Predicting Earthquake Magnitude

Ryan Barry, Brian Farrell, Jacob Pacheco
CSC-322 Machine Learning
Fall 2023

***Abstract*— Predicting the magnitude of earthquakes can be a tricky and time-consuming task. In this study, we present multiple machine learning models that predict the magnitude of an earthquake with various degrees of accuracy. We used linear regression, random forest, K-nearest neighbor, decision trees, and neural networks to predict magnitudes. The input information used in these models were: month, day, latitude, longitude, depth, nst, gap, dmin, rms, horizontal error, depth error, mag error, magnst. We encountered problems with our models prior to standardizing the dataset we used. There was a clear increase in success for our models once we standardized the dataset. We also decided to round the magnitude to the nearest tenth decimal place from the original hundredth place to investigate the change in accuracy with this additional change. The most accurate models were our Neural Network, Random Forest, and KNN. Our Sequential Neural Network has an R squared value of .81. Random Forest resulted in an R squared value of .85 with the standardized data. KNN resulted in a r squared value of .75 with the non-standardized dataset. We concluded that standardization was crucial in achieving high accuracy models.**

Keywords*— *magnitude, regression, neural network, decision tree, random forest, R squared

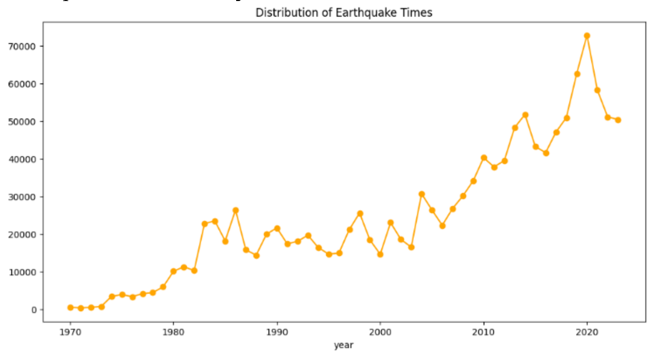
I. Introduction

The problem we investigated was creating machine learning models that can accurately predict the magnitude of an earthquake based on the multiple input data fields. The dataset we used to train and test these models was retrieved from the USGS[1] government agency. We found a dataset of earthquakes on Kaggle but the size of the dataset was not adequate to meet our needs in training the models, so we created a new dataset by querying from the USGS[1] database. This resulted in a dataset with around 1.3 million data points after preprocessing the data. We experimented running our five different models with three different datasets: with non-standardized data, with standardized data, and standardized data with the magnitudes rounded.

The dataset included thirteen different data fields that we determined were the most important to test and train our models. The thirteen different datapoints are as follows: month, day, latitude, longitude, depth, nst, gap, dmin, rms, horizontal error, depth error, mag error, magnst. The month and day values specify the date of the earthquake. The latitude and longitude are the coordinates in which the earthquake occurred. The depth variable is the depth in which the earthquake first began to rupture. Nst refers to the total number of seismic stations that were used to determine the location of the earthquake. The gap measures the distance between the seismic stations that calculate the location of the earthquake. A small gap shows an increased confidence in the accuracy of the measured location, while a larger gap shows the opposite.[1]

The dmin variable is the distance between the epicenter of the earthquake and the nearest seismic station. The rms data field is the root mean squared. The horizontal error value is the measured uncertainty of the location of the earthquake. Similarly, the depth error is the measured uncertainty for the recorded depth of the earthquake and the mag error is the uncertainty of the reported magnitude of the earthquake. The mag nst variable is the total number of seismic stations used to determine the magnitude of the given earthquake. [1]

To retrieve our dataset, we utilized the USGS[1] database and its API. By writing a script that would query and append earthquake entries to a csv file, we were able to collect 3.8 million instances of earthquake type events. This data needed to be cleaned and preprocessed for our models as many rows contained NULL, missing, malformed, or extraneous data. After cleaning, we then had 1.3 million entries. We leveraged python scripting to do this which is shown in our data_cleaner.py. We standardized the data by dividing the values in each row by the maximum value in that column. The timedate column was split into month and day as a year column would be misleading. As technology has developed, more earthquakes have been recorded leading to the data appear as though there have been more earthquakes in recent years. This is shown below.



Developing a machine learning model that can predict the magnitude of an earthquake given a certain amount of data can have future applications. These models and ones like it can become the building blocks of a future model that can predict earthquakes before they occur. This potential model would be very beneficial to humanity and could prevent significant loss of human life. The models we developed in this project are an interesting first step in achieving this ambitious goal.

