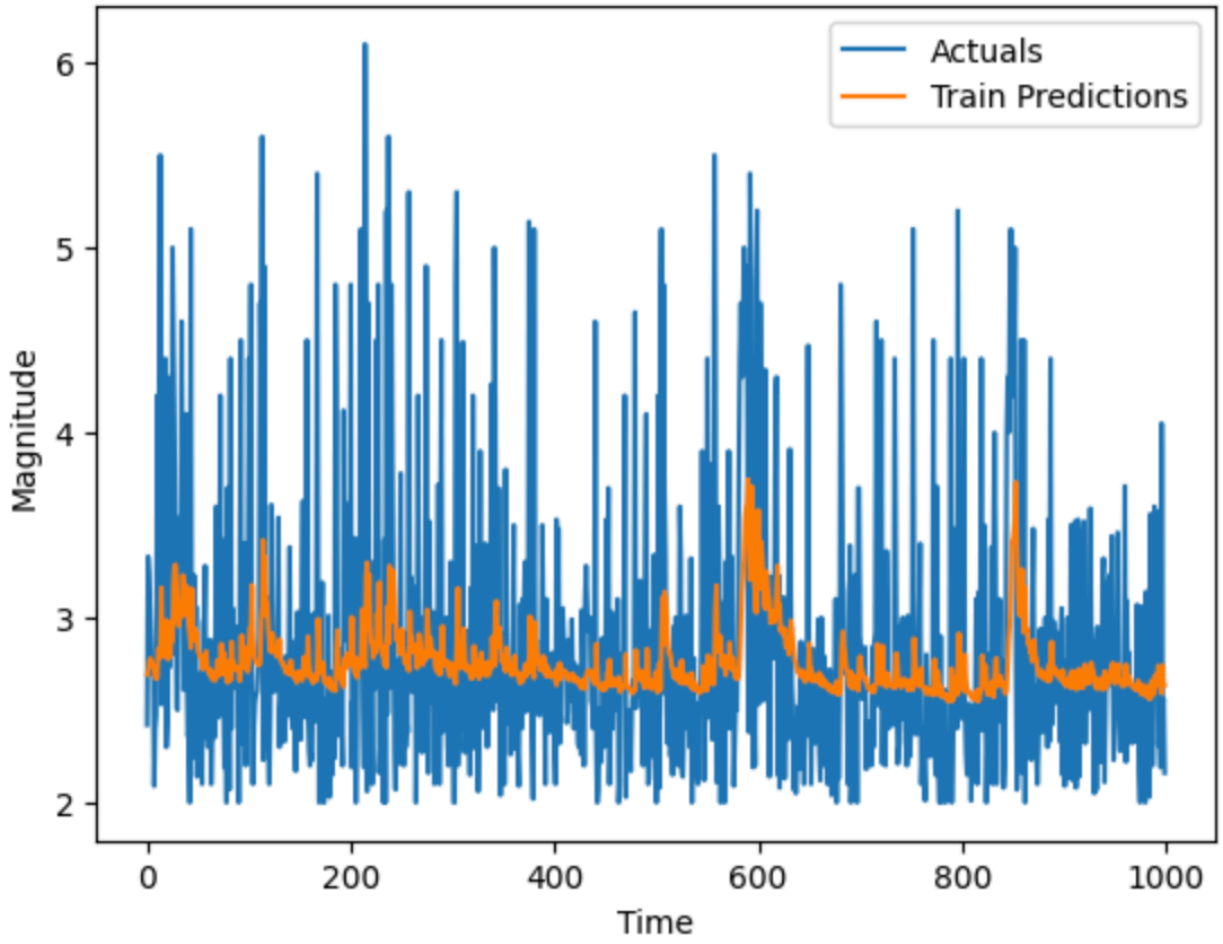

Figure 19. Training results on Dataset II

Prediction and actual values for the validation set of dataset II can be seen side by side in a dataframe in Figure 20. MSE for validation was 0.2620.

```
val_predictions = model1.predict(X_val).flatten()
val_results = pd.DataFrame(data={'Val Predictions':val_predictions, 'Actuals' :y_val})
val_results
```

```
2032/2032 ━━━━━━━━━━━━━━━━━━━━ 200s 99ms/step
```

	Val Predictions	Actuals
0	2.420008	2.00
1	2.415397	2.94
2	2.460172	2.10
3	2.420716	2.40
4	2.437750	2.14
...	...	...
64995	3.225281	3.80
64996	3.310239	4.13
64997	3.413354	3.54
64998	3.365486	3.42
64999	3.326468	3.13

65000 rows × 2 columns

Figure 20. Validation results on Dataset II.

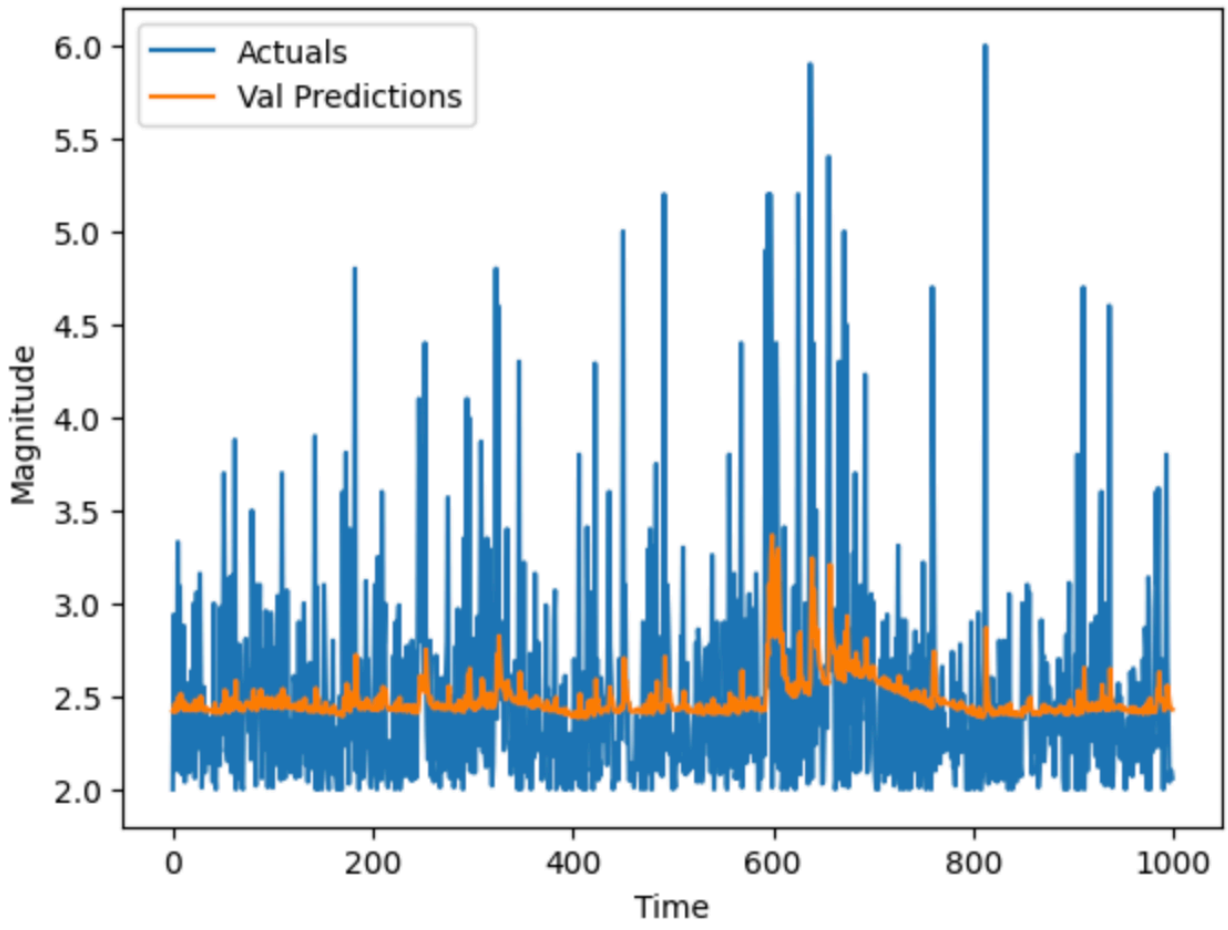

Figure 21. Validation results on Dataset II.

The best model is then tested on test set of dataset II. MSE in this case was equal to 0.2279. Plotting actual and predicted magnitude values show that the trend of the time-series as well as the peaks are successfully identified.

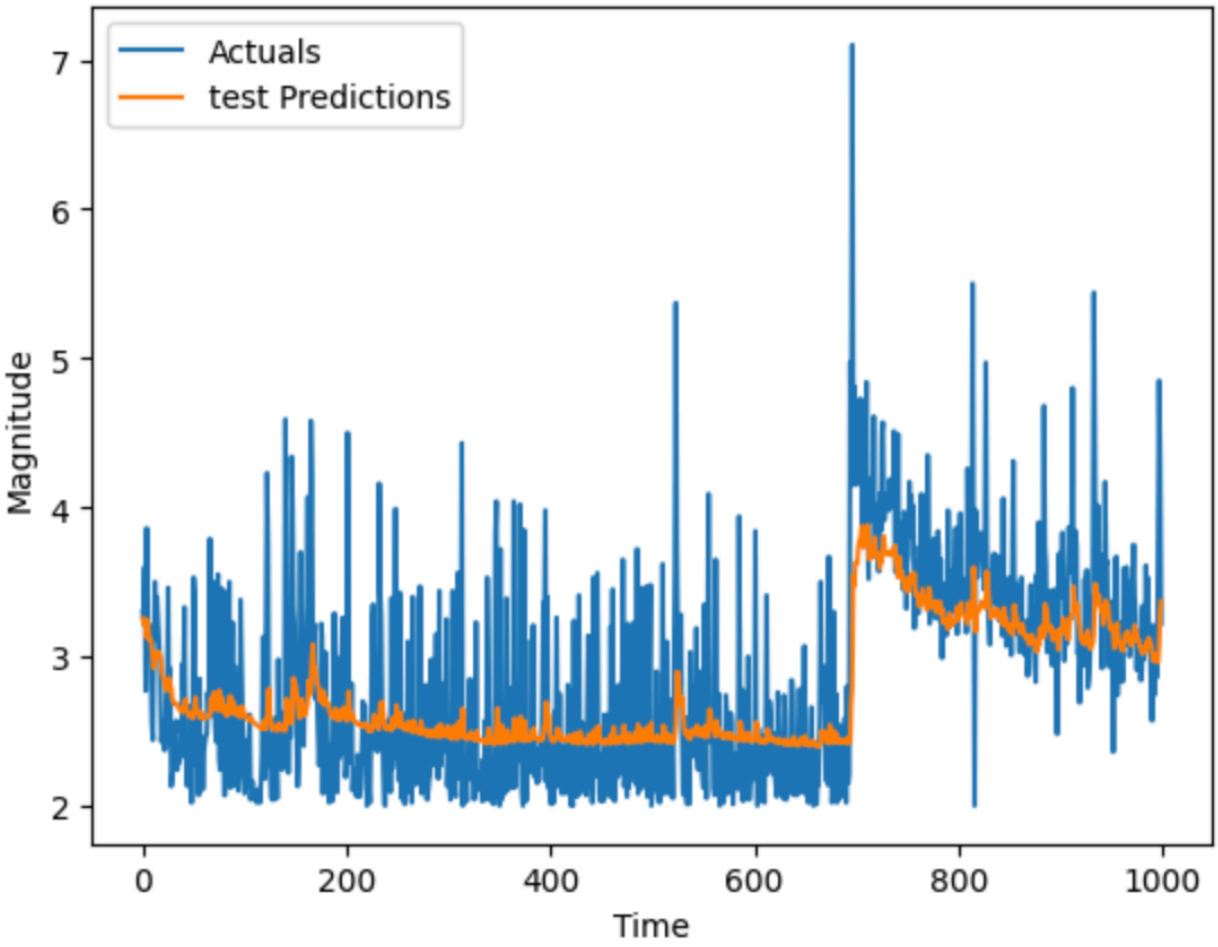

Figure 22. Test results on Dataset II

This model is then tested on the entire dataset III. MSE in this case was equal to 0.9049.

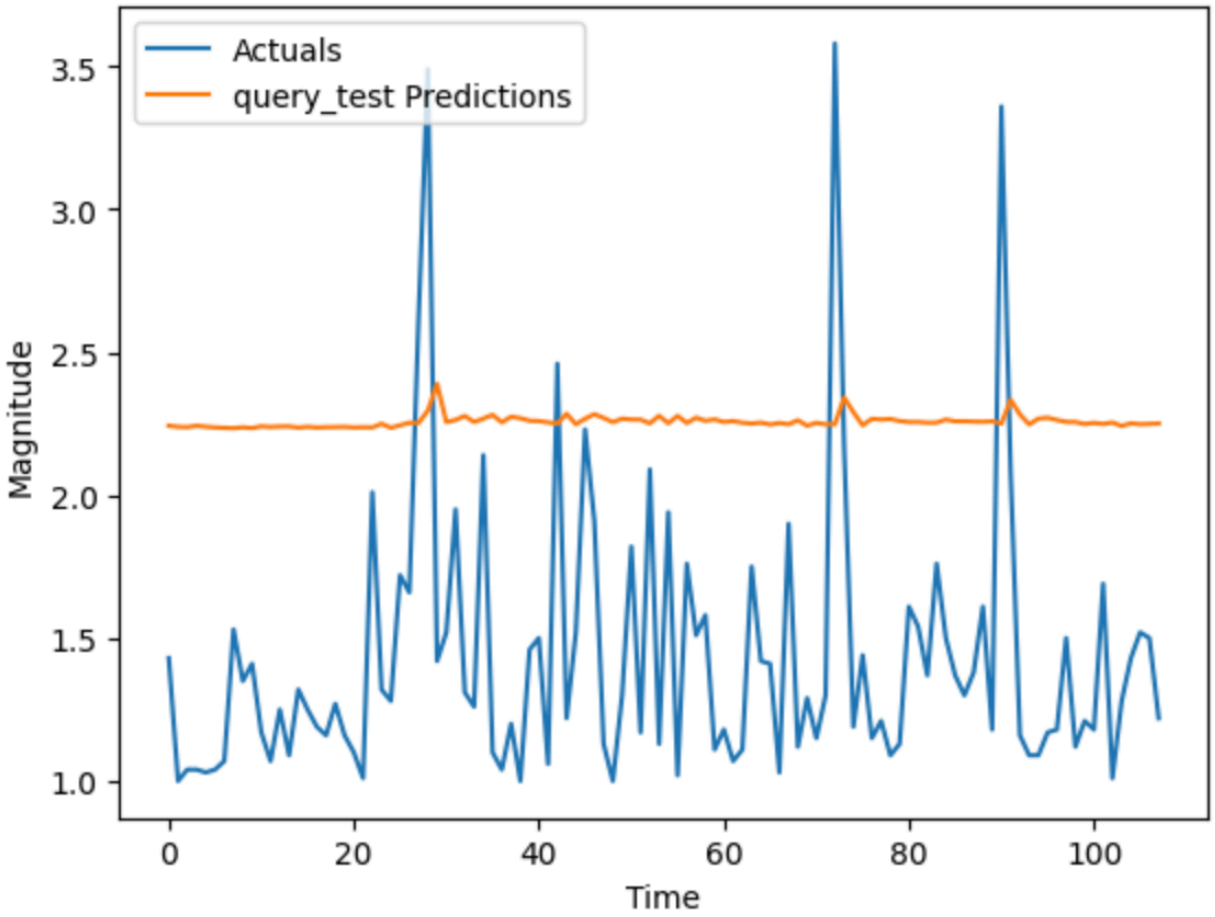

Figure 23. Test results on Dataset III

Lastly, the model is tested on dataset I and the predicted and actual values are shown in a dataframe below. MSE in this case was equal to 0.3737. This plot is showing the actual and predicted magnitude for dataset I. It is seen that the model is successfully predicting the trend and peaks of the time-series data.

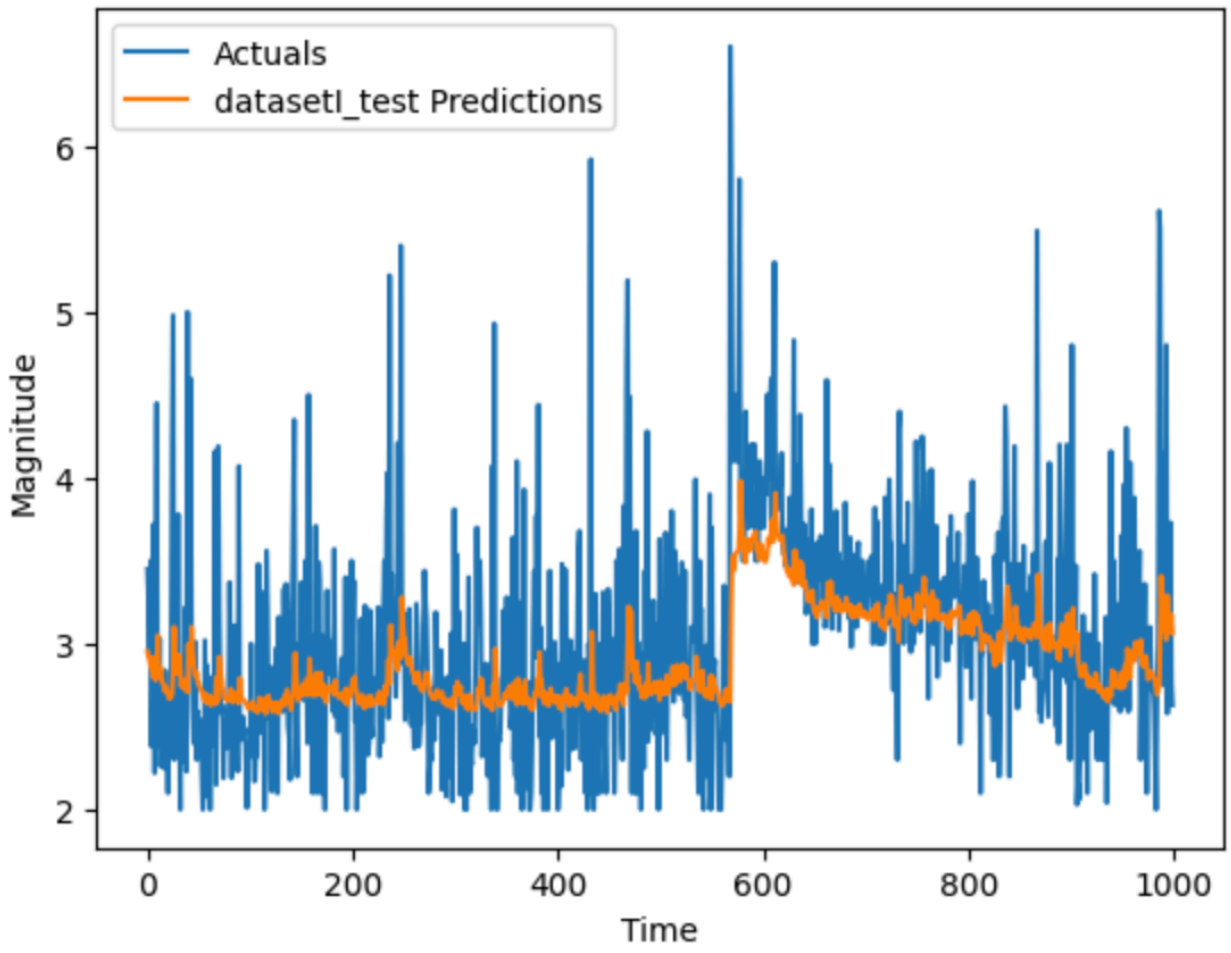

Figure 24. Test results on Dataset I

64-unit LSTM layer - Sigmoid Activation Function - 50 step Look-Back Window

In this attempt, a new activation function, Sigmoid is used to study how activation function might affect the results. In this case, the look-back window size is set to 50. The results show that Sigmoid increased MSE for all datasets and decreased the Model performance.

