Georgia State University

CSC6780 – Fundamentals of Data Science

Fall 2021

Final Project Report

EARTHQUAKE DAMAGE PREDICTION

Valorants

Gaayathri Vaidhyanathan Pazhambalacode

Sai Kowshik Ananthula

Mounika Ravi Hospet

Ardavan Sassani

Table of Contents

1 Business Understanding

1.1 Business Problem

The massive scale of destruction caused by the Gorkha Earthquake in Nepal in April 2015 led to the necessity to assess the scale of damage of post-earthquake structures. Having one of the enormous post-disaster datasets, the task is to develop a model to predict the grade of damage caused on a building post-earthquake. The degree of destruction should be classified as follows: 1 represents low damage, 2 represents moderately damaged, and 3 represents massive destruction.

1.2 Dataset

The dataset considered for this purpose is from a competition titled –"Richter's Predictor: Modeling Earthquake Damage". It contains information about the buildings' structure and their legal ownership. The dataset is divided into three parts: test values.csv, train values.csv, and train labels.csv. Each of them consists of 39 features describing the structural and legal aspects of the building and is uniquely identified by building id. Using this humongous data, the model is trained to predict the extent of damage caused and classifies using the ordinal variable damage grade.

1.3 Proposed Analytics Solution

The proposed approach is to first run the given training data through K-Nearest Neighbor Classifier. The classifier makes use of Euclidean distance to classify. In K-NN, the learning rate may be relatively less with a single layer. Based on the accuracy of K-NN classification, the data is then given to Random Forest method. In the random forest method, using decision trees, the depth increases. Since there is a greater quantity of decision-making in this methodology, it is assumed that the accuracy might increase at this stage.

Based on the accuracy here, we then plan on feeding it to logistic regression. In this, with the use of the sigmoid function, the scale is reduced. There is a greater possibility for an increase in the accuracy using this. If the accuracy is still low, ensembling techniques like Adaboost or xgboost can be used.

2 Data Exploration and Preprocessing

2.1 Data Quality Report

The chosen dataset has 39 features including the '*target variable*' and '*building_id'* which will be dropped during training. Most of the data is of *'Nominal'* scale and majority of the features in the dataset are binary. There are few features like *'foundation_type'*, *'roof_type'*, *'ground_floor_type'* which are of string datatype. As for cardinality of the data, it varies from 260601 to 2.

Table 1. Data Quality Report for Continuous Features

Feature	Count	Missing (%)	Cardinality (#)	Min	Q1	Median	Q3	Max	Mean	Std.Dev	Outlier_lower	Outlier_upper	Number of outliers	Outliers (%)	Note
age	260601	0.0	42	0	10.0	15.0	30.0	995	26.54	73.57	-20.0	60.0	12499	4.80	outlier_h,
area_percentage	260601	0.0	84	1	5.0	7.0	9.0	100	8.02	4.39	-1.0	15.0	13557	5.20	outlier_h,
height_percentage	260601	0.0	27	2	4.0	5.0	6.0	32	5.43	1.92	1.0	9.0	7843	3.01	outlier_h,

Of the categorical features present in the dataset, possible values for *land_surface_condition* are **(t, n, o)**. similarly, for *foundation_type* the values could be **(r, w, u, i, h)**. **(n, q, x)** for *roof_type*, **(f, x, v, z, m)** for *ground_floor_type*. In the below graphs, the x-axis represents the count and y-axis represents the possible values.

