Eartquake_Damage_Predictor

Muhammad Ali

26/12/2020

Introduction

The goal of this project is to predict damage on buildings during the Gorkha earthquake in Nepal. A model was constructed using a dataset provided by drivendata First an analysis of the dataset for insight was performed. The outcome of this stage case two new datasets, a filtered dataset (with less predictors) and an expanded dataset (with more predictors). Models of two different classes (linear and tree based) were built using the three datasets. Finally the model with the greatest accuracy was selected, retrained on the entire dataset, and had its performance analysed by submitting its predictions on an unseen testdataset. The model was successful because it ranked in the top 11% of all models created for this problem.

Dataset

The label for this dataset was damage_grade, which is a three level factor variable representing the damage the building received. A level of '1' represented low damage, '2' represents medium damage, and '3' represents complete damage.

For each building in the dataset, there were 39 features. Three of them had to do with the geographic region in which building exists. The numeric variables included the number of floors, number of families in the building, age, area percentage, height percentage. The categorical variables included foundation type, roof type, ground floor type, other floor type, position, and plan configuration. What is important to note is that the factor variables have there names assigned randomly. Hence the actual levels are meaningless, the only thing one can tell from the data is if two building are of the same or different factor level. For example legal ownership has levels a,r,v, and w. But none of a,r,v, or w have any meaning with legal ownership.

There were then 11 binary variables that had to the material of the building superstructure. Superstructure is basically the part of the building where people will spend most of there time, so it includes beams, columns, finishes and doors. There are 11 other binary variables that had to do with the secondary use of the building, such as if it was used as a school, police office, hotel, industry, etc.

```r
setwd('C:/Users/iceca/Documents/Earthquake_Damage_Predictor')
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(nnet)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(e1071)
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.0.2

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.3

## corrplot 0.84 loaded
```

```r
loadRaw <- modules::use("Helpers/Load_Raw_Data.R")
train <- loadRaw$trainVal()
head(train)
```

```
##   building_id geo_level_1_id geo_level_2_id geo_level_3_id count_floors_pre_eq
## 1      802906              6            487          12198                   2
## 2       28830              8            900           2812                   2
## 3       94947             21            363           8973                   2
## 4      590882             22            418          10694                   2
## 5      201944             11            131           1488                   3
## 6      333020              8            558           6089                   2
##   age area_percentage height_percentage land_surface_condition foundation_type
## 1  30               6                 5                      t               r
## 2  10               8                 7                      o               r
## 3  10               5                 5                      t               r
## 4  10               6                 5                      t               r
## 5  30               8                 9                      t               r
## 6  10               9                 5                      t               r
##   roof_type ground_floor_type other_floor_type position plan_configuration
## 1         n                 f                q        t                  d
## 2         n                 x                q        s                  d
## 3         n                 f                x        t                  d
## 4         n                 f                x        s                  d
## 5         n                 f                x        s                  d
## 6         n                 f                q        s                  d
##   has_superstructure_adobe_mud has_superstructure_mud_mortar_stone
## 1                            1                                   1
## 2                            0                                   1
## 3                            0                                   1
## 4                            0                                   1
## 5                            1                                   0
## 6                            0                                   1
##   has_superstructure_stone_flag has_superstructure_cement_mortar_stone
## 1                             0                                      0
## 2                             0                                      0
## 3                             0                                      0
## 4                             0                                      0
## 5                             0                                      0
## 6                             0                                      0
##   has_superstructure_mud_mortar_brick has_superstructure_cement_mortar_brick
## 1                                   0                                      0
## 2                                   0                                      0
## 3                                   0                                      0
## 4                                   0                                      0
## 5                                   0                                      0
## 6                                   0                                      0
##   has_superstructure_timber has_superstructure_bamboo
## 1                         0                         0
## 2                         0                         0
## 3                         0                          0
## 4                         1                          1
## 5                         0                          0
## 6                         0                          0
##   has_superstructure_rc_non_engineered has_superstructure_rc_engineered
## 1                                    0                                0
## 2                                    0                                0
## 3                                    0                                0
## 4                                    0                                0
## 5                                    0                                0
## 6                                    0                                0
##   has_superstructure_other legal_ownership_status count_families
## 1                        0                      v              1
## 2                        0                      v              1
## 3                        0                      v              1
## 4                        0                      v              1
## 5                        0                      v              1
## 6                        0                      v              1
##   has_secondary_use has_secondary_use_agriculture has_secondary_use_hotel
## 1                 0                             0                       0
## 2                 0                             0                       0
## 3                 0                             0                       0
## 4                 0                             0                       0
## 5                 0                             0                       0
## 6                 1                             1                       0
##   has_secondary_use_rental has_secondary_use_institution
## 1                        0                             0
## 2                        0                             0
## 3                        0                             0
## 4                        0                             0
## 5                        0                             0
## 6                        0                             0
##   has_secondary_use_school has_secondary_use_industry
## 1                        0                          0
## 2                        0                          0
## 3                        0                          0
## 4                        0                          0
## 5                        0                          0
## 6                        0                          0
##   has_secondary_use_health_post has_secondary_use_gov_office
## 1                             0                            0
## 2                             0                            0
## 3                             0                            0
## 4                             0                            0
## 5                             0                            0
## 6                             0                            0
##   has_secondary_use_use_police has_secondary_use_other
## 1                            0                       0
## 2                            0                       0
## 3                            0                       0
## 4                            0                       0
## 5                            0                       0
## 6                            0                       0
```

An Important Note about the Code

Rather then using a single script, the project was broken into modules. Hence the code may be hard to follow since it calls function from other modules. For a better understanding of the module organization, please see the git repo available at https://github.com/icecap360/Earthquake_Damage_Predictor. Furthermore instead of RStudio, Jupyter Notebooks were used, which facilitated data exploration and model creation.

Methods and Analysis

Prior Research

The internet was consulted for expert opinions on what causes building damage during an earthquake. Surprisingly the use of the building, and the number of levels of the building were not important factors in determining building damage. Instead building foundation and material are of greater relevence. Specifically buildings made of wood and steel on solid bedrock are better then buildings made on concrete on loose ground. The primary reason for these guidelines is because researchers found that flexibility is a very important factor for a building to successfully survive an earthquake.

Initial Preprocessing

The dataset is quite large, over 25MB with over 260k training samples. The data came in a format where it was ready to be processed. For example there were no missing values. The training set was split into training and validation (which will be used only for error estimation). Test data without labels was supplied from the organization, and will be used for final error evaluation.

Analysis and Visualization

Note that all analysis and visualization, was done on the training set. This ensured that no information leaked from the validation set into our models, this way the validation set modelled real test data well.

To facilitate analysis, the features of the training set was split into types of numeric and factor.

```r
loadNF <- modules::use('Helpers/Load_Num_Factor.R')
manipulate <- modules::use('Helpers/Manipulate.R')
numTrain <- loadNF$num()[[1]]
factorTrain <- loadNF$factor()[[1]]
labels <- loadNF$factor()[[2]] #num and factor features have the same labels
numTrain <- manipulate$combineLab(numTrain, labels)
```