The LSTM model architecture is same as what was used in the first experiment. The activation function is changed from RELU to Sigmoid to better understand the effects of activation functions on LSTM model performance. At this step, the model metrics, the loss function, optimizer and learning rate are set.
This model is run on the training set of Dataset II for 10 epochs and the best model is saved with training MSE of 0.2401. Best model is loaded to inspect and visualize performance on training data and validation data and to make predictions on test data. Below the predicted and actual values of magnitude for training set of Dataset II are read into a dataframe for better comparison.

In the following plot, it is shown that the trained model with Sigmoid activation function can successfully predict the trend and identify data peaks within training set for dataset II.

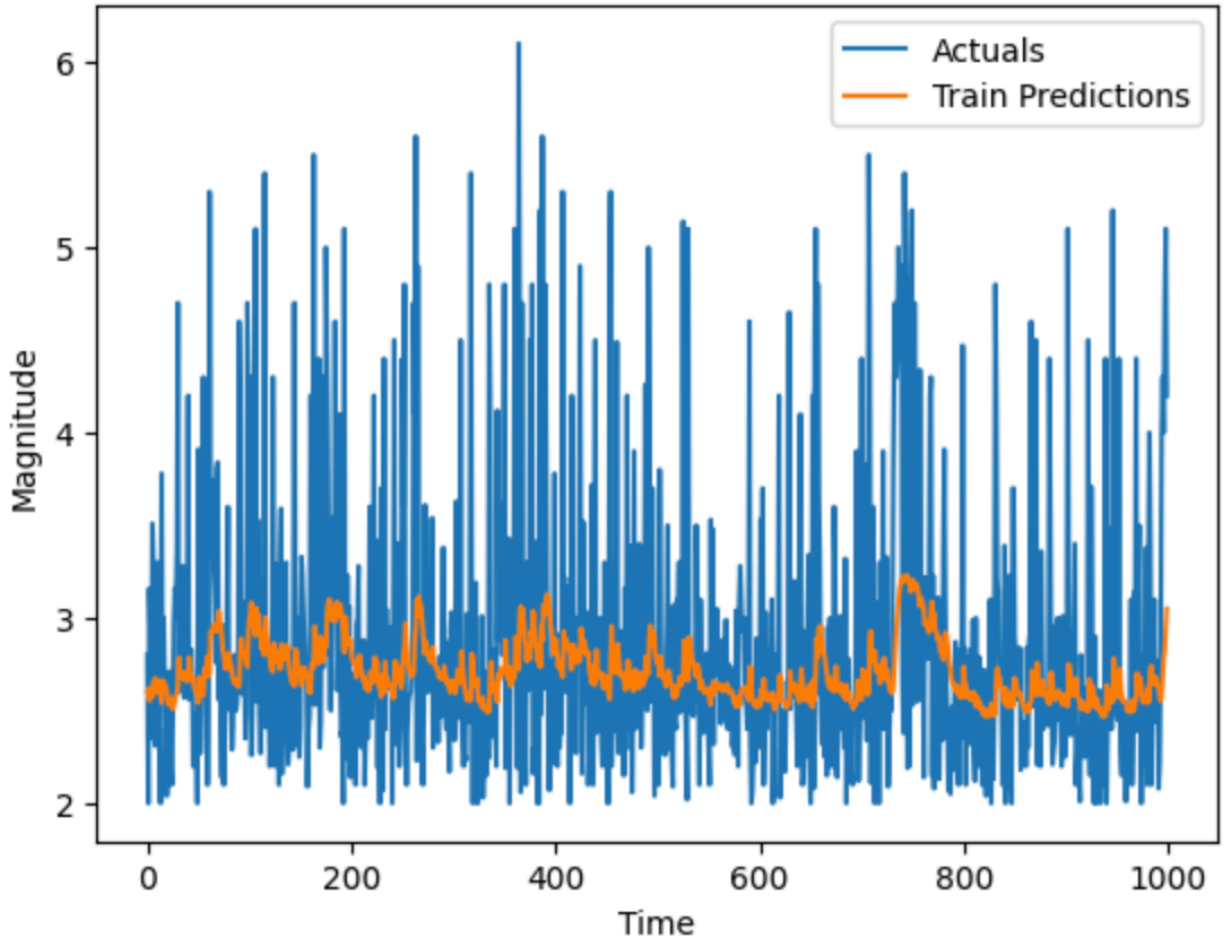

Figure 25. Comparison of predicted and actual values within training set of dataset II

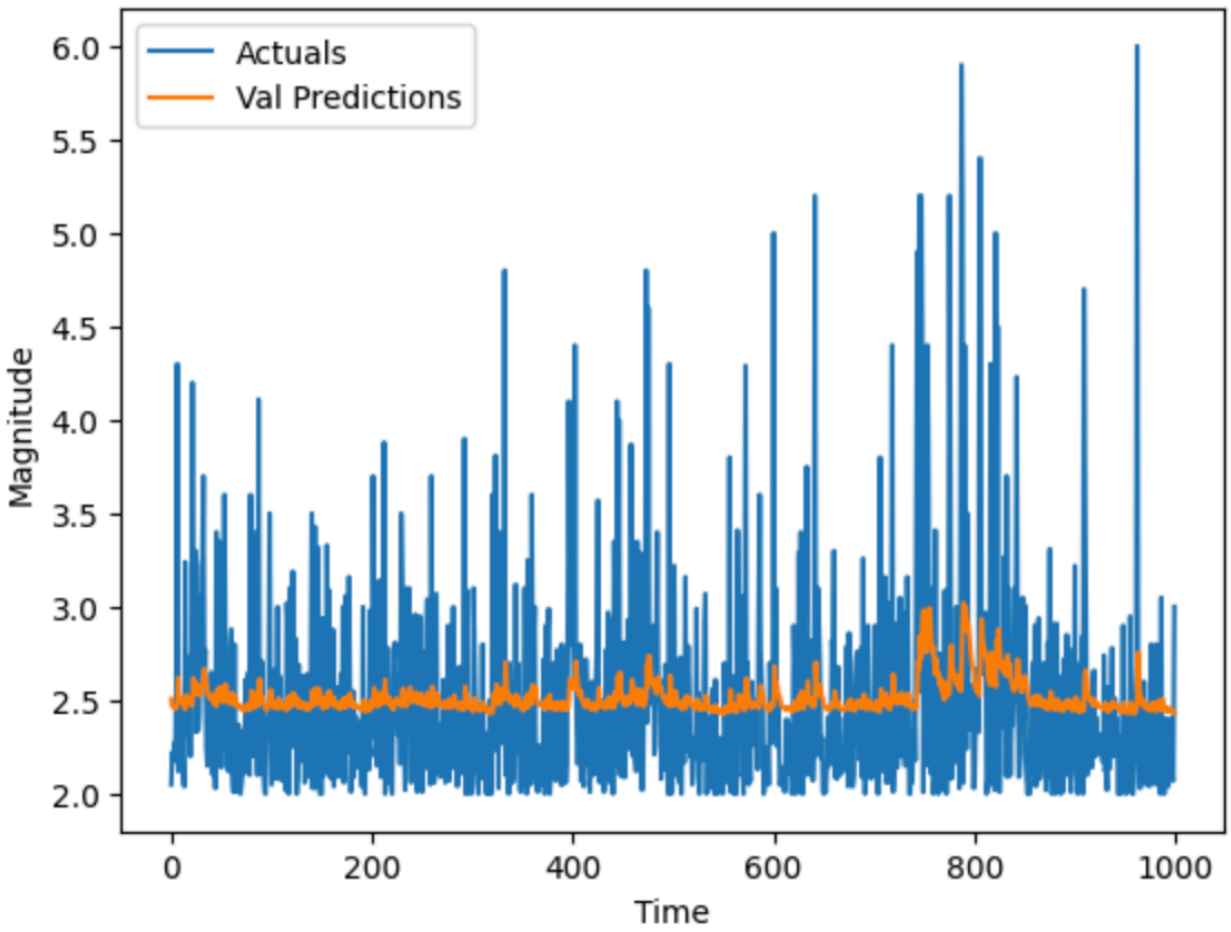

Figure 26. Validation results on Dataset II

The next step in machine learning flow is to test the model on the unseen test data. Here, the best model is making predictions on test set of Dataset II. MSE in this case was equal to 0.2338.

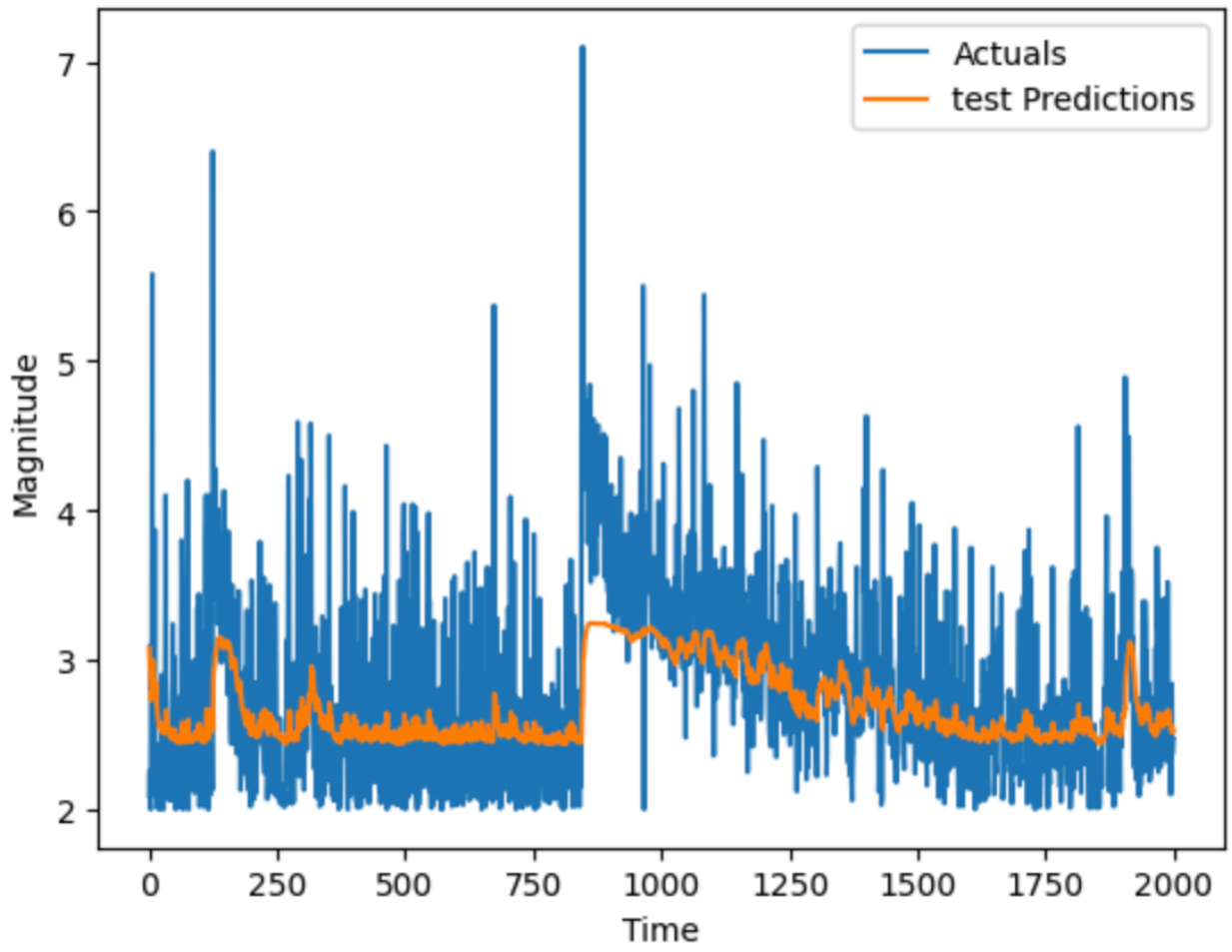

Figure 27. Test results on Dataset II

Model was then tested on the entire Dataset III to observe the effects of Sigmoid activation function on a smaller Dataset. Test results on Dataset III are visualized in the plot below. MSE in this case was equal to 1.1266.

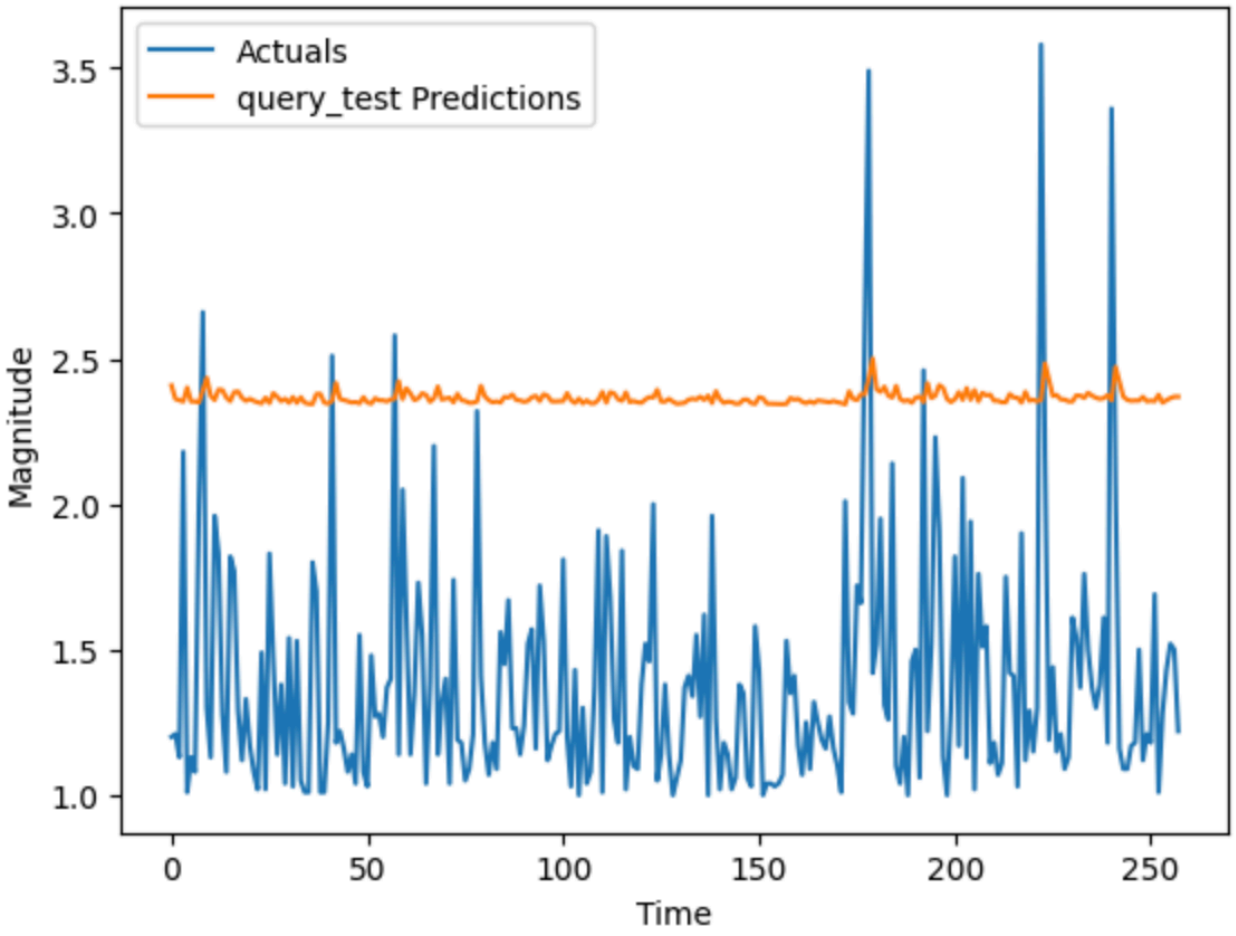

Figure 28. Test results on Dataset III

A similar test was performed on Dataset I to observe the model performance. Model has successfully predicted the trend and peaks on a different medium size Dataset. MSE in this case was equal to 0.4121.