Table 2. Data Quality Report for Continuous Features

Feature	Count	Missing (%)	Cardinality (#)	Mode	Mode Freq	Mode (%)	2nd Mode	2nd Mode Freq	2nd Mode (%)	Note
count_floors_pre_eq	260601	0.0	9	2	156623	60.10	3	55617	21.34	
land_surface_condition	260601	0.0	3	t	216757	83.18	n	35528	13.63	
foundation_type	260601	0.0	5	r	219196	84.11	w	15118	5.80	
roof_type	260601	0.0	3	n	182842	70.16	q	61576	23.63	
ground_floor_type	260601	0.0	5	f	209619	80.44	x	24877	9.55	
other_floor_type	260601	0.0	4	q	165282	63.42	x	43448	16.67	
position	260601	0.0	4	s	202090	77.55	t	42896	16.46	
plan_configuration	260601	0.0	10	d	250072	95.96	q	5692	2.18	
has_superstructure_adobe_mud	260601	0.0	2	0	237500	91.14	1	23101	8.86	
has_superstructure_mud_mortar_stone	260601	0.0	2	1	198561	76.19	0	62040	23.81	
has_superstructure_stone_flag	260601	0.0	2	0	251654	96.57	1	8947	3.43	
has_superstructure_cement_mortar_stone	260601	0.0	2	0	255849	98.18	1	4752	1.82	
has_superstructure_mud_mortar_brick	260601	0.0	2	0	242840	93.18	1	17761	6.82	
has_superstructure_cement_mortar_brick	260601	0.0	2	0	240986	92.47	1	19615	7.53	
has_superstructure_timber	260601	0.0	2	0	194151	74.50	1	66450	25.50	
has_superstructure_bamboo	260601	0.0	2	0	238447	91.50	1	22154	8.50	
has_superstructure_rc_non_engineered	260601	0.0	2	0	249502	95.74	1	11099	4.26	
has_superstructure_rc_engineered	260601	0.0	2	0	256468	98.41	1	4133	1.59	
has_superstructure_other	260601	0.0	2	0	256696	98.50	1	3905	1.50	
legal_ownership_status	260601	0.0	4	v	250939	96.29	a	5512	2.12	
count_families	260601	0.0	10	1	226115	86.77	0	20862	8.01	
has_secondary_use	260601	0.0	2	0	231445	88.81	1	29156	11.19	
has_secondary_use_agriculture	260601	0.0	2	0	243824	93.56	1	16777	6.44	
has_secondary_use_hotel	260601	0.0	2	0	251838	96.64	1	8763	3.36	
has_secondary_use_rental	260601	0.0	2	0	258490	99.19	1	2111	0.81	

has_secondary_use_institution	260601	0.0	2	0	260356	99.91	1	245	0.09
has_secondary_use_school	260601	0.0	2	0	260507	99.96	1	94	0.04
has_secondary_use_industry	260601	0.0	2	0	260322	99.89	1	279	0.11
has_secondary_use_health_post	260601	0.0	2	0	260552	99.98	1	49	0.02
has_secondary_use_gov_office	260601	0.0	2	0	260563	99.99	1	38	0.01
has_secondary_use_use_police	260601	0.0	2	0	260578	99.99	1	23	0.01
has_secondary_use_other	260601	0.0	2	0	259267	99.49	1	1334	0.51
damage_grade	260601	0.0	3	2	148259	56.89	3	87218	33.47

