PREDICTION OF DAMAGE LEVEL OF INFRASTRUCTURE CAUSED BY THE 2015 GORKHA EARTHQUAKE IN NEPAL

Project Proposal

Aprajita ARORA
ESSEC-CentraleSupélec
aprjita.arora@student-cs.fr

Lasse SCHMIDT
ESSEC-CentraleSupélec
lasse.schmidt@student-cs.fr

Abhimanyu SONI
ESSEC-CentraleSupélec
abhimanyu.soni@student-cs.fr

Raghuwansh RAJ
ESSEC-CentraleSupélec
raghuwansh.raj@student-cs.fr

1. Abstract

The growing scientific evidence that climate change may also play a role in geological phenomena such as earthquakes, tsunamis and volcanic eruptions has left us in need of predicting them, in order to timely warn people of potentially damaging earthquakes, allowing people to respond to it appropriately, minimizing loss of life and property. Apart from predicting when an earthquake might occur, decision-makers and stakeholders also need an assessment of the potential damage following earthquake to formulate evacuation and risk reduction strategies.

Classical risk assessment methods are resource and time consuming, and so we have employed machine learning algorithms to detect the extent of damage to buildings post an earthquake based on data about the 2015 Gorkha Earthquake in Nepal.

2. Introduction

"[Without maps], the world of modern science and technologies would hardly exist." [1]

Besides their obvious advantages in terms of geolocation, maps are crucial to modern research as they provide context to demographic information. And when catastrophes strike, their importance cannot be understated. They enable governments and NGOs to organize thousands of emergency workers and help individuals to orientate themselves. However, maps can become outdated in the shortest of time spans. Take for example the 2015 Gorkha earthquake in Nepal: Critical infrastructure, such as bridges or tunnels, was destroyed, and consequently, rescue squads could only advance with great caution and limited speed into the impacted regions. In total, the earthquake caused 9,000 fatalities, 20,000 injuries, and affected 8 million people. Fourteen out of 75 Nepalese districts were declared crisis-hit and in these fourteen districts, more than 500,000 residential buildings, 5000 schools (83% of all schools) and 500 health facilities simply collapsed. The overall financial loss amounts to $7 billion USD [2].

Within this project, we want to predict the damage level of infrastructure during catastrophes such as earthquakes, and thereby, potentially contribute to a faster and more secure rescue response. The prediction will be based on features regarding location and structure of affected buildings.

The project is based on data about the previously described 2015 Gorkha earthquake in Nepal, provided by Kathmandu Living Labs and the Central Bureau of Statistics of the Government of Nepal within an on-going data competition on DRIVENDATA [3]. According to DRIVENDATA, the dataset is one of the largest post-disaster datasets ever collected and contains information about earthquake impacts, household conditions, and socio-economic-demographic statistics. The competition closes on 30 Sept. 2022.

3. Problem definition

According to the rules of the data competition [3], we will train our models to classify the damage level of infrastructure caused by the 2015 Gorkha earthquake in three classes, namely 1 (low damage), 2 (medium damage), and 3 (complete destruction). Consequently, the level of damage is an ordinal vaiable (order matters), and the prediction of the damage level can be framed either as a multi-class classification or as an ordinal regression problem.

We will train each of our models and evaluate them based on the evaluation metric chosen by the competition organizer, namely the *micro-averaged F1 score*. It is defined as following: [4]

$$F_{micro} = \frac{2 \times P_{micro} \times R_{micro}}{P_{micro} + R_{micro}}$$

where P denotes *precision* and R denotes *recall* and

$$P_{micro} = \frac{\sum_{k=1}^{3} TP_k}{\sum_{k=1}^{3} (TP_k + FP_k)}, \; R_{micro} = \frac{\sum_{k=1}^{3} TP_k}{\sum_{k=1}^{3} (TP_k + FN_k)}$$

where TP denotes *true positives*, FP *false positives*, FN *false negatives* and k each of the three classes. Consequently, the *micro-averaged F1 score*, does not consider TN (*true negatives*).

The best models will maximize the micro-averaged F1 score. Besides this score, we will also report each models precision and recall to better understand their classification capabilities.

