2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC) | 978-1-7281-9183-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICAECC50550.2020.9339484

Richter's Predictor: Modelling Earthquake Damage Using Multi-class Classification Models

Aishwarya Kumaraswamy
PES University
aishwaryakalgudi@gmail.com

Bhargava N Reddy
PES University
reddybhargava@outlook.in

Rithvik Kolla
PES University
rithvikkolla99@gmail.com

***Abstract*—Natural calamities like earthquakes cause damage to life and property. Estimation of damage grade to buildings is essential for post-calamity response and recovery, elimination of the tedious process of manual validation and authentication of property damage before granting relief funds to people. By considering basic aspects like building location, age of the building, construction details and it's secondary uses, taken from the Gorkha earthquake dataset, this paper explores various multi-class classification machine learning models and techniques for predicting the damage grade of structures. The proposed architecture of the model involves three major steps, Feature Selection, XGBoost Classifier, and Parameter Tuning. The paper presents the results of the experiments with feature engineering, training variations and ensemble learning. The paper delves into the analysis of each model, to understand the reason behind their performance. This paper also infers the agents that play a major role in deciding the seismic vulnerabilities of the buildings. The proposed classifier in the paper provides significant input to understanding earthquake damage and also provides a paradigm to model other natural disaster damage.**

***Index Terms*—earthquake damage, classifiers, XGBoost, ensemble learning**

I. Introduction

There is a significant need to assess the damage done by earthquakes in calamity-hit regions. Predicting the damage done to buildings helps in identifying the beneficiaries eligible for government assistance for housing reconstruction. It is a critical part of the post-calamity recovery phase. Manual identification and classification require significant time and resources and leads to distress among the population. There have been attempts made to predict the seismic damage using satellite imagery [2], [3] of earthquake-hit regions, but these are compute-intensive. The April 2015 Nepal earthquake, the Gorkha earthquake, was one of the most devastating disasters that has ever hit Nepal. It had a moment magnitude of 7.8 on the Richter Scale. It killed around 9000 people, also causing injuries to 22000 others. In the aftermath of the Gorkha earthquake, Nepal performed an extensive household survey by using mobile technology to evaluate the extents of building damages in those areas affected by the earthquake. The survey, which is now one of the largest post-disaster datasets ever compiled, contains useful information on the impact of the earthquake, the household conditions, and the socio-economic demographics of the disaster. This paper aims to evaluate and estimate the damage caused to each building by the 2015 Gorkha earthquake based on the characteristics of the building location and its construction. The assessment could help two types of end-users:

- Government agencies - Government agencies will benefit from a closer and a faster approximation on the damage and havoc created by the earthquakes without any manual inspection. This can be conducive to effective post-disaster recovery operations.
- Insurers - After a large scale and widespread disaster like the Gorkha earthquake, insurers' claims systems are accumulated with a huge number of new claims. It is incredibly tedious for claim handlers and inspectors to sift through all the data and determine the damage severity. Incorporating AI-based damage evaluation components with claim systems will provide the claim inspectors a single index (damage level), using which they can determine the severity of the damage. It will also quicken claims processing and aid in sending help to the people who are at-risk immediately.

These predictions also help in understanding the effect of earthquakes and the seismic vulnerability of various buildings in different seismic zones. These are required for precise estimation of seismic disaster and also for planning risk reduction.

In this paper, the approach section discusses the architecture of the foundation model. The experiments section couples the multi-class classification model in approach with techniques like feature selection, sampling methods like undersampling, oversampling, SMOTE, feature engineering like normalization, log transformations, and study their performance using the F1 score metric. The paper also experiments with four different machine learning models, Logistic Regressor, KNN, Random Forest, XGBoost,and an ensemble of these learners to find the best combination of classification model. This is then followed by the inference and conclusion from these experiments.

II. Dataset

The dataset [1] contains information on the structure of buildings and other socio-economic information about each building. Each row represents a building that was affected by the Gorkha earthquake. There are 39 features with *building_id* being the unique identifier. In this paper, we are predicting

978-1-7281-9183-6/20/$31.00 © 2020 IEEE

Authorized licensed use limited to: The George Washington University. Downloaded on November 23,2023 at 01:32:17 UTC from IEEE Xplore. Restrictions apply.

damage_grade, which is the level of damage to the building that was hit by the earthquake. There are three grades of damage - low level damage, medium amount of damage and high level damage, each represented by an ordinal variable(1, 2, 3) in the dataset. Information about few of the other important columns:

- *geo_level_1_id, geo_level_2_id, geo_level_3_id* (Integer) - Geographic region in which building exists.
- *area_percentage* (Categorical) - Normalized area of the building
- *age* (Integer) - The age of a building.
- *height_percentage* (Categorical) - Height of a building
- *ground_floor_type* (Categorical) - Type of ground floor
- *roof_type* (Categorical): Roof type of a building.
- *count_floors_pre_eq* (Integer) - Number of levels in the building pre-earthquake.
- *has_superstructure* (Binary) - Columns with prefix 'has_superstructure' are flag variables which indicate the presence of specific superstructures that are not common to all buildings
- *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. Approach

The approach consists of three parts: Feature Selection, XGBoost Model, Parameter Tuning. The below block diagram depicts the architecture of our model

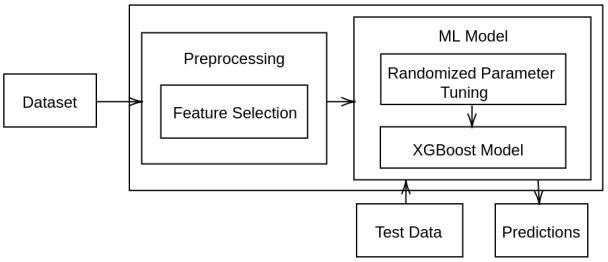

Fig. 1: Block Diagram

A. Feature Selection

Feature selection is used for filtering out irrelevant or redundant features from the dataset as it can negatively impact the performance of an ML model. Choosing the optimal set of relevant features will help reduce the complexity of the model and can also help increase the accuracy and purity of the model. We have used variance threshold as a primary method for feature selection. Using a variance threshold of 0.02 i.e., dropping all the columns with a variance of less than 0.02. It is seen that most of *has_secondary_use_* columns and some of the *has_superstructure_* columns have very low variance. Hence these columns have been dropped for the further analysis.

B. XGBoost

XGBoost [5], short for extreme gradient boosting of decision trees, is used for classification problems, regression problems or for ranking. XGBoost is a boosting algorithm that converts weak learners into strong learners. A tree is grown one after other and attempts to reduce misclassification rate in subsequent iterations. In this, the next tree is built by giving a higher weight to misclassified points by the previous tree. Gradient boosting [7] is a method that goes through cycles to iteratively add models into an ensemble. Simply put, gradient boosting consists of two things, gradient descent and boosting. The objective of any supervised learning algorithm is to define a loss function and minimize it. Similarly, for Gradient Boosting, we have mean squared error (MSE) as loss defined as:

$$Loss = MSE = \sum (y_i - y_i^p)^2 \tag{1}$$

where y_i is the actual value, y_i^p is the predicted value. By using gradient descent and updating our predictions based on a learning rate, the loss function (MSE) is minimized. XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements. Algorithmic Optimization involves:

- Regularization: Penalizes more complex models through both LASSO (L1) and Ridge (L2) regularization [6] to prevent over fitting
- Sparsity Awareness: Admits sparse features for inputs by automatically 'learning' best missing value depending on training loss
- Cross Validation: Algorithm comes with built-in cross-validation method at each iteration

Coming to the model specifications, we need to understand the basic parameters of the model. Let's assume the model has k trees.

$$\hat{y_i} = \sum f_k(x_i), f_k \in SpaceofTrees \tag{2}$$

The objective function is given by:

$$Obj = \sum l(y_i, \hat{y_i} + \sum \Omega(f_k) \tag{3}$$

The former summation calculates the training loss, that is how well the model fits on training data, and the later part is the regularization, measures the complexity of the trees. Optimizing the training loss encourages the predictive models, optimizing the regularization encourages simple models. This is essential as simpler models have smaller variance in future predictions, making predictions stable. Using gradient descent, XGBoost tries to optimize the objective function.

C. Parameter Tuning using Randomized Searching

XGBoost parameters can be divided into three categories:

- General Parameters: Controls the booster type in the model which eventually drives overall functioning.
- Booster Parameters: Controls the performance of the selected booster. These include parameters like nrounds(controls the maximum number of iterations),

Authorized licensed use limited to: The George Washington University. Downloaded on November 23,2023 at 01:32:17 UTC from IEEE Xplore. Restrictions apply.

eta(controls the learning rate), gamma(controls regularization), max_depth(controls the depth of the tree).

- Learning Task Parameters: These parameters specify methods for the loss function and model evaluation. For our problem statement of multi-class classification, we set the appropriate objective function to multi-softmax, and the evaluation metric to calculate model's accuracy on validation data to multi-class log-loss.

