MODELING EARTHQUAKE DAMAGE

Drexel University | Spring 2020 | STAT 642

Group 6: Aishwarya Reddy Toom | Debomita Bhattacharjee | Hemachandar Nagarajan | Pawan Punera

I. EXECUTIVE SUMMARY

The primary objective of this project is to predict the level of damage to buildings caused by the 2015 Gorkha earthquake in Nepal using various supervised analyses techniques.

To make sure our results be optimal and accurate, performing proper data preprocessing and a thorough research about the Gorkha earthquake before our analysis was critical.

II. PROBLEM STATEMENT

o MOTIVATION

An earthquake is what happens when two blocks of the earth suddenly slip past one another. The surface where they slip is called the fault or fault plane. The location below the earth's surface where the earthquake starts is called the hypocenter, and the location directly above it on the surface of the earth is called the epicenter.

Gorkha earthquake occurred at 11:56 Nepal Standard Time on 25 April 2015, with a magnitude of 8.1Ms. Its epicenter was east of Gorkha District at Barpak, Gorkha, and its hypocenter was at a depth of approximately 8.2. It was the worst natural disaster to strike Nepal since the 1934 Nepal–Bihar earthquake. The earthquake triggered an avalanche on Mount Everest, killing 21, making 25 April 2015 the deadliest day on the mountain in history. The earthquake triggered another huge avalanche in the Langtang valley, where 250 people were reported missing. Hundreds of thousands of Nepalese were made homeless with entire villages flattened, across many districts of the country. Centuries-old buildings were destroyed at UNESCO World Heritage Sites in the Kathmandu Valley, including some at the Kathmandu Durbar Square, the Patan Durbar Square, the Bhaktapur Durbar Square, the Changu Narayan Temple, the Boudhanath stupa and the Swayambhunath Stupa.

Each year, thousands of earthquakes occur throughout the world. Although not damaging, few earthquakes provide a wealth of information that enables seismologists and engineers to better assess the distribution, frequency, and severity of seismic hazards throughout the country. Seismograph networks supply earthquake parameter and waveform data that are essential for the real-time evaluation of tectonic activity for public safety, the development of earthquake hazard maps and seismic design criteria used in building codes and land-use planning decisions. Based on information collected from previous earthquakes and the damages it caused we can damages levels a building can face when a similar earthquake strikes again.

Few advantages of these kind of predictions are implementing precautionary measures and better disaster preparedness like:

- Helps in construction of any structure in any area.
- Renovation of old/damage prone buildings.
- Classification of geographic region in which the building exists.

III. EXPLORING THE DATA

o DATA SOURCE

The dataset mainly consists of information on the buildings' structure and their legal ownership. Each row in the dataset represents a specific building in the region that was hit by Gorkha earthquake. There are 200k observations and 39 columns in this dataset, where the building_id column is a unique and random identifier. The remaining 38 features are described in the Data preprocessing section.

Based on aspects of building location and construction, our goal is to predict the level of damage to buildings caused by the 2015 Gorkha earthquake. The data was collected through household surveys using mobile technology by Kathmandu Living Labs and the Central Bureau of Statistics, which works under the National Planning Commission Secretariat of Nepal in the earthquake-affected districts.

This survey is one of the largest post-disaster datasets ever collected, containing valuable information on earthquake impacts, household conditions, and socio-economic-demographic statistics.

Below is the description of the variables:

- **geo_level_1_id, geo_level_2_id, geo_level_3_id** (type: int): geographic region in which building exists, from largest (level 1) to most specific sub-region (level 3). Possible values: level 1: 0-30, level 2: 0-1427, level 3: 0-12567.
- **count_floors_pre_eq** (type: int): number of floors in the building before the earthquake.
- **age** (type: int): age of the building in years.
- **area_percentage** (type: int): normalized area of the building footprint.
- **height_percentage** (type: int): normalized height of the building footprint.
- **land_surface_condition** (type: categorical): surface condition of the land where the building was built. Possible values: n, o, t.
- **foundation_type** (type: categorical): type of foundation used while building. Possible values: h, i, r, u, w.
- **roof_type** (type: categorical): type of roof used while building. Possible values: n, q, x.

- **ground_floor_type** (type: categorical): type of the ground floor. Possible values: f, m, v, x, z.
- **other_floor_type** (type: categorical): type of constructions used in higher than the ground floors (except of roof). Possible values: j, q, s, x.
- **position** (type: categorical): position of the building. Possible values: j, o, s, t.
- **plan_configuration** (type: categorical): building plan configuration. Possible values: a, c, d, f, m, n, o, q, s, u.
- **has_superstructure_adobe_mud** (type: binary): flag variable that indicates if the superstructure was made of Adobe/Mud.
- **has_superstructure_mud_mortar_stone** (type: binary): flag variable that indicates if the superstructure was made of Mud Mortar - Stone.
- **has_superstructure_stone_flag** (type: binary): flag variable that indicates if the superstructure was made of Stone.
- **has_superstructure_cement_mortar_stone** (type: binary): flag variable that indicates if the superstructure was made of Cement Mortar - Stone.
- **has_superstructure_mud_mortar_brick** (type: binary): flag variable that indicates if the superstructure was made of Mud Mortar - Brick.
- **has_superstructure_cement_mortar_brick** (type: binary): flag variable that indicates if the superstructure was made of Cement Mortar - Brick.
- **has_superstructure_timber** (type: binary): flag variable that indicates if the superstructure was made of Timber.
- **has_superstructure_bamboo** (type: binary): flag variable that indicates if the superstructure was made of Bamboo.
- **has_superstructure_rc_non_engineered** (type: binary): flag variable that indicates if the superstructure was made of non-engineered reinforced concrete.
- **has_superstructure_rc_engineered** (type: binary): flag variable that indicates if the superstructure was made of engineered reinforced concrete.
- **has_superstructure_other** (type: binary): flag variable that indicates if the superstructure was made of any other material.
- **legal_ownership_status** (type: categorical): legal ownership status of the land where building was built. Possible values: a, r, v, w.
- **count_families** (type: int): number of families that live in the building.
- **has_secondary_use** (type: binary): flag variable that indicates if the building was used for any secondary purpose.
- **has_secondary_use_agriculture** (type: binary): flag variable that indicates if the building was used for agricultural purposes.
- **has_secondary_use_hotel** (type: binary): flag variable that indicates if the building was used as a hotel.
- **has_secondary_use_rental** (type: binary): flag variable that indicates if the building was used for rental purposes.
- **has_secondary_use_institution** (type: binary): flag variable that indicates if the building was used as a location of any institution.
- **has_secondary_use_school** (type: binary): flag variable that indicates if the building was used as a school.