4. Related Work

In a study by Austin Cooner et al. [9], remote sensing along with multilayer feedforward neural networks and radial basis neural networks was used to detect damage by earthquake in Haiti in 2010. For high spatial resolution imagery classification, textural and structural features including entropy, dissimilarity, Laplacian of Gaussian, and rectangular fit were seen as key variables. Using neural network helped them achieve error rate of less than 40%.

B. Patterson et al. in their research [10] used deep learning to automate post-earthquake reconnaissance image tagging activities by training a computer algorithm to classify each occurrence of damage per building material and structural member type. The authors implemented a DL algorithm to automatically identify multiple damage types and associated structural members in a single image by adapting a pre-trained deep residual network. Their work achieved 88% accuracy when detecting building damage.

Gian P. Delacruz et al. used Generative Adversarial Networks to classify structural damage caused by earthquakes [11]. He combined convolutional neural network (CNN) with a generative neural network. By taking a classifier trained in a GAN and modifying it to classify other images the classifier can take advantage of the GAN training without having to find more training data. The classifier trained in this way was able to achieve an 88% accuracy score when classifying images of structural damage caused by earthquakes.

Maidiawati et al. performed seismic analysis of damaged buildings [12] post earthquake in Palu city in 2018. The field investigation was focused on the damaged RC frame of buildings' structures. . Several types of damage were detected on RC structures such as collapse due to the soft story, damage to beam-column joint, failure of short column, shear failure of the column, and collapsed of brick masonry infills.

In a study by Peter Reinartz et al. [13] a novel disaster building damage monitoring method is presented. This method combines the multispectral imagery and DSMs from stereo matching to obtain three kinds of changes. The proposed method contains three basic steps. The first step is to segment the panchromatic images to get the smallest possible homogeneous regions. In the second step, based on a rule based classification using change information from Iteratively Reweighted Multivariate Alteration Detection (IR-MAD) and height, the changes are classified to ruined buildings, new buildings, and changes without height change (mainly temporary residential area, etc. tents). In the last step, a region based grey level co-occurrence matrix texture measurement is used to refine the third change class. The method is applied to building change detection after the Haiti earthquake.

A research [14] by Ömer Faruk Nemutlu studied the structural damage cases in reinforced concrete and masonry structures were investigated. Many structural deficiencies and mistakes such as non ductile details, poor concrete quality, short columns, strong beams weak columns mechanism, large and heavy overhangs, masonry building damages and inadequate reinforcement arrangements were observed. Requirements of seismic codes are discussed and compared with observed earthquake damage.

5. Methodology

In this chapter, we will present (1) the pipeline we build to create our models, and (2) the different models we created based on this pipeline.

5.1 Exploratory Data Analysis and Data Pre-Processing

The provided dataset has 260,601 entries, no duplicates and no missing entries. Regarding the features, we have in total

a) five numerical features

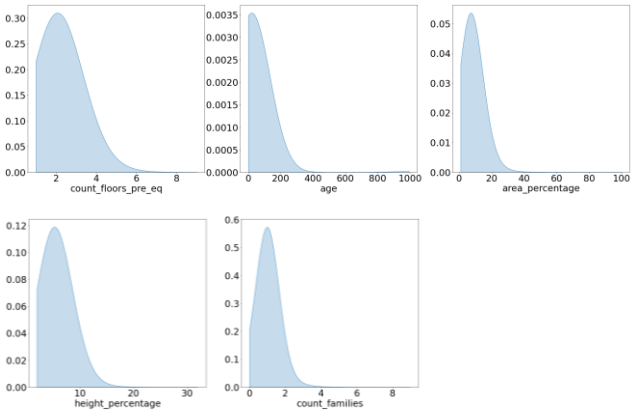
b) twenty-two binary features (one-hot encoded)

c) eight categorical features

d) three geographical features represented by integers

Numerical features

The five numerical features provide basic information about each building, such as its number of floors or its age. To make sure that all of our models will be able to learn from them, we will have a look at the distributional *skewness* of each feature.[1] Skewed distributions within the predictor data may severely impact the performance of models such as linear regression. In contrast, other models, such as tree-based ones, are known for their insensitivity to the characteristics of the predictor data. The sample skewness is the third standardized moment and can consequently be derived from the method of moments. [5]