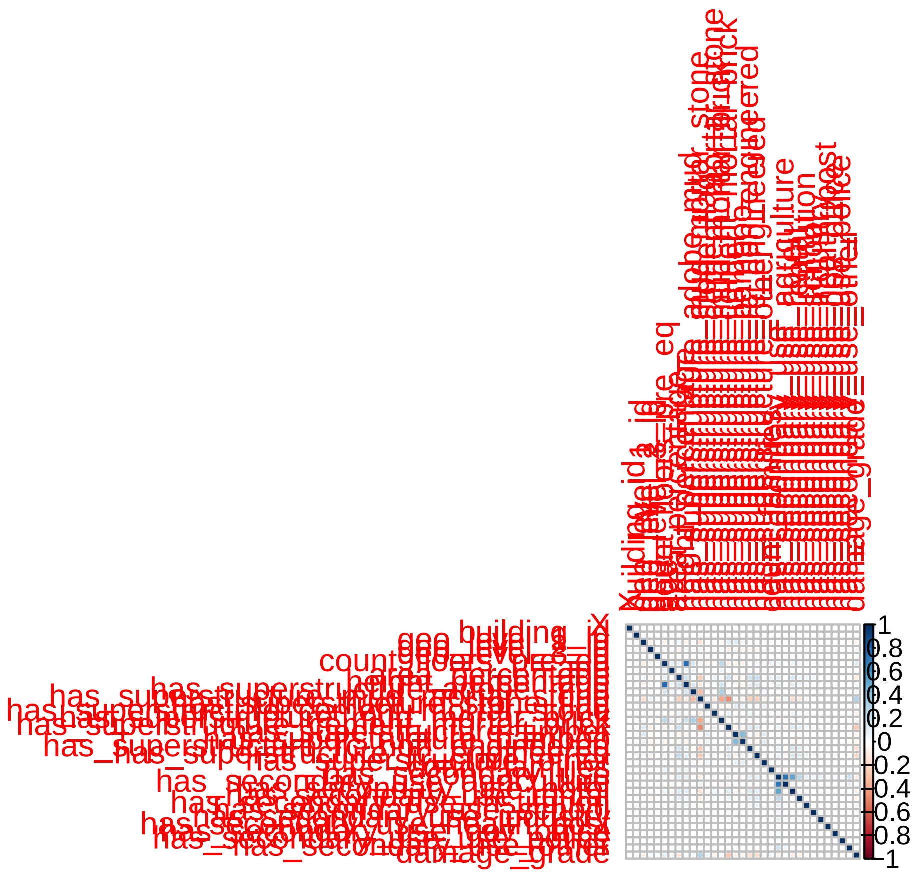
```
## Joining, by = c("X", "building_id")
```

```r
factorTrain <- manipulate$combineLab(factorTrain, labels)
```

```
## Joining, by = "X"
```

First the numerical variables were analysed. The label, which is an ordinal factor, was treated as an integer so that correlations could be calculated with it. Note that the corrplot because of so many features may not be clearly visible, please see the Analysis.ipnyb file in the git repo for the full plot.

```
##
##  Features that show greatest corrrlation with each other
##
```



```
## height_percentage count_floors_pre_eq 0.7715583
## has_superstructure_mud_mortar_stone has_superstructure_adobe_mud -0.3071836
## has_superstructure_mud_mortar_brick has_superstructure_adobe_mud 0.3172745
## has_superstructure_mud_mortar_brick has_superstructure_mud_mortar_stone -0.3758007
## has_superstructure_cement_mortar_brick has_superstructure_mud_mortar_stone -0.4712676
## has_superstructure_bamboo has_superstructure_timber 0.4395434
## has_secondary_use_agriculture has_secondary_use 0.7388177
## has_secondary_use_hotel has_secondary_use 0.5256072
```

```
##
##  Features that show greatest corrrlation with the label
```

```
## [1] "has_superstructure_cement_mortar_brick"
## [2] "has_superstructure_rc_engineered"
## [3] "has_superstructure_rc_non_engineered"
## [4] "area_percentage"
## [5] "count_floors_pre_eq"
## [6] "has_superstructure_mud_mortar_stone"
## [7] "damage_grade"
```

```
## -0.2543188 -0.1794018 -0.1580555 -0.1252037 0.1224673 0.2909351 1
```

Key findings between numerical features:

- Building height and area are heavily correlated.
- has_superstructure_mud_mortar_brick , has_superstructure_mud_mortar_stone , has_superstructure_adobe_mud ; these 3 variables correlate very heavily.

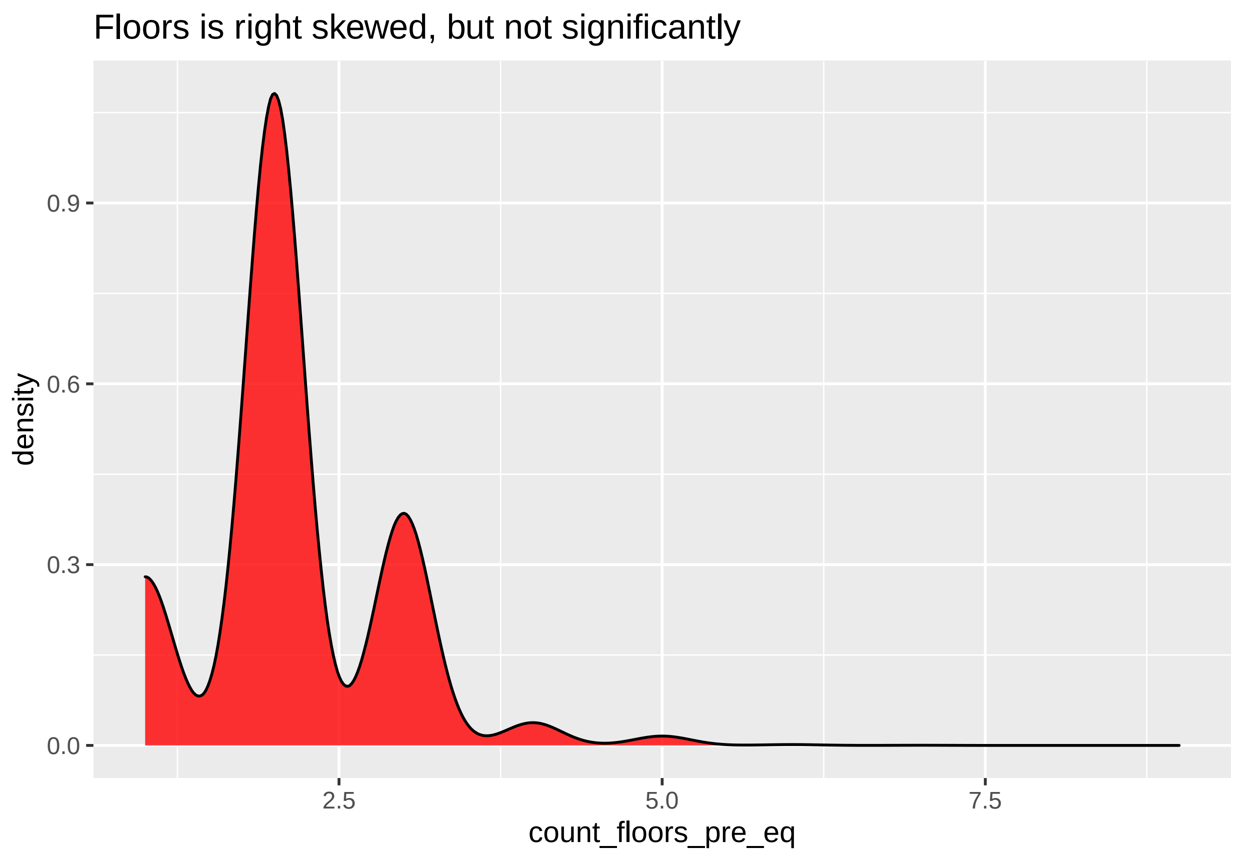
- A building that is used for agriculture or as a hotel has high chance of having a secondary use. Probably because these buildings have different seasonal uses.
- Overall there is very little correlation between the variables.

