Report For Modeling Earthquake Damage

Junkai Liang
Viterbi School of Engineering
University of Southern California
junkaili@usc.edu

Gangda Deng
Viterbi School of Engineering
University of Southern California
gangdade@usc.edu

Abstract

To predict the ordinal variable *damage_grade*, which represents a level of damage to the building that was hit by the earthquake, we use 38 features consisting of information on the buildings' structure and their legal ownership to build a 2-layer hierarchical model composed of ensemble classifiers of several **Gradient Boosting Decision Trees**. We achieve an **micro averaged F1 score** of 0.7542 on the testing set from `www.drivendata.org`, which is the 1^{st} place on the course leaderboard and the 5^{th} place on the public leaderboard.

1 Introduction

1.1 Problem description

The goal is to predict the level of damage to buildings caused by the 2015 Gorkha earthquake in Nepal, based on aspects of building location and construction. The data was collected through surveys by Kathmandu Living Labs and the Central Bureau of Statistics, which works under the National Planning Commission Secretariat of Nepal. The dataset contains valuable information on earthquake impacts, household conditions, and socio-economic-demographic statistics.

An ordinal variable *damage_grade* is used to represent the level of damage to the building we want to predict. There are 3 grades of the damage, listed in Table 1

Table 1: Damage grade values

damage_grade	Representation
1	low damage
2	a medium amount of damage
3	almost complete destruction

Features The dataset mainly consists of information on the buildings' structure and their legal ownership. There are 39 columns in the dataset, where *building_id column* is a unique and random identifier. The remaining 38 features include geographic region, age, superstructure information, second use information and other household conditions.

Metric The metric we want to maximize is the **micro averaged F1 score** on the testing set from *www.drivendata.org*, which balances the precision and recall of a multi-class classifier.

1.2 Model introduction

We use a hierarchical model to predict *damage_grade*. It is mainly composed of 2 layers.

First layer The first layer is a 3-class classifier (`global_classifier`), which classifies the data points into 3 classes: 1, 2, and 3 representing the damage grade, described in Section 3.

Second layer The second layer is 2 binary classifiers. One is to classify the data points into 2 classes: 1 and 2 (`binary_classifier_1_2`); the other one is to classify the data points into 2 classes: 2 and 3 (`binary_classifier_2_3`), described in Section 4.

Combination of 2 layers Based on the results of the first-layer classifier, we further classify the data points using the second-layer classifiers. If a data point is classified as 1 in `global_classifier`, we use the result of `binary_classifier_1_2` as the prediction; if a data point is classified as 2 in `global_classifier`, we combine the results of `binary_classifier_1_2` and `binary_classifier_2_3`, more details in Section 4.3; if a data point is classified as 3 in `global_classifier`, we use the result of `binary_classifier_2_3` as the prediction.

2 Data analysis and pre-processing

2.1 Dataset analysis

Data amount We have 260601 data points in the training set, with 9749 duplicates (about 3.7%). And we have 86868 data points in the testing set to predict, with 1786 duplicates (about 2.1%).

Among all the 260601 data points in the training set, there are 25124 data points with label 1, 148259 data points with label 2, and 87218 data points with label 3. See Figure 1.

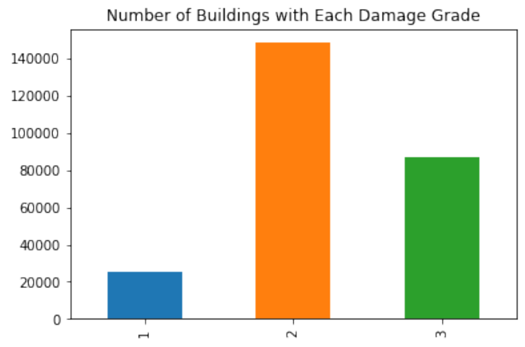

Figure 1: Number of data points with each label.

Feature types Among 38 features of a data point, there are 8 features of type int, 8 features of type categorical, and 22 features of type binary. Among 22 binary features, 11 of them describe the superstructure of the building, the other 11 describe the secondary use of the building.

