Predicting Earthquake Damage in the 2015 Nepal Earthquakes

Alp Serdaroglu
Georgia Institute of Technology
gtID: 903835556
aserdaroglu3@gatech.edu

Abstract

The aim of this study is to build a machine learning model to correctly predict the level of earthquake damage that occurred during the April 2015 Nepal Earthquakes. This study also aims to understand the factors affecting earthquake damage using the trained models. The dataset used in this study is collected through post-catastrophe building surveys and includes information about the building and the level of damage. The target variable is the level of earthquake damage which is an ordinal variable. An autoencoder model is used to reduce the dimensionality of the variables. Finally, using tree-based models and different approaches to tackle the ordinal classification problem, 74% testing accuracy is achieved.

1. Introduction

Earthquakes are among the deadliest natural disasters in the world. Strong earthquakes often cause significant loss of life and property to unprepared communities. On April 25th, 2015, an earthquake with a magnitude of 7.8 Mw on the Richter scale hit Nepal [2]. The epicenter of the earthquake was 85 km northwest of Kathmandu, the capital city of Nepal which had a population of 1 million. With its epicenter being close to a major population center, the results of the earthquake were catastrophic. 8,964 people were killed and 21,952 more were injured in Nepal, India, China, and Bangladesh. Approximately 2.8 million people were left homeless as nearly 600,000 buildings were damaged in Kathmandu combined with the significant damage caused to the rural areas caused by the landslides [2].

After the earthquake, the Central Bureau of Statistics of Nepal and Kathmandu Living Labs conducted one of the biggest post-catastrophe surveys [5]. The survey includes extensive information about the buildings including structural information, age, location, and purpose. The survey also includes information about the damage sustained as a result of the 2015 earthquake [5].

The goal of this study is to utilize this survey along with appropriate machine learning techniques to understand what factors contribute to earthquake damage. It is still not possible to predict when the earthquakes will happen. Thus, earthquakes are much more dangerous than other natural disasters as we cannot avoid earthquake damage with short-term planning. Instead, appropriate long-term planning is essential to prepare against earthquakes. Policy measures such as a suitable construction code or disaster management plans are essential to avoid future catastrophes. Constructing buildings to withstand strong earthquakes is critical to be prepared against earthquakes. Understanding the factors that contribute to the earthquake damage can help policymakers to design appropriate policies that will help prevent further earthquake damages potentially saving lives and money. Although we know how to construct reliable buildings, rebuilding entire urban centers requires a long time and money. Therefore, it is critical to understand what type of buildings get damaged during an earthquake. Furthermore, a successful predictive model can be used to forecast the potential damage from future earthquakes. Nepal is in one of the most active earthquake zones in the world. Thus, it is very likely that there will be more earthquakes in the near future. Thus, we can use a machine learning model to predict potential damages

and understand the scale of the damage in the case of a new earthquake using the new data about the building stock.

2. Dataset

The data on the building damage caused by the earthquake were collected through the surveys of Kathmandu Living Labs and the Central Bureau of Statistics of Nepal [5]. It is available as a practice competition at "DrivenData". The dataset includes information about the buildings such as their location, structural properties (ground floor type, roof type, construction material, etc.), age, primary and secondary purpose of use (government, educational, hospital, etc.) [5]. The target variable is "damage_grade" which represents the level of damage sustained during the earthquake. It is an ordinal variable which means that there is a natural order between the categories. There are 3 possible levels of building damage, 1: low damage, 2: medium damage, and 3: high damage. The dataset includes 260,601 data points. In total, there are 40 variables including the building ID and the target variable. Out of the 38 explanatory variables, 34 of them are categorical or binary variables. The remaining 4 variables are numerical. For a detailed explanation of the variables, please refer to [5].

3. Methodology

3.1. Challenges and Preprocessing

Two main challenges were faced during this study. In the following sections, these challenges and the methods used to tackle them are going to be explained.