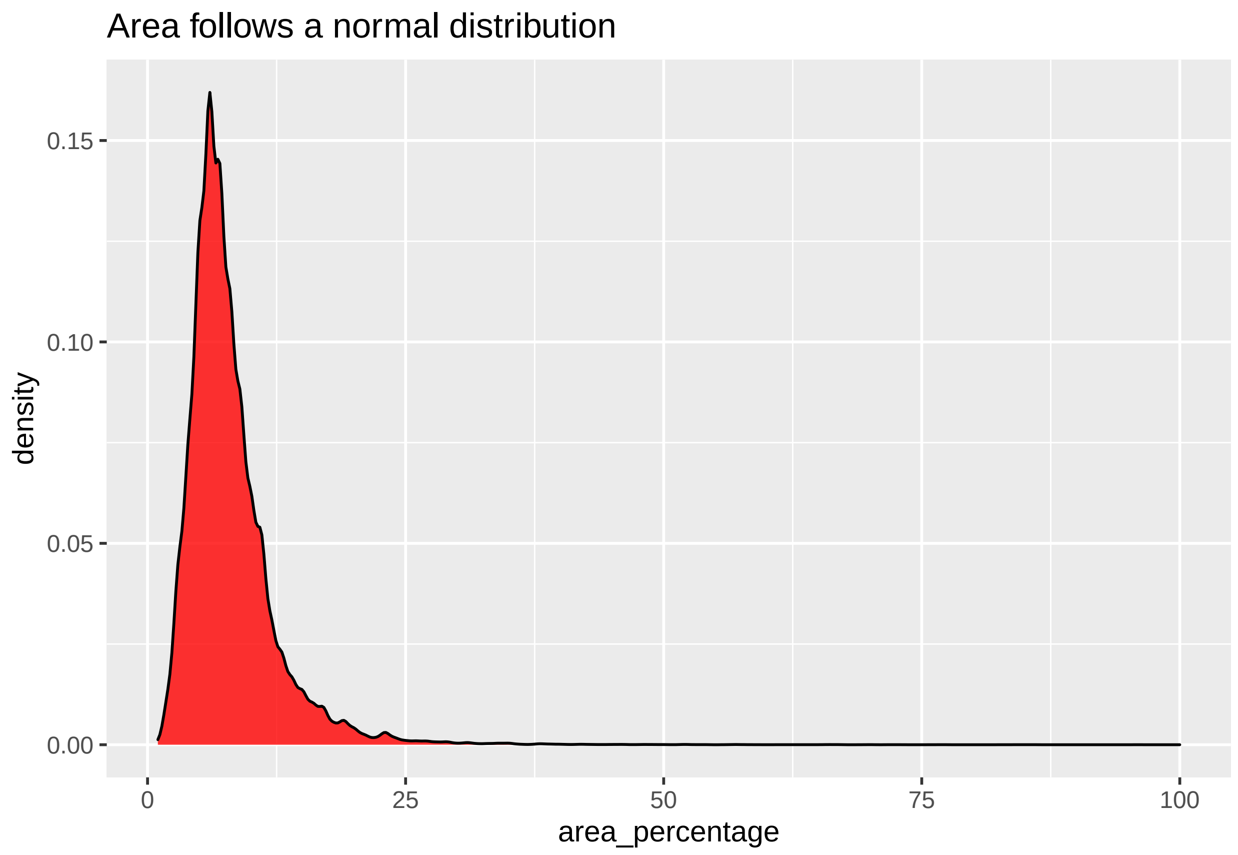
Key findings with regard to label:

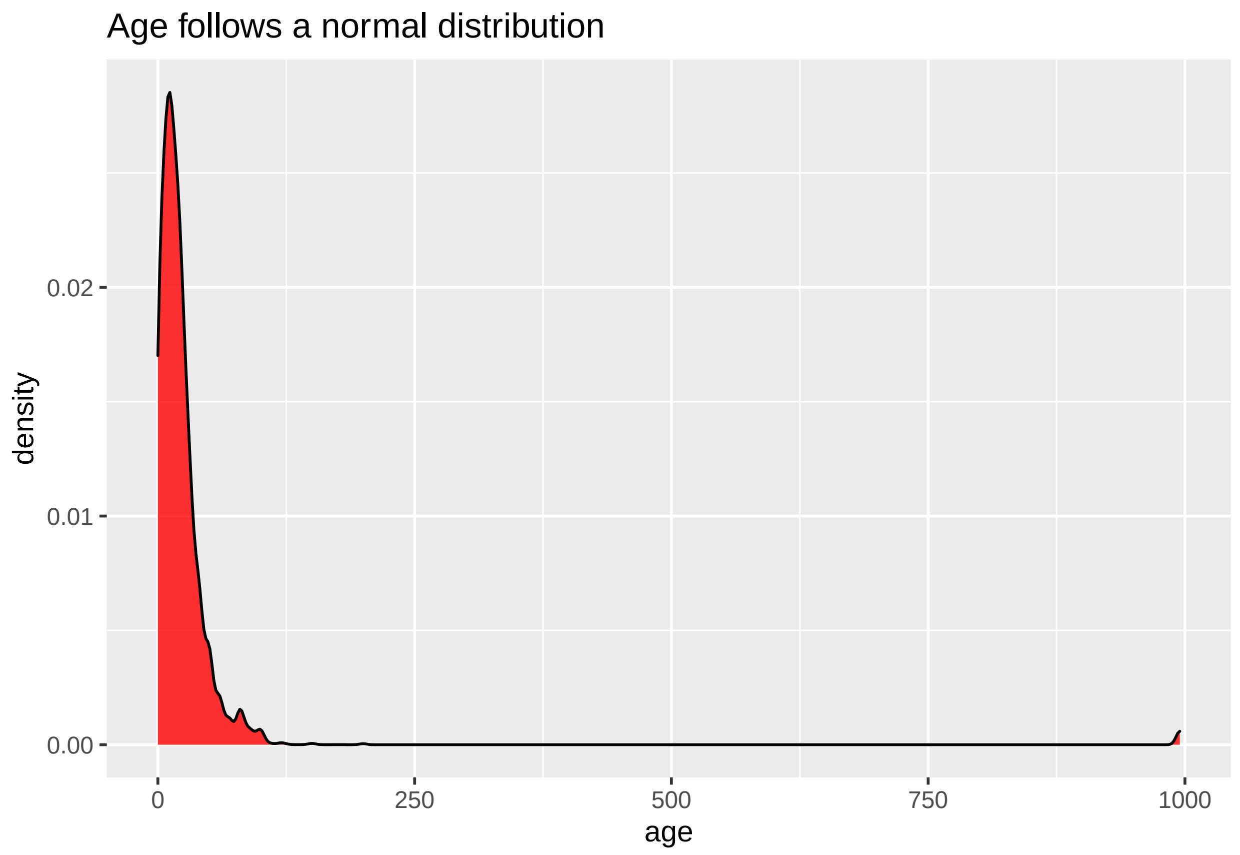
- The more floors a home has the more likely it is to be damaged. This goes against the conventional belief that more floors means greater flexibility/greater safety.
- The more area a home has the safer it is
- Superstructure is a valuable variable, it shows that reinforced concrete, mud mortar stone and cement mortar correlate heavily with label
- None of the features that had to do with building purpose/use correlated with the label.
- Having a cement mortar and mud mortar are significantly different, as such should be seperate variables.

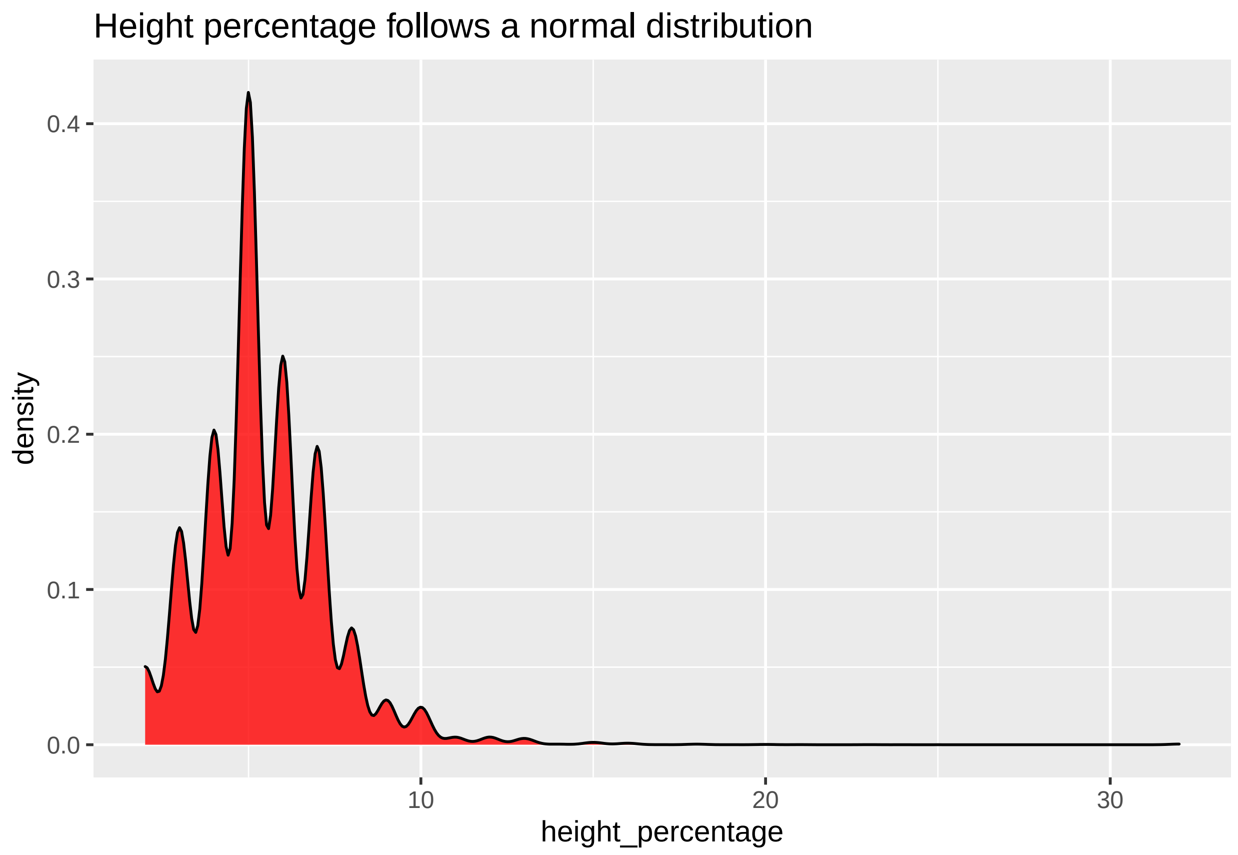
Next the distribution of numerical features was analysed, please read the chart titles for the anslysis.

```
## Note that the odd shape is because floors is discrete
```