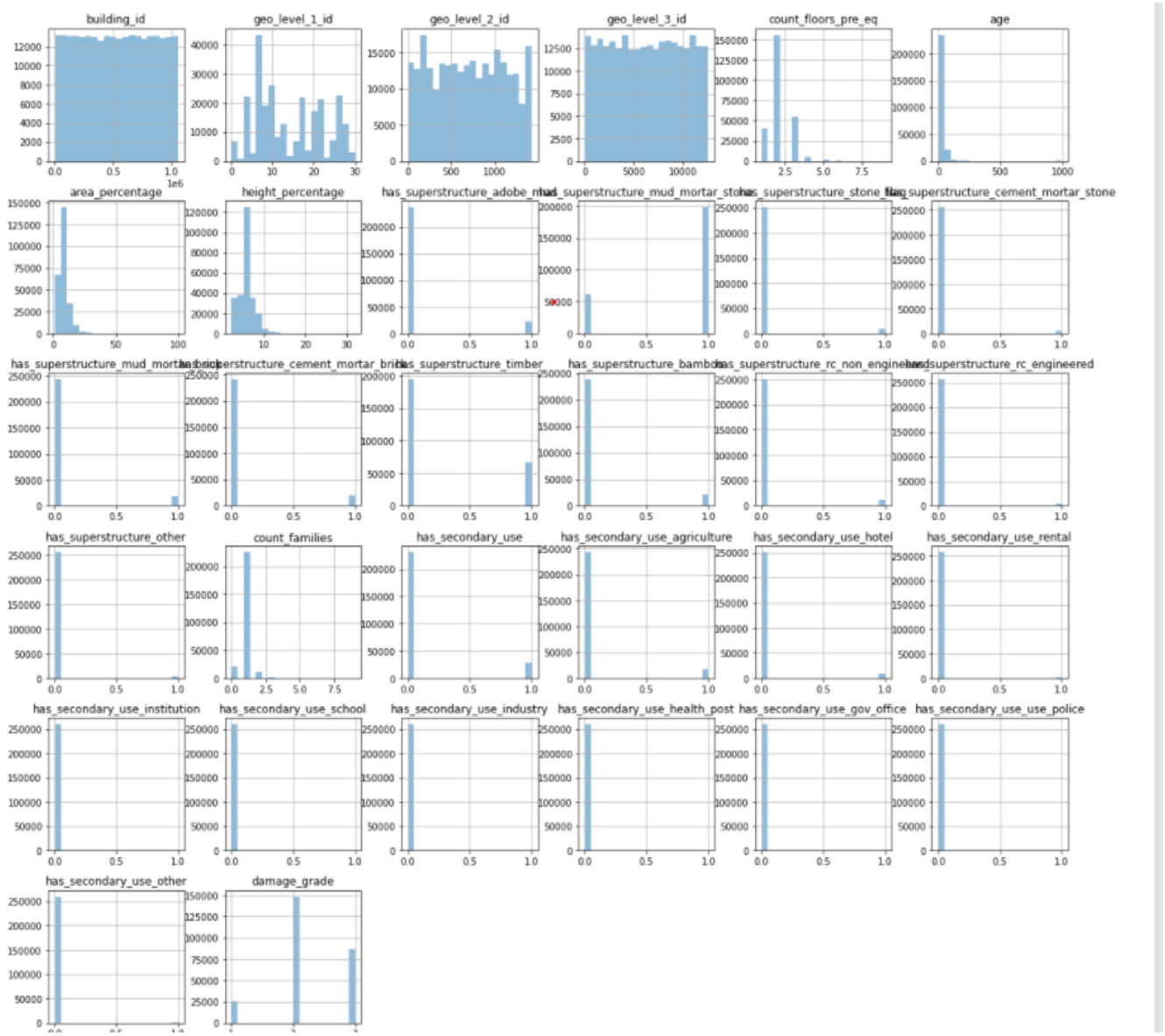


Figure 1: Distribution of numerical data

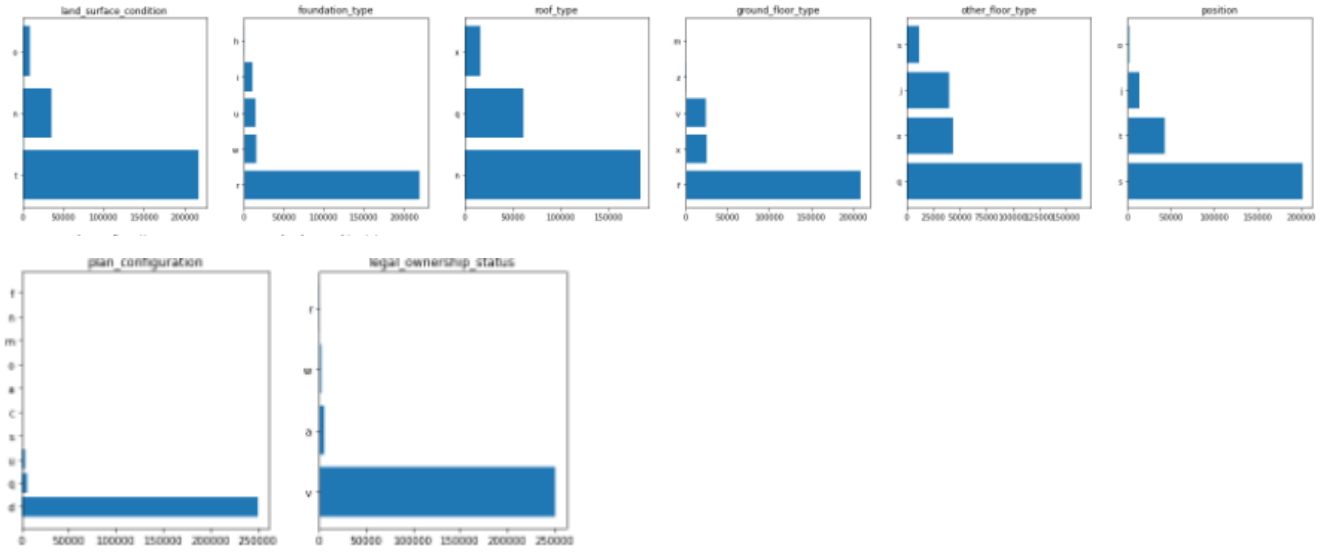
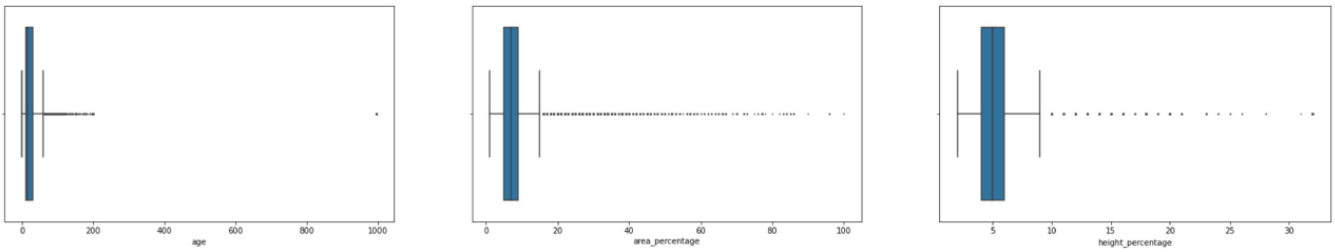


