# Richter's Predictor: Modeling Earthquake Damage

**Berkay Uğur Şenocak**
Department of Computer Engineering
TOBB University of Economics and Technology
Ankara, Turkey
bsenocak@etu.edu.tr

## Abstract

Earthquakes are natural disasters and most of them do some damage to the buildings. Although the earthquakes cannot be prevented, some predictions to increase safety for mankind could be made. In a competition in datadriven.org, the target was to predict building-based damage of an earthquake. To achieve this task, several artificial intelligence approaches were tried. The solution proposed in this report achieved 191th place among 2930 competitors.

## 1 Problem Description

Earthquake is a term used to describe both sudden slip on a fault, and the resulting ground shaking and radiated seismic energy caused by the slip, or by volcanic or magmatic activity, or other sudden stress changes in the earth. Although earthquakes are natural disasters, the destructive effect of earthquakes could be estimated based on many parameters. The main concern is that the damage of earthquake to a specific building is trying to be estimated using many features of the building.

### 1.1 Damage of an earthquake to a specific building

The damage of an earthquake to a specific building is divided into 3 different categories: Low, Medium and High(almost complete destruction).

### 1.2 Parameters of the damage taken

As mention previously, the damage taken for a building depends on many parameters.

Obviously, the one of the main parameters is the material of the building. Also, the geographical location is another important parameter being used in damage calculation. Check Section 2 for the complete list of the parameters.

## 2 Dataset Description

The dataset mainly consists of information on the buildings' structure and their legal ownership.

### 2.1 Data collection

The data in the dataset was collected through surveys by Kathmandu Living Labs and the Central Bureau of Statistics, which works under the National Planning Commission Secretariat of Nepal.

| Feature no. | Feature name | Data type | Description |
|---|---|---|---|
| 1 | building_id | int | unique and random identifier for a building |
| 2, 3, 4 | geo_level_1_id, geo_level_2_id geo_level_3_id | int | geographic region in which building exists |
| 5 | count_floors_pre_eq | int | of floors in the building before the earthquake |
| 6 | age | int | age of the building in years |
| 7 | area_percentage | int | normalized area of the building footprint |
| 8 | height_percentage | int | normalized height of the building footprint |
| 9 | land_surface_condition | categorical | surface condition of the land |
| 10 | foundation_type | categorical | type of foundation used while building. |
| 11 | roof_type | categorical | type of roof used while building. |
| 12 | ground_floor_type | categorical | type of the ground floor. |
| 13 | other_floor_type | categorical | type of constructions used higher floors. |
| 14 | position | categorical | position of the building. |
| 15 | plan_configuration | categorical | building plan configuration. |
| 16 | has_superstructure_adobe_mud | binary | if made of Adobe/Mud. |
| 17 | has_superstructure_mud_mortar_stone | binary | if made of Mud Mortar - Stone. |
| 18 | has_superstructure_stone_flag | binary | if the superstructure was made of Stone. |
| 19 | has_su_cement_mortar_stone | binary | if made of Cement Mortar - Stone. |
| 20 | as_superstructure_mud_mortar_brick | binary | if made of Mud Mortar - Brick. |
| 21 | has_superstructure_cement_mortar_brick | binary | if made of Cement Mortar - Brick. |
| 22 | has_superstructure_timber | binary | if made of Timber. |
| 23 | has_superstructure_bamboo | binary | if made of Bamboo. |
| 24 | has_superstructure_rc_non_engineered | binary | if made of non-engineered reinforced concrete. |
| 25 | has_superstructure_rc_engineered | binary | if made of engineered reinforced concrete. |
| 26 | has_superstructure_other | binary | if made of any other material. |
| 27 | legal_ownership_status | categorical | legal ownership status of the land |
| 28 | count_families | int | number of families that live in the building. |
| 29 | has_secondary_use | binary | if used for any secondary purpose. |
| 30 | has_secondary_use_agriculture | binary | if used for agricultural purposes. |
| 31 | has_secondary_use_hotel | binary | if used as a hotel. |
| 32 | has_secondary_use_rental | binary | if used for rental purposes. |
| 33 | has_secondary_use_institution | binary | if used as a location of any institution. |
| 34 | has_secondary_use_school | binary | if used as a school. |
| 35 | has_secondary_use_industry | binary | if used for industrial purposes. |
| 36 | has_secondary_use_health_post | binary | if used as a health post. |
| 37 | has_secondary_use_gov_office | binary | if used as a government office. |
| 38 | has_secondary_use_use_police | binary | if used as a police station. |
| 39 | has_secondary_use_other | binary | if secondarily used for other purposes. |

Table 1: Features

### 2.2 Types of features

In dataset, three kind of features exist: binary, int and categorical.