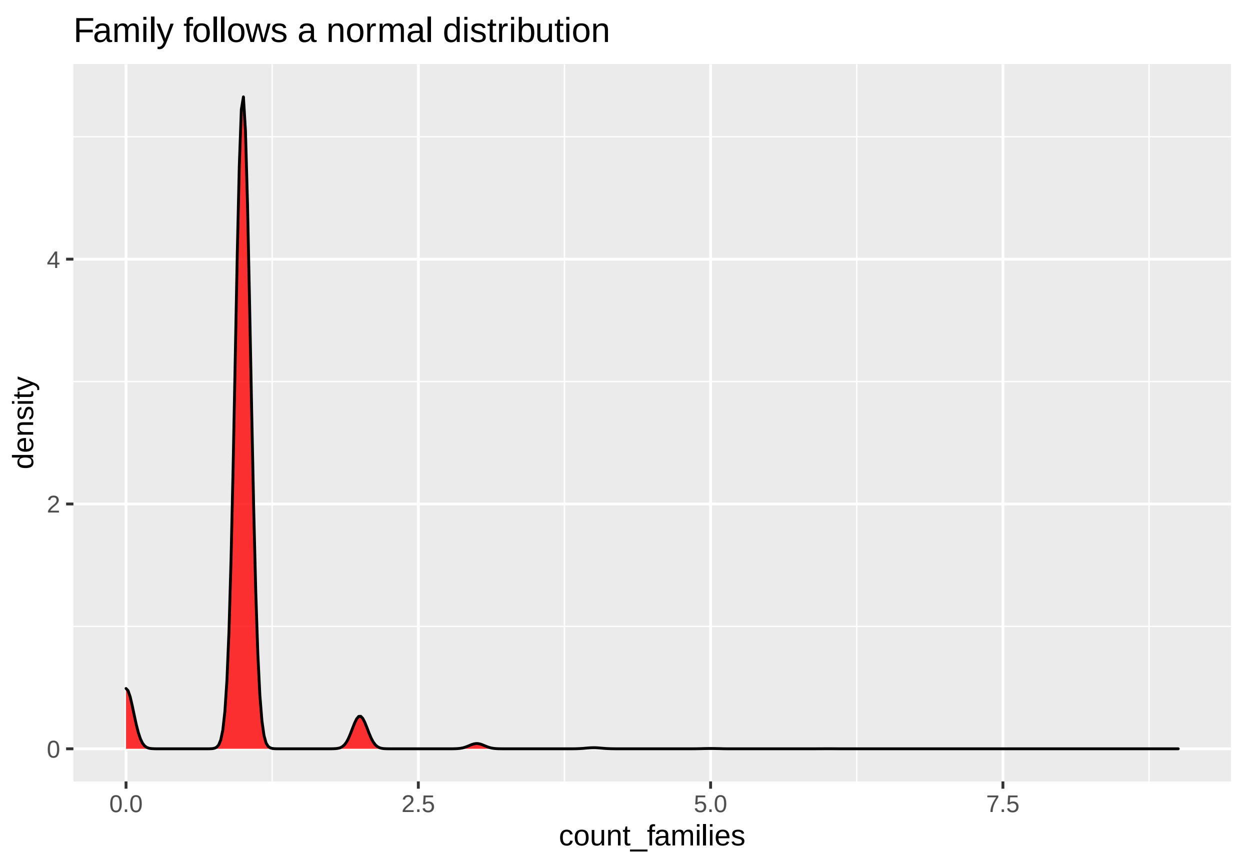



Area follows a normal distribution
density
0.15
0.10
0.05
0.00
0
25
50
75
100
area_percentage


Age follows a normal distribution
density
0.00
0.01
0.02
0
250
500
750
1000
age



```
## Note that the odd shape is because families is discrete
```



Next binary features were analysed. First the binary features were extracted from the numeric dataframe.

```
binaryFeat <- apply(numTrain,2,function(x) { all(x %in% 0:1) })
binTrain <- numTrain[,binaryFeat]
head(binTrain)
```

```
##   has_superstructure_adobe_mud has_superstructure_mud_mortar_stone
## 1                            1                                   1
## 2                            0                                   1
## 3                            0                                   1
## 4                            0                                   1
## 5                            1                                   0
## 6                            0                                   1
##   has_superstructure_stone_flag has_superstructure_cement_mortar_stone
## 1                             0                                      0
## 2                             0                                      0
## 3                             0                                      0
## 4                             0                                      0
## 5                             0                                      0
## 6                             0                                      0
##   has_superstructure_mud_mortar_brick has_superstructure_cement_mortar_brick
## 1                                   0                                      0
## 2                                   0                                      0
## 3                                   0                                      0
## 4                                   0                                      0
## 5                                   0                                      0
## 6                                   0                                      0
##   has_superstructure_timber has_superstructure_bamboo
## 1                         0                         0
## 2                         0                         0
## 3                         0                         0
## 4                         1                         1
## 5                         0                         0
## 6                         0                         0
##   has_superstructure_rc_non_engineered has_superstructure_rc_engineered
## 1                                    0                                0
## 2                                    0                                0
## 3                                    0                                0
## 4                                    0                                0
## 5                                    0                                0
## 6                                    0                                0
##   has_superstructure_other has_secondary_use has_secondary_use_agriculture
## 1                        0                 0                             0
## 2                        0                 0                             0
## 3                        0                 0                             0
## 4                        0                 0                             0
## 5                        0                 0                             0
## 6                        0                 1                             1
##   has_secondary_use_hotel has_secondary_use_rental
## 1                       0                        0
## 2                       0                        0
## 3                       0                        0
## 4                       0                        0
## 5                       0                        0
## 6                       0                        0
##   has_secondary_use_institution has_secondary_use_school
## 1                             0                        0
## 2                             0                        0
## 3                             0                        0
## 4                             0                        0
## 5                             0                        0
## 6                             0                        0
##   has_secondary_use_industry has_secondary_use_health_post
## 1                          0                             0
## 2                          0                             0
## 3                          0                             0
## 4                          0                             0
## 5                          0                             0
## 6                          0                             0
##   has_secondary_use_gov_office has_secondary_use_use_police
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   has_secondary_use_other
## 1                       0
## 2                       0
## 3                       0
## 4                       0
## 5                        0
## 6                        0
```