The development of these models involved a combination of applying material discussed in class with our own independent research. This project required us to consistently brainstorm ways to adjust our data and models to improve the accuracy of our results. This was an educational exercise that allowed us to fully understand the machine learning models and concepts that we have learned throughout the semester.

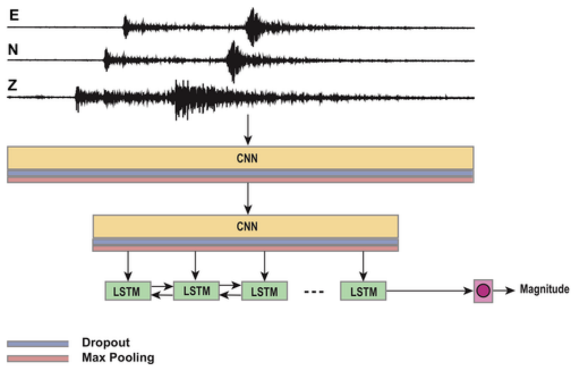
II. Previous Research

A. Mousavi and Beroza

S. Mostafa Mousavi and Gregory C. Beroza published a paper titled “A Machine Learning Approach for Earthquake Magnitude Estimation”. Mousavi and Beroza created a deep learning machine learning model composed of Recurrent Neural Networks and Convolutional Neural Networks. They developed a regressor that was not susceptible to data normalization. This allowed them to train their algorithm with waveform amplitude information. The input to the algorithm is the raw seismogram recorded at seismic stations.

When testing their algorithm on local magnitudes, they reported an error of close to 0 and a standard deviation of around 0.2. The below image is the diagram they included in their paper explaining how their algorithm worked. The raw seismogram data is fed into the convolutional neural networks. The results of the CNNs then feed into a specific type of recurrent neural network called “Long-Short Term-Memory” (LSTM). The LSTM is unique due to its insensitivity to unnormalized inputs.

This approach of designing machine learning models that utilize multiple types of neural networks to allow an input of raw seismogram data led to an extremely accurate model. This model is also a very fast algorithm. However, it does have some downside in that it is most effective with local magnitudes and needs some adaptations when used for larger magnitudes. [2]



B. Asim, Martínez-Álvarez, Basit, Iqbal

Asim, Martínez-Álvarez, Basil and Iqbal published a report titled “Earthquake magnitude prediction in Hindukush region using machine learning techniques”. They used four different machine learning techniques, pattern recognition neural network, recurrent neural network, random forest, and linear programming boost ensemble classifier. The problem is a prediction of earthquakes with a magnitude of 5.5 or greater.

Their pattern recognition neural network model resulted in 58% accuracy in identifying the earthquake magnitude. The Recurrent Neural Network model resulted in a 64% accuracy. Random Forest has a similar 62% accuracy. Lastly, the Linear Programming Boost Ensemble had an accuracy of 65%. While these percentages are not great, the authors concluded that they believed it was a promising start.

C. Ascenio-Cortés, Martínez-Álvarez, Troncosco, Morales-Esteban

Ascenio-Cortés, Martínez-Álvarez, Troncosco, and Morales-Esteban published a paper titled “Medium-large earthquake magnitude prediction in Tokyo with artificial neural networks. They considered earthquakes of magnitude 5 and higher for their research. The authors used multiple machine learning models in their paper, artificial neural network learning machine(ANN), KNN, SVM, Bayesian networks method, and decision trees.

The group used five different datasets to test their different machine learning models. The accuracy for each of the models stayed relatively consistent throughout the dataset. For the first dataset, the Artificial NN had a 72% accuracy. KNN and SVM followed behind with 65% and 45% respectively. The Bayesian Network and Decision trees had accuracies of 30% and 37%. The machine learning algorithms aside from the ANN were only run to show how ANN compares to the already existing commonly used algorithms. ANN stayed consistently at 70% or higher throughout the five datasets and even reached a peak of 80% one of them. This led the authors to conclude that their results were satisfactory. [4]

D. How Our Project Fits In

The goal of our project is not to do something that has never been done before. The above sections clearly show this is not an original idea. Our project is more in line with the Asim and Ascenio-Cortés[4] papers, as opposed to the Mousavi[2] paper. The Mousavi and Beroza algorithm were not only a very complex algorithm, but also involved very complex input data. Such complexity is beyond our current understanding of machine learning.