Binary features indicate that they have value 0 or 1. Integer(int) features indicate that they have integer values. Categorical features indicate that they have a single character denoting the category of that field. Categorical variables(features) have been obfuscated random lowercase ascii characters. The appearance of the same character in distinct columns does not imply the same original value.

### 2.3 Details of features

In total, there are 8 integer, 8 categorical and 22 binary features exist. In Table 1 all details of the features are shown.

## 2.4 Dataset exploration

In this section, integrity of dataset, correlation between features and distributions within dataset are examined.

### 2.4.1 Dataset integrity

There are no missing values for any sample, all the values are filled. Thus, there is no work needs to be done due to lack of data.

### 2.4.2 Correlation between features

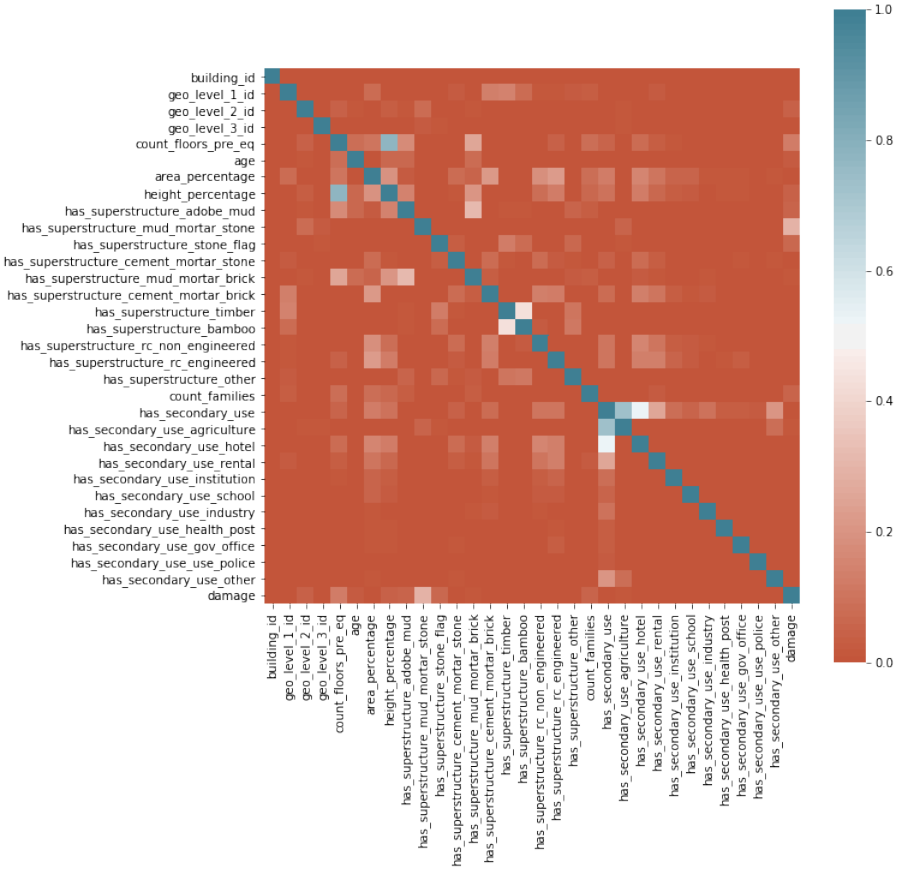


Figure 1: Data correlation matrix

As it may be seen in Figure 1, there are no much correlation between features. Still, a high correlation between $height\_percentage, count\_floors\_pre\_eq$ could be seen.

| damage_grade | # of buildings |
|---|---|
| 1 | 25124 |
| 2 | 148259 |
| 3 | 87218 |

Table 2: damage_grade distribution on training data

| no | feature name | importance |
|---|---|---|
| 1 | has_superstructure_mud_mortar_stone | 0.20683616 |
| 2 | has_superstructure_cement_mortar_brick | 0.09948629 |
| 3 | geo_level_1_id | 0.09206356 |
| 4 | has_superstructure_adobe_mud | 0.078477114 |
| 5 | foundation_type | 0.06116745 |

Table 3: The first five most important features

### 2.4.3 Distribution of data

1- damage_grade: As it can seen in Table 2, the training data is imbalanced. There no equal number of records for each damage_grade.