3.1.1 Ordinal Target Variable

In this study, the goal is to correctly predict the level of damage that a building sustained during the earthquake. As the level of damage is an ordinal variable, treating the problem as a regular classification and regression problem has some shortcomings. Instead, new approaches should be considered to utilize all the information that we can get from the dataset. During model development, three different approaches are considered.

Classification Approach

The first approach used is to consider the problem as a regular multiclass classification problem. Thus, we can apply the regular classification algorithms to predict the level of earthquake damage. However, this approach ignores the ordered relationship within the target variable and we lose some of the information contained in the data. On the other hand, we greatly simplify the problem and eliminate the need for pre and post-processing steps.

Regression Approach

The second approach used is to treat the problem as a regular regression. In this approach, it is assumed that the target variable is numeric and regression algorithms are used to predict the level of damage. This approach requires some post-processing steps. To correctly predict the level of damage, we need to round the initial numerical prediction to the closest label. This may cause some discrepancies in the prediction especially if the prediction has similar distances to more than 2 labels. On the other hand, having numerical predictions provides an additional way to understand how confident we are with our predictions.

Ordinal Classification Approach

The third approach used was proposed by Frank and Hall [6]. In this approach, we reduce the ordinal classification problem with k different labels into k-1 binary classification problems. The current target variable with k labels is transformed into k-1 new target variables. Each target variable corresponds to whether the current target is greater than or equal to a certain level. For this problem, the transformation is given in Tab. 1.

Target Var. (Damage Grade)	Binary Var. 1	Binary Var. 2
1	0	0
2	1	0
3	1	1

Table 1. Encoding used to convert ordinal classification to two binary classification problems

For this case, the first new target variable ($\tilde{y}_1$) corresponds to whether the old target (y) is greater than or

equal "damage.grade = 2". Whereas the second new target variable ($\tilde{y}_2$) represents whether the old target is greater than or equal to "damage.grade (Y) = 3". By training a binary classification model using $\tilde{y}_1$ as target, we can classify data points as $(y \geq 2)$ or $(y \leq 1)$. Using the second target variable we can classify the points as $(y \geq 3)$ or $(y \leq 2)$. Furthermore, when we calculate the classification probabilities from the two binary classification models, we get $\Pr\{Y \leq 1\}$, $\Pr\{Y \geq 2\}$, and $\Pr\{Y \geq 3\}$. Then we can calculate the classification probabilities for the "damage grade".

$$\Pr\{y = 1\} = \Pr\{y \leq 1\} \quad (1)$$
$$= \Pr\{\tilde{y}_1 = 0\} \quad (2)$$
$$\Pr\{y = 2\} = \Pr\{y \geq 2\} - \Pr\{y \geq 3\} \quad (3)$$
$$= \Pr\{\tilde{y}_1 = 1\} - \Pr\{\tilde{y}_2 = 1\} \quad (4)$$
$$\Pr\{y = 3\} = \Pr\{y \geq 3\} = \Pr\{\tilde{y}_2 = 1\} \quad (5)$$

In this study, these three approaches are considered using appropriate algorithms.

3.1.2 Dimensionality Reduction

The second challenge was the high cardinality of categorical variables. Particularly, the location variables ("geo_id_1", "geo_id_2", "geo_id_3") had a high number of unique values. The number of categories for "geo_id_1", "geo_id_2", and "geo_id_3" were 31, 1428 and 12568 respectively. Using one-hot encoding to convert these variables into binary features would create 13040 new features. Combined with the high number of data points (250K), the high number of features would make it computationally challenging to train and tune models. Therefore, dimensionality reduction is used to reduce the number of location-related features to make training computationally easier. To that end, a simple autoencoder is used to reduce the dimensionality of the location-related variables.