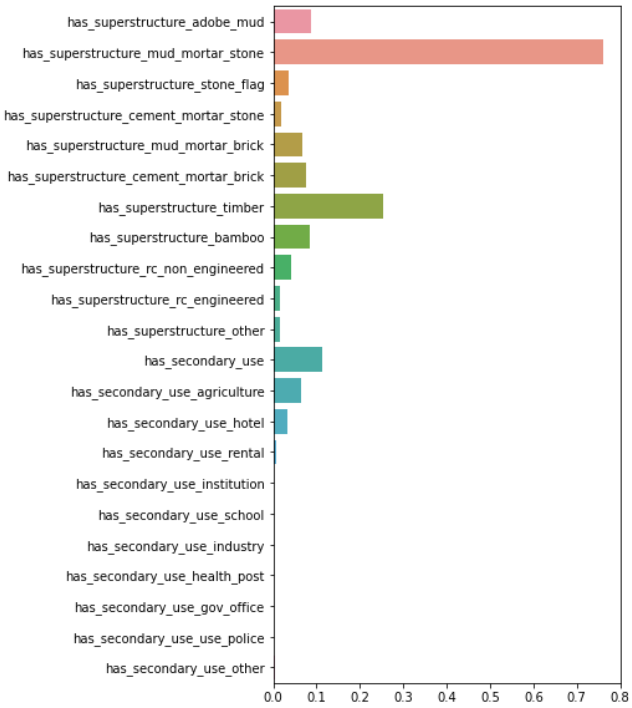
The distributions of all numerical features show a significant right skewness with a concentration of points with low values. To deal

[1] Unskewed distributions are symmetric distributions, meaning that the probability of falling on either side of the distribution's mean is roughly equal.

with this, we (1) applied a log transformation (as all features include 0, we added 1 to each feature before taking the logarithm), and (2) standardized all features by substracting the mean and dividing by the standard deviation. We also tried out another transformation method to remove skewness, proposed by Box and Cox [6]. They proposed a family of transformation indexed by a parameter λ, which can be approximated through maximum likelihood estimation. However, submitting a model on the evaluation platform gave the exactly same scores for both transformation methods, even though the numerical features were among the most important during feature selection.

Binary features

To understand the possible added value from the binary features, we will look at their *sparsity*. A feature can be called *sparse*, if it contains mostly zero values. [6] Consequently, to analyse sparsity of binary features, we can simply have a look at their mean.



We can easily see, that most of the binary features contain more than 90% zeros and thus, can be called sparse. Very likely, they will not play an important role in our models. Besides removing the unmeaningful features during feature selection to avoid overly complex models, there is not much that can be done to improve these binary features.

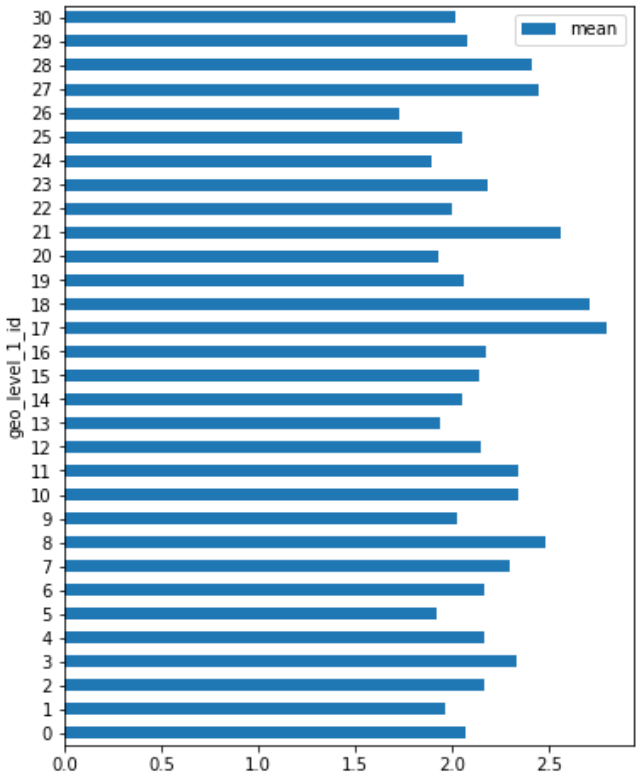
Categorical features

The eight categorical features mostly provide information about the construction of the building, such as the type of the foundation or the roof. In addition, all of them have a low cardinality between three to ten unique values, all represented by strings. As our models cannot work with strings directly, we need to encode these features. Due to their low cardinality, *one-hot encoding* seems to be a very promising solution. Even more so, as this is the same method that was most likely used to create the binary features. During one-hot encoding, we will create a new binary column for each unique value of every categorical feature and write a 1 if this entry contains this specific unique value or a 0 if not. This will increase the number of features by 30, making careful feature selection even more important.