- **has_secondary_use_industry** (type: binary): flag variable that indicates if the building was used for industrial purposes.
- **has_secondary_use_health_post** (type: binary): flag variable that indicates if the building was used as a health post.
- **has_secondary_use_gov_office** (type: binary): flag variable that indicates if the building was used as a government office.
- **has_secondary_use_use_police** (type: binary): flag variable that indicates if the building was used as a police station.
- **has_secondary_use_other** (type: binary): flag variable that indicates if the building was secondarily used for other purposes.

  We are going to predict damage_grade class, which represents a level of damage to the building that was hit by the earthquake. There are 3 grades/classes of the damage:

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

○ DATA SAMPLING

Sampling can be particularly useful with data sets that are too large to efficiently analyze in full -- for example, in big data analytics applications or surveys. Identifying and analyzing a representative sample is more efficient and cost-effective than surveying the entirety of the data or population. Random samples are the best method of selecting your sample from the population of interest. The advantages are that your sample should represent the target population and eliminate sampling bias.

The initial dataset consists of 200k observations. We performed random sampling on this dataset in order to randomly pick 10% of the initial dataset with the same proportion of target variable that exists in the initial dataset. We sampled the data without replacement because we have enough data for the sampling to be performed.

- **DESCRIPTIVE STATISTICS & EXPLORATORY DATA ANALYSIS**

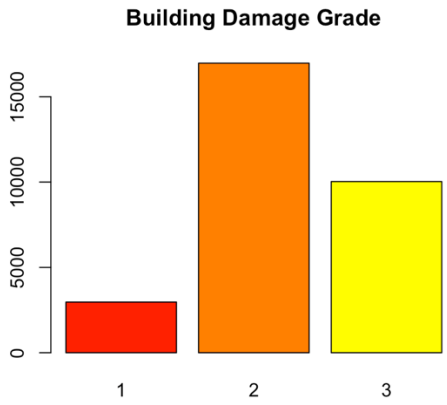
```
'data.frame':   29971 obs. of  41 variables:
 $ X                                 : int  4 7 16 20 32 41 43 51 59 103 ...
 $ building_id                       : int  16 28 47 63 95 128 132 151 177 364 ...
 $ geo_level_1_id                    : int  4 8 10 17 17 7 26 22 17 7 ...
 $ geo_level_2_id                    : int  651 1297 310 490 663 382 39 913 566 1091 ...
 $ geo_level_3_id                    : int  105 9721 5331 7900 8570 2045 9133 2241 12387 10826 ...
 $ count_floors_pre_eq               : int  2 2 2 1 3 2 1 3 3 2 ...
 $ age                               : int  80 0 15 5 5 25 5 20 20 20 ...
 $ area_percentage                   : int  5 2 6 8 8 11 11 12 7 7 ...
 $ height_percentage                 : int  4 6 5 2 7 5 3 7 7 5 ...
 $ land_surface_condition            : Factor w/ 3 levels "n","o","t": 1 3 1 2 3 3 3 3 1 3 ...
 $ foundation_type                   : Factor w/ 5 levels "h","i","r","u",..: 3 3 3 3 3 3 3 3 3 4 ...
 $ roof_type                         : Factor w/ 3 levels "n","q","x": 1 1 1 1 1 1 3 1 1 2 ...
 $ ground_floor_type                 : Factor w/ 5 levels "f","m","v","x",..: 1 1 1 1 1 1 3 1 1 1 ...
 $ other_floor_type                  : Factor w/ 4 levels "j","q","s","x": 2 4 2 1 2 4 1 4 4 2 ...
 $ position                          : Factor w/ 4 levels "j","o","s","t": 3 3 3 3 3 4 4 3 3 3 ...
 $ plan_configuration                : Factor w/ 10 levels "a","c","d","f",..: 3 3 3 3 3 3 3 3 3 3 ...
```

After sampling the data, we are left with 29k observations of 41 variables. Here we can see a few variables which are unnecessary are can be removed in the preprocessing step. We can also observe that there are different terminologies for levels of a few categorical variables. (For example: land_surface_condition has levels n,o,t).

```
       X          building_id      geo_level_1_id  geo_level_2_id geo_level_3_id
 Min.   :     4   Min.   :     16   Min.   : 0.00   Min.   :   0   Min.   :    3
 1st Qu.: 65994   1st Qu.: 264674   1st Qu.: 7.00   1st Qu.: 352   1st Qu.: 3089
 Median :130087   Median : 524840   Median :12.00   Median : 712   Median : 6305
 Mean   :130465   Mean   : 526336   Mean   :13.94   Mean   : 706   Mean   : 6268
 3rd Qu.:195241   3rd Qu.: 788950   3rd Qu.:21.00   3rd Qu.:1055   3rd Qu.: 9414
 Max.   :260593   Max.   :1052908   Max.   :30.00   Max.   :1426   Max.   :12565

 count_floors_pre_eq      age         area_percentage   height_percentage land_surface_condition
 Min.   :1.000       Min.   :  0.00   Min.   :  1.000   Min.   : 2.000    n: 4110
 1st Qu.:2.000       1st Qu.: 10.00   1st Qu.:  5.000   1st Qu.: 4.000    o:  942
 Median :2.000       Median : 15.00   Median :  7.000   Median : 5.000    t:24919
 Mean   :2.129       Mean   : 25.95   Mean   :  8.045   Mean   : 5.426
 3rd Qu.:2.000       3rd Qu.: 30.00   3rd Qu.:  9.000   3rd Qu.: 6.000
 Max.   :8.000       Max.   :995.00   Max.   :100.000   Max.   :32.000

 foundation_type roof_type ground_floor_type other_floor_type position  plan_configuration
 h:  152         n:21026   f:24070           j: 4592          j: 1515   d      :28728
 i: 1181         q: 7104   m:   61           q:19095          o:  280   q      :  694
 r:25196         x: 1841   v: 2822           s: 1341          s:23257   u      :  420
 u: 1701                   x: 2895           x: 4943          t: 4919   s      :   40
 w: 1741                   z:  123                                      c      :   33
                                                                        a      :   25
                                                                        (Other):   31
 has_superstructure_adobe_mud has_superstructure_mud_mortar_stone has_superstructure_stone_flag
 Min.   :0.00000              Min.   :0.000                       Min.   :0.0000
 1st Qu.:0.00000              1st Qu.:1.000                       1st Qu.:0.0000
 Median :0.00000              Median :1.000                       Median :0.0000
 Mean   :0.08618              Mean   :0.764                       Mean   :0.0345
 3rd Qu.:0.00000              3rd Qu.:1.000                       3rd Qu.:0.0000
 Max.   :1.00000              Max.   :1.000                       Max.   :1.0000

 has_superstructure_cement_mortar_stone has_superstructure_mud_mortar_brick
 Min.   :0.00000                        Min.   :0.0000
 1st Qu.:0.00000                        1st Qu.:0.0000
 Median :0.00000                        Median :0.0000
 Mean   :0.01909                        Mean   :0.0663
 3rd Qu.:0.00000                        3rd Qu.:0.0000
 Max.   :1.00000                        Max.   :1.0000
```