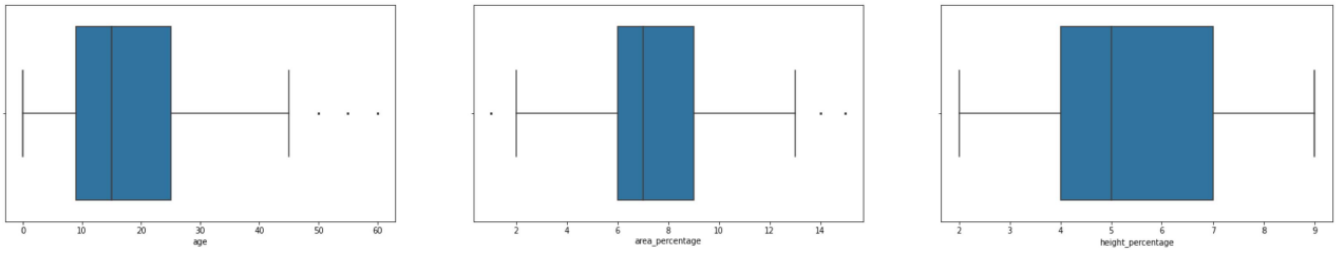
Figure 2: Distribution of Categorical data

2.2 Missing Values and Outliers

In the chosen dataset, there are no missing values for any of the features present.
Figure 2(a) represents data with outliers that was detected based on the **Inter Quartile Range(IQR)** and Figure 2(b) represents the data after the outliers were handled, by dropping the data that is outside the IQR.



a



b

Figure 2: Boxplot of Continues features; a) before outlier handling; b) after outlier handling

2.3 Normalization

For Normalization, we have used *Standard Scalar* form sklearn library. Standard scalar will make sure

that mean of the whole data is equal to 0 and its standard deviation equal to 1 i.e., μ=0 and σ=1. We have 39 columns in data out of which 37 will be considered for training the data. So, we will take each column of X and will normalize using the standard scalar module.