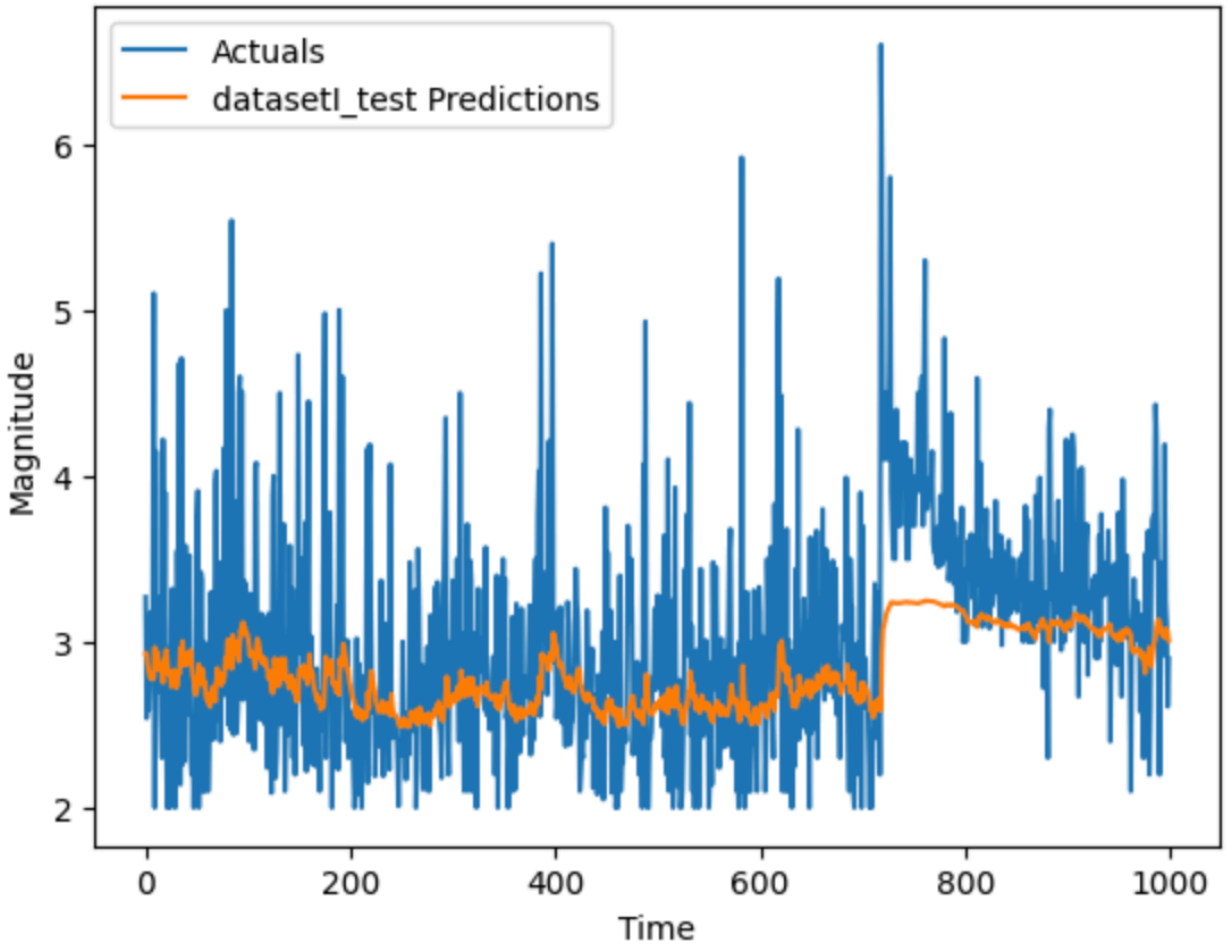

Figure 29. Visualization of test results on Dataset I with Sigmoid activation function.

64-unit LSTM layer - Tanh Activation Function - 50 step Look-Back Window

In this section, the activation function is changed from Sigmoid to Tanh with the purpose of studying the effects of different activation functions on LSTM performance. Look-back window size is set to 50. It was observed that Tanh dropped the model performance when compared to Relu and Sigmoid. Best model with minimum MSE should be saved to be applied to test Datasets.
In the case of LSTM with Tanh activation function, best model was trained with 0.2380 MSE.

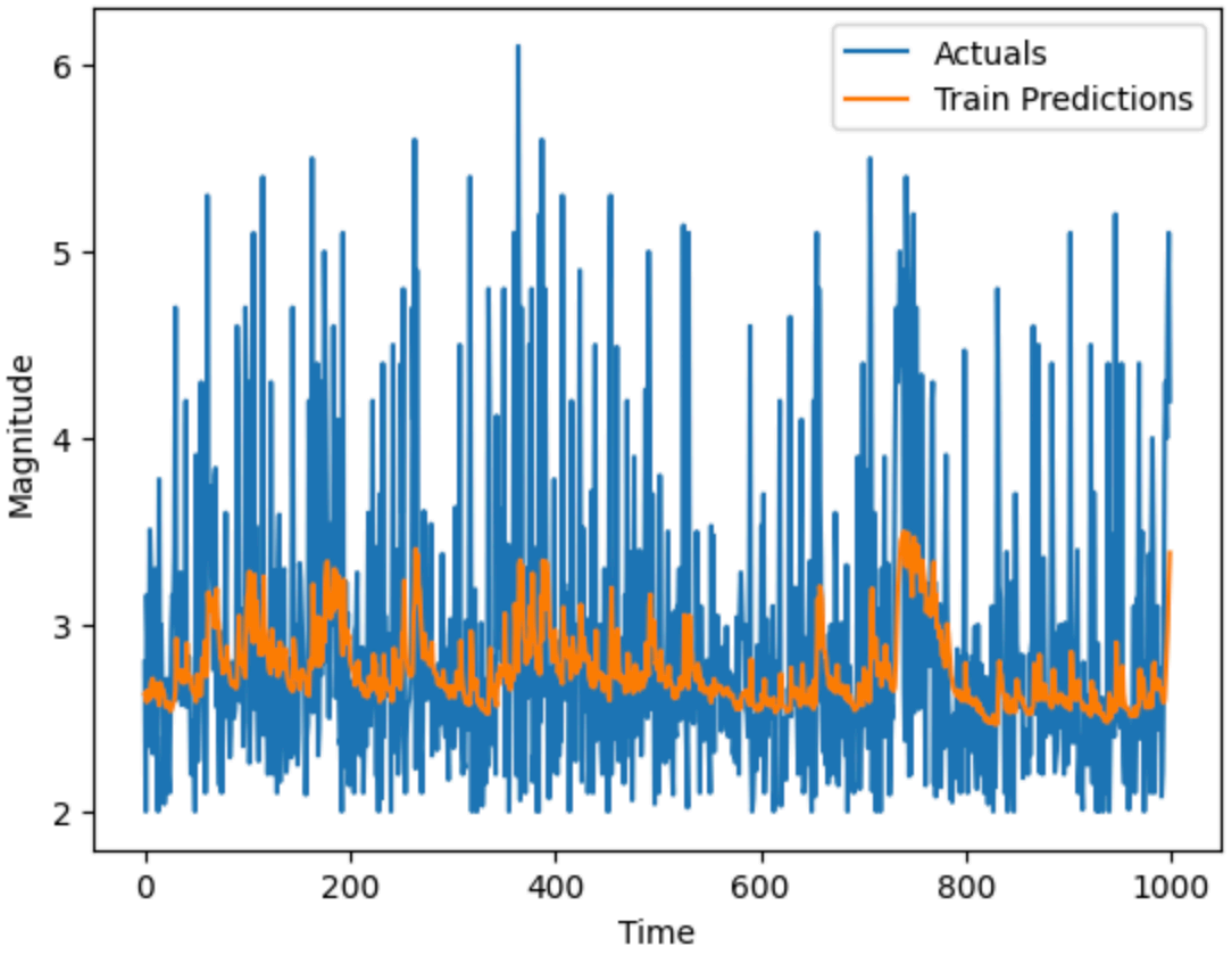

Figure 30. Comparison of predicted and actual values within training set of Dataset II

For the model with Tanh activation function on validation set of Dataset II, MSE was 0.2613.

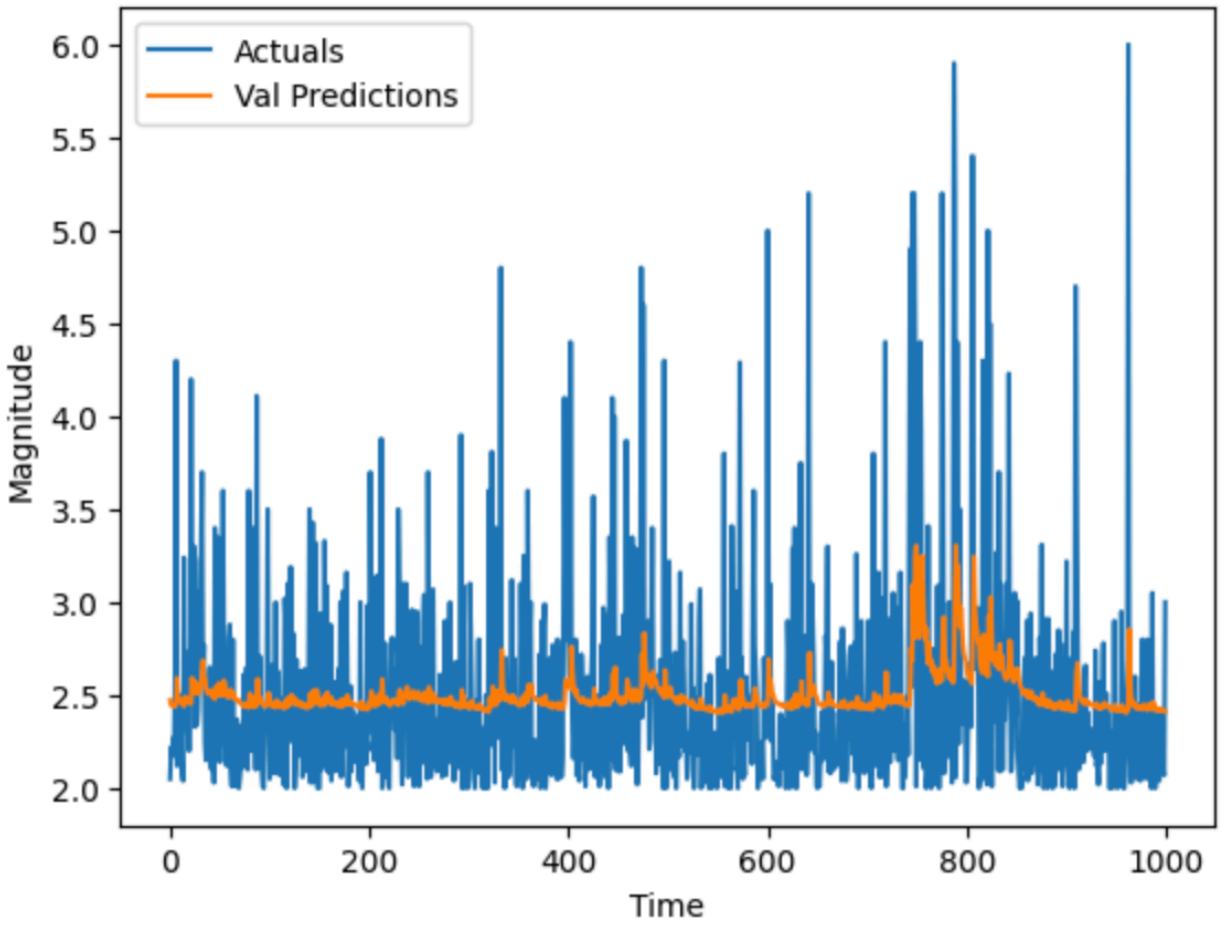

Figure 31. Validation results on Dataset II

The model was then tested on test set of Dataset II, and it is observed that the time-series trend is successfully determined, and peak data points are identified. MSE for this case is equal to 0.2299.

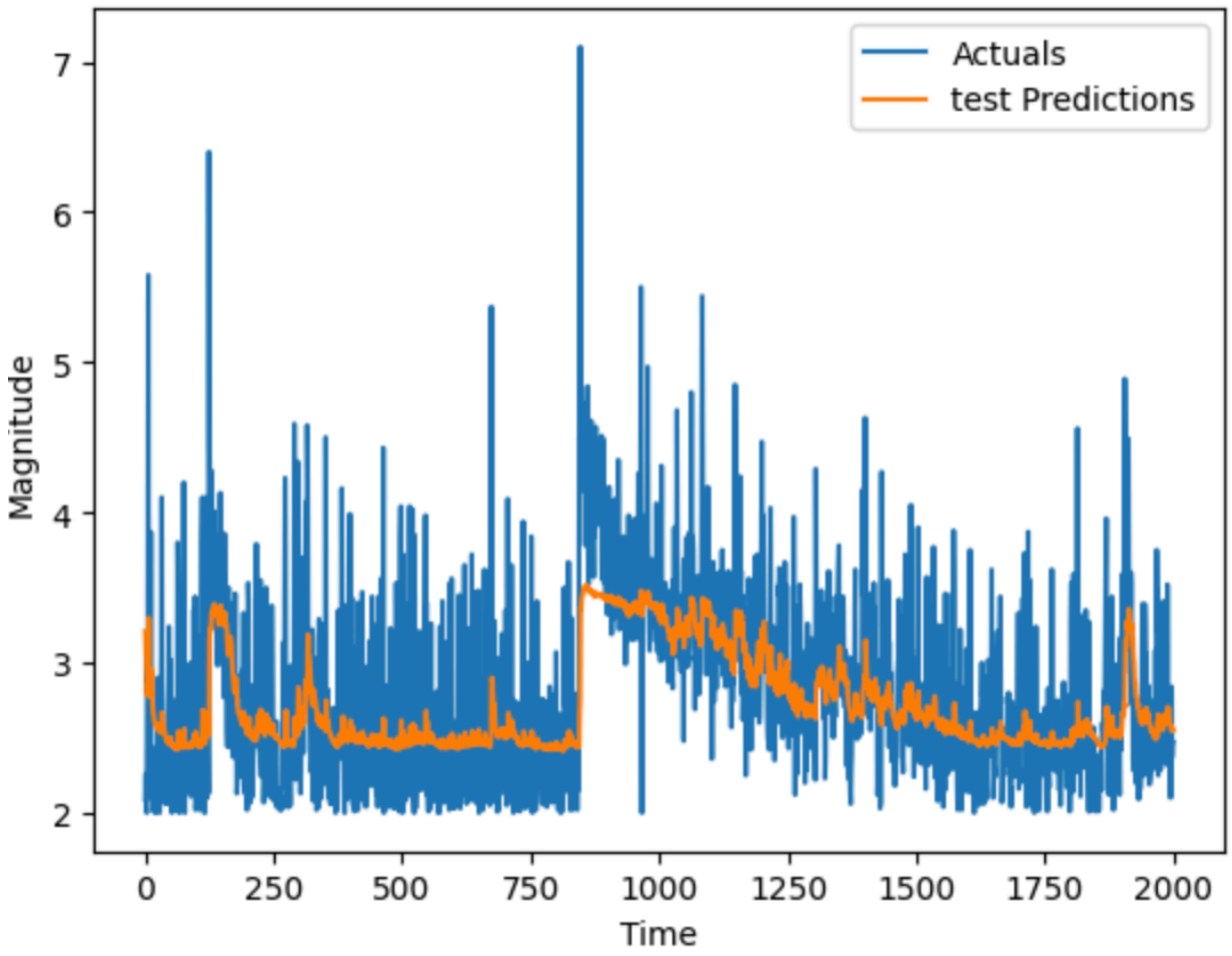

Figure 32. Test results on Dataset II

Trained model with Tanh activation function was tested on Dataset III to observe the effects of utilizing Tanh activation function on a smaller Dataset. MSE for testing on Dataset III is equal to 1.0608.

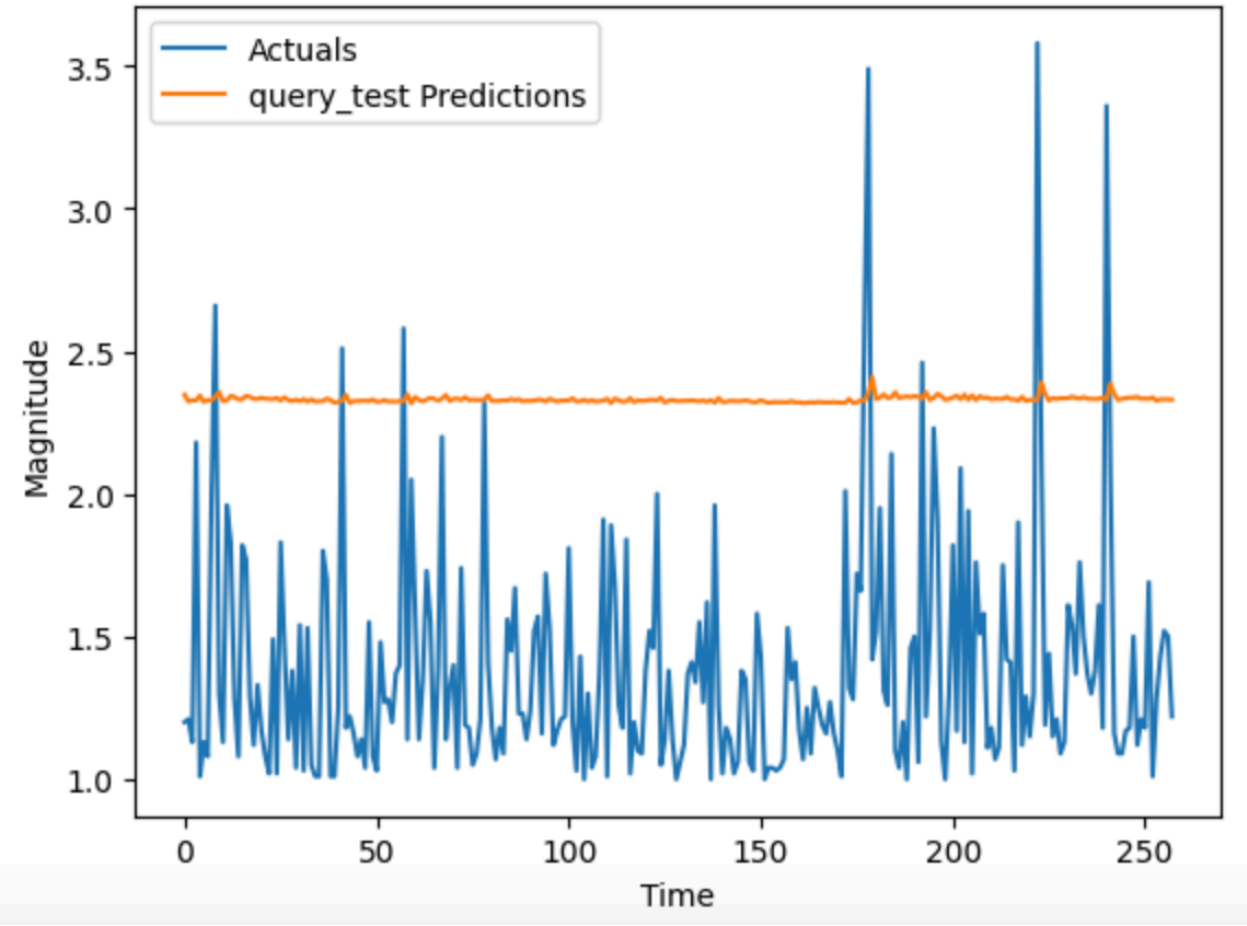

Figure 33. Test results on Dataset III

The trained model was tested on Dataset I. MSE for testing on Dataset I is equal to 0.3938. The results are plotted against time in the plot below and the model performance can be compared with the actual values. Similar to previous cases, trend and data peaks are successfully predicted.