Our goal with this project is to achieve Asim’s [5] group and Ascenio-Cortés’[4] group achieved: a good starting point. Neither of these two groups concluded their paper with a fully functional machine learning model that correctly predicts the magnitude to a level they are fully happy with. However, they viewed their research as a steppingstone to creating that model in the future. That is what we aspired to do with this research paper. We wanted to learn enough about the process of using machine learning models that we can create a significantly more successful model in the future.

E. Model References

Linear Regression aims to predict the relationship between two variables by assuming a linear connection between the independent and dependent variables. It seeks to find the optimal line that would minimize the sum of squared differences between the predicted and actual values. [5]

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression). [6]

Decision Tree algorithms work by creating a tree like model of decisions and their consequences by recursively splitting input data into subsets based on the most significant feature at each node of the tree. [7]

Random Forest is a collection of decision trees that contains additional randomness to the model when growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that results in a better model. [8]

Neural Networks are an attempt to mimic the human brain's structure, consisting of interconnected nodes organized into layers including input, hidden and output layers. Each of these nodes processes data based on weights and thresholds, activating if the output surpasses a specified threshold value. [9]

III. Proposed Work

A. *Linear Regression* [5]

We used the sci-kit learn library for our linear regression model. The features used for the input for the model were the previously explained thirteen variables. We ran the model on three different variations of our data set. First, we ran it on the non-standardized data. The mean squared error for this run was .41. The r squared score was 0.33.

Secondly, we ran the model using the standardized data. The mean square error improved to 0.35 and R squared value to 0.43.

Lastly, the model was run with the standardized data as well as the magnitude being rounded to the nearest whole number. The mean squared error was 0.43 and the R squared value was 0.37.

Despite the model becoming more accurate with standardization, we concluded that our linear regression was not very successful in predicting the magnitude. We did not anticipate this model to perform very well since our data does not follow a linear pattern.

B. *K Nearest Neighbor*[6]

Our K-nearest neighbor algorithm implemented the sci-kit learn K Neighbors Regression to run KNN on the dataset. We used the mean squared error to evaluate the effectiveness of the model. Like our linear regression model, we ran the model three times with the different processed datasets. We created a for loop that tried the KNN algorithm with different values of k. We tried k values of 1, 5, 9, 13, 17, 21.

The best of the k values for the non-standardized data resulted in a R squared of 0.55 with k = 21. The best k value for the standardized data resulted in a R squared of 0.75 with k = 13. R squared for the best result with the rounded magnitudes was .52 with k = 9.

The KNN model was one of our more accurate models. The best result came when the standardized non-rounded data was used as an input into the model. The result was 75% accurate when run with a k value of 13. This was a surprise when working on the different models. We did not anticipate KNN to be one of our top models when we began planning which models, we would implement.

C. *Decision Tree* [7]

The decision tree model that we created was from the sci-kit learn library. We started by using the built-in gain function to calculate the gain for each of the thirteen different variables we have been using for our models.

The gain values are as follows: 3.472, 3.652, 2.184, 0.248, 0.294, 1.861, 0.242, 0.554, 0.509, 0.457, 0.293. These are for month, day, latitude, longitude, depth, nst, gap, dmin, rms, horizontal error, depth error, mag error, mag nst respectively.

The target for our machine learning model was once again the magnitude. The features that we ran the model on was the thirteen different variables. We chose a test train split of 80/20 in favor of the testing. We determined this was the best split of the different ratios that we tried.

The decision tree accuracy when using the non-standardized data was 0.692.

The accuracy for the decision tree when running the model with standardized data and the rounded magnitudes was 0.706. This was a surprise to us because we anticipated the rounding of the 'magnitudes was going to increase the accuracy of the model.

The R squared value for the decision tree when using the standardized data was around 0.71. This was the most accurate of the different datasets for the decision tree.

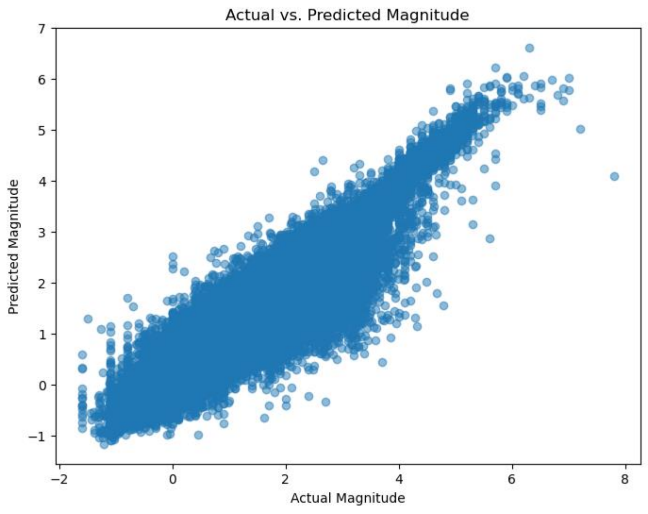
The decision tree was one of our more accurate models in predicting the magnitude. The decision tree did not work out quite as well as we were hoping for, but it still produced some promising results. The limited success we achieved with this model prompted us to investigate the random forest model.