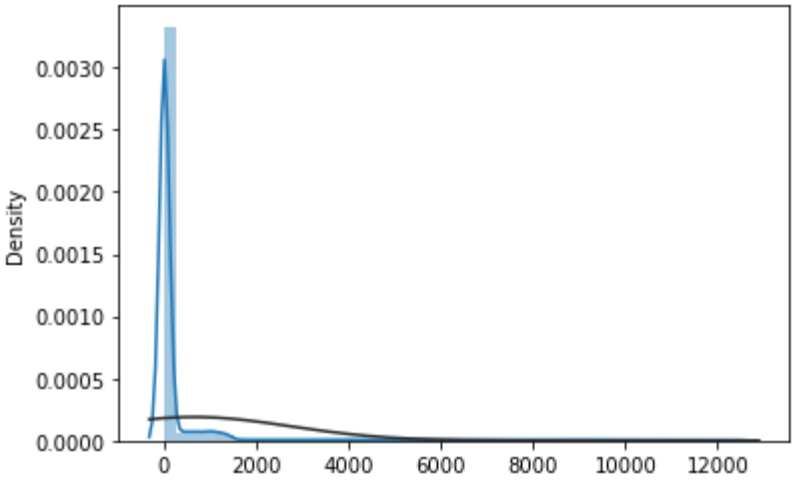


Figure 3(a): Before Normalization

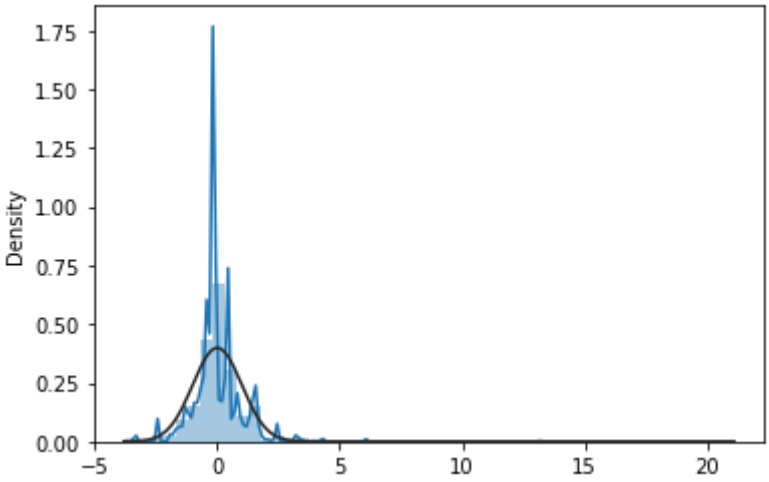


Figure 3(b): After Normalization

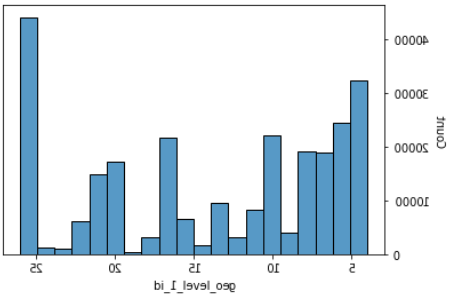


Figure 3(c) sample feature column before Scaling.

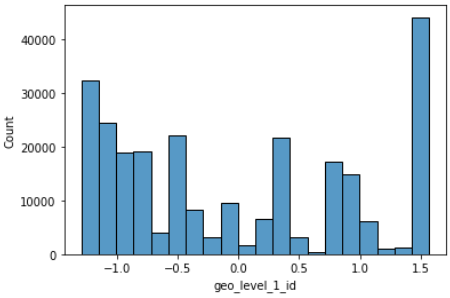


Figure 3(d) sample feature column after Scaling.

Most of our feature variables are binary so we have to normalize all the other variables that are non-binary.

2.4 Transformations

There are some features that have literal values of object type. As we know these literals values cannot be understood by machine learning model so we have to convert them into numerical. To do this task we have used *OrdinalEncoder* which is a module in sklearn to convert 8 literal features into numerical values.

Feature	Count	Missing (%)	Cardinality (#)	Mode	Mode Freq	Mode (%)	2nd Mode	2nd Mode Freq	2nd Mode (%)	Note
land_surface_condition	260601	0.0	3	t	216757	83.18	n	35528	13.63	
foundation_type	260601	0.0	5	r	219196	84.11	w	15118	5.80	
roof_type	260601	0.0	3	n	182842	70.16	q	61576	23.63	
ground_floor_type	260601	0.0	5	f	209619	80.44	x	24877	9.55	
other_floor_type	260601	0.0	4	q	165282	63.42	x	43448	16.67	
position	260601	0.0	4	s	202090	77.55	t	42896	16.46	
plan_configuration	260601	0.0	10	d	250072	95.96	q	5692	2.18	
legal_ownership_status	260601	0.0	4	v	250939	96.29	a	5512	2.12	