If we look at the summary statistics of the variables, we can see that there are no missing values and we can also see that the scales are different for different continuous variables which needs to be handled if any algorithm is sensitive to that. Another thing we can notice that there are

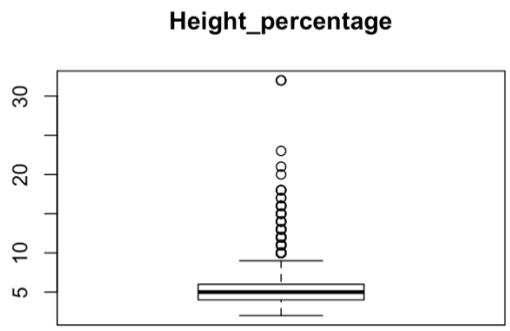
outliers. For example, if we look at the distribution of age, the max is 995 which can bias the classification.

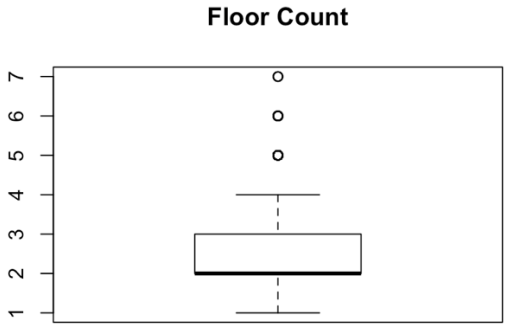


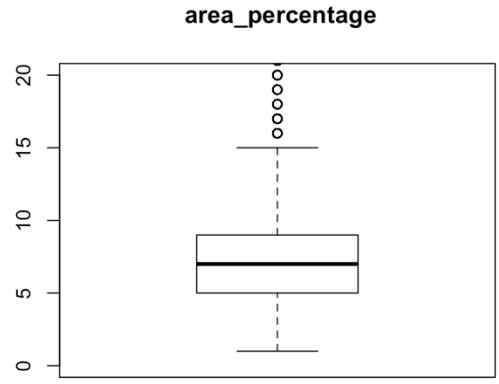
We can see that the sampled dataset has imbalanced data and also the number of observations in level 1 damage level are very low. In order to balance it out we need to perform sampling in order to get accurate results.

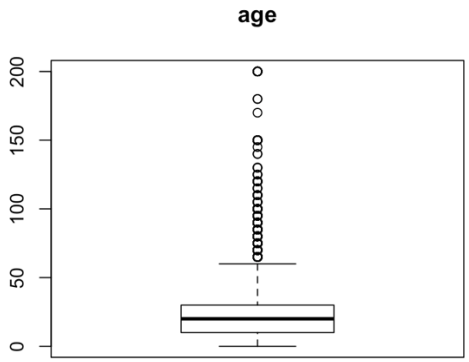
In our dataset the dependent variable is of int class where 3>2>1. We need to change it to factor class for our classification model.

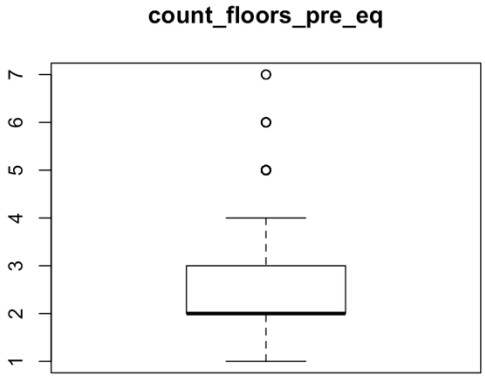
Outlier detection:





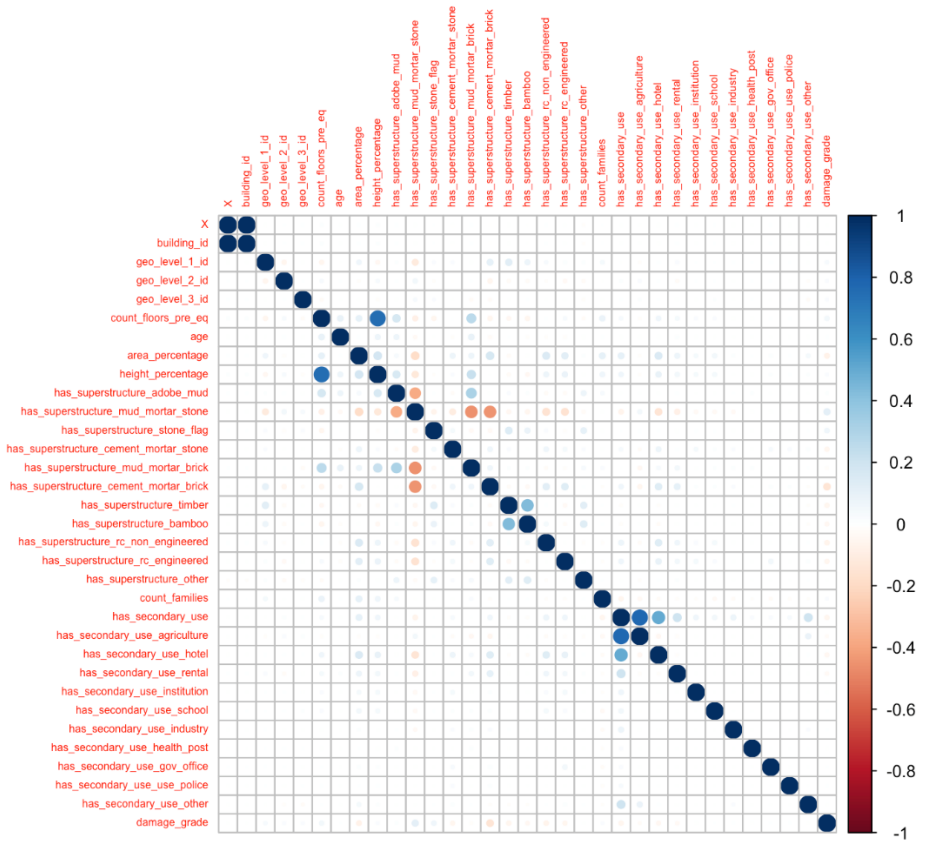




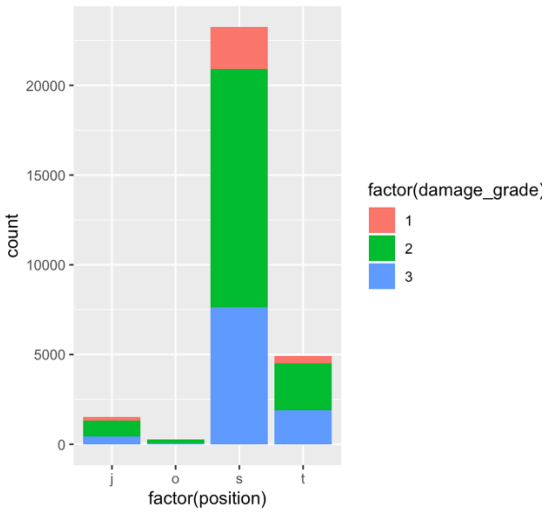


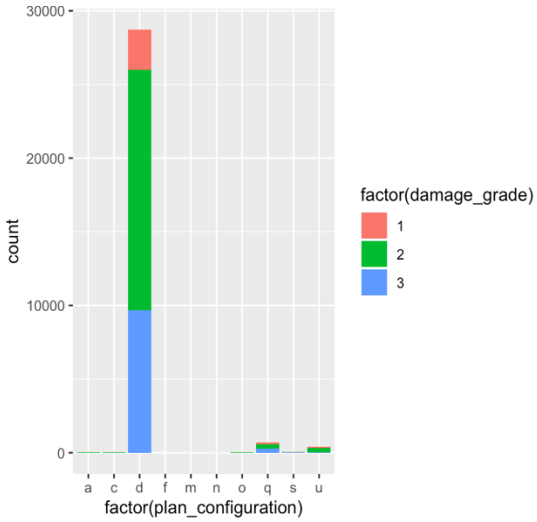
We observed that few of the numeric variables in the dataset have outliers by looking at the boxplot. Though algorithms like random forest are not sensitive to outliers, if we are using algorithms which might be affected by outliers then they should be removed.

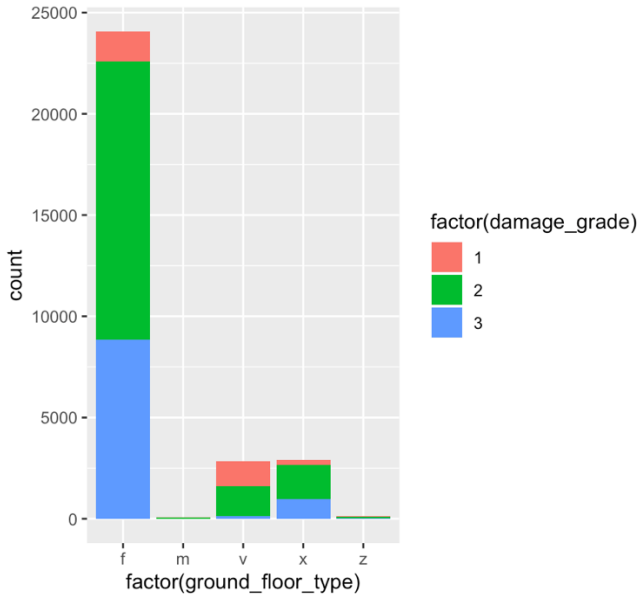
Correlation plot for numeric variables:

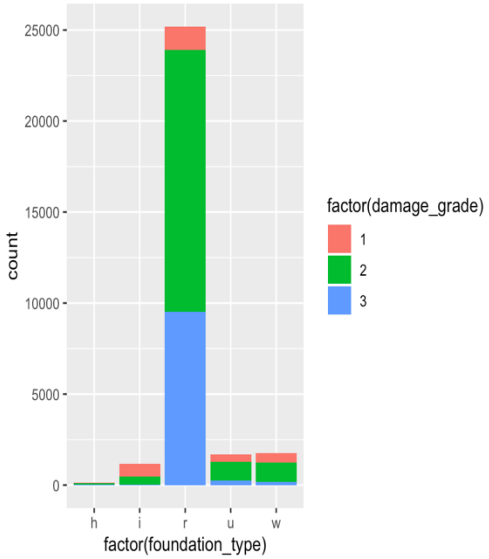


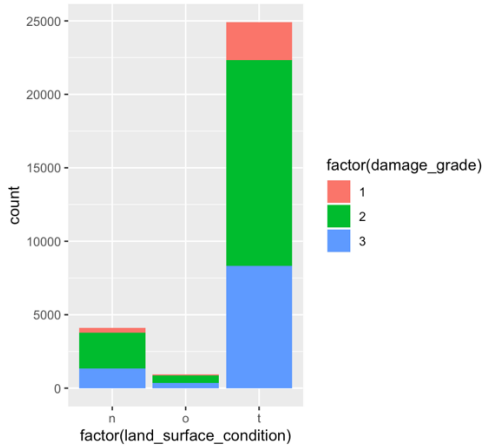
We can see 2 unnecessary variables which is X and building_id which can be removed in the feature selection stage. Apart from that there is a high correlation between height_percentage and count_floors_pre_eq, has_secondary_use and has_secondary_use_agriculture, has_superstructure_mud_mortar_brick and has_superstructure_mud_mortar_stone, We can go ahead and remove one of the highly correlated variables.











