Richter's Predictor: Modeling Earthquake Damage

Team Name: 404

Aishwarya Kumaraswamy
PES1201701575
PES University
aishwaryakalgudi@gmail.com

Bhargava N Reddy
PES1201701016
PES University
bharghava.n.reddy@gmail.com

Rithvik K
PES1201700152
PES University
rithvikkolla99@gmail.com

***Abstract*—Richter's Predictor is used to is to predict the level of damage to buildings caused by the 2015 Gorkha earthquake in Nepal. Considering some basic aspects like building location, age of the building, construction details and it's secondary uses, the model is expected to predict the level of damage as one of low, medium or high.**

***Index Terms*—earthquake damage, classifier**

I. Introduction

Following the 7.8 Mw Gorkha Earthquake on April 25, 2015, Nepal carried out a massive household survey using mobile technology to assess building damage in the earthquake-affected districts. The primary goal of this survey was to identify beneficiaries eligible for government assistance for housing reconstruction, this survey collected a lot of other useful socio-economic information that can be used to predict up to what extent the building has been damaged and draw conclusions about what factors cause low damage and what causes high damage or almost complete destruction. In order to do this, the model considers the factors that help in deciding how strong a building structure is or how prone the location is to the earthquake that took place.

II. Dataset

The dataset [1] consists of information on the buildings' structure and their legal ownership. Each row represents a building that was affected by the Gorkha earthquake. There are 39 columns in the dataset with *building_id* being the unique and random identifier. We are trying to predict the ordinal variable *damage_grade*, which represents a level of damage to the building that was hit by the earthquake. There are three grades of damage -

- 1 represents low damage
- 2 represents a medium amount of damage
- 3 represents almost complete destruction

Some information about the other columns -

- *geo_level_1_id, geo_level_2_id, geo_level_3_id* (Integer): Geographic region in which building exists, from largest (level 1) to most specific sub-region (level 3).
- *count_floors_pre_eq* (Integer): Number of floors in the building before the earthquake.
- *age* (Integer): Age of the building in years
- *area_percentage* (Categorical): Normalized area of the building footprint
- *height_percentage* (Categorical): Normalized height of the building footprint
- *land_surface_condition* (Categorical): surface condition of the land where the building was built
- *foundation_type* (Categorical): Type of foundation used while building
- *roof_type* (Categorical): Type of roof used while building
- *ground_floor_type* (Categorical): Type of the ground floor
- *other_floor_type* (Categorical): Type of constructions used in higher than the ground floors (except of roof)
- *position* (Categorical): Position of the building
- *plan_configuration* (Categorical): Building plan configuration
- *has_superstructure* (Binary): Columns with prefix 'has_superstructure' are flag variables which indicate presence of specific superstructures that are not common to all buildings
- *legal_ownership_status* (Categorical): Legal ownership status of the land where building was built
- *count_families* (Integer): Number of families living in that building
- *has_secondary_use* (Binary): Columns with the prefix 'has_secondary_use' are flag variables which indicate if the buildings were used for any specific secondary purposes

III. Performance Metric

The performance of the multiclass classifiers can be measured using the following:

- Micro Averaged F1 Score
- Receiver Operating Characteristic (ROC)
- Area Under Curve (AUC)

IV. Classification Models

A. Logistic Regression

Logistic regression is a statistical model used to model a binary independent variable, although many more complex extensions exist. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative distribution function of logistic distribution. Logistic regression is mainly used for binary classification. Our ordinal variable *damage_grade* has

3 grades and would require multi-class classification models. Also the data is imbalanced, as in most of the buildings in our dataset fall into *damage_grade* 2. This is why logistic regression does not work very well in our case.

B. Random Forest

Random Forest Classifier is ensemble algorithm. In next one or two posts we shall explore such algorithms. Ensemble algorithms are those which combines more than one algorithms of same or different kind for classifying objects. Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object. Random forest classifiers, gave an improvement in accuracy when compared to logistic regression. In general, random forest classifiers are known to perform rather poorly on imbalanced data.

Sl No	Model	Accuracy
1	Logistic Regressor	54.701%
2	Random Forest	76.94%

V. Our Approach

The performance of machine learning algorithms is typically evaluated using predictive accuracy. However, this is not appropriate when the data is imbalanced. The imbalance can be fixed using the following approaches:

A. Under-sampling the Majority Class

To remedy the problem of imbalanced data, the approach of undersampling [2] involves taking a sample of the majority class label. Undersampling can be carried out in any of the following ways:

- Random undersampling: A simple under-sampling technique is to under-sample the majority class randomly and uniformly. This can potentially lead to loss of information.
- Near Miss: In order to prevent potential information loss, "near neighbor" method calculates the distances between all instances of the majority class and the instances of the minority class. Then k instances of the majority class that have the smallest distances to those in the minority class are selected.

B. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE [3] shows that a combination of our method of over-sampling the minority class and under-sampling the majority class can achieve better classifier performance (in ROC space) than only under-sampling the majority class. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. The synthetic examples cause the classifier to create larger and less specific decision regions.

We want to explore the XGBoost classifier, Neural Networks along with one of the above approaches. Tree boosting has been shown to give state-of-the-art results on many standard classification benchmarks. XGBoost [4] is an ensemble learning method, and one of it. XGBoost build trees one at a time, where each new tree helps to correct errors made by previously trained tree. It works well with unbalanced data. With correct tuning parameters, they generally give better results than Random Forests.

REFERENCES

[1] *https://www.drivendata.org/competitions/57/nepal-earthquake/data/* Nepal's Earthquake Damage

[2] Ajinkya More (2016), Survey of resampling techniques for improving classification performance in unbalanced datasets.

[3] Nitesh V. Chawla, Kevin W. Bowyer (2002), SMOTE: Synthetic Minority Over-sampling Technique.

[4] Tianqi Chen, Carlos Guestrin (2016) XGBoost: A Scalable Tree Boosting System.