After converting three location variables into binary variables using one-hot encoding, an autoencoder is trained to reduce the dimensionality from 13,040 (total number of location categories) to 16. The autoencoder had one hidden layer. The input layer had 13040 neurons (dimension of the inputs). In the hidden layer, we used 256 neurons with a ReLU activation function. Finally, the output layer had 16 neurons. All layers were dense. The sigmoid activation function was also considered for the hidden layer but the ReLU activation function resulted in better testing loss. Due to the size of the dataset, memory issues were experienced during the training of the autoencoder. Since it was not possible to train the autoencoder on the whole dataset, a smaller portion of the dataset is used to train the autoencoder. Then, a separate testing set is used to measure the training loss. Thus, we can understand how well the autoencoder generalizes to the rest of the dataset. The training set is formed by randomly selecting 50,000 data points and the testing set is formed by randomly selecting separate 10,000 data points. The autoencoder was trained for 10 epochs using the ADAM optimizer. The training and testing loss per epoch can be seen in Fig. 1. Mean Squared Error is used as the loss function.

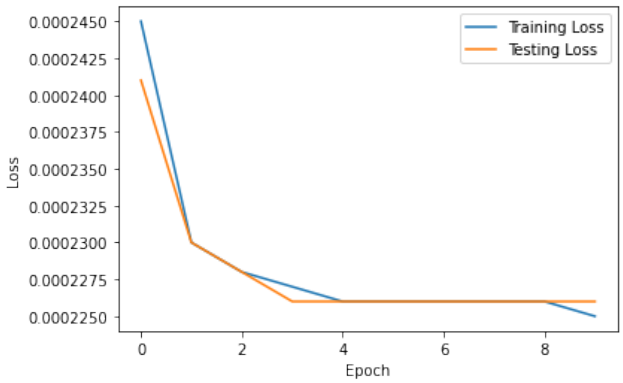

Figure 1. Training and Testing Loss per Epoch

After fitting the autoencoder to the training dataset, the optimal weights are used to encode the remainder of the dataset.

3.1.3 Preprocessing

Other than creating two new target variables for the ordinal classification approach and using an autoencoder to reduce the dimensionality of the location variables, one-hot encoding is used to convert categorical variables to binary variables. The resulting final dataset has 87 features and 260601 data points.

3.2. Models and Hyperparameter Tuning

Due to the high number of categorical variables in the dataset, this study focuses on tree-based methods which are known to deal with categorical variables

Approach	Max. Tree Depth	Min. Samples to Split
Classification	30	50
Regression	20	50
Ordinal Class.	30	50

Table 2. Best Parameters for the Decision Trees

well [1]. Since tree-based methods iteratively segment the data into smaller datasets while increasing the prediction accuracy [3]. This segmentation can be done more intuitively and effectively in the case of binary or categorical variables. During the hyperparameter tuning, prediction accuracy is used as the criterion to select the best models.

3.2.1 Decision Trees

The first and simplest model considered is the decision tree. This model is used both for the classification and regression approaches. Also, using the parameters of the best classification tree, the ordinal classification approach proposed by Frank and Hall [6] was implemented. To tune the hyperparameters of the model 5-fold cross-validation is used. The considered hyperparameters and their mean prediction accuracies can be seen in the Appendix. Optimal parameters determined using cross-validation are given in Tab. 2.

3.2.2 AdaBoost

The second model considered is the AdaBoost. In this part, we tried to improve prediction accuracy by using the best decision tree from the previous part as the weak learner. As an ensemble method, AdaBoost uses boosting to improve the performance of a model by combining weak learners. This model is used both for the classification and regression approaches. The parameters are tuned using 5-fold cross-validation. Furthermore, in the ordinal classification approach, the best parameters from the classification approach are used to train the two binary classification models. The details about the parameters considered can be found in the Appendix. Optimal parameters can be seen in Tab. 3.

Approach	Number of Estimators
Classification	100
Regression	10
Ordinal Class.	100

Table 3. Best Parameters for AdaBoost

3.2.3 Random Forest

The third model considered is the Random Forest. As an ensemble method, Random Forest uses multiple decision trees and a bagging approach to improve the prediction accuracy of a single decision tree. Random Forest is used both for the classification and regression approaches. Also, the ordinal classification approach was implemented using the best-performing model parameters from the classification approach. The parameters are tuned using 3-fold cross-validation. The details about the parameters considered can be found in the Appendix. Optimal parameters can be seen in Tab. 4.