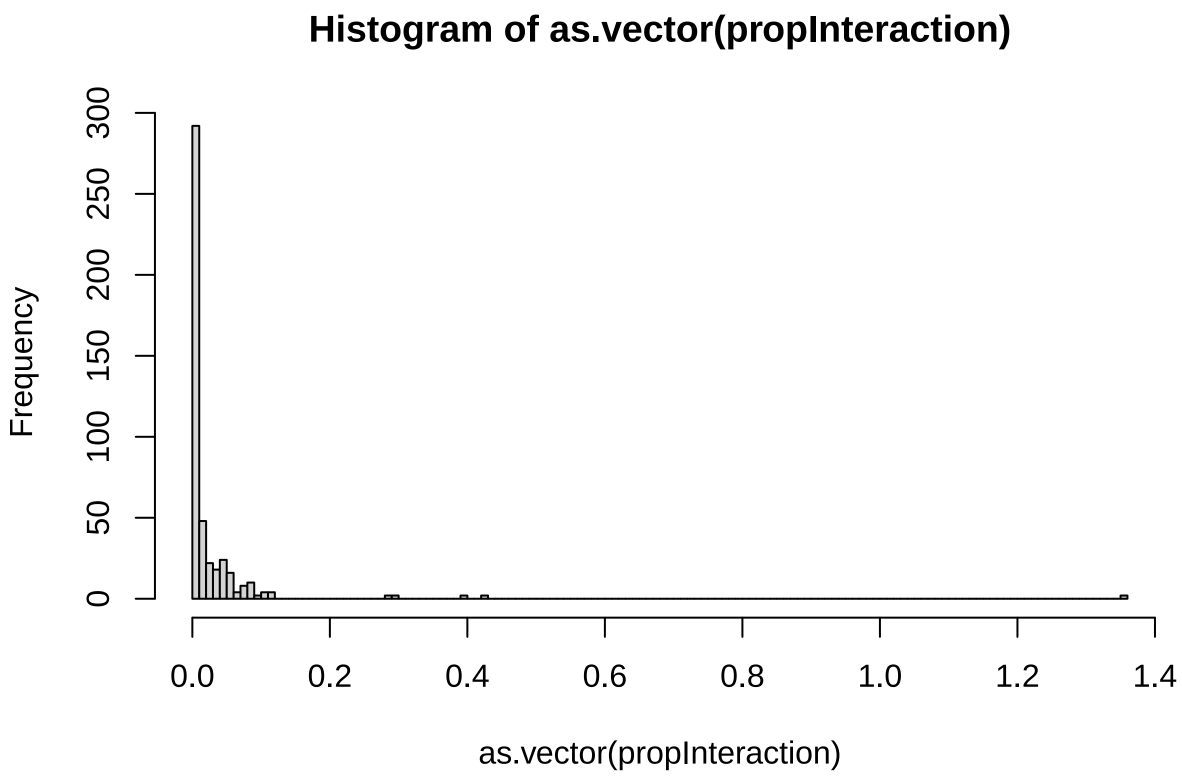
The first question is how common is it for buildings to have one of these binary features.

```
## How many distinct buildings have these features?
```

```
##          has_superstructure_adobe_mud    has_superstructure_mud_mortar_stone
##                                 20842                                 178711
##         has_superstructure_stone_flag has_superstructure_cement_mortar_stone
##                                  8002                                   4311
##   has_superstructure_mud_mortar_brick has_superstructure_cement_mortar_brick
##                                 15999                                  17687
##             has_superstructure_timber              has_superstructure_bamboo
##                                 59770                                  19988
##   has_superstructure_rc_non_engineered        has_superstructure_rc_engineered
##                                  9976                                   3730
##              has_superstructure_other                      has_secondary_use
##                                  3530                                  26265
##         has_secondary_use_agriculture                has_secondary_use_hotel
##                                 15105                                   7896
##              has_secondary_use_rental          has_secondary_use_institution
##                                  1900                                    229
##              has_secondary_use_school             has_secondary_use_industry
##                                    89                                    254
##         has_secondary_use_health_post           has_secondary_use_gov_office
##                                    42                                     35
##          has_secondary_use_use_police                has_secondary_use_other
##                                    20                                   1202
```

Another question about the binary variables is how exclusive and independent are they? How often is a building a member of two distinct binary features? For each pair of variables, percentage of buildings that had both features was calculated, and then plotted on a histogram. The histogram shows that the majority of interactions (300) occur 0% of the time. Hence the binary variables are fairly exclusive. In other words a building is not very likely to be of 2 categories. Hence the formulas of the models need not investigate interaction between binary variables, as any relationship found will be among very few data samples.

```
## After calculating the frequency of buildings between every possible pair of binary features, a histog
```

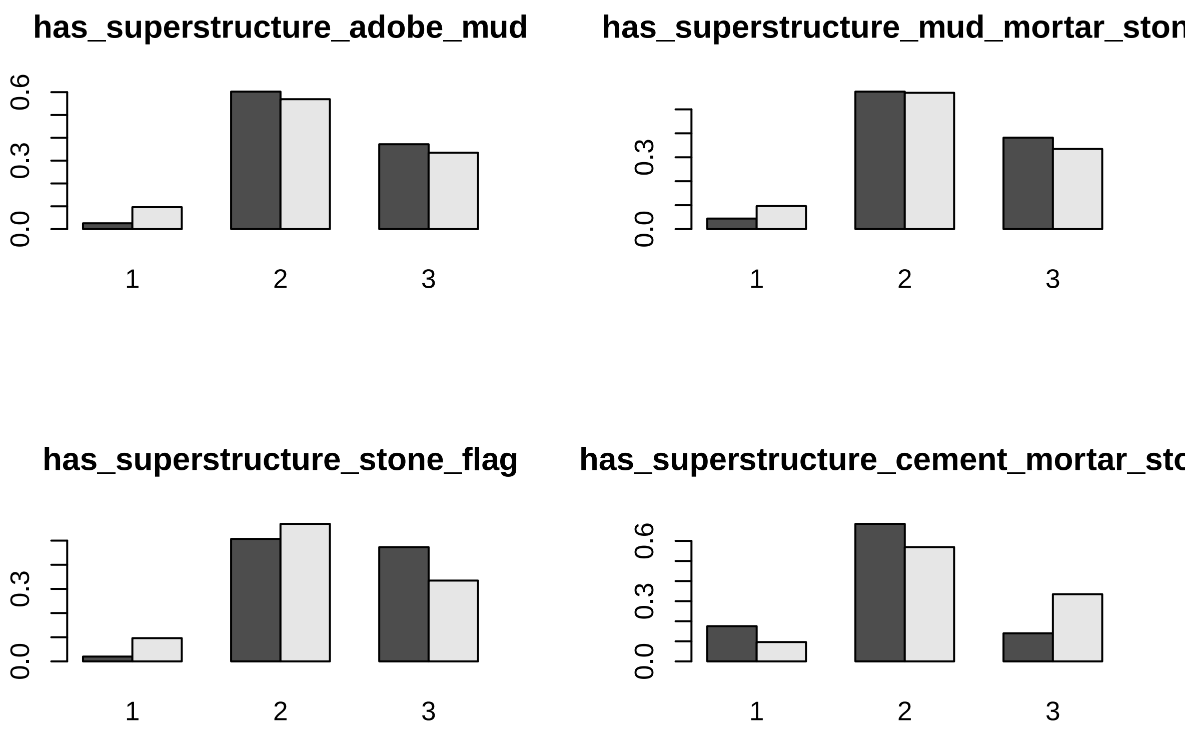


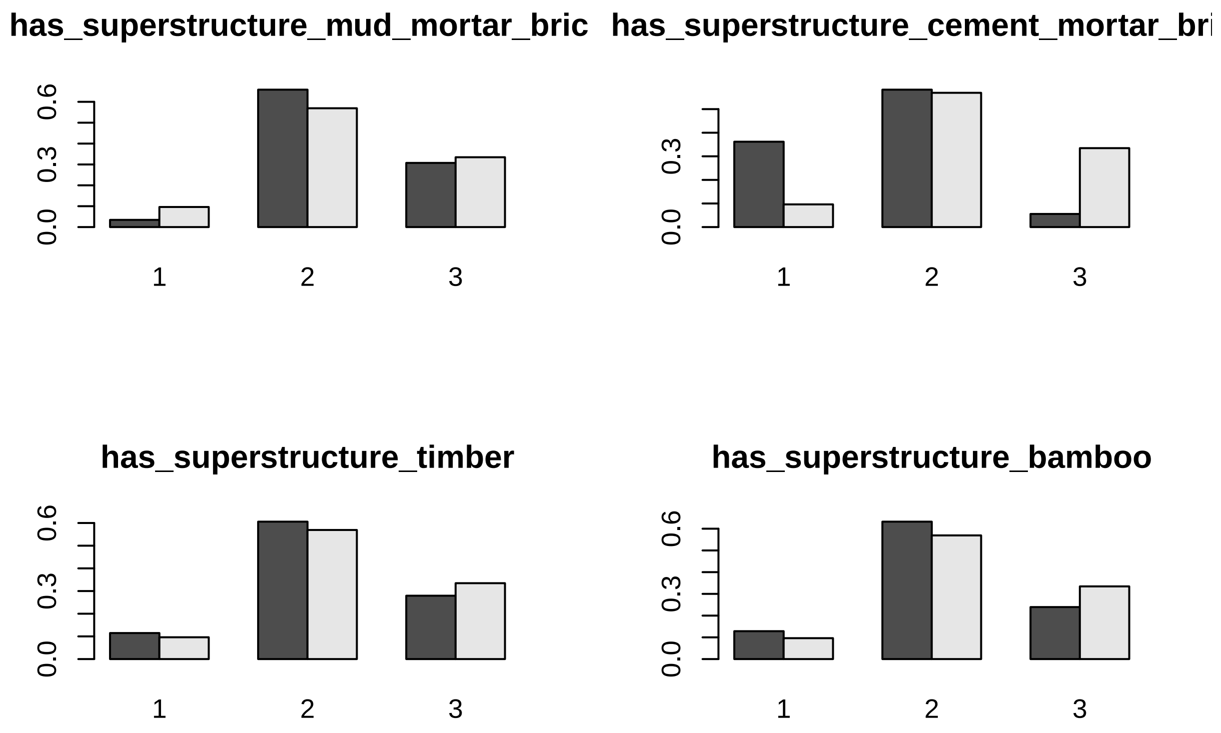
Another question that arises is how helpful is a particular binary feature in predicting the label. The purpose of these figures is to compare the distribution of damage_grade among those that have a particular feature and that of the whole data. Hence to determine if a feature is a good predictor, look for different distribution among the colors

```
binTrain %>%
    mutate(damage_grade =labels$damage_grade) ->
    binTrain2
nameBin2 <- names(binTrain2)
numFeat <- length(nameBin2)
plotBinLab <- function(start,end, nameBin2, binTrain2) {
    n <- floor(sqrt(end-start+1))
    par(mfrow = c(n,n))
    for (i in start:end) {
        freqFeat <- prop.table(table(binTrain2[,nameBin2[i]], numTrain$damage_grade), 1)
        freqData <- prop.table(table(binTrain2$damage_grade))
        freq <- rbind(freqFeat[2,],freqData)
        barplot( freq , beside=TRUE, main = nameBin2[i] )
    }
}
cat('The purpose of these figures is to compare the distribution of damage_grade among those that have a
    particular feature and that of the whole data. Hence to determine if a feature is a good predictor,
    diffferences among the colors')
```