Geographic Features There are 3 features describing the geographic information of the building, *geo_level_1_id*, *geo_level_2_id*, and *geo_level_3_id*, from largest (level 1) to most specific sub-region (level 3). There are 31 distinct values of *geo_level_1_id*, from 0 to 30; 1414 distinct values of *geo_level_2_id*, from 0 to 1427; 11595 distinct values of *geo_level_3_id*, from 0 to 12567.

The data points with same *geo_level_2_id* have same *geo_level_1_id* and similarly, the data points with same *geo_level_3_id* have same *geo_level_2_id* and *geo_level_1_id*. However, these three features are not correlated in terms of numerical values and continuity, so we treat them as categorical features although they are in type of int.

2.2 Data pre-processing

Missing data There are no missing data in the training dataset or the testing dataset, so we don't have to fill the missing cells.

Categorical feature There are 8 native categorical features with at most 10 distinct values. We apply One-Hot Encoding to them if needed.

The 3 geographic features discussed in Section 2.1 are also treated as categorical features. However, the origin int value can be seen as Label Encoding, so we don't need to perform any additional pre-processing on them.

3 First-layer 3-class classifier

The model has a hierarchical structure consisting of two layers. Both layers take the original features as input and output the probabilities of each label. Since the only difference between the classifiers is the dimension of the label space, we use the same training methods, ensembling techniques, and hyperparameters (of each base model) for all classifiers, for simplicity. In this section, we will introduce the first-layer 3-class classifier which classifies the data points into 3 classes: 1, 2, and 3 representing the damage grade. If not otherwise specified, the architecture of binary classifiers should be the same as the 3-class classifier.

3.1 K-fold Model Selection

The K-fold model selection is a critical technique to avoid overfitting and reduce the variance of the base model. We set K to 5 and applied a 5-fold model selection to all the classifiers. By splitting the training set evenly into K fold according to index, we are able to train K different models with the guidance of the validation set. For the k-th model, where $k = 0, 1, ..., K$, we leave the k-th fold data as the validation set and train the model on the reset of the data. After training K models with early stopping, we average the predictions made on the test set by each model. This technique effectively reduces variance by ensembling predictions from different models trained on partial-overlapping data.

Note that we do not directly average models tuned from the K-fold cross-validation grid search. The estimated training time is unacceptable for exploring a large hyperparameter space. Instead, we fix the train-validation split and tune the hyperparameters on the validation set using the hyperparameter optimization framework `Optuna` Akiba u. a. (2019). We tried on 10+ hyperparameters and 1000+ trails for each base model. However, most hyperparameters are not effective regarding the accuracy of the test set, and only using a fixed partial dataset as a validation set sacrifices the model's generalizability. We believe using a K-fold cross-validation grid search on a selected hyperparameter set would result in higher accuracy for each base model.

3.2 Ensemble Base Models with Generalized Weighted Mean

Ensembling is a powerful yet simple technique to significantly improve the accuracy of each base model. Previous work *www.drivendata.org* shows that simple Gradient Boosting Decision Trees (GBDT) have the potential to achieve one of the best scores on the leaderboard. A straightforward idea is to ensemble multiple trained GBDTs to get a stronger classifier. We choose `CatBoost` Dorogush u. a. (2018) and `LightGBM` Ke u. a. (2017) as our base models, and blend the predictions of these two base models using generalized weighted mean Bullen (1987). We tune the weights on a leave-out validation set using `Optuna` Akiba u. a. (2019).

The following equation shows the formula of generalized weighted mean,

$$\bar{x} = \left(\sum_{i=1}^{n} w_i x_i^p \right)^{1/p}$$

where x_i is the probability predicted by the i-th base model, $w_i \in (0, 1)$ is the weight of i-th base model, and $p \in (-2, 2)$ is a exponent factor. Both w_i and p are tuned by the hyperparameter.

4 Second-layer binary classifiers

In this section, we will introduce second-layer binary classifiers used in the hierarchical model.

4.1 Motivation of binary classifiers

The goal is to predict the ordinal variable representing a level of damage. The variable can take values from 1 to 3, representing the lowest to the highest grade of damage.