D. *Random Forest* [8]

The set up for the random forest algorithm was similar to the decision tree. We used the Random Forest Regressor import from sci-kit learn to implement the model. The target was the magnitude of the earthquakes while the input features were the thirteen variables. The training split was also 80% training with the remaining 20% being the testing data.

After running the algorithm, we calculated the mean squared error and the R squared score for each of the three different datasets. The model had a mean squared error of 0.164 and an R squared value of 0.765 with the non-standardized data. For the standardized data with rounded magnitudes, the mean squared error was 0.164 and the R-squared value was 0.765.

The best of the three datasets was the unrounded standardized data. The mean square error for this random forest was 0.09 and the R squared value was 0.86. This was our most accurate model overall. After calculating the mean square error and R squared values, we plotted the predicted magnitudes and the actual magnitudes. The plot is shown below. As the plot shows, the random forest model was very accurate in predicting the magnitudes. There are some outliers, but it successfully predicted most of the magnitudes correctly.



E. *Neural Network* [9]

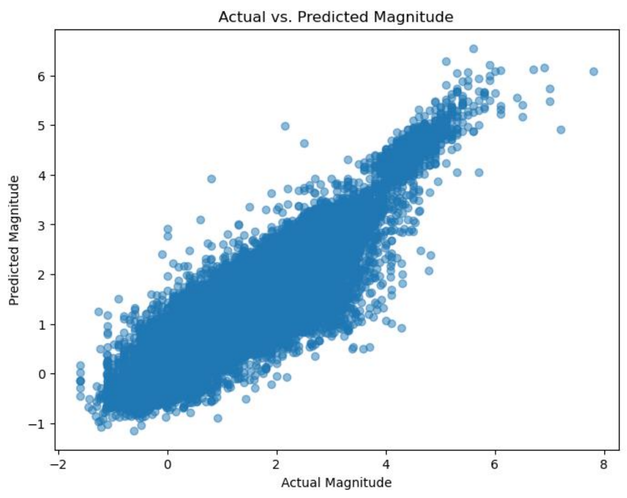
The initial design of our neural network model was a sequential neural network which consists of 13 inputs, 2 dense layers, and a single output layer. The number of dense layers for our model was changed when we experimented with different aspects to increase the accuracy. The two dense layers have 32 and 16 neurons respectively. We chose relu as our activation function. We calculated the loss with mean squared error and calculated the accuracy with the R squared score. We used 10% of the dataset for testing and 90% for training. We experimented with multiple batch sizes to find the optimal one. We tried batch sizes of 2, 8, 16, 32, 64. The batch size of 16 proved to produce the best results. We also set the number of epochs to 10 for these initial tests.

We ran our neural network on the three different datasets: non-standardized data, standardized data, standardized data with rounded magnitudes. We ran each model with a batch size of 2 on each of the three datasets. They resulted in accuracies of 69%, 72%, and 63% respectively. We concluded that the non-rounded standardized dataset was going to produce the best results. Our further experimentation was done exclusively with this dataset.

We tried the different batch sizes with the standardized dataset. The best result our model was with a batch size of 16 which produced an R squared of 0.74 and a loss of 0.163. Batch sizes of 2, 4, 8, 32, and 64 were also used with our model, but did not produce better results than the batch size of 16. The accuracies of these other batch sizes ranged from 0.70 to 0.73. We also experimented with different training splits, trying 20% testing and 5% testing. However, this did not improve the accuracy of the model. We experimented with setting different learning rates for the model as well. This did not lead to any improvement in the accuracy either. The last two experimentations we did on the model was adding more dense layers to the model and increasing the number of epochs. Adding more dense layers was the most successful additions we encountered. We added 5 layers with their number of neurons being 64, 128, 256, 512, 1024. Each time we added a new dense layer, the R squared value increased. For the final test, we increased the number of epochs to 50 while keeping the 5 additional dense layers. This resulted in the best accuracy for our neural network model with an R squared value of 0.81 and a mean squared error of 0.11. The below figure shows the predicted versus actual magnitude plot for the neural network with the dense layers up to 1024 neurons and 50 epochs. The plot looks very similar to the plot for random forest, as they showed similar levels of accuracy.