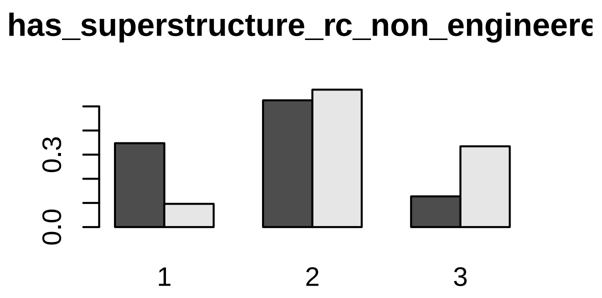
```
## The purpose of these figures is to compare the distribution of damage_grade among those that have a
##     particular feature and that of the whole data. Hence to determine if a feature is a good predicto
##     diffferences among the colors
```

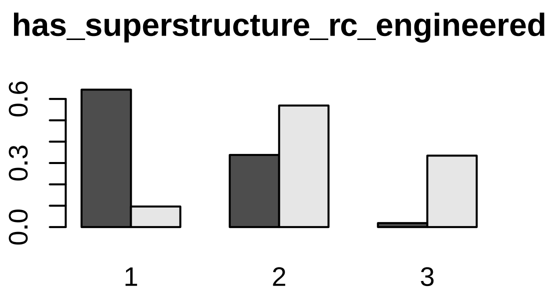
```r
plotBinLab(1,8, nameBin2, binTrain2)
```

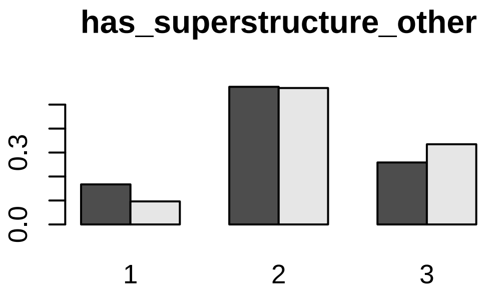


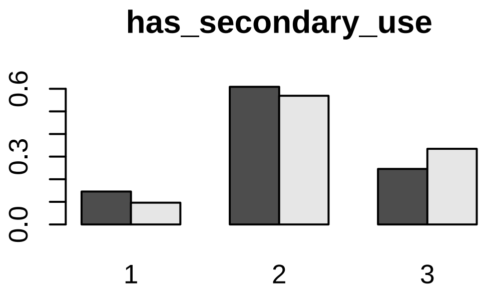


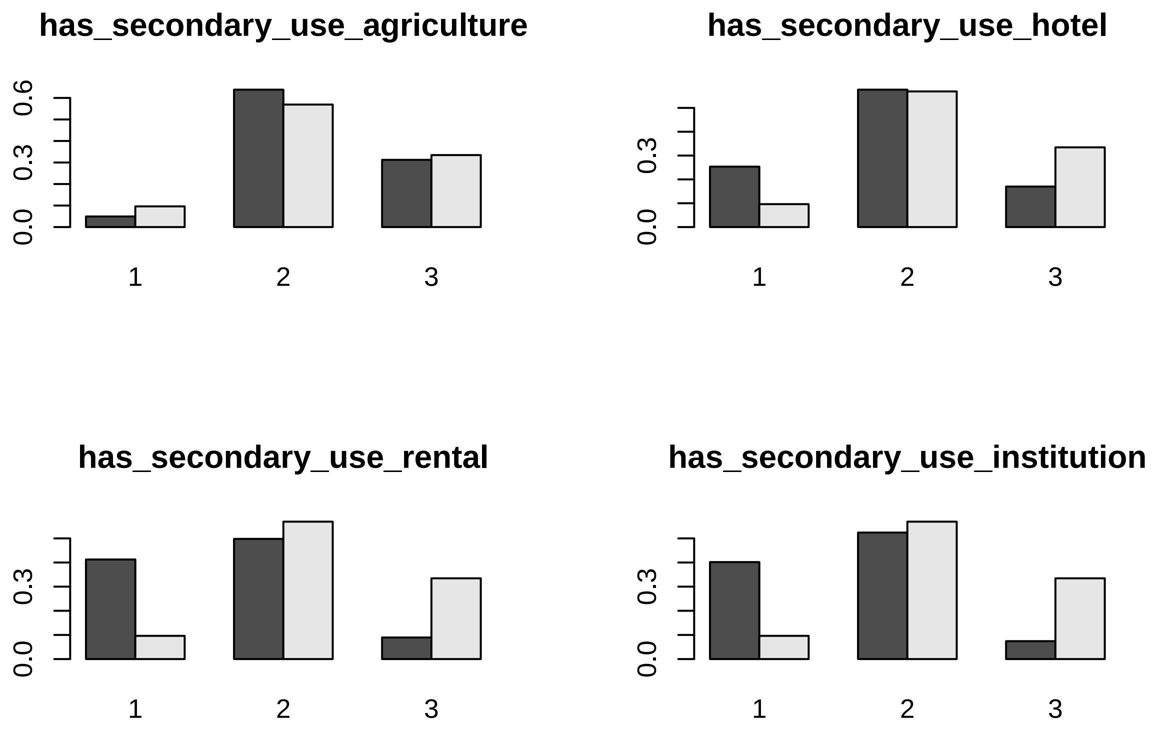
```r
plotBinLab(9,16, nameBin2, binTrain2)
```