Geographical features

The three geographical features (geo_level_1_id, geo_level_2_id, geo_level_3_id) are intuitively the most promising features and should help our models greatly in predicting the damage level as they denote the geographic region in which the building exists, from largest (level 1) to most specific sub-region (level 3). Obviously, a building that is very close to the epicenter of an earthquake is much more at risk than a building several kilometers away.

However, even though the geographical features are represented by integers, we chose to interpret them as categorical data. Reasoning: we do not know the geographical ordering of the different regions. For example, geo_level_1_id has a cardinality of 31 (values range from 0 to 30) but we do not know where these regions are actually located. This is underlined by the following analysis of geo_level_1_id.



In this graph, we plotted the target mean of each of the 31 level 1 regions, and we can see that that there is no correlation with the target (higher number in geo_level_1_id does not imply higher / lower mean damage level). This makes sense, as the world is not a straight line, but a 3D-space. For example, maybe the level 1 regions of 17, 18 and 21 form a circle around the epicenter of the earthquake, explaining that these regions have the highest target mean. Unfortunately, we were not able to identify the actual geographical regions implied by the three geographical identifiers. Also, any model will have trouble to learn from these three features as they also have very high cardinality (level 1 has 31 unique values, level 2 1428 and level 3 12568).

To enable our models to learn more from these geographical features, we will need to apply categorical encoding. In this case, the previously used method of *one-hot encoding* is not an option to the extremely high cardinality of the underlying features. With such high cardinalities, we would increase the number of features by a factor of 190 (from about 70 to more than 14,000 columns). Also, nearly all 14,000 newly created columns will mostly be extremely sparse. Thus, to work with this encoding method would be in this case (1) very computationally intensive, and (2) our models would have a very hard time to actually learn anything from this geographical data as all features are extremely sparse.

However, there is one very promising encoding method for this case that resolves both problems: *target encoding*. In target encoding, we will calculate the target mean and standard deviation of each unique value of all three geographical features. However, when considering this method for the level 3 feature as well, we again have the problem of high cardinality: with 12,568 unique values it is very likely that for some level 3 id's we will only have one or two buildings. This can result in both high bias and high variance for the mean and standard deviation of all unique values with low frequency in the training data. Also, this method would expose us to unseen geographical id's in the testing data. Actually, there are five unseen level 2 id's and 327 unseen level 3 id's in the test data – too many to be able to simply ignore this issue.

However, this is solvable by considering that the geographical id's actually have a hierarchy: one specific level 3 id always refers to the same level 2 and level 1 id. By leveraging the intuition of Micci-Barreca [7] about the application of target encoding to categorical data with hierarchical structure, we can 'blend' the mean of all geographical id's depending on their frequency.

For example, if we were to blend the mean of a specific level 2 id with that of its corresponding level 1 id, the calculation works as following [7]:

$$S_i = \lambda(n_i)\,\bar{x}_i + (1 - \lambda(n_i))\,\bar{x}_Y$$

where the mean of a subgroup is the weighted average of the subgroup's mean target variable and the mean of the prior, and where S_i^2 denotes the mean of the ith level 2 id and n_i the number of buildings with the ith level. The so-called blending factor