Approach	Max. Depth	Min Samples to Split	Number of Estimators
Classification	100	50	150
Regression	100	50	100
Ordinal Class.	100	50	150

Table 4. Best Parameters for Random Forest

3.2.4 CatBoost

Finally, the CatBoost (Categorical Boosting) model is used as a benchmark model with the default parameter of the Python implementation. CatBoost was developed by Yandex and is known to handle categorical variables efficiently [4]. It is a gradient-boosting algorithm using decision trees.

4. Results and Conclusions

In this section, we present the performance metrics of the best algorithms along with our conclusions. A summary of the performance metrics of the best models can be seen in Tab. 5.

From the results, we can see that Random Forest models achieve the best testing accuracy. All three Random Forest models achieve testing accuracy of more than 73%. On the other hand, the models that achieve the

Model	Approach	Training Accuracy	Testing Accuracy
Decision Tree	Classification	77.91%	68.80%
Decision Tree	Regression	70.42%	67.63%
Decision Tree	Ordinal Class.	78.05%	68.75%
AdaBoost	Classification	89.43%	69.40%
AdaBoost	Regression	72.28%	68.55%
AdaBoost	Ordinal Class.	96.72%	66.94%
Random Forest	Classification	79.18%	73.70%
Random Forest	Regression	79.33%	73.11%
Random Forest	Ordinal Class.	79.42%	73.88%
CatBoost	Classification	74.83%	72.43%
CatBoost	Ordinal Class.	76.61%	73.28%

Table 5. Summary Metrics for Best Models

best training accuracy are the AdaBoost models. Two of these models have a training accuracy of more than 89%. However, these models fail to generalize well to the whole population as they have considerably lower testing accuracies. This is probably due to the overfitting. Decreasing the number of estimators or using a less complex decision tree as the weak learner will help us to solve the overfitting problem and improve the testing accuracy. Finally, Random Forest models perform better than the CatBoost models which we used as a benchmark with the default parameters.

Another important observation is that the accuracies of the classification and ordinal classification approaches are close to each other. Thus, for this dataset, ignoring the natural ordering of the target classes does not significantly reduce the performance of our model. This means ordered relationship does not bring considerable predictive power to the model. This can be caused by the high number of categorical variables and requires further investigation.

The model with the best training accuracy is the Random Forest model with the ordinal classification approach. It achieves a training accuracy of nearly 80% and a testing accuracy of approximately 74%. The confusion matrix of this model for the test dataset can be seen in Tab. 6.

From the confusion matrix, we can see that the predictive power of this model changes based on the label. The recall for labels 1 and 3 are 44% and 60% respectively which is very low compared to the recall of label 2 which is equal to 87%. The predictive power of our model changes because the dataset is imbalanced. Approximately 57% of the training dataset has damage grade 2 while only 10% of the training set has damage grade 1. Thus, the model has better accuracy in predicting the damage grades 3 and 2.

Reference/Predicted	1	2	3
1	2764	3511	68
2	959	32289	3707
3	91	8680	13082

Table 6. Confusion Matrix for the Best Model

Finally, we look at the feature importances of the best model. The top 10 important features are displayed in Fig. 2. A detailed list of feature importances can be found in the Appendix. The feature importance is computed by the sklearn's Random Forest implementation and they represent the mean amount of decrease in leaf impurity per variable [7].

The most important feature is "location" as more than 50% of the predictive power comes from these 16 variables. It is followed by the "foundation type" and "has superstructure mud mortar stone" which means that the superstructure of the building was made of Mud Mortar - Stone [5]. When we look at the least important 10 observations, we see that all of them represent the purpose of the building (i.e. hospital, hotel, school, etc.). Thus, we can say that the purpose of the building does not play an important role in determining earthquake damage.