has_superstructure_rc_non_engineere
0.0
0.3
1
2
3

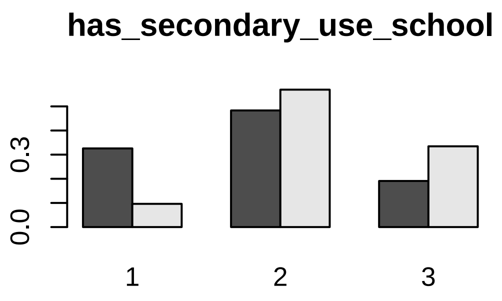

has_superstructure_rc_engineered
0.0
0.3
0.6
1
2
3

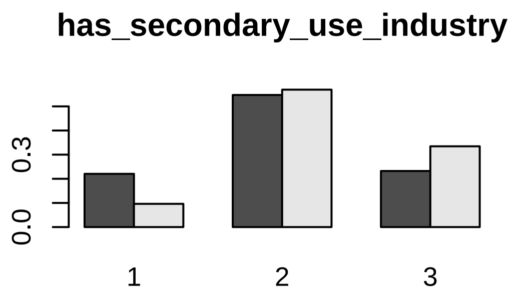

has_superstructure_other
0.0
0.3
1
2
3

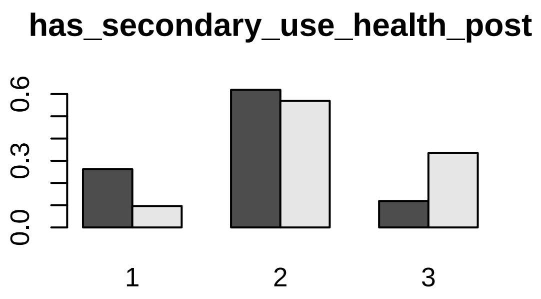

has_secondary_use
0.0
0.3
0.6
1
2
3

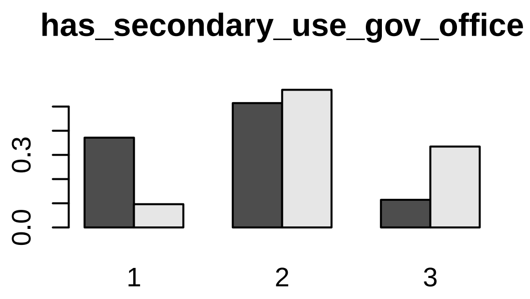


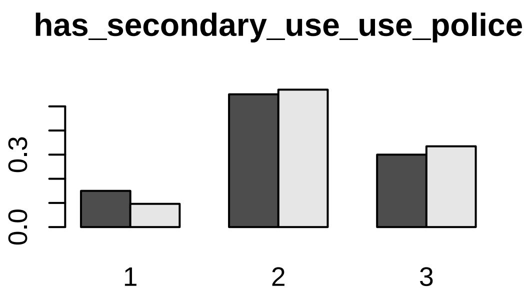
```r
plotBinLab(17,22, nameBin2, binTrain2)
```

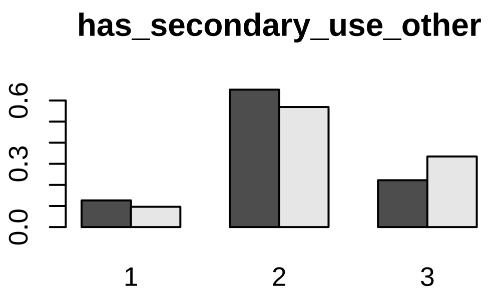

has_secondary_use_school
0.0
0.3
1
2
3


has_secondary_use_industry
0.0
0.3
1
2
3


has_secondary_use_health_post
0.0
0.3
0.6
1
2
3


has_secondary_use_gov_office
0.0
0.3
1
2
3





Key Points Which binary features had great influence on the label? Looking at the graphs, the answer is:

- has_superstructure_stone_flag
- has_superstructure_cement_mortar_stone
- has_superstructure_cement_mortar_brick
- has_superstructure_rc_non_engineered
- has_superstructure_rc_engineered
- has_secondary_use_hotel
- has_secondary_use_rental
- has_secondary_use_institution
- has_secondary_use_school
- has_secondary_use_industry
- has_secondary_use_gov_office

Finally, factor variables are analyzed. In order to determine if a factor has an effect on the label damage_grade (another factor), tables were used. The first table is a frequency table of the factor variable, showing which levels within a factor are common and which are rare. The second table is a frequency table between the factor and the label damage_grade but with the marginal (as a percentage) calculated along the columns. This helps determine the distribution of the factor among the same damage_grade level. A feature is helpful if its levels shows a different distribution of percentages for each distinct label level (i.e. each column). The third table is similar to the second, except that it calculates the marginals along the rows. This table helps determine the distribution of damage_grade among the same feature value. A helpful feature should have a different distribution of damage_grade for each distinct feature value (i.e. for each row)).

```r
factorTrain$X <- NULL
factorTrain$building_id <- NULL
```

```
facs <- names(factorTrain)
facs <- setdiff(facs, c('damage_grade'))

for (name in facs) {
    cat('NEW FEATURE, ANALYZING' , name, '\n')
    feature <- factorTrain[,name]
    print(table(feature))
    cat('\n Distribution of feature among buildings of the same damage grade (marginal percentage taken
    tab <-round(prop.table(table(feature,factorTrain$damage_grade),2)*100,0)
    print(tab)
    cat('\n Distribution of buliding with the same feature value (marginal percentage taken along rows)
    tab <-round(prop.table(table(feature,factorTrain$damage_grade),1)*100,0)
    print(tab)
    cat('\n \n')
}
```

```
## NEW FEATURE, ANALYZING land_surface_condition
## feature
##      n      o      t
##  31916   7474 195150
##
##  Distribution of feature among buildings of the same damage grade (marginal percentage taken along c
## feature  1  2  3
##       n 10 14 13
##       o  2  3  3
##       t 87 82 83
##
##  Distribution of buliding with the same feature value (marginal percentage taken along rows)
## feature  1  2  3
##       n  7 60 32
##       o  7 57 36
##       t 10 56 34
##
##
## NEW FEATURE, ANALYZING foundation_type
## feature
##      h      i      r      u      w
##   1308   9525 197246  12882  13579
##
##  Distribution of feature among buildings of the same damage grade (marginal percentage taken along c
## feature  1  2  3
##       h  1  0  1
##       i 24  3  0
##       r 43 85 95
##       u 15  6  2
##       w 17  6  2
##
##  Distribution of buliding with the same feature value (marginal percentage taken along rows)
## feature  1  2  3
##       h 24 41 35
##       i 57 41  2
##       r  5 57 38
##       u 26 60 14
##       w 29 61 10
```

```
##
##
## NEW FEATURE, ANALYZING roof_type
## feature
##      n      q      x
## 164557  55433  14550
##
##  Distribution of feature among buildings of the same damage grade (marginal percentage taken along co
## feature  1  2  3
##       n 54 72 72
##       q 16 23 27
##       x 31  5  1
##
##  Distribution of buliding with the same feature value (marginal percentage taken along rows)
## feature  1  2  3
##       n  7 58 34
##       q  6 55 38
##       x 47 48  5
##
##
## NEW FEATURE, ANALYZING ground_floor_type
## feature
##      f      m      v      x      z
## 188540    450  22131  22513    906
##
##  Distribution of feature among buildings of the same damage grade (marginal percentage taken along co
## feature  1  2  3
##       f 50 81 89
##       m  0  0  0
##       v 41  9  2
##       x  8 10 10
##       z  1  0  0
##
##  Distribution of buliding with the same feature value (marginal percentage taken along rows)
## feature  1  2  3
##       f  6 57 37
##       m 18 68 14
##       v 42 53  5
##       x  8 58 33
##       z 20 52 28
##
##
## NEW FEATURE, ANALYZING other_floor_type
## feature
##      j      q      s      x
##  35878 148704  10817  39141
##
##  Distribution of feature among buildings of the same damage grade (marginal percentage taken along co
## feature  1  2  3
##       j 35 14 12
##       q 29 66 68
##       s 22  4  1
##       x 14 16 19
##
```

```
##  Distribution of buliding with the same feature value (marginal percentage taken along rows)
## feature  1  2  3
##       j 22 51 27
##       q  4 60 36
##       s 45 49  6
##       x  8 54 38
##
##
## NEW FEATURE, ANALYZING position
## feature
##      j      o      s      t
##  11939   2095 181892  38614
##
##  Distribution of feature among buildings of the same damage grade (marginal percentage taken along co
## feature  1  2  3
##       j  7  5  4
##       o  0  1  1
##       s 79 78 76
##       t 14 15 19
##
##  Distribution of buliding with the same feature value (marginal percentage taken along rows)
## feature  1  2  3
##       j 13 60 28
##       o  5 69 26
##       s 10 57 33
##       t  8 53 39
##
##
## NEW FEATURE, ANALYZING plan_configuration
## feature
##      a      c      d      f      m      n      o      q      s      u
##    225    306 225001     21     42     37    143   5181    317   3267
##
##  Distribution of feature among buildings of the same damage grade (marginal percentage taken along co
## feature  1  2  3
##       a  0  0  0
##       c  0  0  0
##       d 93 96 97
##       f  0  0  0
##       m  0  0  0
##       n  0  0  0
##       o  0  0  0
##       q  3  2  3
##       s  0  0  0
##       u  3  2  1
##
##  Distribution of buliding with the same feature value (marginal percentage taken along rows)
## feature  1  2  3
##       a 26 61 12
##       c 27 62 11
##       d  9 57 34
##       f  0 76 24
##       m 14 76 10
##       n 14 54 32
##        o 27 58 15
##        q 14 45 41
##        s 15 66 19
##        u 21 66 13
##
##
## NEW FEATURE, ANALYZING legal_ownership_status
## feature
##      a      r      v      w
##   5000   1306 225842   2392
##
##  Distribution of feature among buildings of the same damage grade (marginal percentage taken along co
## feature  1  2  3
##        a  6  2  1
##        r  1  0  1
##        v 93 97 97
##        w  1  1  1
##
##  Distribution of buliding with the same feature value (marginal percentage taken along rows)
## feature  1  2  3
##        a 27 56 17
##        r 15 49 36
##        v  9 57 34
##        w  5 48 47
##
##
```