We mainly treat this as a 3-class classification problem and first apply the 3-class classifier `global_classifier` discussed in Section 3. We analyze the result of `global_classifier` and generate the confusion matrix on the validation set. See Table 2

Table 2: Confusion matrix of global_classifier

Number of data points	Prediction label: 1	Prediction label: 2	Prediction label: 3
True label: 1	14106	10658	360
True label: 2	5581	125772	16906
True label: 3	601	30064	56553

We can find the most mistakes are mis-classification between label 1 and 2, and between label 2 and 3. This is intuitive, because it is less likely for a proper classifier to predict the building with lowest damage as having the highest damage, or vice versa.

So we use 2 binary-classifiers in the second layer to improve the performance of the model, as binary-classification is easier than multiclass-classification and binary-classifiers commonly have better results. One of the binary-classifiers is to classify data points between label 1 and 2 (`binary_classifier_1_2`) and the other one is to classify data points between label 2 and 3 (`binary_classifier_2_3`).

4.2 Training binary classifiers

Training data The training dataset has 3 labels, so we need to pre-process the data before feeding them to the binary-classifiers. There are 2 approaches.

The first one is to relabel the training data: re-label the data points with label 3 as label 2 when training `binary_classifier_1_2` and re-label the data points with label 1 as label 2 when training `binary_classifier_2_3`.

The second one is to filter the training data: only use data with label 1 and label 2 to train `binary_classifier_1_2` and only use data with label 2 and label 3 to train `binary_classifier_2_3`.

Although there are more training data if we adopt the first approach, the second approach has better results. This is because in the training dataset, data points with label 2 has the largest amount (shown in Figure 1). If we adopt the first approach, data imbalance between label 1 and label 2 (in `binary_classifier_1_2`), and between label 2 and label 3 (in `binary_classifier_1_2`) is exacerbated, which leads to a decrease in the performance of the binary classifiers.

So we adopt the second approach, i.e., filtering the training data to pre-process the data fed to the binary-classifiers.

Training method We apply similar cross-validation and ensemble tricks described in Section 3 to train the binary classifiers.

4.3 Prediction with two layers of the model

When predicting with the 2-layer model, we decide which binary classifier(s) to use based on the result of the first layer classifier, i.e., `global_classifier`.

If a data point is classified as label 1, i.e., the lowest damage, it is not likely to have the highest damage, i.e., label 3. So we re-classify it with `binary_classifier_1_2` and take the result as the final prediction.

Similarly, if a data point is classified as label 3, i.e., the highest damage, it is not likely to have the lowest damage, i.e., label 1. So we re-classify it with `binary_classifier_2_3` and take the result as the final prediction.

Otherwise, if a data point is classified as label 2, i.e., intermediate damage, the true label of it may be any of the 3 possible labels. So we need to pass it through both `binary_classifier_1_2` and `binary_classifier_2_3`, and combine the results.

Combination of results of binary classifiers There are 4 possible result combinations.

`binary_classifier_1_2` predicts 1; `binary_classifier_2_3` predicts 2: both classifiers predict the building as having low damage, so it is most likely to have the lowest damage, i.e., label 1.

`binary_classifier_1_2` predicts 2; `binary_classifier_2_3` predicts 3: both classifiers predict the building as having high damage, so it is most likely to have the highest damage, i.e., label 3.

`binary_classifier_1_2` predicts 2; `binary_classifier_2_3` predicts 2: both classifiers predict the building as having intermediate damage, so it is most likely to have intermediate damage, i.e., label 2.

`binary_classifier_1_2` predicts 1; `binary_classifier_2_3` predicts 3: there are conflicts between the binary classifiers, so we keep the result of the 3-class classifier `global_classifier`, i.e., label 2.

Pseudo code Here is the pseudo code for the entire prediction process. See Algorithm 1