To tune the booster parameters, we employ Randomized Search algorithm. Randomized Search implements a randomized search over parameters, where each setting is sampled from a distribution over possible parameter values. This has two main benefits over an exhaustive search like Grid Search, a budget can be chosen independent of the number of parameters and possible values, adding parameters that do not influence the performance does not decrease efficiency. Randomized Search picks the each parameter randomly and finds the error rate of a machine learning model with those set of parameters. After calculating the error rates for different sets of parameters for the given number of iterations, it returns the set of parameters with the least error rate for the machine learning model. Randomized Search was the approach chosen for parameter tuning rather than Grid Search. The time taken to run Grid Search is exponential as it goes through every possible combination of parameters to find out the set that gives the least error rate. Even though Grid Search promises to give the best possible set of parameters, the inefficiency in time complexity is a disadvantage compared to Randomized Search.

IV. Experiments

A. Feature Engineering

To improve the accuracy of our classification model, we performed feature engineering techniques. Some of these techniques include feature normalization, log transformation, one hot encoding of categorical variables. The results are shown in Table I. Normalizing the data is done so that all the input variables have the same treatment in the model and the coefficients of a model are not scaled with respect to the units of the inputs. The scale of certain features in the dataset is high where as some have small scales. Normalization ensures high scaled features do not completely dominate the others. Log [8] Transformation is used to fix the skewness in data. On applying log transformation on the attribute age, which is right skewed, the accuracy decreased by a very small percentage.

Sl No	Model	Accuracy
1	normalized data + XGBoost	73.975%
2	log transformed data + XGBoost	74.377%
3.	one-hot encoded categories + XGBoost	73.982%
4.	XGBoost	74.44%

TABLE I: Feature Engineering

XGBoost is a tree-based decision algorithm that seeks for the best split point in each feature. The split point is determined by the percentage of labels correctly classified using a feature, which is resilient to feature scaling. Therefore, normalization, or transformation does not have any significant impact on XGBoost.

B. Variations in Training using Sampling techniques

To solve the problem of imbalance in the class labels, we looked into techniques like under-sampling, oversampling and SMOTE. Under-sampling [9] is an approach of that involves taking a sample of the majority class label. Oversampling involves supplementing the training data with multiple copies of some of the minority classes. SMOTE [10] is a combination of method of over-sampling the minority class and under-sampling the majority class to achieve better classifier performance than only under-sampling the majority class or only over-sampling the majority class. The table shows the performance of the the above mentioned sampling techniques on the classification model.

As we can see from the table, the accuracy rate of the classifier decreased, when we used sampling techniques. Under-sampling fails as it has an disadvantage of discarding potentially useful data. The main disadvantage with oversampling, is that by making exact copies of existing examples, over-sampling makes over-fitting likely. In fact, with oversampling it is quite common for a learner to generate a classification rule to cover a single, replicated, example.There is also an added disadvantage increasing the learning time when the number of training examples increases. In SMOTE method of sampling, it does not take into consideration neighboring examples can be from other classes, while generating synthetic examples, This increases the overlapping of classes and introduces additional noise.

Sl No	Model	Accuracy
1	Under Sampling + XGBoost	72.62%
2	Over Sampling + XGBoost	71.22%
3.	SMOTE + XGBoost	73.03%
4.	XGBoost	74.44%

TABLE II: XGBoost Variations

C. Basic Models

The following table shows how each of models performed individually on the dataset without any data/feature engineering.

Sl No	Model	Accuracy
1	Logistic Regressor	54.70%
2	k-Nearest Neighbors	71.06%
2	Random Forest	71.95%
3.	XGBoost	74.44%

TABLE III: Basic Models

Logistic regression is usually used as a benchmark before moving on to more complex algorithms. Logistic Regression assumes that there is a linear relationship between the dependent and the independent variables and this is a weakness of this model. It is mostly used for binary classification and does not work well for multi-class classification problems, and is

Authorized licensed use limited to: The George Washington University. Downloaded on November 23,2023 at 01:32:17 UTC from IEEE Xplore. Restrictions apply.

also vulnerable to overfitting.

k-NN [11] captures the idea of similarity, hinges on the point that similar data points are close to each other and that they exist in close proximity. k-NN can be useful in solving problems that have solutions that depend on identifying similar objects. This principle can be useful in our case as we try to predict the damage grade of a building as the features causing a particular damage grade could also cause those data points to lie in a close proximity. k-NN classifer performs much better than the logistic regression model, as we can see from the table above that the k-NN classifier produces an accuracy of around 71% . But k-NN model does suffer from the class imbalance problem in the dataset, as more than 50% of the data is of class 2. Due to this data imbalance there could be more data points of class 2 in the proximity of a data point than the number of data points of its actual class, see Fig 3.