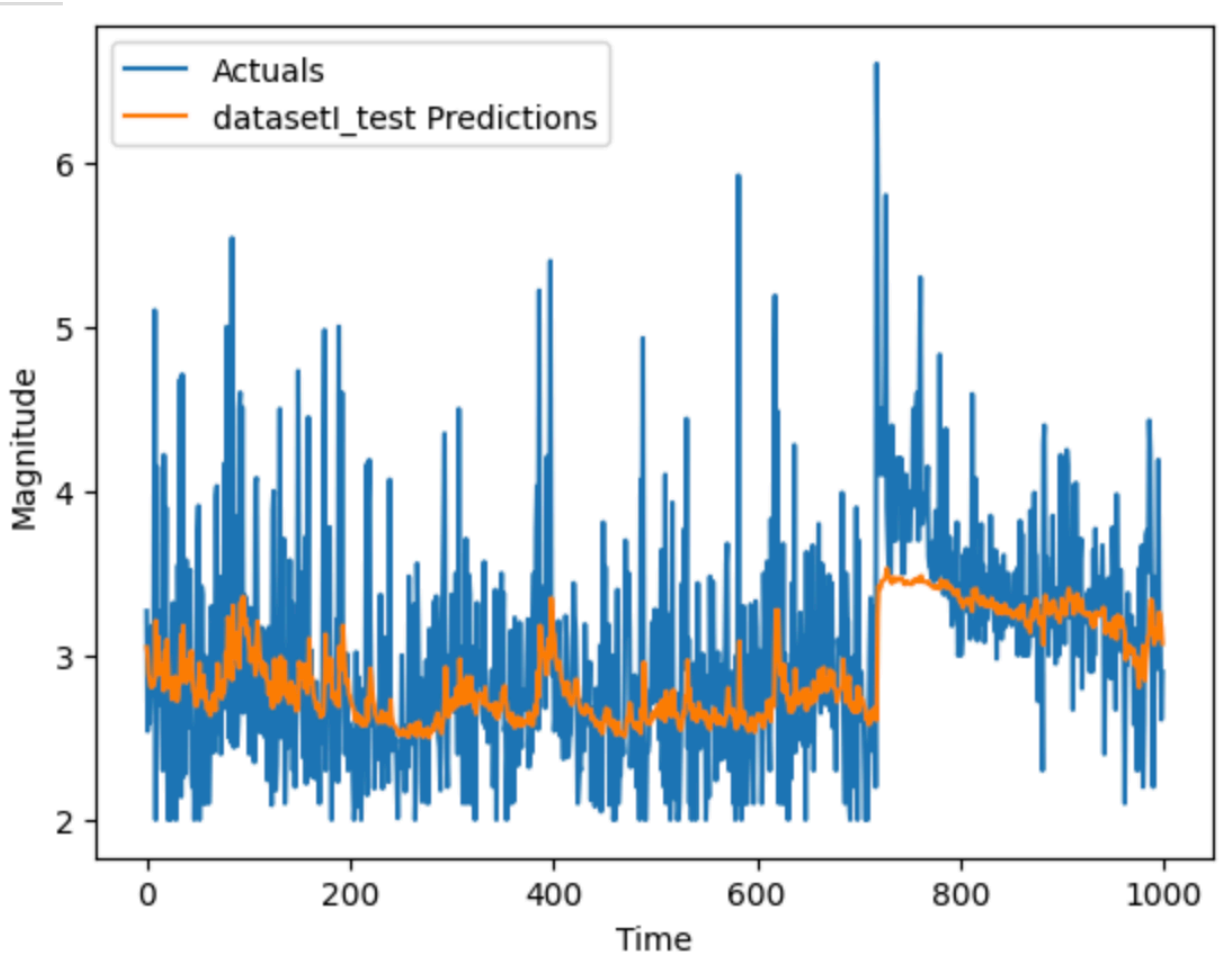

Figure 34. Visualization of test results on Dataset I with Tanh activation function.

4.2 Interpolation

In this section, interpolation will be applied to Datasets to assess the effect of resampling data with different intervals on LSTM performance on irregular time-series Datasets. Due to the nature of earthquakes and studies of earthquake swarms, it is assumed that two consecutive magnitudes are closely correlated, and the nearest neighbor has been used to fill the missing values. In the case of single earthquake data, the earthquake magnitude measured exactly before the missing value has been used to represent the missing value for the magnitude.

4.2.1 4-hour Interpolation Sampling Temporal Interval on Dataset II

As an initial attempt of interpolation, Dataset II was resampled with a temporal interval of 4 hours. The Dataset was resampled with a temporal interval of 4 hours. Column mag is read into a dataframe, and input and outputs are created. LSTM look-back window is set to 50, and new

dimensions of data is demonstrated. Dataset is divided into training (%70), validation (%20) and test (%10) sets.

LSTM sequential model is then created, trained and tested on Dataset II and Dataset III.
This model was run for 10 epochs and trained on resampled data from Dataset II. It was tested on Dataset II and III. MSE for the best model with interpolation with 4-hour interval was 0.2460 for training. Comparison of actual values and results are shown below.

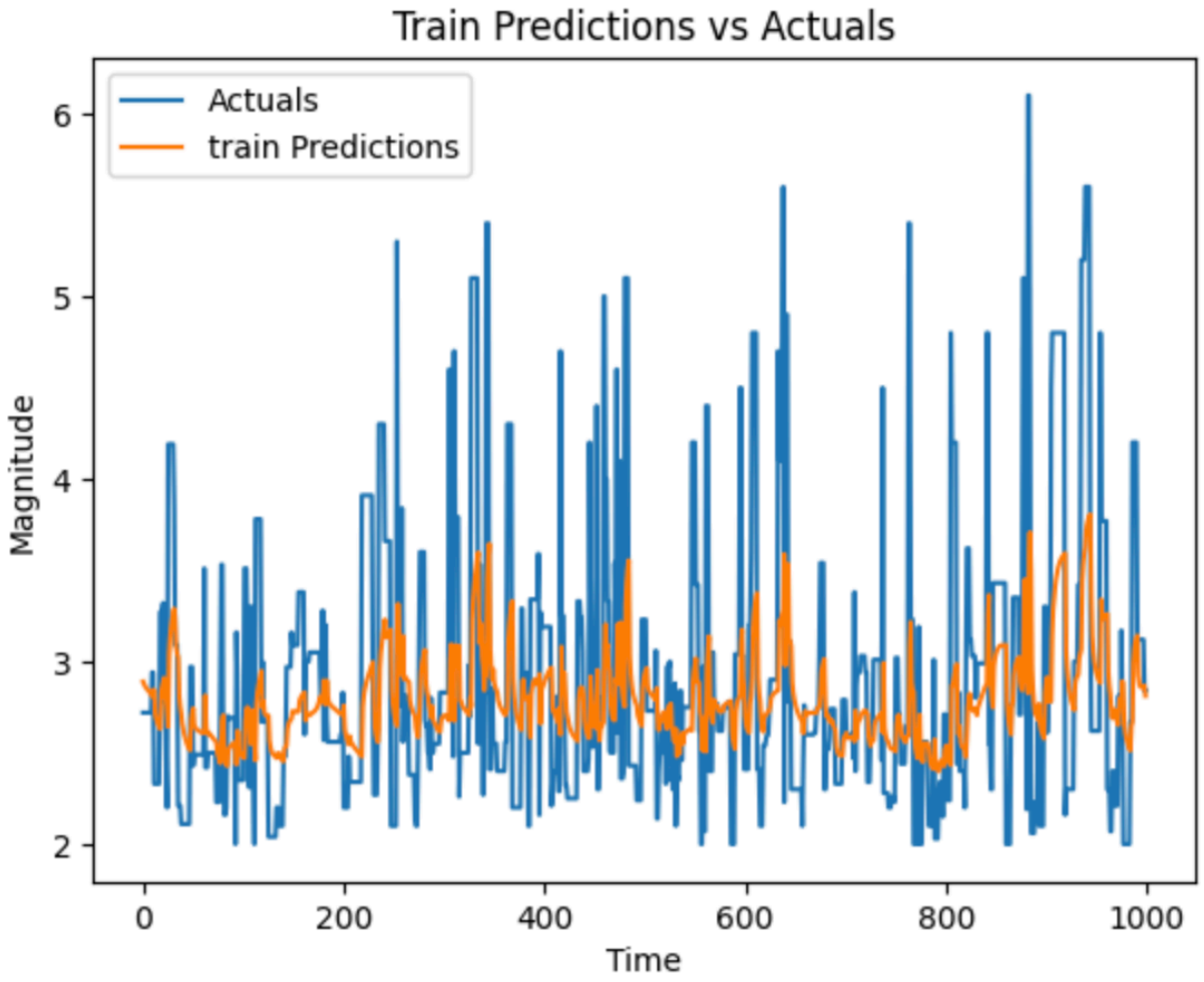

Figure 35. Visualization of training results on Dataset II.

After training the LSTM network on training set with interpolation with 4-hour interval, model was applied to validation set of Dataset II and the resulting MSE for the best model was 0.2442.

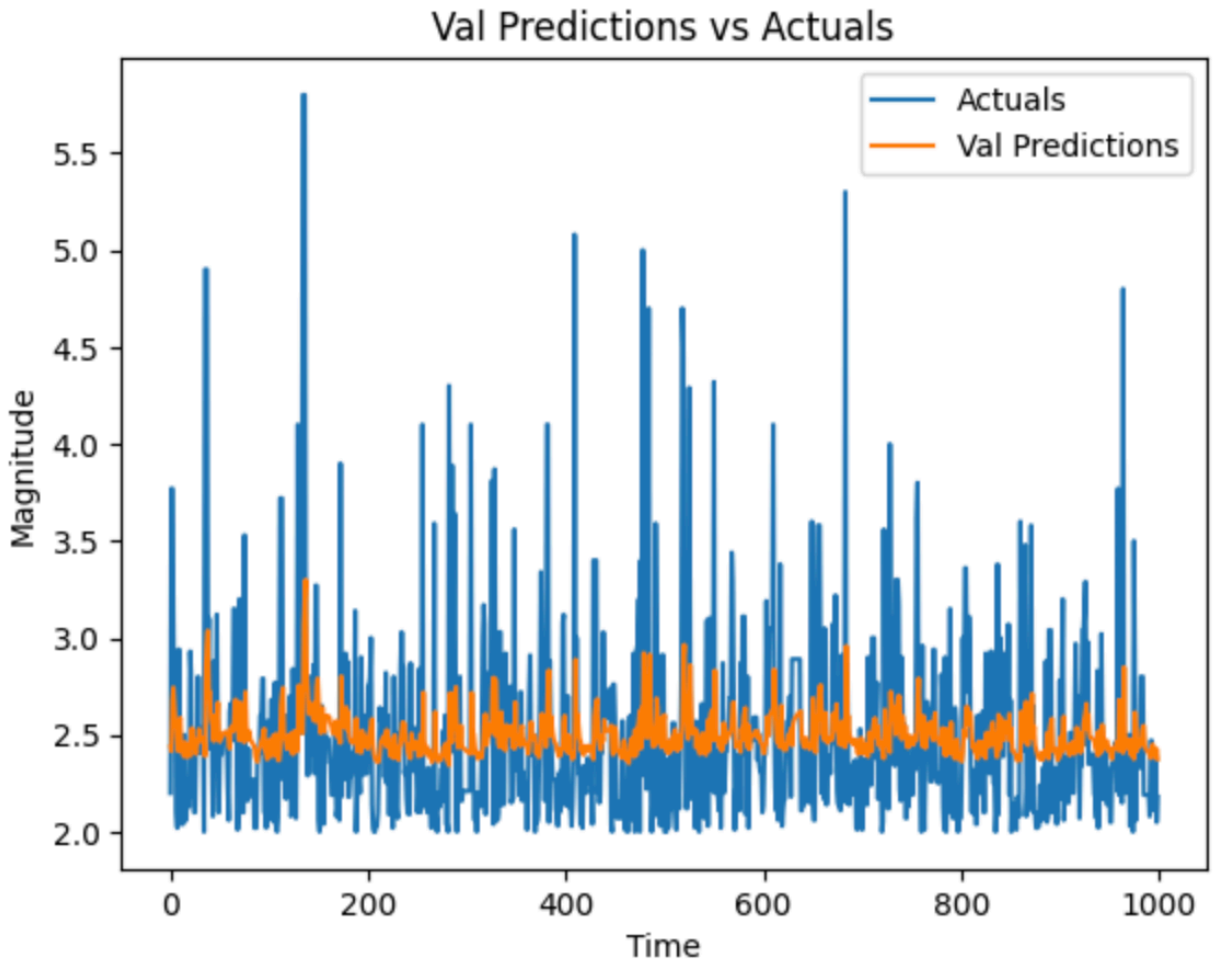

Figure 36. Visualization of validation results on Dataset II.

Finally, the trained LSTM network was tested on the test set of Dataset II and the obtained MSE was equal to 0.2303.

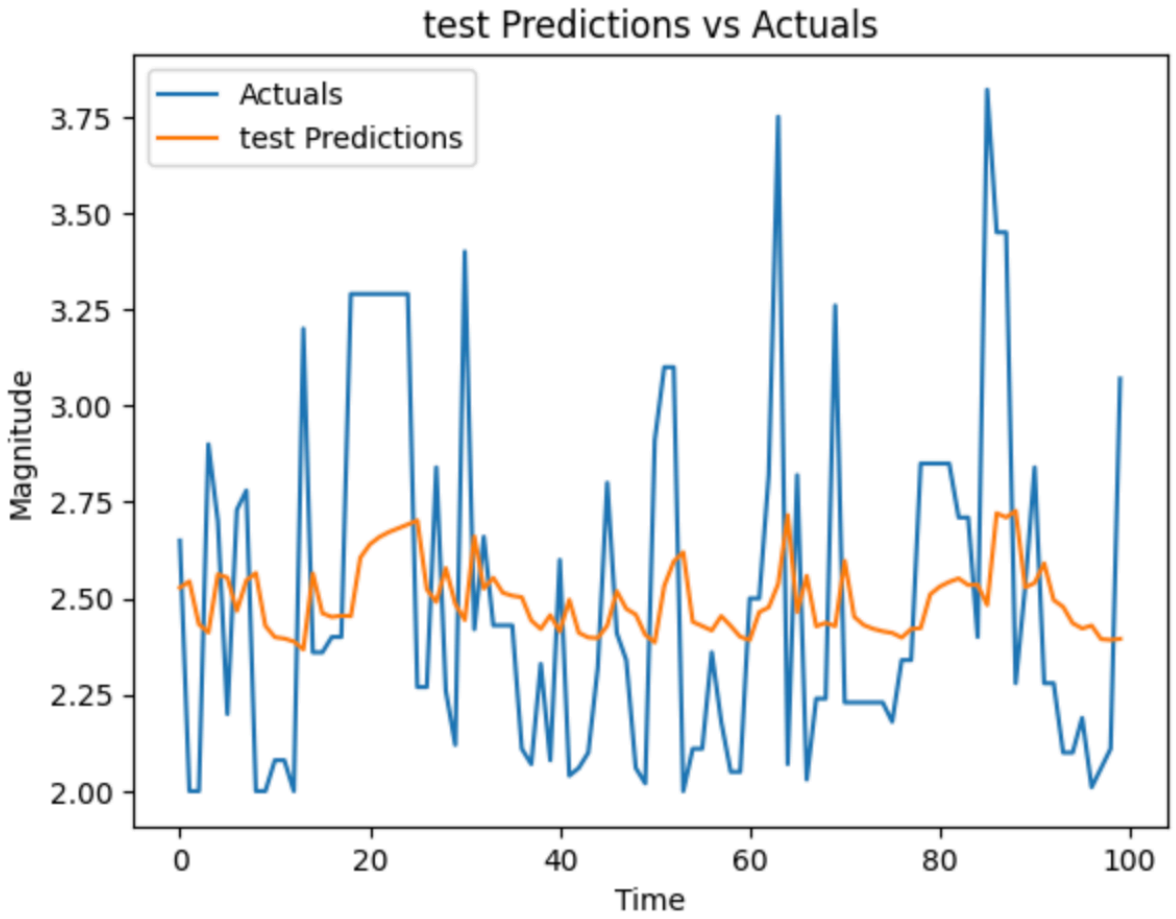

Figure 37. Visualization of test results on Dataset II.

This model was then tested on Dataset III to study effects of testing a model trained with interpolated data on seismic data that was not interpolated. MSE for this case was 0.5695. We realize that interpolation significantly improved the results and decreased the MSE on Dataset II, while it did not yield to the best results for Dataset III.

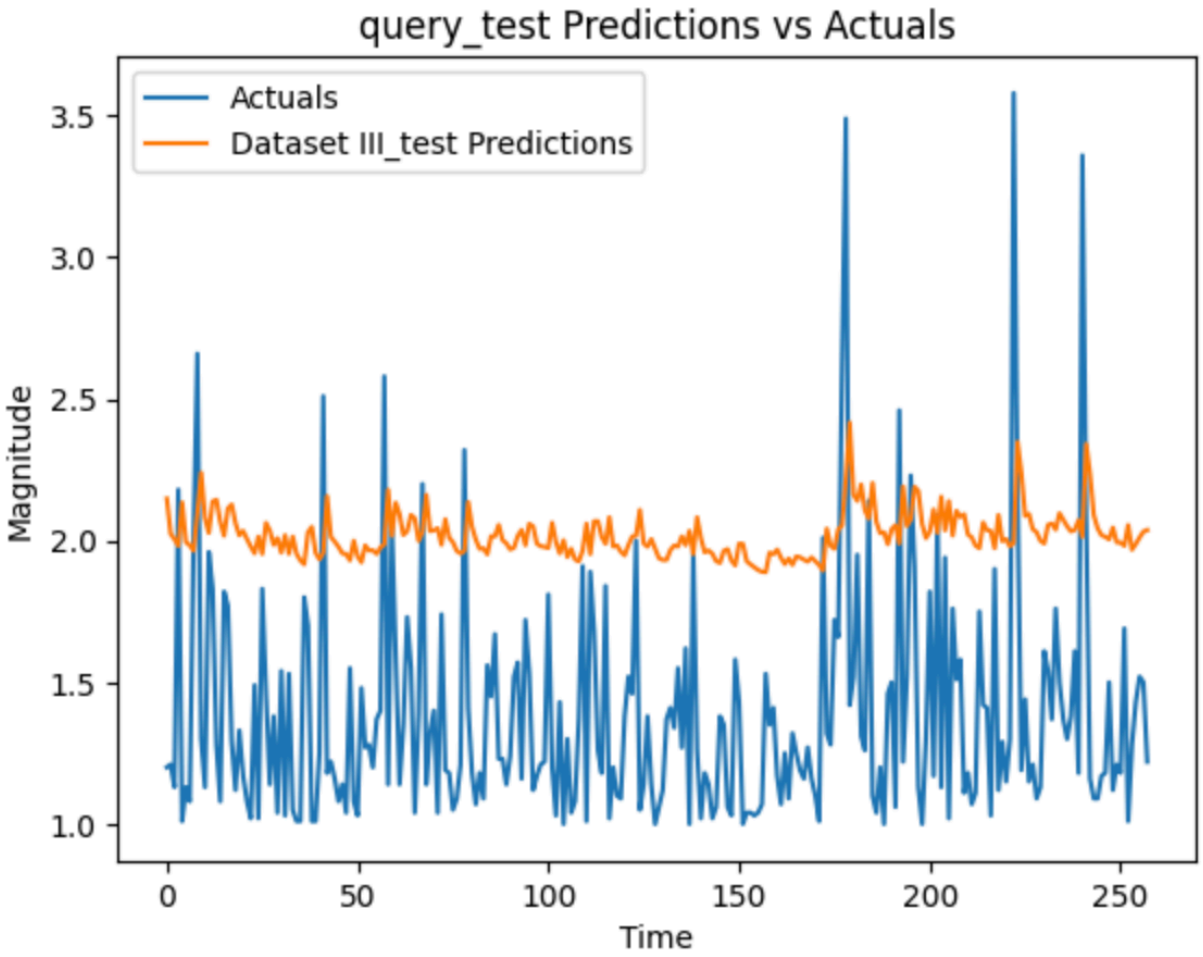

Figure 38. Visualization of test results on Dataset III.