Our neural network model was the second most accurate model we created, only falling behind random forest. This was not surprising as we anticipated the neural network to be our best model at the beginning of this project. This was the model we decided to spend the most time experimenting with to try and get the highest possible accuracy. We had a lot of faith in this model and the model proved our faith to be well placed.

As previously stated, the Ascenio-Cortés[4] group used an artificial neural network that had an accuracy of 72%. The Asim[2] group used a recurrent neural network that had an accuracy of 64%. Our neural network proved to be more accurate than these models in published papers. The success of our neural network is likely attributable to the very large datasets we were able to compile. This allowed us the luxury of doing a 90/10 training testing split while still testing the models on a significant amount of data. Our models were able to successfully train on much more data than if we had just used the Kaggle generated dataset like we initially planned.



IV. Conclusion

The five different models that we used were not as accurate as we originally anticipated at the beginning of the project. Our goal was to develop multiple models that would be accurate in predicting the magnitude of the earthquakes. We succeeded in this goal as we have two models that had an R squared value of 0.80 or higher.

The standardization of the dataset proved to be an essential part of our project. Every model, except for KNN, had an improved R squared value when used with the standardized dataset. However, rounding the magnitudes in our dataset to the tenth place rather from the original hundredths place led to unsuccessful results. Every model produced worse results with the rounded data compared to the standardized dataset with the unrounded dataset. This was surprising as we anticipated rounding to make predictions easier. The reduced specificity in the magnitudes likely caused this decrease in accuracy.

Linear regression was not very accurate with an R squared value of 0.43. This was expected as we only intended linear regression to be used as a benchmark in which to judge our later models. Our KNN and decision tree models were more successful than linear regression but were still not very accurate with R squared values around 0.70 and 0.75. Our neural network and random forest algorithms produced our best results.

This project allowed us to fully explore multiple different machine learning algorithms. It also forced us to look for ways to better train our algorithms after we created and ran them for the first time. This benefitted us because it led to us learning more about each individual model. Specifically, we spent a significant amount of time experimenting with many different parts of our sequential neural network. Since our early experimentation improved the accuracy, we decided to continue experimenting with different parts of the model to get the R squared value as high as possible.

Our project is a good building block for future research into predicting earthquake magnitudes. It is a great start in determining which machine learning models show promise in continuing research for. Based on our models, sequential neural networks, and random forest algorithms merit further investigation.

V. Future Work

While we were happy with the results of our models, most notably the neural network and random forest models, we believe that there is still room for improvement. Our neural network and random forest algorithms present the most promise for future work. We could also investigate creating an ensemble model that combines our sequential neural network with our random forest model. This should increase the accuracy of the models and deliver better R squared values.

Another aspect of this problem that merits some research is using raw seismogram information like Mousavi and Beroza[2] used for their neural networks. This is an intriguing method that proved to be very successful. Therefore, it could be a model that expands the overall accuracy for our models.

References

[1] *The United States Geological Survey*. USGS.gov | Science for a changing world. (2023a, December 1). https://www.usgs.gov/

[2] Mousavi, S.M. and Beroza, G.C., 2020. A machine- learning approach for earthquake magnitude estimation. Geophysical Research Letters, 47(1), p.e2019GL085976.

[3] Asim, K.M., Martínez-Álvarez, F., Basit, A. et al. Earthquake magnitude prediction in Hindukush region using machine learning techniques. Nat Hazards 85, 471-486 (2017).

[4] Asencio-Cortés, G., Martínez-Álvarez, F., Troncoso, A. et al. Medium–large earthquake magnitude prediction in Tokyo with artificial neural networks. Neural Comput & Applic 28, 1043–1055 (2017). https://doi.org/10.1007/s00521-015-2121-7

[5] Analytics Vidhya. (2021, October). Everything You Need to Know About Linear Regression. https://www.analyticsvidhya.com

[6] Towards Data Science. Machine Learning Basics with the K-Nearest Neighbors Algorithm. https://towardsdatascience.com

[7] Analytics Vidhya. (2021, August). Decision Tree Algorithm. https://www.analyticsvidhya.com

[8] BuiltIn. Random Forest Algorithm. https://builtin.com/data-science/random-forest-algorithm

[9] IBM. Neural Networks. https://www.ibm.com/topics/neural-networks