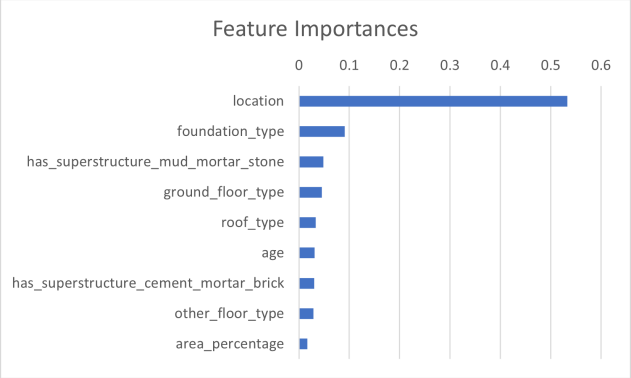

Figure 2. Top 10 Important Features

4.1. Future Improvements

There are possible improvements that can be implemented in the future to improve the performance of the models. First, we can try to run a more detailed hyperparameter tuning process. Since there were computational constraints in this study, the number of alternatives considered was limited. Second, a more sophisticated autoencoder model can be used for dimensionality reduction of the location variables. This may allow us to preserve more of the variance in the reduced features. We can also try to run models without any dimensionality reduction. However, this will probably result in long training times. Finally, we can balance the dataset before training any models. Having an imbalanced data set reduced the performance of the model in the labels with fewer instances. Thus, we can use methods like SMOTE to artificially generate new observations and balance the dataset.

References

[1] Timothy C. Au. Random forests, decision trees, and categorical predictors: The "absent levels" problem. *Journal of Machine Learning Research*, 2018. 4

[2] Brittanica. Nepal earthquake of 2015. https://www.britannica.com/topic/Nepal-earthquake-of-2015. 1

[3] C3.ai. Tree-based models. https://c3.ai/glossary/data-science/tree-based-models/#: :text=What 4

[4] Catboost. Catboost overview. https://catboost.ai/. 4

[5] Driven Data. Richter's predictor: Modeling earthquake damage. https://www.drivendata.org/competitions/57/nepal-earthquake/page/134/. 1, 2, 5

[6] Eibe Frank and Mark Hall. A simple approach to ordinal classification. *Machine Learning: ECML 2001*, 2001. 2, 4

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 5

Appendix

A. Cross Validation Results

For regression, the mean score is calculated using R^2 and for classification, it is calculated using accuracy.

A.1 Decision Tree

Approach	Maximum Depth	Minimum Samples to Split	Mean Score
Regression	10	2	23.84%
Regression	10	10	23.90%
Regression	10	20	24.04%
Regression	10	30	24.08%
Regression	10	50	24.17%
Regression	20	2	20.07%
Regression	20	10	25.55%
Regression	20	20	28.57%
Regression	20	30	30.11%
Regression	20	50	31.69%
Regression	30	2	1.92%
Regression	30	10	17.90%
Regression	30	20	25.04%
Regression	30	30	28.42%
Regression	30	50	31.49%
Regression	40	2	2.67%
Regression	40	10	16.44%
Regression	40	20	24.33%
Regression	40	30	28.01%
Regression	40	50	31.40%
Regression	50	2	2.64%
Regression	50	10	16.37%
Regression	50	20	24.32%
Regression	50	30	28.03%
Regression	50	50	31.40%
Classification	10	2	60.49%
Classification	10	10	60.50%
Classification	10	20	60.49%
Classification	10	30	60.52%
Classification	10	50	60.52%
Classification	20	2	66.84%
Classification	20	10	67.12%
Classification	20	20	67.33%
Classification	20	30	67.42%
Classification	20	50	67.55%
Classification	30	2	65.73%
Classification	30	10	66.93%
Classification	30	20	67.77%
Classification	30	30	68.16%
Classification	30	50	68.48%
Classification	40	2	65.17%
Classification	40	10	66.70%
Classification	40	20	67.59%
Classification	40	30	68.07%
Classification	40	50	68.45%
Classification	50	2	65.08%
Classification	50	10	66.70%
Classification	50	20	67.60%
Classification	50	30	68.07%
Classification	50	50	68.45%