Random Forest [12] classifier uses the ensemble learning technique, creates a set of decision trees on randomly selected data samples(subset) of training set, gets prediction from each tree and selects the best solution. Random Forest is less prone to noise than a single decision tree as the set of many decision trees that the Random Forest uses reduces the effect of noise resulting in more accurate results. It is also equipped to handle non-linear parameters efficiently. But even such a robust classifier could not tackle the problem of data imbalance as we can see just slightly better than the k-NN model.

XGBoost [5], is a powerful method on real-life datasets and are great for anomaly detection in supervised learning settings where data is imbalanced. From the table, we can see that XGBoost clearly outperforms Random Forests and k-Nearest Neighbors due to it's ability to work well on imbalanced data.

D. Ensemble Learning

A collection of models working together on a single dataset is called an ensemble [13] and hence this method is often referred to as ensemble learning. Working on a ensemble of models can improve the stability and predictive power of the model. Typically, the models for the combination process are drawn from the same family of models, but this need not always be case.

The final labels predictions for the ensemble model is calculated by taking a vote. Voting is of two types:

- Hard voting, also known as majority voting where the predicted target label of the ensemble is the one with the highest number of votes from the individually predicted labels.
- Soft Voting [14] uses the probabilities calculated by the classifiers and arrives at the best result by averaging them out. The label with highest probability would then be selected as predicted label of the ensemble.

The ensemble model uses soft voting, an improvement over hard voting as it considers each classifier's uncertainty and probabilities while predicting the final target label.

The following table shows the accuracy's produced by different combinations of models.

Sl No	Model	Accuracy
1	KNN	71.06%
2	Random Forest	71.95%
3	XGBoost	74.44%
4	KNN + Random Forest	73.37%
5	KNN + XGBoost	74.20%
6	KNN + Random Forest + XGBoost	74.59%

TABLE IV: Ensemble Learning

The ensemble of k-NN and Random Forest performs slightly better than individual models as they come from different families of classification models.The ensemble of k-NN and XGBoost performs worse than the XGBoost as k-NN is affected by the data imbalance in the data set and this is strongly influences the predictions during voting. Combining k-NN, Random Forest and XGBoost performs slightly better than just XGBoost alone because of the inclusion of Random Forest classifier which might cancel out or reduce the affect of k-NN's influence on the XGBoost.

V. Results

Fig. 2 shows the features ordered according to it's importance by the XGBoost Model. Geological level ID, area percentage, age, height of the buildings play an important role in predictions. These features are intuitive as well.

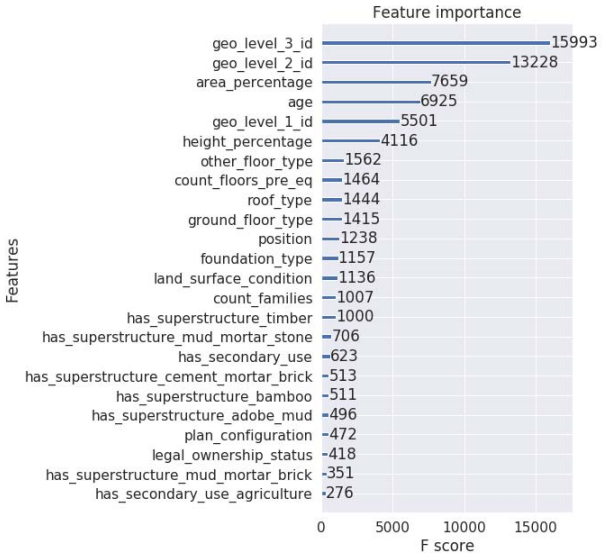

Fig. 2: Feature Importance

The k-NN model (statistics shown in Fig. 3) produces an accuracy of 71.07%. The number of true positives it predicts for damage_grade of 1 and 2 is more than that of Random Forest and XGBoost models. Hence, this might be a good choice for the ensemble of models.

The Random Forest model (statistics shown in Fig. 4) produces an accuracy of 72.12%. The number of true positives it predicts for damage_grade 3 is more than that of k-NN and XGBoost models. Hence, this might also be a reasonably good choice for the ensemble of models.