Figure 4(a) Categorical Data before using Ordinal encoder

Feature	Count	Missing (%)	Cardinality (#)	Mode	Mode Freq	Mode (%)	2nd Mode	2nd Mode Freq	2nd Mode (%)	Note
land_surface_condition	260601	0.0	3	2.0	216757	83.18	0.0	35528	13.63	
foundation_type	260601	0.0	5	2.0	219196	84.11	4.0	15118	5.80	
roof_type	260601	0.0	3	0.0	182842	70.16	1.0	61576	23.63	
ground_floor_type	260601	0.0	5	0.0	209619	80.44	3.0	24877	9.55	
other_floor_type	260601	0.0	4	1.0	165282	63.42	3.0	43448	16.67	
position	260601	0.0	4	2.0	202090	77.55	3.0	42896	16.46	
plan_configuration	260601	0.0	10	2.0	250072	95.96	7.0	5692	2.18	
legal_ownership_status	260601	0.0	4	2.0	250939	96.29	0.0	5512	2.12	

Figure 4(b) Categorical Data after using Ordinal encoder

3 Model Selection

After understanding the business requirement and pre-processing the data clean for training the model to predict as desired, the training data was split into 70% data for training and 30% data size for testing. One model we used to train the data was Gaussian Naïve Bayes classifier. This was not quite an efficient procedure as it had an accuracy of only 47%. The data was later trained using Logistic Regression model. It is an error-based learning mechanism This model was chosen as it reduces the scale of the data with the help of sigmoid function having a better probability to get more accurate results. This gave an

accuracy of 57.6% on the training data and 57.2% on the testing data.

To enhance the accuracy, a different approach i.e., similarity-based learning, namely k-nearest neighbor classifier was used. It is a lazy learning technique that calculates distance between the testing sample and k nearest neighbors of the testing sample and classifies the point accordingly. The reason for choosing this model is because, it has quite an ease of use for multi-class classification with no prior underlying assumptions made on the data. This approach gave out better accuracy with about 82.3% on training data whereas 68.8% on testing data for 3 neighbors. This was then given to Decision Tree Classifier which had approximately given out a similar accuracy of 69%.

In the attempt to improve the performance of the model, ensembling techniques were made use of. The Decision Tree Classifier model was fed to Adaboost classifier with the hope to increase the accuracy. Despite this attempt, no peak in accuracy was observed for 50 estimators but, with increase in number of estimators to about 1000-2000 it went up to 70%. When a different ensembling technique i.e., XGBoost Classifier was used, the accuracy went up to around 73-74%. The given model stands out to be in top 5%, in terms of F1-score, in the competition organized by Drivendata titled – "Richter's Predictor: Modeling Earthquake Damage".

4 Model Evaluation

To evaluate the models, classification report has been generated for every model trained for the given problem statement. Below are the figures of classification reports of training using Gaussian Naïve Bayes, Logistic Regression, KNN, Decision Tree with emsembling techniques Adaboost and XGBoost.