2- Damage taken according to age of buildings: As it can see below, medium grade is the most common damage type.

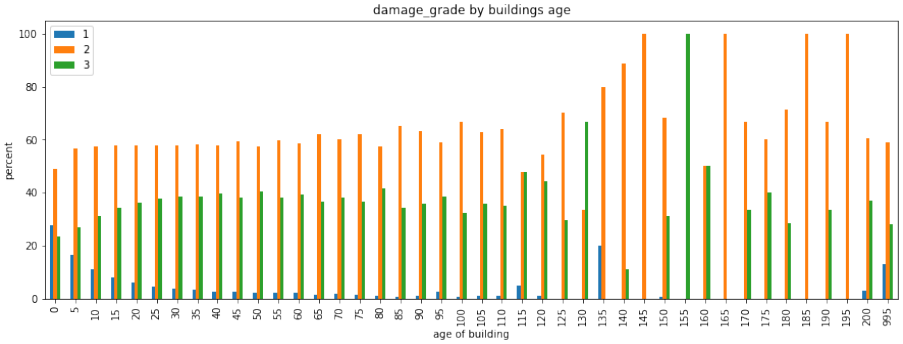


Figure 2: Damage labels with age of buildings

## 2.5 Feature importance

Table 3 shows the first five most important(with most information gain) features. We can claim that these five features are most-effective features as calculating earthquake damage.

# 3 Methodology

In this section, the methodology as trying to estimate earthquake damage is discussed.

## 3.1 Problem type

The problem is a multi-class classification problem since it might be one of three different classes as labels for each sample.

## 3.2 Candidate solution approaches

In this part, some candidate solution approaches are explained and proposed.

### 3.2.1 Hierarchical classification

Hierarchical classification is basically means that mapping input data into defined subsumptive output categories. The classification occurs first on a low-level with highly specific pieces of input data.

### 3.2.2 kNN

Since kNN is a non-parametric(except k) model, without the effort to improve the model, it might solve this problem efficiently as its nature supports multi-class classification.

### 3.2.3 Naive Bayes

Since Naive Bayes is a successful classifier based upon the principle of maximum a posteriori (MAP), this approach is naturally extensible to the case of having more than two classes, and was shown to perform well in spite of the underlying simplifying assumption of conditional independence.

### 3.2.4 Neural networks

Multi-class perceptrons provide a natural extension to the multi-class problem. Instead of just having one neuron in the output layer, with binary output, one could have N binary neurons leading to multi-class classification. In practice, the last layer of a neural network is usually a softmax function layer, which is the algebraic simplification of N logistic classifiers, normalized per class by the sum of the N-1 other logistic classifiers.

## 3.3 Transformation to binary

As discussed in class, there are two types of reduce multi-class classification to binary classification problem

```
1: One class vs rest
2: One class vs one class
```

Using this approach, any binary classifier, some examples in Section 3.2, could be used to solve this problem too.

## 3.4 Feature elimination/extraction

For this problem, 38 features are available in dataset. Although 38 are available, it is not necessary that every feature needs to be used during training.

In Table 3 the first five most important features are shown. Likewise, some number of the less important features could be discarded from dataset. Indeed, the final performance metric might increase.

## 3.5 Scaling

For this problem, 38 features are available in dataset. There are different types of features such as categorical, integer and binary which range in the different scale from each other. For example,

although binary feature would be in range [0-1], age, which is an integer feature, ranges from 0 to 995.

Without preprocessing, some features may affect the classification results more than others because of the bias in the scale of the features that is between zero and one.

Thus, the features need to be scaled so that all the features come into the common scale, that is 0-1. Features in the common scale lack of the bias in the scales, therefore, the classification result may be affected positively.