Authorized licensed use limited to: The George Washington University. Downloaded on November 23,2023 at 01:32:17 UTC from IEEE Xplore. Restrictions apply.

```
k-NN Classifier
Accuracy of k-NN Classifier: 0.7107609265678362

Predicted     1      2     3    All
Actual
1           182   1427   591   2200
2          1097   8721  3573  13391
3           631   5078  2196   7905
All        1910  15226  6360  23496

              precision    recall  f1-score   support

           1       0.59      0.49      0.54      7481
           2       0.72      0.83      0.77     44440
           3       0.72      0.58      0.64     26260

    accuracy                           0.71     78181
   macro avg       0.68      0.63      0.65     78181
weighted avg       0.71      0.71      0.70     78181
```

Fig. 3: Performance metrics: k-NN Classifier

```
Random Forest Classifier
Accuracy of Random Forest Classifier: 0.7212621992555736

Predicted     1      2     3    All
Actual
1           152   1398   650   2200
2          1016   8628  3747  13391
3           576   5051  2278   7905
All        1744  15077  6675  23496

              precision    recall  f1-score   support

           1       0.65      0.49      0.56      7481
           2       0.73      0.83      0.77     44440
           3       0.72      0.61      0.66     26260

    accuracy                           0.72     78181
   macro avg       0.70      0.64      0.66     78181
weighted avg       0.72      0.72      0.72     78181
```

Fig. 4: Performance metrics: Random Forest Classifier

```
XGBoost Classifier
Accuracy of XGBoost Classifier: 0.7438508077410112

Predicted     1      2     3    All
Actual
1           155   1432   613   2200
2           978   8628  3785  13391
3           564   5080  2261   7905
All        1697  15140  6659  23496

              precision    recall  f1-score   support

           1       0.68      0.51      0.58      7481
           2       0.74      0.85      0.79     44440
           3       0.76      0.63      0.69     26260

    accuracy                           0.74     78181
   macro avg       0.73      0.66      0.69     78181
weighted avg       0.74      0.74      0.74     78181
```

Fig. 5: Performance metrics: XGBoost Classifier

The XGBoost model (statistics shown in Fig. 5) produces an accuracy of 74.38%. As this model performs slightly better than the previous two models, it might be a good model to build on, for our ensemble model.

The ensemble classifier (statistics shown in Fig. 6) performs better than the other 3 models individually did. The number of true positives for damage_grade 1 decreases, but it increases slightly for damage_grades of 2 and 3.

```
Ensemble Classifier: kNN + Random Forest + XGBoost
Accuracy of Ensemble Classifier: 0.7457182691446771

Predicted     1      2     3    All
Actual
1           142   1449   609   2200
2           916   8709  3766  13391
3           525   5111  2269   7905
All        1583  15269  6644  23496

              precision    recall  f1-score   support

           1       0.70      0.48      0.57      7481
           2       0.74      0.86      0.79     44440
           3       0.76      0.63      0.69     26260

    accuracy                           0.75     78181
   macro avg       0.74      0.66      0.69     78181
weighted avg       0.75      0.75      0.74     78181
```

Fig. 6: Performance metrics: Ensemble learning

VI. Conclusion and Future Scope

The results from the study showed that predictions based on the XGBoost algorithms have an overall accuracy of 74.59% in assigning the tags on the test set. This kind of a holistic approach to post-disaster recovery operations helps in cutting down cost and the precious amount of time. More work is needed to enable widespread adoption of machine learning algorithms for earthquake-induced building damage predictions. Working with datasets consisting of damage grade caused due to other natural calamities like floods, hurricanes would be the future scope. More additional features like soil conditions, building remodelling histories, etc. will help improve the overall accuracy. Datasets with features considering the seismic magnitude of the earthquake, seismic history of a region will help in generalizing the prediction models to different regions.

References

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

[2] Paolo Gamba and Fabio Casciati, GIs and Image Understanding for Near-RealTime Earthquake Damage Assessment

[3] Lichen Wang, Ken'ichi Kawaguchi, Pujin Wang (2020). Damaged ceiling detection and localization in large-span structures using convolutional neural networks

[4] Mohamed Aly (Nov. 2005), Survey on Multiclass Classification Methods

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

[6] L.E.Melkumova, S.Ya.Shatskikh (2017) Comparing Ridge and LASSO estimators for data analysis.

[7] Jerome. H. Friedman(1999) Greedy Function Approximation: A Gradient Boosting Machine

[8] Changyong FENG, Hongyue WANG, Naiji LU, Tian CHEN, Hua HE, Ying LU, Xin M. TU (2014) Log-transformation and its implications for data analysis

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

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

[11] Gongde Guo, Hui WangHui Wang, David A. Bell, Yaxin Bi (2004) KNN Model-Based Approach in Classification

[12] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001)

[13] Dean W. Abbott (1999), Combining Models to Improve Classifier Accuracy and Robustness

[14] Haishen WangYan YangHongjun WangDahai Chen (2003) Soft-Voting Clustering Ensemble.

Authorized licensed use limited to: The George Washington University. Downloaded on November 23,2023 at 01:32:17 UTC from IEEE Xplore. Restrictions apply.