1-hour Interpolation Sampling Temporal Interval on Dataset II

At second attempt of interpolation, Dataset II was resampled with a temporal interval of 1 hour.
Column mag is read into a dataframe, and input and outputs are created. LSTM look-back window is set to 50, and new dimensions of data is demonstrated. Dataset is divided into training (%70), validation (%20) and test (%10) sets.
LSTM sequential model is then created, trained and tested on Dataset II and Dataset III.
This model was run for 10 epochs and trained on resampled data from Dataset II. It was tested on Dataset II and Dataset III. MSE for the best model with interpolation with 1-hour interval was 0.1848 for training set of Dataset II. Comparison of actual values and results are shown below.

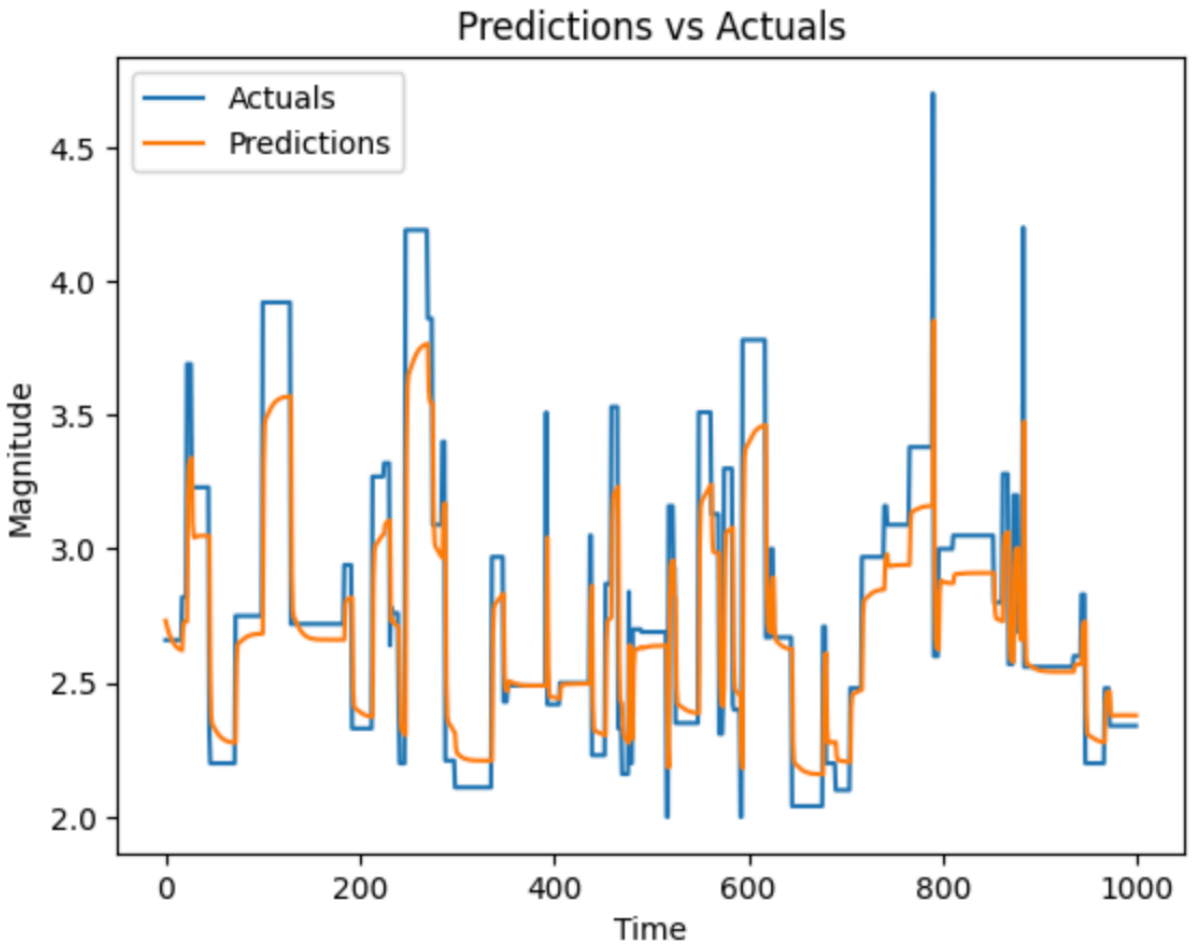

Figure 39. Visualization of Training results on Dataset II.

This model was then used on the validation set of Dataset II and the MSE for validation data was equal to 0.1708.

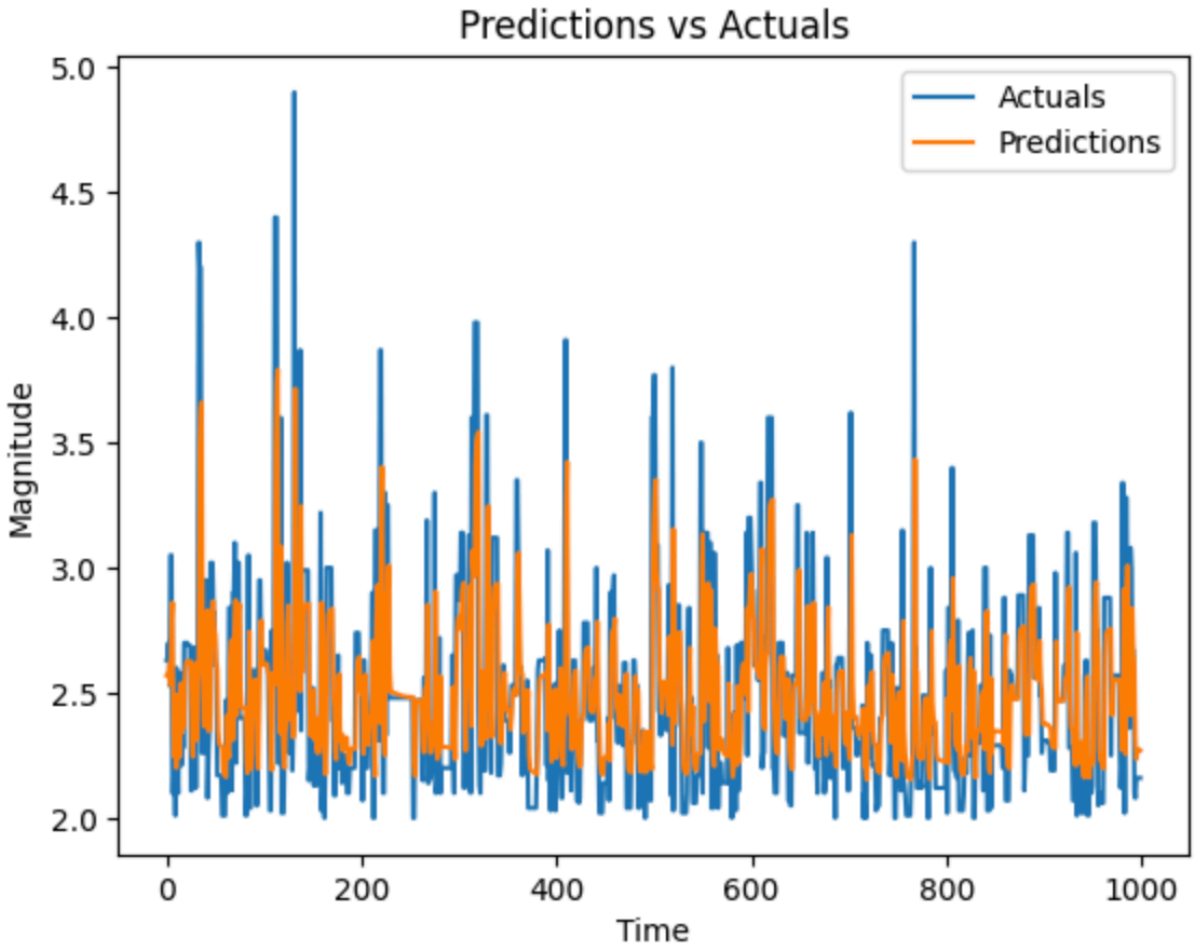

Figure 40. Visualization of validation results on Dataset II.

Next, the trained model was tested on the test set of Dataset II. The result MSE for test set was equal to 0.1565.

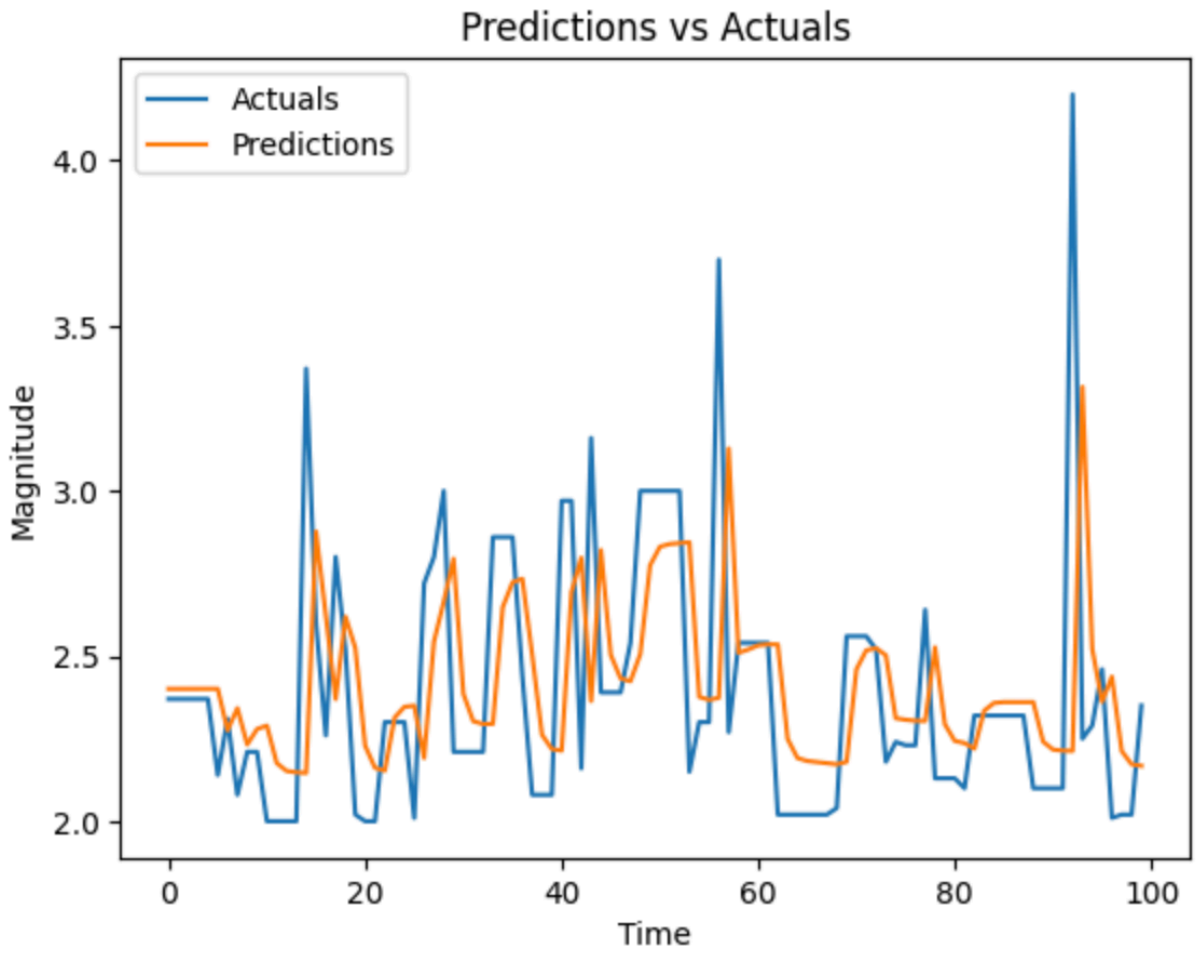

Figure 41. Visualization of test results on Dataset II.

This model was then tested on Dataset III to study effects of testing a model trained with interpolated data on seismic data that was not interpolated. In this case, where a model that was trained on interpolated data, was tested on a Dataset that was not interpolated, the resulting MSE was equal to 0.2878.

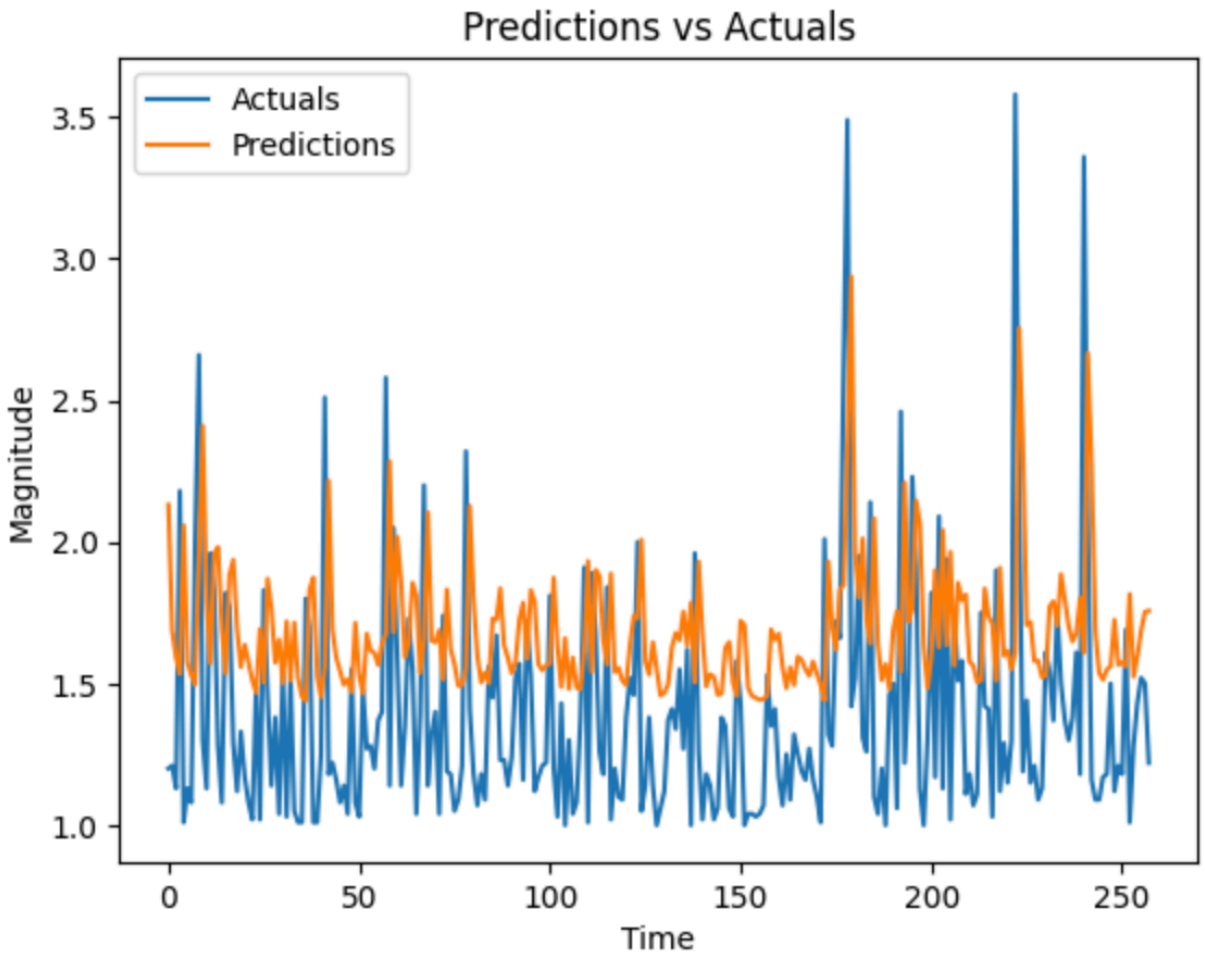

Figure 42. Visualization of test results on Dataset III.

This Model was then tested on Dataset III after interpolation with 1-hour temporal intervals. MSE for this case was equal to 0.1754.