Algorithm 1 The predicting process of the model.

```
Train the 3-class classifier global_classifier and the binary classifiers binary_classifier_1_2, binary_classifier_2_3 with the training data.
Input: Data point with feature $\boldsymbol{x}$.
Output: Prediction label $y \in \{1, 2, 3\}$ representing damage grade.
$global\_classifier\_result \leftarrow global\_classifier(\boldsymbol{x})$
if $global\_classifier\_result = 1$ then
    $y \leftarrow binary\_classifier_{1,2}(\boldsymbol{x})$
else if $global\_classifier\_result = 3$ then
    $y \leftarrow binary\_classifier_{2,3}(\boldsymbol{x})$
else
    $binary\_result_{1,2} \leftarrow binary\_classifier_{1,2}(\boldsymbol{x})$
    $binary\_result_{2,3} \leftarrow binary\_classifier_{2,3}(\boldsymbol{x})$
    if $binary\_result_{1,2} = 1$ and $binary\_result_{2,3} = 2$ then
        $y \leftarrow 1$
    else if $binary\_result_{1,2} = 2$ and $binary\_result_{2,3} = 3$ then
        $y \leftarrow 3$
    else if $binary\_result_{1,2} = 2$ and $binary\_result_{2,3} = 2$ then
        $y \leftarrow 2$
    else if $binary\_result_{1,2} = 1$ and $binary\_result_{2,3} = 3$ then
        $y \leftarrow global\_classifier\_result$
    end if
end if
return prediction label $y$.
```

4.4 Results of the second layer

We use the testing dataset consisting of 86868 data points to test the model and compare the results of the first-layer 3-class classifier, i.e., `global_classifier` and the second layer, i.e., the final output of the model and show the differences in Table 3

Table 3: Difference between two layers on testing set

Number of data points	First layer output: 1	First layer output: 2	First layer output: 3
Final output: 1	6421	251	
Final output: 2	356	54700	595
Final output: 3		678	23867

The conflicts, i.e., `global_classifier` predicts 2, `binary_classifier_1_2` predicts 1, and `binary_classifier_2_3` predicts 3, only appeared **twice** during the prediction on the testing set. This indicates that the results of all three classifiers are highly reliable.

Using the second-layer binary classifiers to enhance the model makes the micro averaged F1 score on the testing set improve from 0.7534 to 0.7542.

5 Results

We went through several stages to build and optimize the model, and ultimately achieved a micro averaged F1 score on the testing set of 0.7542.

Stage 1 Use `CatBoost` multiclass-classifier with hand-set hyper-parameters. Achieved score: 0.7519.

Stage 2 Use `CatBoost` multiclass-classifier with tuned hyper-parameters. Achieved score: 0.7523.

Stage 3 Combine the results of multiclass-classifiers of `CatBoost` and `LightGBM` and tune the weight. Achieved score: 0.7534.

Stage 4 Construct the 2-layer structure and use binary classifiers to enhance the model. Achieved score: 0.7542.

References

[Akiba u. a. 2019] AKIBA, Takuya ; SANO, Shotaro ; YANASE, Toshihiko ; OHTA, Takeru ; KOYAMA, Masanori: Optuna: A Next-generation Hyperparameter Optimization Framework. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019

[Bullen 1987] BULLEN, Peter S.: Handbook of means and their inequalities, 1987

[Dorogush u. a. 2018] DOROGUSH, Anna V. ; ERSHOV, Vasily ; GULIN, Andrey: CatBoost: gradient boosting with categorical features support. In: *arXiv preprint arXiv:1810.11363* (2018)

[Ke u. a. 2017] KE, Guolin ; MENG, Qi ; FINLEY, Thomas ; WANG, Taifeng ; CHEN, Wei ; MA, Weidong ; YE, Qiwei ; LIU, Tie-Yan: Lightgbm: A highly efficient gradient boosting decision tree. In: *Advances in neural information processing systems* 30 (2017)

A Appendix

A.1 Instruction for running the code

The code is available at `https://github.com/Hydrapse/ds-richter`.

Firstly, execute all the code blocks in notebook *danny_catboost_1.ipynb* and *danny_lightboost_1.ipynb* to get the intermediate results of 3-class classifier base models.

Secondly, execute all the code blocks in notebook *junkai_2classifier_catboost.ipynb* and *junkai_2classifier_lightboost.ipynb*. Remember to execute twice with the `split_set` equals to "1_2" and "2_3" to get the base-model intermediate results of both binary classifiers.

Then, execute all the code blocks in notebook *junkai_predict.ipynb* to ensemble the base-model intermediate results. Remember to set proper intermediate result filenames of 3-class classifier and binary classifiers.

Finally, execute all the code blocks in notebook *junkai_3_2_predict.ipynb*, which will read the results generated in former steps. The final prediction will be made through hierarchical structure of the model we introduced in this report.