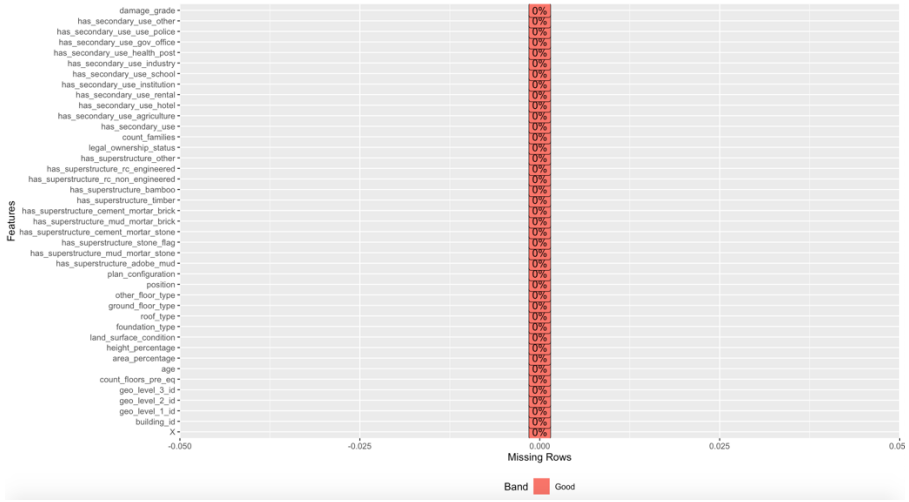
From these bar plots we can see the count of damage_levels 1,2,3 for different levels of different categorical variables.

For example: The damages are highest for plan_configuration d, in which damage level 2 is the highest.

- **DATA PREPOSSESING**

The first step before building model for the given dataset is to perform data preprocessing Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

1. Data Quality - Checking for missing values



We can see that there are no missing values in the dataset. Hence, there is no need for imputation.

2. Data quality - Checking for outliers

   As observed in the EDA there are outliers in a few variables which need to be handled. This is an alternative step for algorithms which can handle outliers. We went ahead and got rid of the outliers present in the variables age, height_percentage, area_percentage, count_floors_pre_eq and count_families.

3. Handling Categorical data

   List of categorical variables: land_surface_condition, foundation_type, roof_type ground_floor_type,other_floor_type,position,plan_configuration, legal_ownership_status

   We used the dummyVars() function, which creates a full set of dummy variables. This is an alternative step for algorithms like random forest but in the case of models like single regression models dummy variables are useful because they enable us to use a single regression equation to represent multiple groups.

4. Removing Irrelevant and redundant variables

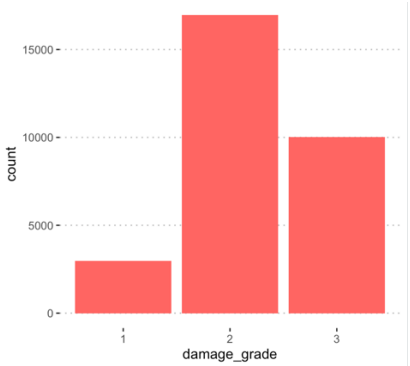
We removed irrelevant variables X and building_id. Apart from these we can also get rid of the variables which have high correlation with other variables in the dataset. We calculated the correlation for all the variables in the dataset and we found 4 variables: count_floors_pre_eq, position_s, has_secondary_use, plan_configuration_d, which are highly correlated to 4 other variables in the dataset (corr>0.75). Hence, we removed these variables.

5. Balancing the data

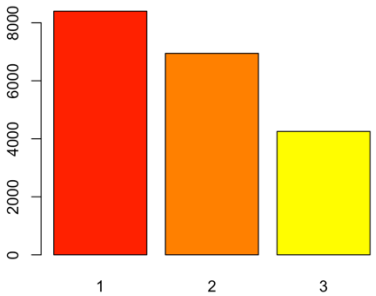
SMOTE (*Synthetic Minority Oversampling Technique*) is a powerful sampling method that goes beyond simple under or over sampling.

Smote synthetically generates new minority instances between existing instances. The new instances created are not just the copy of existing minority cases instead; the algorithm takes sample of feature space for each target class and its neighbors and then generates new instances that combine the features of the target cases with features of its neighbors.
This approach increases the features available to each class and makes the samples more general. SMOTE takes the entire dataset as an input, but it increases the percentage of only the minority cases.

As you can see from the below bar plots. The count of classes before and after SMOTE.