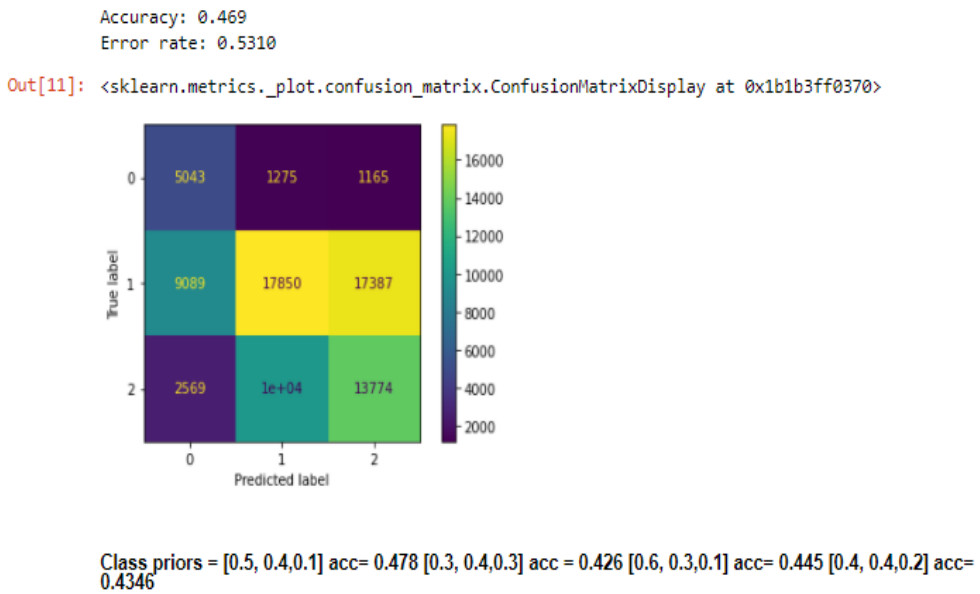


Figure 5(a): Confusion matrix for gaussian naïve bayes

Classification report :

	precision	recall	f1-score	support
1	0.30	0.67	0.42	7483
2	0.61	0.40	0.49	44326
3	0.43	0.52	0.47	26372
accuracy			0.47	78181
macro avg	0.45	0.53	0.46	78181
weighted avg	0.52	0.47	0.47	78181

Figure 5(b): Classification report for gaussian naïve bayes

The Accuracy for Training Set is 58.07422431750905
The Accuracy for Test Set is 57.80048860976452
Classification report :

	precision	recall	f1-score	support
1	0.54	0.25	0.34	7483
2	0.58	0.91	0.71	44326
3	0.53	0.11	0.18	26372
accuracy			0.58	78181
macro avg	0.55	0.42	0.41	78181
weighted avg	0.56	0.58	0.50	78181

)ut[4]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1b1b9462610>

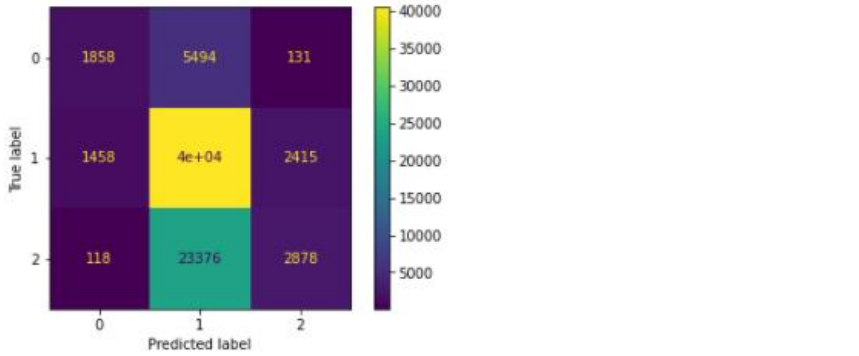


Figure 5(c): Classification report for logistic regression

```
The Accuracy for Training Set is 79.181559039579
The Accuracy for Testing Set is 61.46378276051726
Classification report :
              precision    recall  f1-score   support

           1       0.42      0.46      0.44      7483
           2       0.67      0.70      0.68     44326
           3       0.57      0.52      0.54     26372

    accuracy                           0.61     78181
   macro avg       0.55      0.56      0.56     78181
weighted avg       0.61      0.61      0.61     78181
```

```
ut[5]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1b1b9584a60>
```

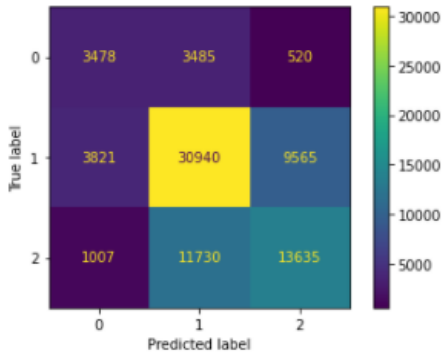


Figure 5(d): Classification report for k-nearest neighbor

```
Testing Accuracy: 0.6938386564510559
Error Rate is: 0.3061613435489441
Classification report :
              precision    recall  f1-score   support

           1       0.56      0.45      0.50      7483
           2       0.71      0.80      0.75     44326
           3       0.69      0.59      0.64     26372

    accuracy                           0.69     78181
   macro avg       0.65      0.61      0.63     78181
weighted avg       0.69      0.69      0.69     78181
```

```
Out[6]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1b1b9584af0>
```