### 3.6 Hyper-parameter optimization

For any model able to solve this problem, there would be some hyper-parameters within model. Thus, to increase performance metric, the hyper-parameters needs to be optimized.

$GridSearch\ Cross\ Validation$ or $Randomized\ Search\ Cross\ Validation$ are some methods to complete this task.

#### 3.6.1 Grid Search Cross Validation

Grid Search Cross Validation or in short $GridSearchCV$ can be thought of as an exhaustive search for selecting a model. In Grid Search, a grid of hyper-parameter values is set up and for each combination, a model is being trained and the testing data is being scored. In this approach, every combination of hyper-parameter values is tried which can be very inefficient.

#### 3.6.2 Randomized Search Cross Validation

Randomized Search Cross Validation or in short $RandomizedSearchCV$ sets up a grid of hyper-parameter values and selects random combinations to train the model and score. This allows you to explicitly control the number of parameter combinations that are attempted. The number of search iterations is set based on time or resources.

### 3.7 Upsampling & Downsampling

As shown in Figure 1 the data given is imbalanced. Classes that make up a large proportion of the data set are called majority classes. Those that make up a smaller proportion are minority classes. To make it more balanced, there are two ways:

#### 3.7.1 Upsampling

Upsampling means duplicating the minority class until there is not a minority class.

#### 3.7.2 Downsampling and Upweighting

Downsampling means training on a disproportionately low subset of the majority class examples.
Upweighting means adding an example weight to the downsampled class equal to the factor by which you downsampled.
In short, downsampling means discarding some samples of majority class until there is not a majority class.

### 3.8 Ensemble models using bagging & boosting

Some number of classifiers as an ensemble model could be used in this specific task. In addition to the models in Section 3.2, $AdaBoost$, $LightGBM$ or $XGBoost$ could be for this approach too.

# 4 Results

In this section, results for the methodology used are discussed.

## 4.1 Performance metric

For this specific project, the performance metric is $F1\ score$ and for the $average$ parameter, $micro$ is selected.

The parameter $micro$ is used for calculating metrics globally by counting the total true positives, false negatives and false positives.

$\mathrm{F}_{micro} = (2 * P_{micro} * R_{micro})/(P_{micro} + R_{micro})$

## 4.2 Performance baseline

The baseline in the competition is Random Forest with score 0.5815.

Thus, any solution to submit needs to have higher F1 score than 0.5815.

## 4.3 First attempts with basic models

Very first attempts were total failures. None of the basic models(kNN, Decision Tree, Naive Bayes) could exceed the baseline score 0.5815.

## 4.4 First attempts with a more complex model

In comparison to Section 4.3, more complex models such as XGBoost were obviously more successful. It had 0.7301 F1 score.

## 4.5 Increasing the F1 score

The highest F1 score among the approaches was 0.7301 from Section 4.4. To increase it more, there are several ways to try.

### 4.5.1 Upsampling & Downsampling

For this problem, it is obvious that the dataset is very unbalanced. It could be seen in Table 2. So, both upsampling and downsampling were tried to make it balanced but there was no significant increase on F1 score. It was almost the same.

### 4.5.2 Feature extraction

Like it is mentioned in Section 3.4, some feature extraction methods were tried.

**Algorithm 1:** Feature extraction

**Result:** Model M
declare M as a new model with all features;
sorted = features' information gain in ascending order;
**while** *sorted is not empty* **do**
    extract the feature in the first index
    Assign M as a model of remaining features;
    predict test labels using M;
    calculate f1 score of M;
    store the f1 score and M;
**end**
return model M with the highest f1 score;

Once algorithm 1 was run, it returned a model with 26 features and whose F1 score 0.7267 was slightly lower than the previous XGBoost model. Therefore, it did not work the way as expected.

### 4.6 Feature scaling & normalization

As explained in Section 3.5, feature scaling is tried to increase F1 score.
$MinMaxScaler$ is used to achieve this task.

$MinMaxScaler$ transforms features by scaling each feature to a given range. In this attempt, given range was $[0, 1]$.

Eventually, the trained model with scaled features resulted 0.7293 F1 score, still, slightly lower than XGBoost model. It made F1 score decrease.

### 4.7 Hyper-parameter optimization

As mentioned in Section 3.6 two different approaches are used to increase F1 score.