Before SMOTE



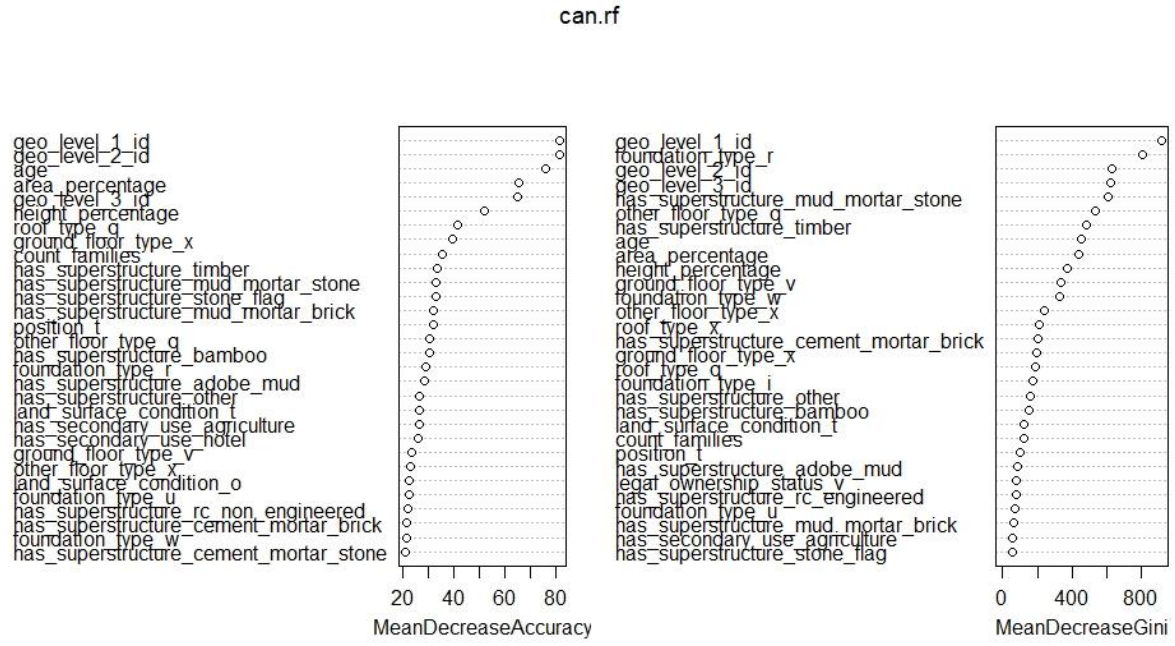
After SMOTE

- **METHOD USED FOR VARIABLE SELCTION**

  - **Random Forest**

Our dataset consists of 39 variables (excluding dummy variables). Out of these variables, only a few of them are highly correlated to each other but that leads to the issue of multicollinearity as mentioned earlier. Some of the variables show strong scientific relationships making them redundant. In order to make sure are predicting the correct value for ordinal variable damage_grade we are using random forest method as feature selection method.

In Random forest method we can measure the importance of a variable by measuring the degree of association between the classification result and a specific variable. One of the means for doing so is Mean Decrease Accuracy. For instance, to estimate a variable's importance for variable x, let the out-of-bag observations pass down all the trees and record the accuracy result. Then the values of variable x are permuted, or we can delete the column. The calculation is carried out tree by tree repeatedly. Mean decrease accuracy is averaged over all trees and is used to measure the importance of variable x. If the prediction accuracy drops a lot, then it suggests that variable x is strongly associated with the response.
After implementing this method, the variables we found to be important in our dataset are as follows:



From the above two plot we finalized 8 variables geo_level_1_id, geo_level_2_id, geo_level_3_id, age, area_percentage, height_percentage, has_superstructure_mud_mortar_stone and foundation_type_r.

6. Standardizing continuous variables:

Standardization is the process of putting different variables on the same scale. This process allows you to compare scores between different types of variables. Typically, to standardize variables, you calculate the mean and standard deviation for a variable. Then, for each observed value of the variable, you subtract the mean and divide by the standard deviation.
This process produces standard scores that represent the number of standard deviations above or below the mean that a specific observation fall.

Standardization can be important for algorithms like ANN because if one feature is ranging from 0 to 0.1 and other features range from 1-1000 the ANN can give more importance to certain features.

It's a bit like if like your features are kg and cm or kg and m, it might make a difference when it shouldn't.

IV. METHODOLOGY (MODELS)

Partitioning data into training and training sets allows you to develop highly accurate models that are relevant to data that you collect in the future, not just the data the model was trained on. We partitioned the data into a series of test/training partitions (80/20 split) using createDataPartition() function.

Multi-label Classification

Otherwise known as multiclass classification where we must classify the target variable into more than two classes. In our case it is of three classes/levels pertaining to damage levels in building after an earthquake.

We have implemented two models for the given data set in order to classify the target variable damage_grade. In this section we are going to compare the results between two models and check which one is the better fit for the dataset.

Model 1: Classification using Decision Tree

We have implemented Decision tree as our first model. Decision tree methodology is a commonly used data mining method for establishing classification systems based on multiple covariates or for developing prediction algorithms for a target variable. This method classifies a population into branch-like segments that construct an inverted tree with a root node, internal nodes, and leaf nodes.

We are using decision trees due to following distinct advantages in classification

- It is one of the CART algorithms which handles multiclass classification.
- It is inexpensive to construct.
- It is extremely fast in classification.
- It simplifies complex relationships between input variables and target variables and gives their variable importance.
- Decision trees are robust to outliers since the partitioning happens based on the proportion of samples within the split ranges and not on absolute values.

After that the modelling data was split in two parts, one for training and the other for testing in 80:20 ratio. The model was trained on the training dataset and was tested on the same to checking the performance of the model. Here no parameters were tuned.
Without tuning any parameters, the model's performance is shown in the below picture.

```
> confusionMatrix(data= tree.is.preds,reference = train$damage_grade, mode = "prec_recall")
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3
         1 5254  920  198
         2 1212 4099 2092
         3  254  538 1114

Overall Statistics

               Accuracy : 0.6675
                 95% CI : (0.6601, 0.6749)
    No Information Rate : 0.4285
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.474

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Precision              0.8245   0.5537  0.58447
Recall                 0.7818   0.7376  0.32726
F1                     0.8026   0.6326  0.41959
Prevalence             0.4285   0.3544  0.21708
Detection Rate         0.3351   0.2614  0.07104
Detection Prevalence   0.4064   0.4721  0.12155
Balanced Accuracy      0.8285   0.7056  0.63138
```

As shown the model gave an accuracy of 67%. And had good F1 measures for Class 1 and Class2. Recall for Class 1 is high at 0.78. Recall refers to the percentage of total relevant results correctly classified by your algorithm. Meaning the model is able to correctly classify only 30% of Class3. This performance can be improved by Pruning the tree.

Pruning the tree will reduce the risk of overfitting, it will help in avoiding insignificant splits and introduces a complexity parameter which imposes a penalty to the tree for having many

splits. Complexity parameters tuned with K-fold cross validation will produce better metrics than the above model.

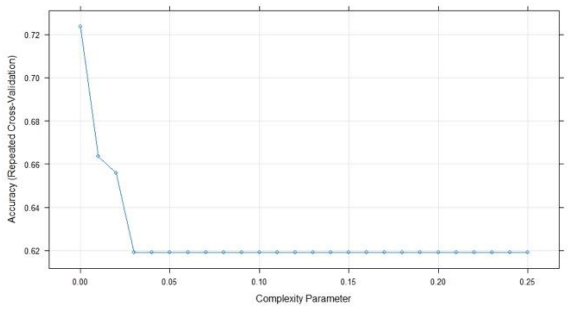
The picture below shows the accuracy of the model tuned by grid search with 5 fold validation.

```
> confusionMatrix(DTFit)
Cross-Validated (10 fold, repeated 3 times) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    1    2    3
         1 36.4  5.0  1.1
         2  5.6 23.7  8.4
         3  0.9  6.7 12.2

 Accuracy (average) : 0.7237
```