$$\lambda(n_i) = \frac{n_i}{m + n_i}$$

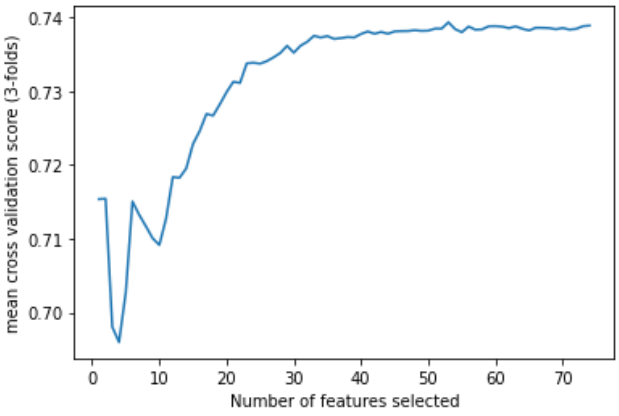
where $m = \frac{\sigma^2}{\tau^2}$ with σ^2 as the variance of the subgroup and τ^2 as the variance of the prior. This means, that if (1) we had a level 2 id with very few observations with completely different damage levels (high sigma) and (2) the corresponding level 1 id had low variance (low tau), the overall mean for the level 2 id would be highly impacted by the mean of the level 1 id and only slightly by the mean of the level 2 id.

There is one significant drawback to utilizing this encoding method though: overfitting. If these encodings are not consistent with the test data, we might run into severe problems. Even more so as the use of summary statistics can drastically underestimate the variation in the actual population. [8] We will have to pay significant attention to this during the model building and evaluation process.

5.2 Feature selection

After the described pre-processing methods, we are left with 74 features. As we already detected that, for example, lots of the binary features are very sparse, feature selection is very important. When considering all features, our model gets unneccessarily complex and might even loose some of its ability to generalize. Consequently, we looked at the F1-score of our baseline model depending on the number of features we use. To select the features in a meaningful way, we used *Recursive Feature Elimination with Cross-Validation (RFECV).* It is basically a backward selection of the preditors and begins by building a model on the entire feature space and computing an importance score for each feature. As we set the step size to one, the least important predictor is now removed, the model re-built, and importance scores re-calculated. To decrease the effect of pure chance as much as possible and consequently select the features that optimise the score (without too much computational power), the feature importances per iteration are evaluated within 3-fold cross validation.

The RFECV method finds its maximum mean 3-fold cross validation micro-averaged F1 score at 53 features with 73.94%. Somewhat surprisingly, if we only selected the most important feature (target encoded mean of geo_level_3_id), we already get a score of 71.54%. Below, we show a plot of the RFECV process.



5.3 Model building

a. Logistic Regression

Logistic regression is an extension of Linear Regression with the sigmoid function applied as an activation function to convert the outputs to a value between 0 and 1.

The first model we tried was multi class logistic regression, since it's a linear model scaling of the data has to be done. After the necessary scaling and transformations, the model gave an overall f1 micro score of **0.56** on the cross-validation sets.

Class	precision	recall	f1-score
1	0.51	0.06	0.11
2	0.57	0.95	0.71
3	0.40	0.04	0.08

The Performance of the model was quite low. The model has failed to classify class 1 and class 3which can be seen by the recall. The model was not able to capture the non -linearity of the data or it may be to too simple. Hence, we move on to more robust Linear multilayer perceptron aka Neural Network.

b. Neural Network

For this model we had to decide on the following things:

1. Number of hidden layers to be incorporated and the respective neurons in each hidden layer-1 hidden layer with 50 neurons yield a respectable score.
2. The loss function that can incorporate multi class classification loss- categorical_crossentropy.
3. The evaluation metrics for evaluating the model performance-categorical accuracy
4. Activation functions used in each layer and in the output layer-Relu in Hidden layer and Softmax in output layer.
5. Hyper Parameters like number of epochs, batch size-100 epochs with batch size of 50
6. Finally, the output layer has to be transformed using one hot encoding to fit as per the keras pipeline. Here we have three classes for target variable, so the number of neurons in output layer has to be 3.

This is the performance report of the neural network model.

Class	precision	recall	f1-score
1	0.58	0.60	0.59
2	0.71	0.88	0.78
3	0.85	0.50	0.63

Once the model was setup, we evaluated the model on the data and the score was significantly good. We hit a f1 score of **0.68** on the cross validation and the similar performance was also seen on the data driven leader board-**0.7239** which is a very good improvement in comparison to the previous model.

At this point we have tried different types of linear model with increasing order of complexity. Finally, we decided to move towards Ensemble techniques and tree-based model to capture the hidden non linearity of the data.

c. XGBOOST

The first ensemble model we tried was XGBOOST. At first, we modelled a basic XG-Boost model without much of the preprocessing of the data, we just converted the categorical columns into their integer forms and put into the model. The accuracy achieved was **0.66** which was a good start, then we tried to incorporate the class imbalance using the in-built class weight parameter in XG boost and the f1 score improved to **0.68**

After applying the necessary transformations, Scaling and encoding as explained in the above parts of report, The F1 score improved drastically to **0.76, which was our current highest performance on the cross validations.**

Then we moved towards the Hyperparameter tuning. After the analysis, these are the model parameters that found to be giving the highest score.