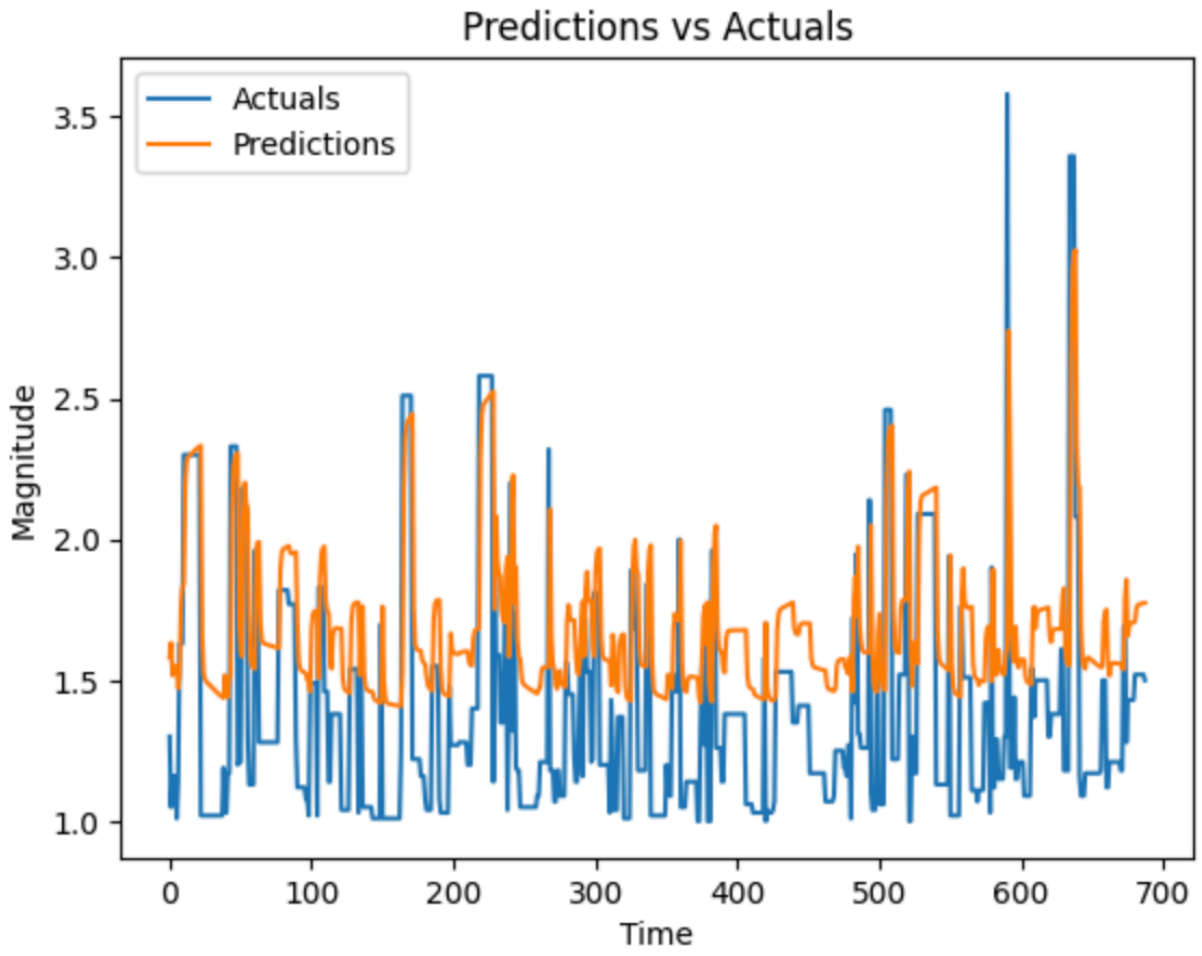

Figure 43. Visualization of test results on interpolated Dataset III.

This Model was then tested on Dataset I with and without interpolation with 1-hour temporal intervals. When model was tested on Dataset I without interpolation, the resulting MSE was equal to 0.4675.

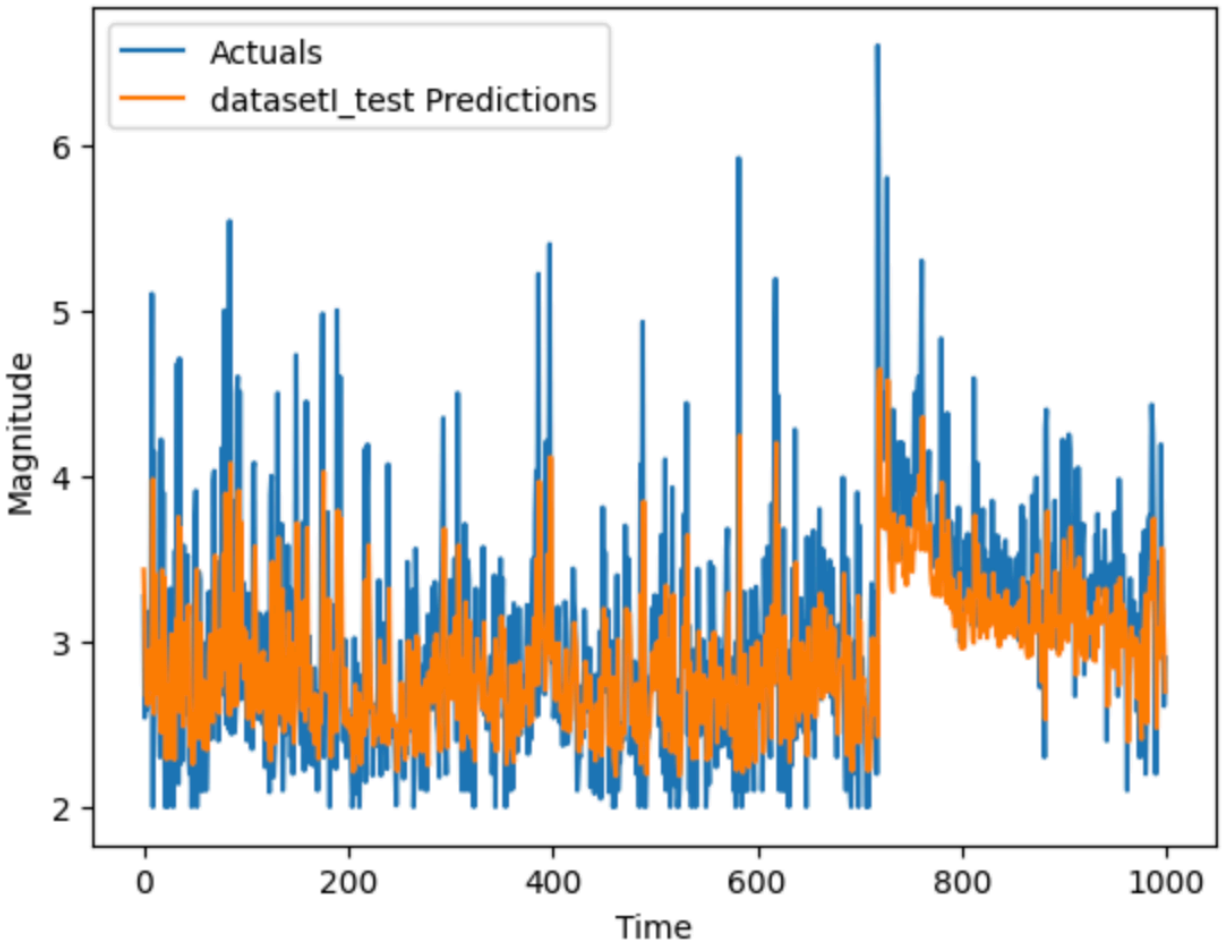

Figure 44. Visualization of test results on Dataset I without interpolation.

To better understand how this model performs on interpolated data, it was tested on Dataset I with interpolation with 1-hour temporal intervals. MSE in this case was equal to 0.0884. Below figures show the comparison of actual values versus predicted values.

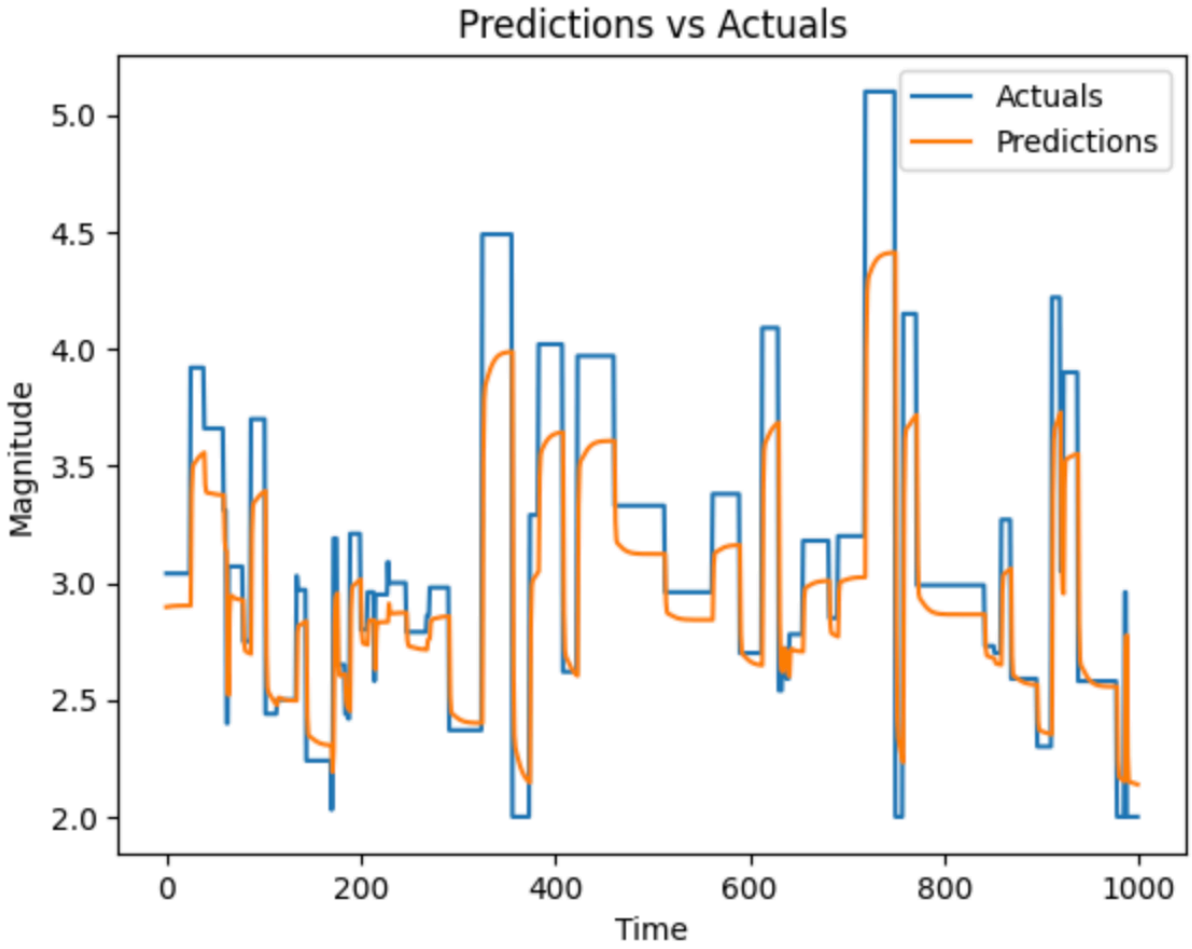

Figure 45. Visualization of test results on Dataset I with interpolation.

1-hour Interpolation Sampling Temporal Interval on Dataset III

Finally, LSTM was run for 100 epochs on interpolated Dataset III. Dataset III was resampled with a temporal interval of 1 hour and column mag is read into a dataframe, and input and outputs are created. LSTM look-back window is set to 50, and new dimensions of data is demonstrated. Dataset is divided into training (%70), validation (%20) and test (%10) sets.

LSTM sequential model is then created, trained and tested on Dataset III, Dataset II and Dataset I. This model ran for 100 epochs and trained on resampled data from Dataset III. It was tested on Dataset III, II and I. MSE for the best model with interpolation with 1-hour interval for training set was equal to was 0.0952. Comparison of actual values and results are shown below.

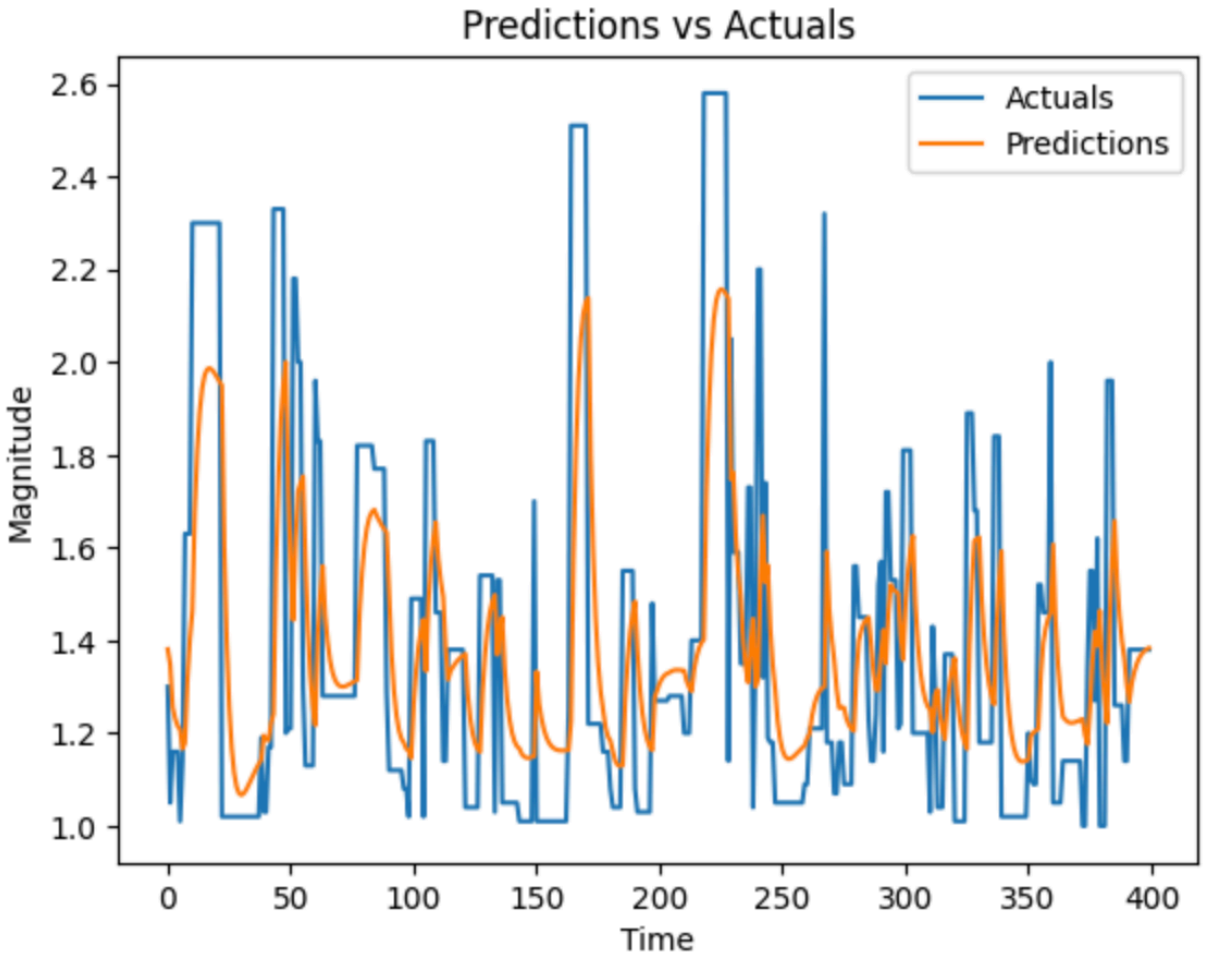

Figure 46. Visualization of training results on Dataset III.

For the next step, this model was applied to the validation set of Dataset III. Validation MSE for this case was equal to 0.0838. A comparison of the actual values versus predicted values can be seen in the table below.

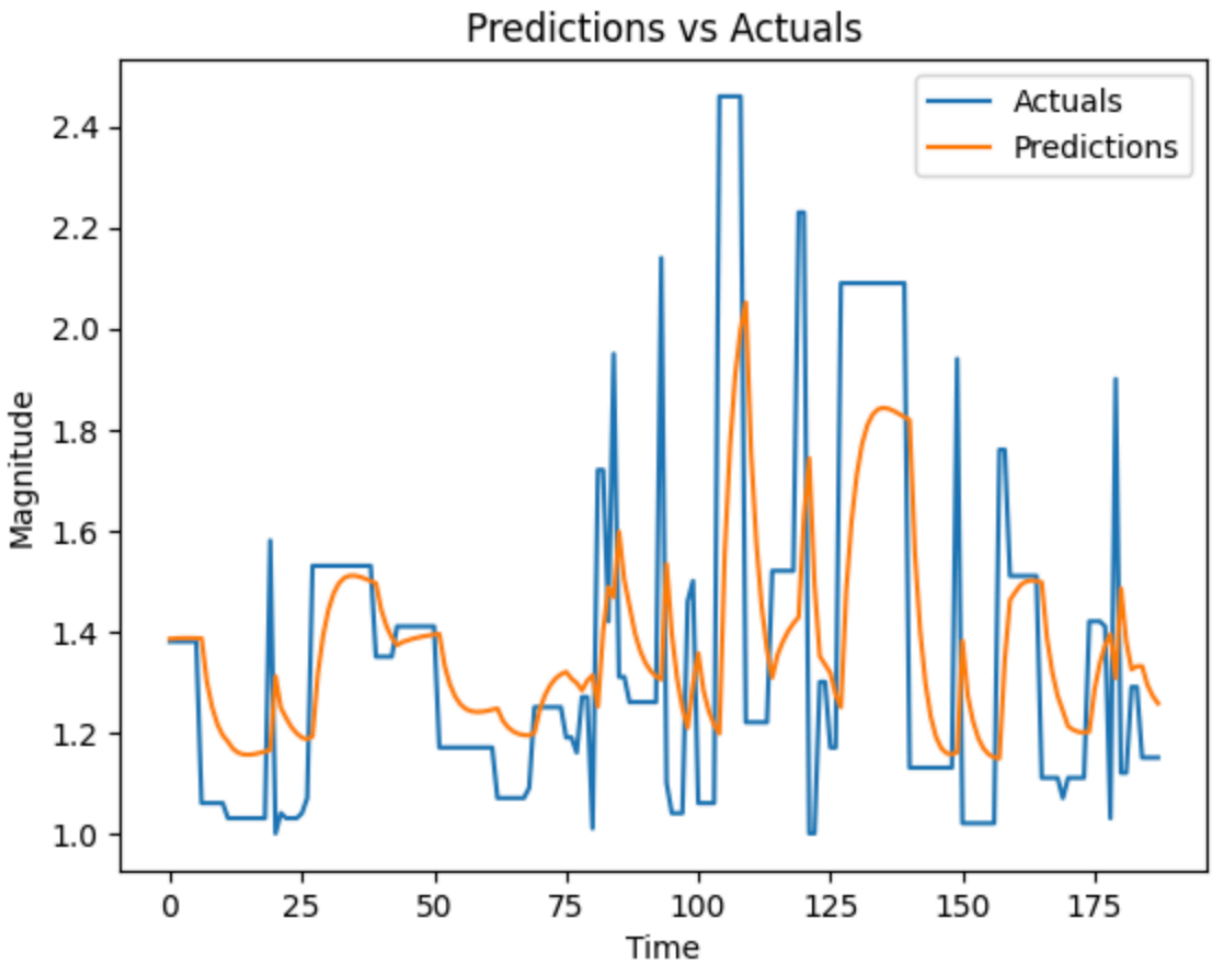

Figure 47. Visualization of validation results on Dataset III.

At this step, trained model was applied to the test set of Dataset III with an MSE of 0.1804.