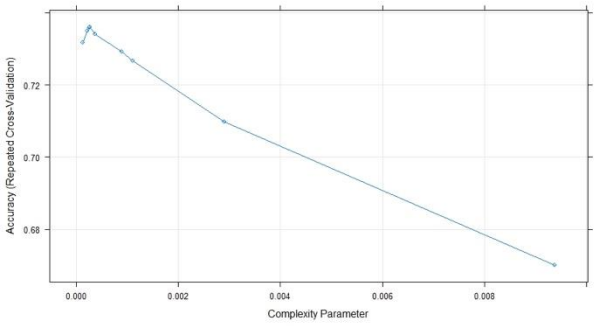
The picture below shows the accuracy of the model tuned by random search with 5-fold validation.

```
> confusionMatrix(DT2Fit)
Cross-Validated (10 fold, repeated 3 times) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    1    2    3
         1 36.6  4.8  0.9
         2  5.7 25.0  8.8
         3  0.5  5.7 12.0

 Accuracy (average) : 0.7361
```

Grid search



Random Search



Based on the accuracy metric and the plots of Complexity parameter vs Accuracy. The random search tuned model as the better one. We picked this as the classification model.

Training set performance

```
> train_perf
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3
         1 5931  577  128
         2  727 4316 1114
         3   62  664 2162

Overall Statistics

               Accuracy : 0.7913
                 95% CI : (0.7849, 0.7977)
    No Information Rate : 0.4285
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6737

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Precision              0.8938   0.7010   0.7486
Recall                 0.8826   0.7767   0.6351
F1                     0.8881   0.7369   0.6872
Prevalence             0.4285   0.3544   0.2171
Detection Rate         0.3782   0.2752   0.1379
Detection Prevalence   0.4232   0.3926   0.1842
Balanced Accuracy      0.9020   0.7974   0.7880
```

Testing set performance

```
> test_perf
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3
         1 1417  186   32
         2  246  996  347
         3   17  207  471

Overall Statistics

               Accuracy : 0.7359
                 95% CI : (0.7218, 0.7497)
    No Information Rate : 0.4287
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5867

 Mcnemar's Test P-Value : 1.835e-10

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Precision              0.8667   0.6268   0.6777
Recall                 0.8435   0.7171   0.5541
F1                     0.8549   0.6689   0.6097
Prevalence             0.4287   0.3544   0.2169
Detection Rate         0.3616   0.2541   0.1202
Detection Prevalence   0.4172   0.4055   0.1773
Balanced Accuracy      0.8730   0.7413   0.7406
```

As you can see from the above performance metrics. The model performance increased. Especially the classification of Class 3(High damage level) at 0.55. Both the accuracy level shows the model is not overfit. And the model has good F1 measure over all the three classes.

Model 2: Classification using Artificial Neural Networks

Artificial neural networks (ANNs) are biologically inspired computer programs designed to simulate the way in which the human brain processes information. ANNs gather their knowledge by detecting the patterns and relationships in data and learn (or are trained) through experience, not from programming. An ANN is formed from hundreds of single units, artificial neurons or processing elements (PE), connected with coefficients (weights), which constitute the neural structure and are organized in layers. The power of neural computations comes from connecting neurons in a network.

Why we chose Artificial Neural Networks

- Though they are hard to interpret they produce strong performance and are frequently used in Data mining.
- Best applied when input and output data are well-defined, yet their relationships are very complex.
- ANNs can generalize — After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.

Like Decision Tree model, the dataset was split into 80%(training) and 20%(testing). Using the nnet() function a neural network of size 3(nodes) was created and trained on the training dataset.

Its performance was calculated and is shown below.

```
> confusionMatrix(outpreds, test_ann$damage_grade, mode="prec_recall")
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3
         1 1356  217   90
         2  319 1141  735
         3    5   31   25

Overall Statistics

               Accuracy : 0.6435
                 95% CI : (0.6283, 0.6585)
    No Information Rate : 0.4287
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4215

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Precision              0.8154   0.5198 0.409836
Recall                 0.8071   0.8215 0.029412
F1                     0.8112   0.6367 0.054885
Prevalence             0.4287   0.3544 0.216892
Detection Rate         0.3460   0.2911 0.006379
Detection Prevalence   0.4243   0.5601 0.015565
Balanced Accuracy      0.8350   0.7024 0.508841
```

As you can see from the above performance. The model was bad. It had an accuracy of 64% and the recall value of Class 3 is 0.02 meaning it is able to correctly identify only 2% of the class 3 damage. In order to get better performance. We tuned to model.

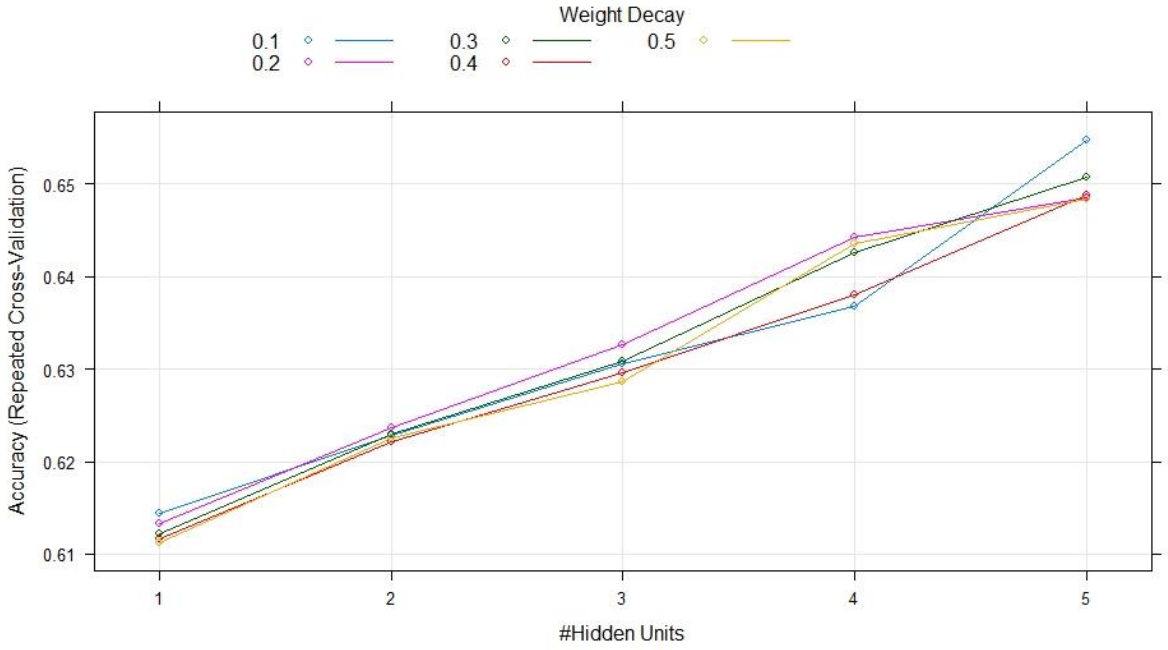
The below two pictures show the performance of the tuned modeled on the training and testing datasets.