A.2 Adaboost

Approach	Number of Estimators	Mean Score
Classification	20	67.61%
Classification	50	69.70%
Classification	100	69.86%
Regression	10	34.99%
Regression	20	33.73%
Regression	30	32.06%

A.3 Random Forest

Approach	Maximum Depth	Minimum Samples to Split	Number of Estimators	Mean Score
Regression	100	50	50	42.74%
Regression	100	50	100	42.95%
Classification	100	2	50	71.49%
Classification	100	2	100	71.64%
Classification	100	2	150	71.62%
Classification	100	50	50	72.55%
Classification	100	50	100	72.72%
Classification	100	50	150	72.76%
Classification	50	2	50	71.53%
Classification	50	2	100	71.62%
Classification	50	2	150	71.73%
Classification	50	50	50	72.57%
Classification	50	50	100	72.75%
Classification	50	50	150	72.72%

B. Detailed Metrics for Best Models

In the following table, recall and precision values are calculated using the testing set.

Approach	Model	Training Accuracy	Testing Accuracy	Recall Damage Grade 1	Recall Damage Grade 2	Recall Damage Grade 3	Precision Damage Grade 1	Precision Damage Grade 2	Precision Damage Grade 3
Classification	Decision Tree	78%	69%	48%	78%	59%	56%	71%	67%
Regression	Decision Tree	70%	68%	35%	85%	56%	72%	70%	72%
Ordinal Class.	Decision Tree	78%	69%	61%	86%	70%	73%	79%	78%
Classification	AdaBoost	89%	69%	43%	81%	57%	66%	70%	68%
Regression	AdaBoost	72%	69%	32%	87%	47%	70%	67%	73%
Ordinal Class.	AdaBoost	97%	67%	48%	72%	64%	58%	72%	61%
Classification	Random Forest	79%	74%	43%	88%	59%	73%	72%	78%
Regression	Random Forest	79%	73%	39%	87%	59%	73%	72%	77%
Ordinal Class.	Random Forest	79%	74%	44%	87%	60%	72%	73%	78%
Classification	CatBoost	75%	72%	44%	87%	57%	71%	71%	76%
Ordinal Class.	CatBoost	77%	73%	48%	86%	59%	71%	72%	76%

C. Feature Importances of the Best Model

Feature	Importance
location	53.30%
foundation_type	9.09%
has_superstructure_mud_mortar_stone	4.87%
ground_floor_type	4.52%
roof_type	3.35%
age	3.13%
has_superstructure_cement_mortar_brick	3.06%
other_floor_type	2.87%
area_percentage	1.64%
height_percentage	1.48%
count_floors_pre_eq	1.43%
has_superstructure_timber	1.11%
has_superstructure_rc_engineered	1.06%
has_superstructure_adobe_mud	1.01%
has_superstructure_mud_mortar_brick	0.96%
has_superstructure_stone_flag	0.89%
position	0.81%
land_surface_condition	0.67%
has_superstructure_rc_non_engineered	0.67%
count_families	0.64%
has_superstructure_bamboo	0.59%
plan_configuration	0.53%
legal_ownership_status	0.48%
has_secondary_use	0.41%
has_superstructure_other	0.33%
has_superstructure_cement_mortar_stone	0.29%
has_secondary_use_agriculture	0.26%
has_secondary_use_hotel	0.22%
has_secondary_use_other	0.11%
has_secondary_use_rental	0.11%
has_secondary_use_industry	0.05%
has_secondary_use_institution	0.03%
has_secondary_use_school	0.01%
has_secondary_use_use_police	0.01%
has_secondary_use_health_post	0.01%
has_secondary_use_gov_office	0.00%