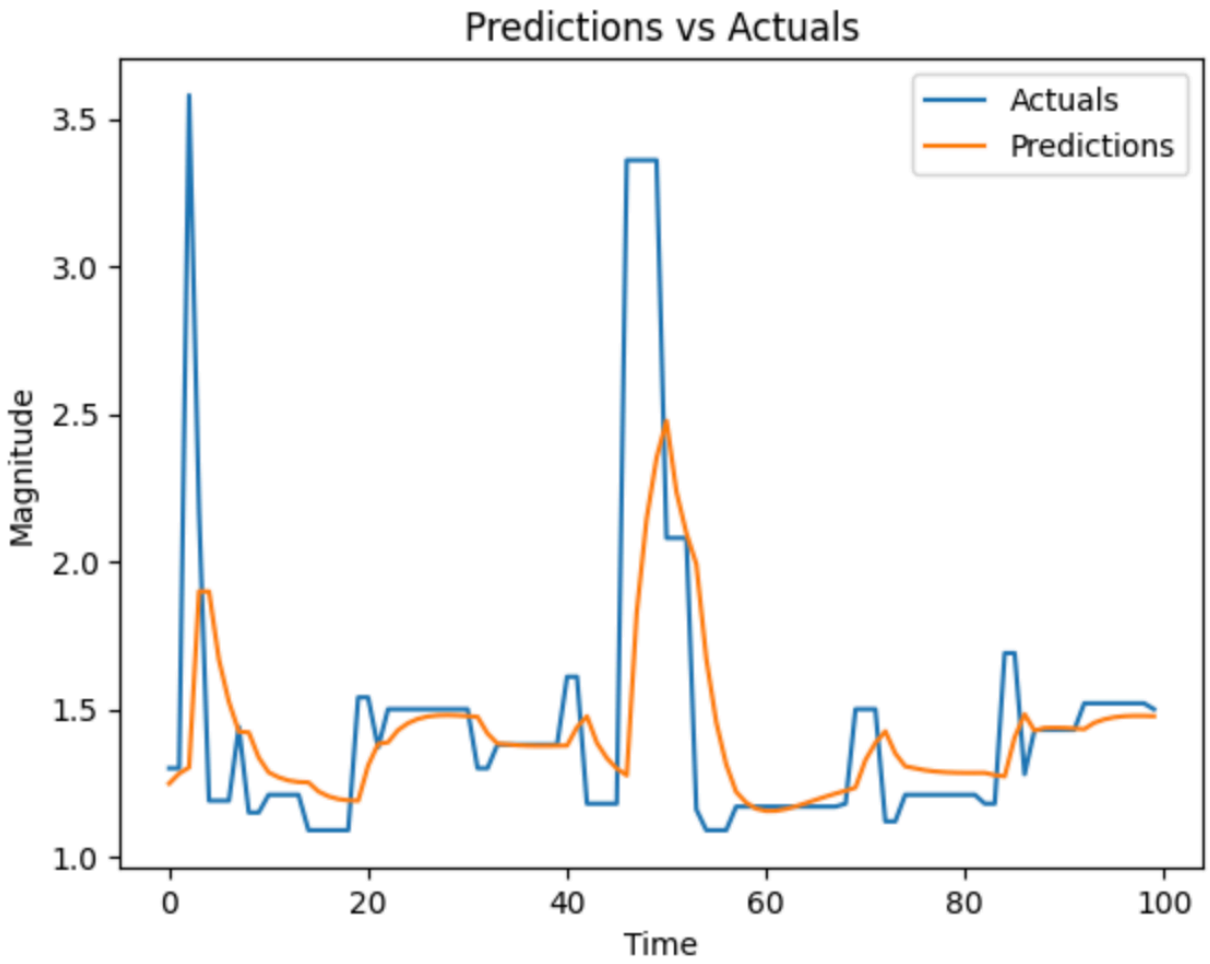

Figure 48. Visualization of Test results on Dataset III.

This model was then tested on Dataset II to study effects of testing a model trained with interpolated data on seismic data that was not interpolated. After that, Dataset II was interpolated for testing with the model trained on interpolated data. When model was tested on Dataset II without interpolation, it resulted in an MSE of 0.4953.

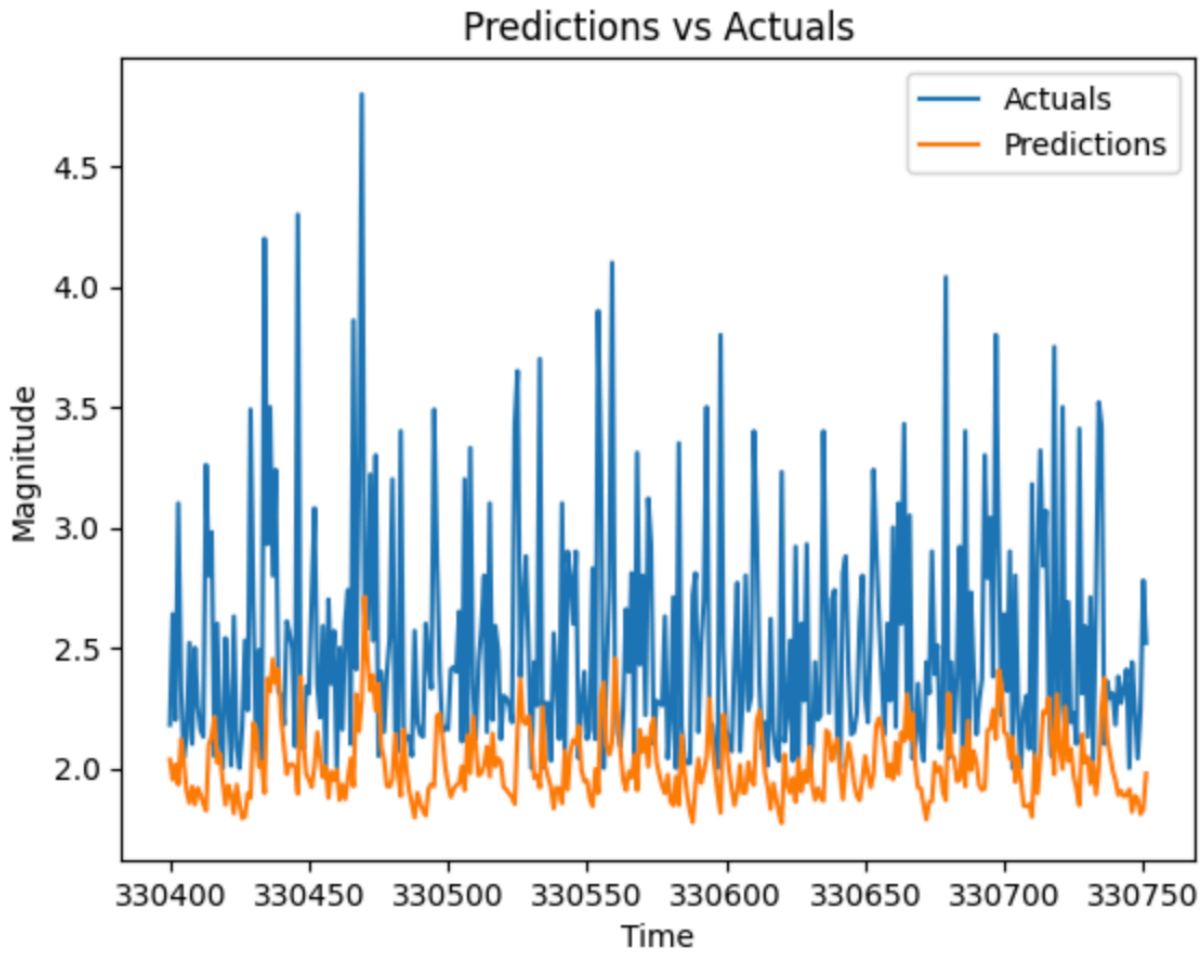

Figure 49. Visualization of test results on Dataset II. Model trained on interpolated Dataset III.

Dataset II was interpolated with 1-hour temporal interval. Results of testing the model trained on interpolated data on resampled Dataset II are represented below. In this case MSE was equal to 0.4194.

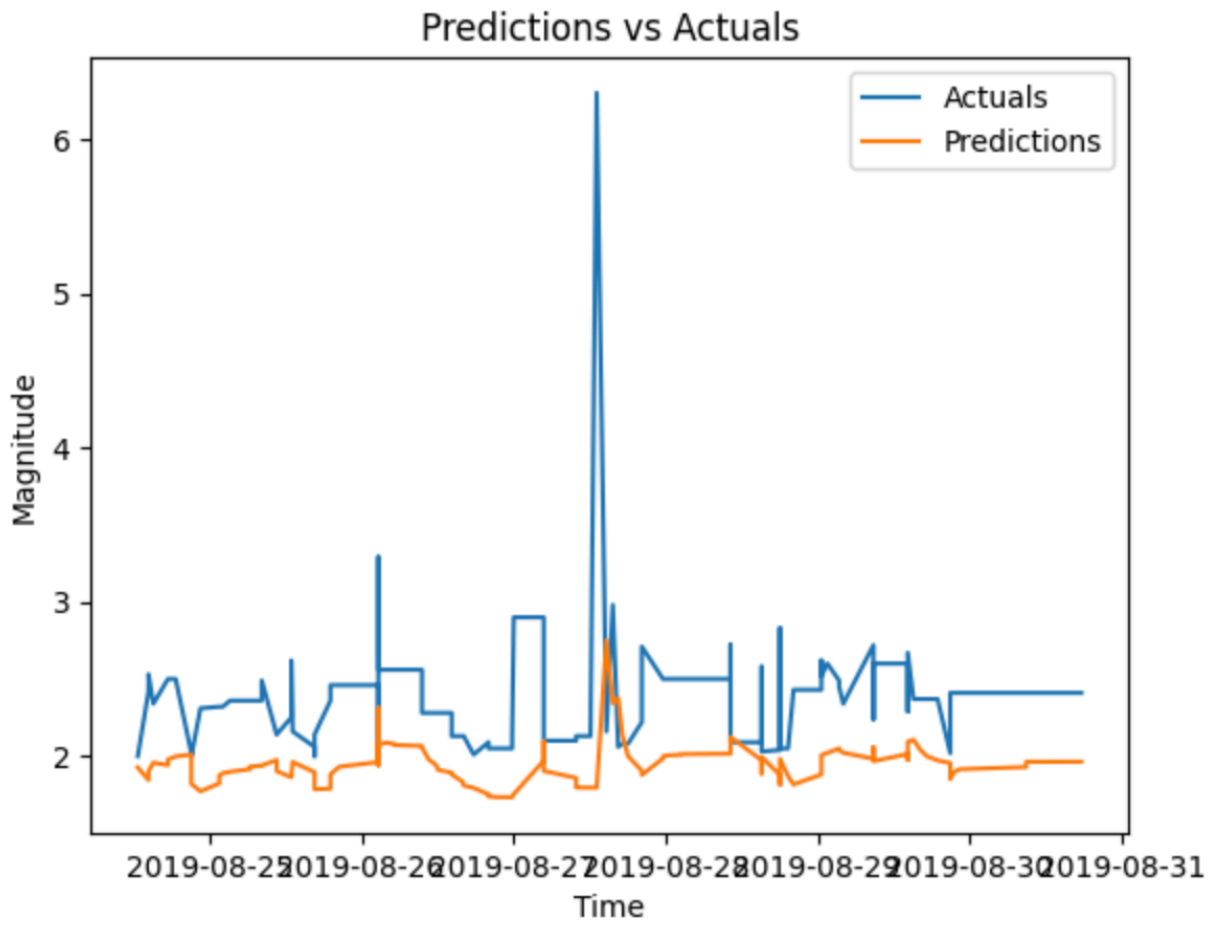

Figure 50. Visualization of test results on interpolated Dataset II. Model trained on interpolated Dataset III

Best model was tested on Dataset I to study effects of testing a model trained with interpolated data on seismic data that was not interpolated. After that, Dataset I was interpolated for testing with the model trained on interpolated data. In the case of testing. The model on Dataset I without interpolation, the MSE was equal to 0.8647.

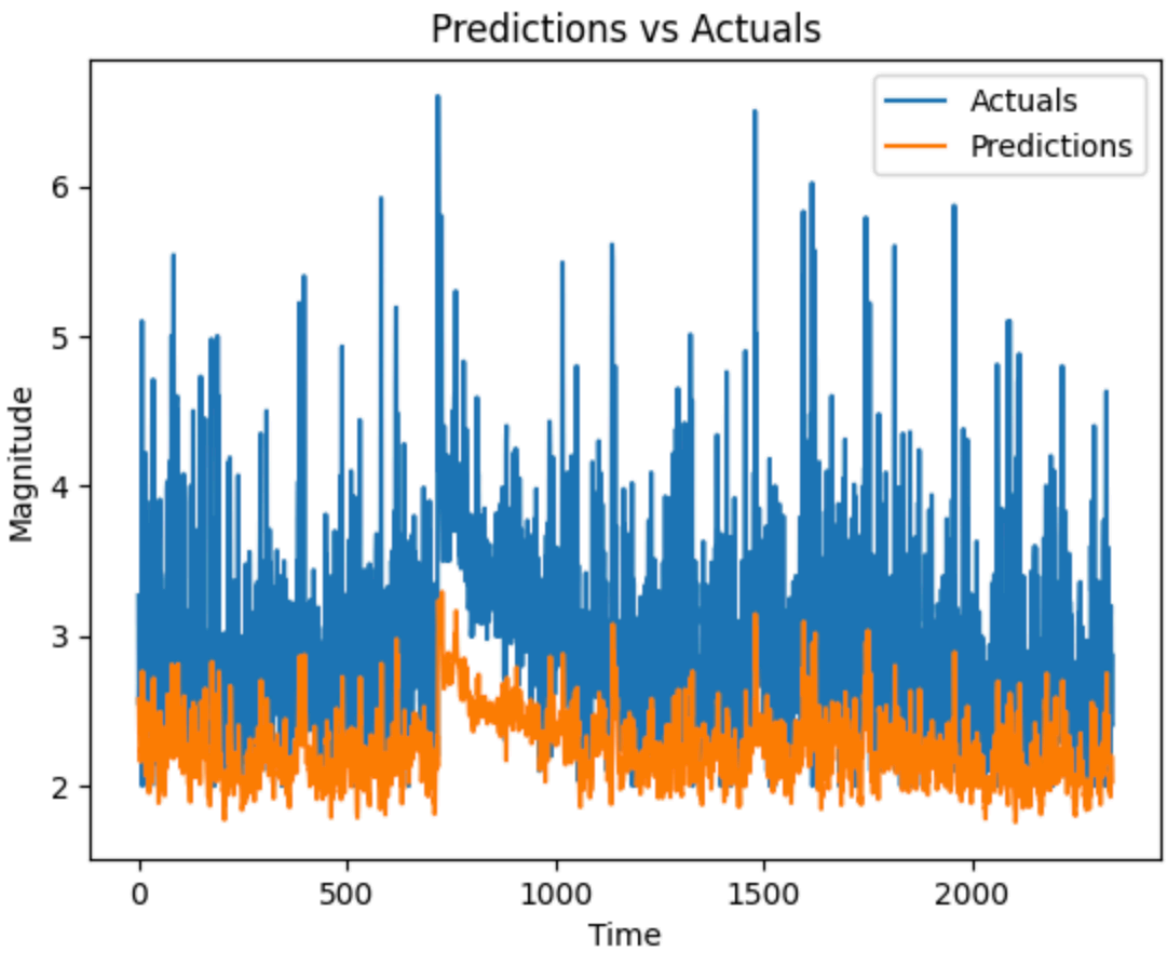

Figure 51. Visualization of test results on Dataset I. Model trained on interpolated Dataset III

After Dataset I was interpolated with 1-hour temporal interval, the obtained MSE was equal to 0.5857. Results of testing the model trained on interpolated data on resampled Dataset I are represented below.

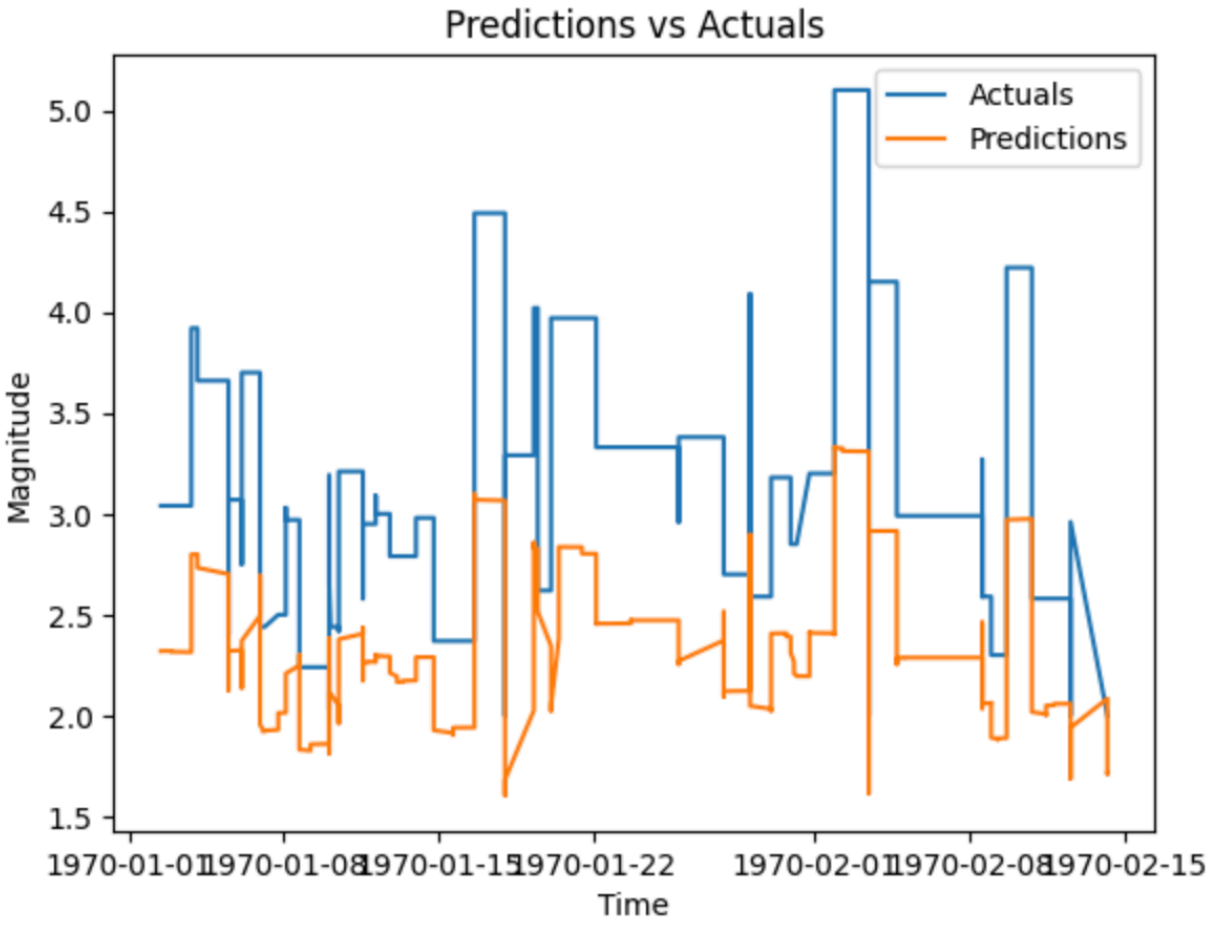

Figure 52. Visualization of test results on interpolated Dataset I. Model trained on interpolated Dataset III

Evaluation

A comparison of different scenarios is provided in the Tables below. Table 1 shows the Mean Squared Error for different model architecture with no interpolation on data. In all cases, learning rate was set to 0.0001 and Adam optimizer was utilized. These selections were made after trial and error with different learning rates and optimizers. All models were trained on Dataset II, which covers a 50-year period, and in each case, model was trained for 10 epochs. As it is demonstrated in Table 1, 128-unit LSTM with RELU activation function and 200 step Look-Back Window had the best performance on training data with MSE of 0.2372. Mean Squared Errors are also provided for test set of Dataset II, and for testing on entire Dataset II and Dataset I. For test sets that were part of Dataset II, 128-unit LSTM with RELU activation function and 200 step Look-Back Window had the best performance. In the case where trained model was tested on a completely new and unknown dataset, 128-unit LSTM with RELU activation function and 200 step Look-Back Window had the best performance when tested on the medium size dataset, Dataset I. When tested on the smallest dataset, dataset III, 64-unit LSTM with Tanh activation function and 500 step Look-Back Window had the best performance. The next best performance in this case was 64-unit LSTM with RELU activation function and 50 step Look-Back Window had the best performance. This result implies that due to the smaller size of the test dataset, Dataset III, which is equal to 30 days, a smaller look-back window is associated with better performance.