Learning rate(eta)	gamma	n_estimators	reg_lambda
0.1433	0.5688	153	0.1511

The best score that we achieved was **0.74**, which was slightly less than the unoptimized XG-Boost but the model overall achieved a better leader board score on the data driven platform.

Class	precision	recall	f1-score
1	0.69	0.59	0.64
2	0.77	0.84	0.80
3	0.77	0.67	0.71

d. Random Forest

Similar to XGBOOST model, first we evaluated the basic random forest with 100 trees without much of preprocessing of the data. The score we achieved was **0.71** which was again a good start, then we tried some variation of the model like random forest with class weight, randoms forest model with bootstrap class weights and the score was revolving around 0.70-0.71

Then we applied the necessary transformation and scaling to the data and again evaluated the model. The f1 score improved to 0.74 which was in line with our intentions.

After that we moved towards hyperparameter tuning of the random forest model. The parameters that gave the optimum performance were following.

n_estimators	max_features	max_depth	min_sample_split	min_sample_leaf	bootstrap
400	Auto	20	5	4	True

After the HPT, the performance went up to 0.7592. Here is the classification report for the hyper tuned random forest model.

Class	precision	recall	f1-score
1	0.71	0.55	0.62
2	0.76	0.86	0.80
3	0.77	0.66	0.71

Finally, we tried to implement transform forest with a variation that utilizes the ordinal scale of the target variable. Along with identifying and describing the magnitude, the ordinal scale shows the relative rank of variables. The primary advantage of using ordinal scale is the ease of comparison between variables.

So far, the Random Forest Model yielded the best results.

e. CatBoost

Another ensemble learning model tried was CatBoost. It is most suitable for data with categorical variables and heterogenous data. By heterogenous, we mean that the dataset has several features belonging to several datatypes. It performs well for such data as it implicitly takes care of categorical data conversion for modelling purposes.

For categorical data with low cardinality, CatBoost implicitly uses one-hot encoding. For other categorical features, "Ordered Target Statistic" technique is used, which is based on target encoding (previously explained during Feature Engineering).

Class	precision	recall	f1-score
1	0.70	0.58	0.63
2	0.76	0.85	0.80
3	0.78	0.66	0.72

f. Light Gradient Boosting

The final ensemble model we tried was Light GBM. It's a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification, and many other machine learning tasks.

We initially implemented the Light GBM model with gradient boosting without any feature engineering and without any categorical features. We dropped all the categorical variables and still got an F1 score on test data to **be 0.701** which seemed to be very promising. Hence, we started working on the model.

After performing the required preprocessing and encoding as explained in the feature engineering part of report, we trained the classifier again using the K fold cross validation technique with K=5. Its mean F1-score was found to be 0.7622.

Class	precision	recall	f1-score
1	0.69	0.60	0.64
2	0.77	0.85	0.80
3	0.77	0.66	0.71

g. K Nearest neighbor Classifier

K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint. The kNN Classifier model was built directly over the preprocessed and feature-engineered and the base-model was built using 10 and 20 nearest neighbors and gave a micro-F1 score of around **0.72** on 5-fold cross-validation.

The kNN model is a non-parametric learning approach so not much hyper-parameter tuning can be done. We tried to find the best value of k: number of nearest neighbours that choose the label for the new datapoint. From our experiments, the value of **k= 25,** maximizes the predictive performance and gives an F1 score of **0.72** for a 70-30 test-train split.

Here is the required performance report for the kNN classifier.

Class	precision	recall	f1-score
1	0.67	0.40	0.50
2	0.73	0.84	0.78
3	0.73	0.63	0.67

6. Evaluation

Through out our Project we tried different modelling techniques ranging for Classifiers like KNN which do grouping based on Euclidean distance to Linear models like logistic regression, neural networks to ensemble technique involving different bagging and boosting algorithms.

The Euclidean distance classifier was very straight forward and hence failed to catch the intrinsic relation between target variable and the features.