Training Performance

```
> confusionMatrix(inpreds, train_ann$damage_grade, mode="prec_recall")
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3
         1 5426  969  291
         2 1267 4468 3008
         3   27  120  105

Overall Statistics

               Accuracy : 0.6377
                 95% CI : (0.6301, 0.6452)
    No Information Rate : 0.4285
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.412

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Precision              0.8115   0.5110 0.416667
Recall                 0.8074   0.8040 0.030846
F1                     0.8095   0.6249 0.057440
Prevalence             0.4285   0.3544 0.217078
Detection Rate         0.3460   0.2849 0.006696
Detection Prevalence   0.4264   0.5576 0.016070
Balanced Accuracy      0.8334   0.6909 0.509436
```

Testing Performance

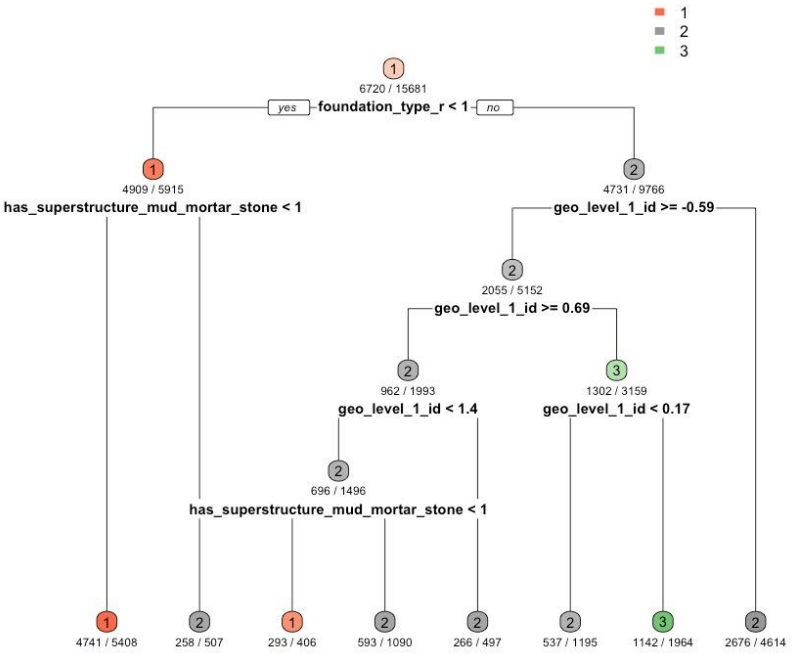
```
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3
         1 5371 1090  267
         2 1344 4285 3025
         3    5  182  112

Overall Statistics

               Accuracy : 0.6229
                 95% CI : (0.6153, 0.6305)
    No Information Rate : 0.4285
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3883

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Precision              0.7983   0.4951 0.374582
Recall                 0.7993   0.7711 0.032902
F1                     0.7988   0.6031 0.060491
Prevalence             0.4285   0.3544 0.217078
Detection Rate         0.3425   0.2733 0.007142
Detection Prevalence   0.4291   0.5519 0.019068
Balanced Accuracy      0.8239   0.6698 0.508835
```



Based on the above performance pictures and the weight decay plot you can see that ANN performance is bad for this three-level classification. The blue line peaks at 5 hidden units.

V. DISCUSSION & CONCLUSION

In this paper we have implemented two models to predict the level of damage caused by Gorkha earthquake on a largescale dataset with more than 29k observations which was collected through household surveys using mobile technology by Kathmandu Living Labs and the Central Bureau of Statistics. Based on this dataset and our models we can conclude that Decision Tree gave the better performance both in terms of accuracy and Recall value of Class 3 which represents High level of damage grade in building.



From the analysis it is clear that the features that determine the level of damage occurring in any building due to earthquake are geographical location of the building, foundation type, height percentage and the type of mortar used along with the type of masonry.

This analytical feature coincides with the practical world as well. These are the one a civil engineer investigates while constructing any structure.

Business Implications:

- This analysis will help Construction companies and civil engineers who aim to build any structure in an earthquake prone zone.

- The features selected from the decision tree coincide with practical ones and should be taken into account.

- This also gives us an idea on the building which need renovation so that they are saved from further disasters.

VI. REFERENCES

1. DrivenData. 2020. Richter's Predictor: Modeling Earthquake Damage. [online] Available at: <https://www.drivendata.org/competitions/57/nepal-earthquake/page/136/>.

2. Usgs.gov. 2020. The Science Of Earthquakes. [online] Available at: <https://www.usgs.gov/natural-hazards/earthquake-hazards/science/science-earthquakes?qt-science_center_objects=0>.

3. 2020. [online] Available at: <http://www.cenc.ac.cn/publish/cenc/887/2015042520075821523226/index.html>.

4. Nytimes.com. 2020. Earthquake Aftershocks Jolt Nepal As Death Toll Rises Above 3,400. [online] Available at: <https://www.nytimes.com/2015/04/27/world/asia/katmandu-nepal-fear-loss-and-devastation.html?emc=edit_th_20150427&nl=todaysheadlines&nlid=58413496&_r=0>.

5. Techopedia.com. 2020. What Is Data Preprocessing? - Definition From Techopedia. [online] Available at: <https://www.techopedia.com/definition/14650/data-preprocessing> .

6. Yan-yan SONG, Y., 2020. Decision Tree Methods: Applications For Classification And Prediction. [online] PubMed Central (PMC). Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/>.

7. Sciencedirect.com. 2020. Decision Trees - An Overview | Sciencedirect Topics. [online] Available at: <https://www.sciencedirect.com/topics/computer-science/decision-trees>.

8. Medium. 2020. Introduction To Neural Networks, Advantages And Applications. [online] Available at: <https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207>.