#### 4.7.1 Grid Search Cross Validation

Grid Search Cross Validation or GridSearchCV in short is one of the common approaches to tuning(optimization) hyper-parameter.

For a XGBoost model, with the candidate parameters below, GridSearchCV algorithm has been run.

```
parameters = {'nthread':[4],
              'objective':['binary:logistic'],
              'learning_rate': [0.1],
              'max_depth': [3, 4, 5],
              'min_child_weight': [1],
              'subsample': [0.8],
              'lambda':[0.5, 1],
              'gamma': [1.5, 2, 5],
              'colsample_bytree': [0.8, 1.0],
              'n_estimators': [500, 1000],
              'missing':[-999], 'seed': [1337]}
```

Also, $n\_jobs$ parameter given to GridSearchCV was 2.

Thus, it took about 9 hours to find the best parameter combinatior for XGBoost. When XGBoost has been trained with the parameters returned by GridSearchCV, the F1 score was 0.7301. It was the exact same F1 score with the version of XGBoost as no GridSearchCV was used.

#### 4.7.2 Randomized Search Cross Validation

Randomized Search Cross Validation or $RandomizedSearchCV$ is, like $GridSearchCV$, a common hyper-parameter tuning method.

In this attempt, parameters below are given to the algorithm:

```
parameters ={
     'n_jobs': [-1],
     'n_estimators': np.arange(100,1000,100),
     'max_depth':np.arange(10, 100, 15),
     'learning_rate': [0.03, 0.01, 0.12]
 }

 p.s. np stands for numpy library
```

This phase took about 23 hours to complete.

With the best parameters returned, XGBoost has been trained. The F1 score was $0.7441$, the highest F1 score until that moment.

| approach | # of attributes | F1 score |
|---|---|---|
| kNN | 38 | 0.5130 |
| XGBoost with feature extraction | 26 | 0.7267 |
| XGBoost with feature extraction & scaling | 26 | 0.7293 |
| Default XGBoost | 38 | 0.7301 |
| XGBoost with GridSearchCV | 38 | 0.7301 |
| XGBoost with max_depth=10 | 38 | 0.7424 |
| XGBoost with RandomizedSearchCV | 38 | 0.7441 |

Table 4: F1 score comparison

### 4.8 Ensemble models

Using 3 different models and $VotingClassifier$, AdaBoost, XGBoost and Light GBM an ensemble model was created.
The final F1 score was slightly lower than the default XGBoost model.

To increase the accuracy, the number of models in the ensemble model has been increased.
Unfortunately, due to hardware limitation, more than 3 models together were not supported in the computer where training is being done.

### 4.9 Comparing the approaches

As it might be seen in the Table 4, the best solution for the problem is using $XGBoost\ with\ Randomized\ Search\ Cross\ Validation$ for the dataset given.

## 5 Conclusion

The problem was predicting the damage given by earthquake to the some buildings with some features.

To solve this problem efficiently, after the given dataset belonging to the problem was being analyzed in detail, a methodology was determined and several ways were tried.

As result, it was seen that the most successful approach was using $XGBoost\ with\ Randomized\ Search\ Cross\ Validation$ for the dataset given among the methods proposed.

Due to hardware limitation, some approaches(e.g. Ensemble model including more than 3 models) were not suitable for the problem at the moment. Also, due to time limitation, during GridSearchCV and RandomizedSelectCV, the number of candidate parameters were limited. With more time, the number of candidates could be increased so it could increase the F1 score.

With the current best model, F1 score is 0.7441 on test data. In future work, approaches which were unable to use and unused models(e.g. neural networks) could be trained to acquire better F1 score.

In the competition, which I attended alone, with the F1 score 0.7441 I was 191th among 2930 competitors. The first competitor/group had 0.7558 F1 score at the time this report is written.

### Acknowledgments

Besides the theoretical information given in the course, the project of the course adds a lot to the student who takes the course.

# References


[1] Earthquake definition.

[2] Hierarchical classification

[3] K-nearest neighbors algorithm

[4] Naive Bayes

[5] Multi-class classification

[6] Feature scaling

[7] Grid search cross validation

[8] Randomized search cross validation

[9] Downsampling and upweighting

[10] AdaBoost

[11] LightGBM

[12] XGBoost

[13] Ensemble models

[14] MinMaxScaler

[15] Precision and recall