Then we moved to multi class logistic regression model which was better than KNN but it still could not catch the complex relationship between the dependent and independent features. Neural Network being the more robust model performed significantly better than logistic regression model though it still could not fully capture the non-linearity present in the data.

Finally, we tried Ensemble methods and the performance was promising right from the start even without hyper parameter tuning. The random forest model that utilizes ordinality of the target variable stood out to be the best classifier for the data. The preprocessing of the geographical ids from integer to target encoded helped the model significantly enough to get us a very good score. Here is the required table that summarizes model performance both on our training data (using cross validation) and on the leaderboard of the competition (based on the test data). All metrics are measured in the micro-averaged F1 score.

model	Logistic Regression	Neural Network	XGBoost
CV	0.564	0.7235	0.7630
leaderboard	0.661	0.7239	0.7442

model	Random Forest	CatBoost	Light GBM	kNN
CV	0.7598	0.7615	0.7622	0.7241
leaderboard	0.7489	0.7446	0.7443	0.7239

Our best performing model was the hypertuned random forest model, which placed us on rank 138 of 5,050 competitors (top 3%).

BEST	CURRENT RANK	# COMPETITORS
0.7489	138	5050

REFERENCES

[1] Thrower, N. J. W (2008). *Maps and civilization: Cartography in culture and society* (3rd ed.). University of Chicago Press.

[2] Gautam, D. & Rodrigues, H. (2017). *Impacts and insights of Gorkha earthquake in Nepal*. Amsterdam: Elsevier. DOI: https://doi.org/10.1016/B978-0-12-812808-4.00001-8

[3] Drivendata (2021). *Richter's Predictor: Modeling Earthquake Damage*. URL: https://www.drivendata.org/competitions/57/nepal-earthquake/

[4] Bex, T. (2021). Comprehensive Guide to Multiclass Classification Metrics. URL: https://towardsdatascience.com/comprehensive-guide-on-multiclass-classification-metrics-af94cfb83fbd

[5] Kuhn, M. & Johnson, K. (2016). *Applied Predictive Modeling*. New York: Springer Nature. DOI: https://doi.org/10.1007/978-1-4614-6849-3

[6] Ertel, W. (2017). Introduction to Artificial Intelligence. Cham: Springer Nature. DOI: https://doi.org/10.1007/978-3-319-58487-4

[7] Micci-Barreca (2001). A preprocessing scheme for high.cardinality categorical attributes in classification and prediction problems. ACM SIGKDD Explorations Newsletter, Vol.3, No.1. DOI: https://doi.org/10.1145/507533.507538

[8] Kuhn, M. & Johnson, K. (2019). Feature Engineering and Selection: A Practical Approach for Predictive Models. New York: Chapman and Hall. DOI: https://doi.org/10.1201/9781315108230

[9] Cooner, A., Shao, Y. & Campbell, J. (2016). *Detection of Urban Damage Using Remote Sensing and Machine Learning Algorithms: Revisiting the 2010 Haiti Earthquake*. Remote Sensing. 8. 868. 10.3390/rs8100868.

[10] Patterson, B. & Leone, G. & Pantoja, M. & Behrouzi, A. (2018). *DEEP LEARNING FOR AUTOMATED IMAGE CLASSIFICATION OF SEISMIC DAMAGE TO BUILT INFRASTRUCTURE*.

[11] Delacruz, G. (2020). *Using Generative Adversarial Networks to Classify Structural Damage Caused by Earthquakes*.

[12] Diawati, M. et al. (2020). *SEISMIC ANALYSIS OF DAMAGED BUILDINGS BASED ON POST-EARTHQUAKE INVESTIGATION OF THE 2018 PALU EARTHQUAKE*. International Journal of GEOMATE. 18. 116-122. 10.21660/2020.70.9490.

[13] Reinartz, P., Tian, J. & Nielsen, A. (2013). *Building damage assessment after the earthquake in Haiti using two post-event satellite stereo imagery and DSM*. International Journal of Image and Data Fusion. 6. 057-060. 10.1109/JURSE#.2013.6550665.

[14] Nemutlu et al. (2021). *Damage assessment of buildings after 24 January 2020 Elazığ-Sivrice earthquake*. Earthquakes and Structures. 20. 325-335. 10.12989/eas.2021.20.3.325.