Table 1: Model Performance evaluation.

LSTM Nodes	Activation Function	Look-Back Window	Training MSE	Validation MSE	Dataset II test MSE	Dataset I test MSE	Dataset III test MSE
64	RELU	50	0.2379	0.2617	0.2297	0.3875	0.7120
64	RELU	200	0.2374	0.2627	0.2287	0.3735	1.0762
128	RELU	200	0.2372	0.2620	0.2279	0.3737	0.9049
64	Sigmoid	50	0.2401	0.2623	0.2338	0.4121	1.1266
64	Tanh	50	0.2380	0.2613	0.2299	0.3938	1.0608

The following tables demonstrate the Mean Squared Errors for varying resampling intervals and training epochs when training dataset was interpolated. Additional test sets were used both with and without interpolation. Similar to experiments with data without interpolation, learning rate was set to 0.0001 and Adam optimizer was utilized in all instances. These selections were made after trial and error with different learning rates and optimizers. All instances used a 64-unit LSTM with RELU activation function and 50 step Look-Back Window model. Next 3 tables list the models used on each dataset based on performance in decreasing order (increasing MSE).

Table 2: Model Performance evaluation for Dataset I

LSTM Nodes	Activation Function	Look-Back Window	Dataset I test MSE
64	RELU	200	0.3735
128	RELU	200	0.3737
64	RELU	50	0.3875
64	Sigmoid	50	0.4121
64	Tanh	50	1.0608

Table 3: Model Performance evaluation for Dataset II

LSTM Nodes	Activation Function	Look-Back Window	Dataset II test MSE
128	RELU	200	0.2279
64	RELU	200	0.2287
64	RELU	50	0.2297
64	Tanh	50	0.2299
64	Sigmoid	50	0.2338

Table 4: Model Performance evaluation for Dataset III

LSTM Nodes	Activation Function	Look-Back Window	Dataset III test MSE
64	Tanh	50	0.3938
64	RELU	50	0.7120
128	RELU	200	0.9049
64	RELU	200	1.0762
64	Sigmoid	50	1.1266

For the second experiment, the model with 64-unit LSTM with RELU activation function and 50 step Look-Back Window was trained on Dataset II with interpolation with 4-hour resampling window. This model was then tested on the entire Dataset III as an additional test set. Results show that this model performed better on Dataset III, when it was trained on Dataset II with interpolation (MSE = 0.5695) compared to when it was trained on Dataset II without interpolation (MSE = 0.7120)

Table 5: Model Performance evaluation with Interpolation

	Train Dataset	**Train Dataset length**	**Resampling Interval**	**# of Epochs**	**Train II MSE**	**Val II MSE**	**Test II MSE**	**Test III MSE**
With Interpolation	Dataset II	50 years	4-hour	10	0.2460	0.2442	0.2303	
Without Interpolation								0.5695

In the second experiment, the model with 64-unit LSTM with RELU activation function and 50 step Look-Back Window was trained on Dataset II with interpolation with 1-hour resampling window. This model was then tested on the entire Dataset I with no interpolation, Dataset I with interpolation with 1-hour resampling interval, Dataset III with no interpolation and Dataset III with interpolation with 1-hour resampling interval as additional test sets.

Table 6: Model Performance evaluation with Interpolation

	Train Dataset	**Train Dataset length**	**Resampling Interval**	**# of Epochs**	**Train/ Val II MSE**	**Test II MSE**	**Test I MSE**	**Test III MSE**
With Interpolation	Dataset II	50 years	1-hour	10	0.1848 / 0.1708	0.1565	0.0884	0.1754
Without Interpolation							0.4675	0.2878

For the third experiment, the model with 64-unit LSTM with RELU activation function and 50 step Look-Back Window was trained on Dataset III with interpolation with 1-hour resampling window. Smaller size of Dataset III (30-day period) allows for increasing the number of epochs to 100. This model was then tested on the entire Dataset I with no interpolation, Dataset I with interpolation with 1-hour resampling interval, Dataset II with no interpolation and Dataset II with interpolation with 1-hour resampling interval as additional test sets. Results are shown in Table 7 below. As it is seen, Dataset III has the smallest MSE in this case. Dataset II and Dataset I have better results where interpolated. It can be seen that performance of model declines when trained on the smallest dataset.

Table 7: Model Performance evaluation with Interpolation

	Train Dataset	**Train Dataset length**	**Resampling Interval**	**# of Epochs**	**Train/ Val III MSE**	**Test III MSE**	**Test I MSE**	**Test II MSE**
With Interpolation	Dataset III	30 days	1-hour	100	0.0952 / 0.0838	0.1804	0.5857	0.4194
Without Interpolation							0.8647	0.4953

Conclusion

This work was initiated with 3 objectives:

Objective 1 was to collect and prepare observational data to create the suitable time-series dataset for this research.

This objective was achieved by creating 3 different time-series from seismic data collected from the USGS, Dataset I, Dataset II, and Dataset III.

Objective 2 was to propose Long Short-Term Memory (LSTM) architectures to predict the earthquake magnitude in seismic time-series.

Different LSTM models with different architectures and parameters were created and tested in experiment 1 through 5. Mean Squared Error for best model was 0.2372
for training and 0.2279 for testing.

Objective 3 was to find a solution to address the irregularity of the time-series used in this study. Hypothesis was that LSTM model performance will be improved with interpolation, and that smaller resampling intervals improve the model performance.

To address this objective, Dataset I, Dataset II and Dataset III were interpolated with different intervals and tested with the LSTM model in experiments 1 through 3. It was observed that interpolation improved model performance compared to where there was no interpolation. It was also observed that smaller resampling intervals result in model performance improvement and reduction of Mean Squared Error. Mean Squared Error for the best case with interpolation was 0.0884. Mean Squared Error with interpolation was % 61 smaller than Mean Squared Error without interpolation.

It is observed that large datasets can be used with Long Short-Term Memory deep learning models to effectively predict the magnitude of earthquakes. This study can be perceived as an informational tool for awareness about the possibility of large forthcoming earthquakes.

For future studies, historic datasets for individual seismic faults can be used for training LSTM models to gain predictive information not only for the magnitude but also for the location of the future earthquakes. This can contribute to saving lives and property, as well as reducing or eliminating injuries and destruction.

Different sizes of datasets, different activation functions and different sizes of LSTM look-back windows were used in this study to observe how they impact the results. The impact of interpolation was studied to understand whether it can change the results significantly. Therefore, this project can also be used as an instructional example to teach utilizing Long Short-Term Memory deep learning models for time-series analysis.

References

[1] S. M. Mousavi and G. C. Beroza, "Machine learning in earthquake seismology", *Annu. Rev. Earth Planet. Sci.*, vol. 51, no. 1, pp. 105-129, May 2023. Available at: https://www.annualreviews.org/content/journals/10.1146/annurev-earth-071822-100323;jsessionid=qN6FL2PDgMETfrCBnDNUISgQ_I05tzz_nqXVzR7c.annurevlive-10-241-10-103. Accessed March 18, 2024.

[2] Wald, Lisa, "The Science of Earthquakes", Available at: https://www.usgs.gov/programs/earthquake-hazards/science-earthquakes. Accessed March 20, 2024.

[3] Bolt, Bruce A. "earthquake". *Encyclopedia Britannica*, Available at: https://www.britannica.com/science/earthquake-geology. Accessed March 20, 2024.

[4] Robert J. Geller, "Earthquake prediction: a critical review", *Geophysical Journal International*, Volume 131, Issue 3, December 1997, Pages 425–450, Available at: https://doi.org/10.1111/j.1365-246X.1997.tb06588.x. Accessed March 24, 2024.

[5] National Research Council. 1976. "Predicting Earthquakes: A Scientific and Technical Evaluation, With Implications for Society". Washington, DC: The National Academies Press. Available at https://doi.org/10.17226/18533. Accessed March 24, 2024.

[6] "Can animals predict earthquakes?", Available at: https://www.usgs.gov/faqs/can-animals-predict-earthquakes. Accessed March 24, 2024.

[7] T. Bhandarkar, V. K, N. Satish, S. Sridhar, R. Sivakumar, and S. Ghosh, "Earthquake trend prediction using long short-term memory RNN", International Journal of Electrical and Computer Engineering (IJECE), vol. 9, no. 2, pp. 1304–1312, Apr. 2019, Available at: https://www.researchgate.net/publication/332545886_Earthquake_trend_prediction_using_long_short-term_memory_RNN. Accessed March 30, 2024.

[8] Robert D. Cicerone, John E. Ebel, James Britton, "A systematic compilation of earthquake precursors", Tectonophysics, Volume 476, Issues 3–4, 2009, Pages 371-396, ISSN 0040-1951. Available at https://doi.org/10.1016/j.tecto.2009.06.008. Accessed April 2, 2024.

[9] Ridzwan, Nurafiqah & Md Yusoff, Siti Harwani. (2023). "Machine learning for earthquake prediction: a review (2017–2021) ". Earth Science Informatics. 16. 1-17. Available at https://www.researchgate.net/publication/369326135_Machine_learning_for_earthquake_prediction_a_review_2017-2021. Accessed April 5, 2024

[10] "What is Deep Learning?" Available at: https://aws.amazon.com/what-is/deep-learning/#:~:text=Deep%20learning%20is%20a%20method,inspired%20by%20the%20human%20brain. Accessed April 7, 2024.

[11] "What is Deep Learning?" Available at: https://www.mathworks.com/discovery/deep-learning.html, Accessed April 7, 2024.

[12] M. H. A. Banna *et al*., "Attention-Based Bi-Directional Long-Short Term Memory Network for Earthquake Prediction," in *IEEE Access*, vol. 9, pp. 56589-56603, 2021, doi:

10.1109/ACCESS.2021.3071400. Available at: https://ieeexplore.ieee.org/document/9395582. Accessed June 5, 2024.

[13] Cheraghi A, Ghanbari A (2017) "Study of risk analysis and earthquake magnitude and timing prediction via tectonic and geotechnical properties of the faults and identifying risky areas in terms of seismicity in larestan city using artificial neural network. " QUID: Investigación, Ciencia y Tecnología, No. Extra 1, 2017, Págs. 1137-1142, (1). Available at: https://dialnet.unirioja.es/servlet/articulo?codigo=6158766. Accessed July 21, 2024.

[14] Q. Wang, Y. Guo, L. Yu and P. Li, "Earthquake Prediction Based on Spatio-Temporal Data Mining: An LSTM Network Approach," in *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 1, pp. 148-158, 1 Jan.-March 2020, doi: 10.1109/TETC.2017.2699169. Available at: https://ieeexplore.ieee.org/document/7913634. Accessed April 10, 2024.

[15] M. H. A. Banna *et al*., "Application of Artificial Intelligence in Predicting Earthquakes: State-of-the-Art and Future Challenges," in *IEEE Access*, vol. 8, pp. 192880-192923, 2020, doi: 10.1109/ACCESS.2020.3029859. Available at: https://ieeexplore.ieee.org/document/9218936. Accessed April 12, 2024.

[16] Stratis Kanarachos, Stavros-Richard G. Christopoulos, Alexander Chroneos, Michael E. Fitzpatrick, "Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and Hilbert transform", Expert Systems with Applications, Volume 85, 2017, Pages 292-304. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0957417417302737. Accessed April 15, 2024.

[17] S. M. Mousavi, W. Zhu, W. Ellsworth and G. Beroza, "Unsupervised Clustering of Seismic Signals Using Deep Convolutional Autoencoders," in *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 11, pp. 1693-1697, Nov. 2019, doi: 10.1109/LGRS.2019.2909218. Available at: https://ieeexplore.ieee.org/document/8704258. Accessed April 18, 2024.

[18] Chatfield, Chris. (2005). "Time-Series Forecasting". Significance. 2. 131 - 133. 10.1111/j.1740-9713.2005.00117. x. Available at: https://rss.onlinelibrary.wiley.com/doi/epdf/10.1111/j.1740-9713.2005.00117.x. Accessed June 10, 2024.

[19] P. Kavianpour, M. Kavianpour, E. Jahani and A. Ramezani, "Earthquake Magnitude Prediction using Spatia-temporal Features Learning Based on Hybrid CNN- BiLSTM Model," *2021 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS)*, Tehran, Iran, Islamic Republic of, 2021, pp. 1-6, Available at: https://ieeexplore.ieee.org/document/9729358. Accessed July 25, 2024.

[20] Olah, C. "Understanding LSTM networks", 2015. Available at: http://colah. github. io/posts/2015-08-Understanding-LSTMs. Accessed June 12, 2024.

[21] A. Pulver and S. Lyu, "LSTM with working memory," *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, USA, 2017, pp. 845-851, doi: 10.1109/IJCNN.2017.7965940. Available at: https://ieeexplore.ieee.org/abstract/document/7965940, Accessed April 20, 2024.

[22] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). "Long Short-term Memory". Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735. Available at: https://doi.org/10.1162/neco.1997.9.8.17. Accessed April 25, 2024.

[23] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to forget: continual prediction with LSTM," *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, Edinburgh, UK, 1999, pp. 850-855 vol.2, doi: 10.1049/cp:19991218. Available at: https://ieeexplore.ieee.org/document/818041. Accessed April 25, 2024.

[24] G. Liu, F. Xiao, C. -T. Lin and Z. Cao, "A Fuzzy Interval Time-Series Energy and Financial Forecasting Model Using Network-Based Multiple Time-Frequency Spaces and the Induced-Ordered Weighted Averaging Aggregation Operation," in *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 11, pp. 2677-2690, Nov. 2020, doi: 10.1109/TFUZZ.2020.2972823. Available at: https://ieeexplore.ieee.org/document/8988162. Accessed April 26, 2024.

[25] B. Bhargava and S. Pasari, "Earthquake Prediction Using Deep Neural Networks," *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2022, pp. 476-479, Available at: doi: 10.1109/ICACCS54159.2022.9785011. Accessed May 3, 2024.

[26] Cao, Chen, Xiangbin Wu, Lizhi Yang, Qian Zhang, Xianying Wang, David A. Yuen, and Gang Luo. 2021. "Long Short-Term Memory Networks for Pattern Recognition of Synthetical Complete Earthquake Catalog" *Sustainability* 13, no. 9: 4905. Available at: https://doi.org/10.3390/su13094905. Accessed May 3, 2024.

[27] "What We Do - Earthquake Hazards Program"?". Available at: https://www.usgs.gov/programs/earthquake-hazards/what-we-do-earthquake-hazards-program. Accessed May 2, 2024.

[28] "API Documentation - Earthquake Catalog". Available at: https://earthquake.usgs.gov/fdsnws/event/1/]. Accessed March 20, 2024

[29] "Comma-Separated Values", Available at: https://en.wikipedia.org/wiki/Comma-separated_values. Accessed July 2, 2024.

[30] https://github.com/saiedmighani/earthquake_time_series_LSTM. Accessed March 28

[31] https://www.python.org/. Accessed July 1, 2024.

[32] https://docs.anaconda.com/navigator/, Accessed July 1, 2024.

[33] "Pandas Documentation". Available at: https://pandas.pydata.org/docs/index.html. Accessed July 1, 2024.

[34] "NumPy". Available at: https://numpy.org/. Accessed July 1, 2024.

[35] "Matplotlib: Visualization with Python". Available at: https://matplotlib.org/. Accessed July 1, 2024.

[36] "API Documentation - Earthquake Catalog". Available at: https://earthquake.usgs.gov/fdsnws/event/1/. Accessed March 30, 2024.

[37] https://earthquake.usgs.gov/earthquakes/search/. Accessed March 30, 2024.

[38] https://colab.research.google.com/. Accessed April 15, 2024

[39] https://www.tensorflow.org/, Accessed April 15, 2024

[40] Philip B. Weerakody, Kok Wai Wong, Guanjin Wang, Wendell Ela, "A review of irregular time series data handling with gated recurrent neural networks", Neurocomputing, Volume 441, 2021, Pages 161-178, ISSN 0925-2312, Available at: https://doi.org/10.1016/j.neucom.2021.02.046. Accessed July 24, 2024

[41] epot, Mathieu, Jean-Baptiste Aubin, and François H.L.R. Clemens. 2017. "Interpolation in Time Series: An Introductive Overview of Existing Methods, Their Performance Criteria and Uncertainty Assessment" *Water* 9, no. 10: 796. Available at: https://doi.org/10.3390/w9100796. Accessed July 24, 2024

[42] "Using JupyterLab". Available at: https://docs.anaconda.com/ae-notebooks/user-guide/basic-tasks/apps/use-jupyterlab/. Accessed April 15, 2024

[43] "What is a GPU?". Available at: https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html. Accessed March 15, 2024

[44] "Accelerate AI development with Google Cloud TPUs". Available at: https://cloud.google.com/tpu. Accessed March 15, 2024