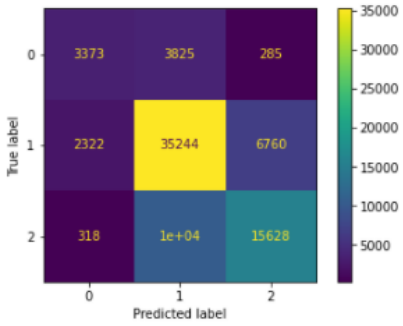


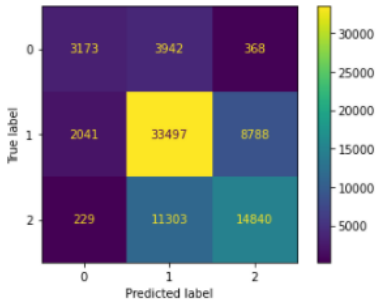
Figure 5(e): Classification report for decision tree classifier

```
Running with estimators:50 getting an accuracy of: 65.8855732211151
Classification report :
              precision    recall  f1-score   support

           1       0.58      0.42      0.49      7483
           2       0.69      0.76      0.72     44326
           3       0.62      0.56      0.59     26372

    accuracy                           0.66     78181
   macro avg       0.63      0.58      0.60     78181
weighted avg       0.65      0.66      0.65     78181
```

Out[8]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1b1bbd3e6d0>



with learning rate as default and 50 estimators accuracy is 65.9

with learning rate 0.02 and 700 estimators accuracy is 66.8 with learning rate 0.002 and 1000 estimators accuracy is 70.3 with learning rate 0.002 and 2000 estimators accuracy is 70

Figure 5(f): Classification report of Adaboost Classifier

```
Testing Accuracy: 0.7391309909057188
Error Rate is: 0.26086900909428123
Classification report :
              precision    recall  f1-score   support

           1       0.67      0.51      0.58      7483
           2       0.74      0.84      0.79     44326
           3       0.75      0.63      0.69     26372

    accuracy                           0.74     78181
   macro avg       0.72      0.66      0.68     78181
weighted avg       0.74      0.74      0.73     78181
```

Out[10]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1b1bbe64f70>

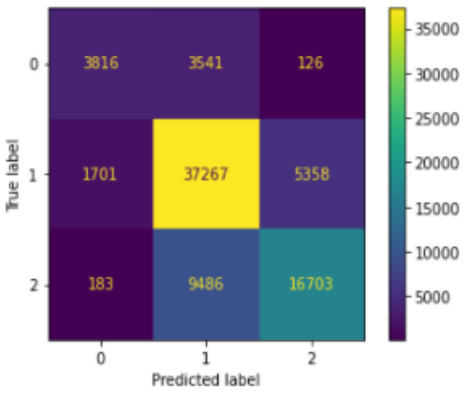


Figure 5(g): Classification report of XGBoost Classifier

As it can be observed from the above figures, the Decision Tree Classification when fed to the XGBoost classifier as an ensemble gives us better performance in every aspect thereby making it a more suitable model.

5 Result

As observed in the above classification report of all models, when the data is classified using Decision Tree Classifier and then ensembled using the XGBoost classifier, it resulted in highest accuracy of the testing data. Along with accuracy, the other evaluation parameters such as precision, recall, f1-score and support also remain high for this model for every target class (1, 2, 3) in the data provided.

6 Conclusion

For the given problem statement, we had to determine the damage grade (damage_grade 1, damage_grade 2, damage_grade 3) caused by the earthquake by taking the data from the destruction caused during the Gorkha earthquake in Nepal to analyze. After using various training techniques like similarity-based learning (k-nn), error-based learning (logistic regression), information-based learning (decision tree), and probability-based learning (Gaussian Naïve Bayes), it was observed that the highest performance was when trained using Decision tree classifier which was further ensembled using XGBoost technique. It predicted pretty accurately all the three classes of damage grade caused by the earthquake for the testing data provided. Among the several participants in the Drivedata competition for the same, this F1-score is one among the top 5% making it quite efficient to predict the scale of damage caused using several parameters such as the type of material used in the building, its positioning, floor plan, area, height, etc.