New Datasets

```
loadPr<- modules::use('Helpers/Load_Preprocessed.R')
train <- loadPr$loadTrain()[[1]]
trainLab <- loadPr$loadTrain()[[2]]
val <- loadPr$loadVal()[[1]]
valLab <- loadPr$loadVal()[[2]]
loadRaw<- modules::use('Helpers/Load_Raw_Data.R')
test <- loadRaw$testVal()

removeId <- function(data) {
    data$X <- NULL
    return(data)
}
removeLevelsPlan <- function(data) {
    newLevel = "other"
    data$plan_configuration <- plyr::revalue(data$plan_configuration,
                c("a"=newLevel, "c"=newLevel, "f"=newLevel,
                "m"=newLevel, "n"=newLevel, "o"=newLevel,
                "s" = newLevel))
    return(data)
}
labelToFactor <- function(data) {
    data$damage_grade <- as.factor(data$damage_grade)
    return(data)
}
```

```
saveData <- function(tr, trLab, val, valLab, test, prefix) {
    tr <- removeId(tr)
    trLab <- labelToFactor(removeId(trLab)) #labels do not have plan configuration to removeLevelsPlan
    val <- removeId(val)
    valLab <- labelToFactor(removeId(valLab)) #labels do not have plan configuration to removeLevelsPla
    test <- removeId(test) #do not remove the building_ids of the test, needed for submission
    if ("plan_configuration" %in% names(tr)) { #training set was used here but any dataset could be use
        tr <- removeLevelsPlan(tr)
        val <- removeLevelsPlan(val)
        test <- removeLevelsPlan(test)
    }
    rootDir <- 'Further_Preprocess_Analysis/Data/'
    write.csv(tr, paste(rootDir, prefix, '_train.csv', sep=''))
    write.csv(trLab, paste(rootDir, prefix, '_train_lab.csv', sep=''))
    write.csv(val, paste(rootDir, prefix, '_val.csv', sep=''))
    write.csv(valLab, paste(rootDir, prefix, '_val_lab.csv', sep=''))
    write.csv(test, paste(rootDir, prefix, '_test.csv', sep=''))
}
```

In total 3 datasets were constructed. One being the orignal dataset.

```
saveData(train, trainLab, val, valLab, test, 'original')
```

One being a filtered dataset, which removes al features except those deemed relevent by the Analysis section. For example many of the building use variables are removed (because they showed little correlation with label and research said they were unimportant).

```
featureEngineerRemove <- function(data) {
    newLevel <- "other"
    #according to analysis, combine levels that were strikingly similar
    data$foundation_type <- plyr::revalue(data$foundation_type, c("u"=newLevel, "w"=newLevel))
    data$roof_type <- plyr::revalue(data$roof_type, c("n"=newLevel, "q"=newLevel))
    return(subset(data,select= c(
        #first add all binary/categorical features that were found to be effective in the analysis stag
                    foundation_type , roof_type , ground_floor_type, other_floor_type, legal_ownership_
                   has_superstructure_stone_flag, has_superstructure_cement_mortar_stone,
                    has_superstructure_cement_mortar_brick, has_superstructure_mud_mortar_stone, has_su
                    has_superstructure_rc_engineered, has_superstructure_timber,has_secondary_use, has_
                    has_secondary_use_institution, has_secondary_use_industry,
                    plan_configuration,
        #now add all other features that were considered relevent according to research conducted on th
                    geo_level_1_id, geo_level_2_id, geo_level_3_id, count_floors_pre_eq, age,
                    area_percentage, height_percentage, land_surface_condition, count_families, buildin
}
saveData(featureEngineerRemove(train), trainLab, featureEngineerRemove(val), valLab, featureEngineerRem
```

One being the expanded dataset, which contains features extracted from the dataset that may be helpful. For example a binary feature was added for all buildings that had any form of concrete in them.

```
featureEngineerAdd <- function(data) {
    data$has_superstructure_tree =
        data$has_superstructure_bamboo | data$has_superstructure_timber
    data$has_superstructure_mortar =
        data$has_superstructure_mud_mortar_stone | data$has_superstructure_cement_mortar_stone | data$ha
    data$has_superstructure_cement =
        data$has_superstructure_cement_mortar_stone | data$has_superstructure_timber
    data$has_superstructure_brick =
        data$has_superstructure_mud_mortar_brick | data$has_superstructure_cement_mortar_brick
    data$has_superstructure_mud =
        data$has_superstructure_adobe_mud | data$has_superstructure_mud_mortar_stone | data$has_supersti
    data$has_superstructure_concrete =
        data$has_superstructure_rc_non_engineered | data$has_superstructure_rc_engineered
    data$has_superstructure_stone =
        data$has_superstructure_mud_mortar_stone | data$has_superstructure_stone_flag | data$has_superst
    return(data)
}
saveData(featureEngineerAdd(train), trainLab,
         featureEngineerAdd(val), valLab, featureEngineerAdd(test), 'expanded')
```

Models

As previously stated, 3 different training datasets were constructed, one called 'original', other called 'filtered', and the other called 'expanded'. However model creation code is identical for all 3 datasets. Here only the original dataset is loaded and analyzed. To analyze other datasets simply comment and uncomment the data selection line appropiately (the code below selects the dataset). Note that for the purposes of this report, none of code in the model testing phase is evaluated/run, due to computational considerations.

```
#currently analyzing the original dataset
loadFin<- modules::use('Helpers/Load_Final_Data.R')
train <- loadFin$original()[[1]]
trainLab <- loadFin$original()[[2]]
val <- loadFin$original()[[3]]
valLab <- loadFin$original()[[4]]
test <- loadFin$original()[[5]]
#train <- loadFin$filtered()[[1]]
#trainLab <- loadFin$filtered()[[2]]
#val <- loadFin$filtered()[[3]]
#valLab <- loadFin$filtered()[[4]]
#test <- loadFin$filtered()[[5]]
#train <- loadFin$expanded()[[1]]
#trainLab <- loadFin$expanded()[[2]]
#val <- loadFin$expanded()[[3]]
#valLab <- loadFin$expanded()[[4]]
#test <- loadFin$expanded()[[5]]
```

Some further proprocessing occurs, along with the creation of a 'Full' dataset, which joins the values and the labels.

```
#when R reads the csv, it thinks damage_grade is an integer, so convert it back to a factor
trainLab$damage_grade <- as.factor(trainLab$damage_grade)
valLab$damage_grade <- as.factor(valLab$damage_grade)

manipulate<- modules::use('Helpers/Manipulate.R')
trainFull <- manipulate$combineLab(train, trainLab)
valFull <- manipulate$combineLab(val, valLab)
```

A saving predictions helper function is now constructed.

```
#saving predictions helper function
savePredictions <- function(model) {
    preds <- cbind(test$building_id, predict(model, subset(test, select=-c(building_id))))
    colnames(preds) <- c("building_id", "damage_grade")
    write.csv(preds, 'Models/Predictions/Random_Forest.csv', row.names=FALSE)
}
trainFull$building_id <- NULL
valFull$building_id <- NULL
```

First we construct linear models. One such model is the linear discriminant analysis (LDA).

```
model.LDA <- lda(as.factor(damage_grade)~., data = trainFull)
predictionsVal <- predict(model.LDA, valFull)
accuracy <- mean(predictionsVal$class == valFull$damage_grade)
cat('Accuracy for the LDA was ', accuracy)
```

The second class of models is tree models, one such model is the random forests.

```
createRandomForest <- function(mtry_, ntree_, trainFull, valFull){
    model.RF <- randomForest(as.factor(damage_grade) ~ ., data=trainFull, ntree=ntree_, mtry=mtry_, imp
    #dput(list(model.RF$confusion, modelRF$oob.times, modelRF$, "Models/Random_Forest_Output.txt")
    cat('RESULTS FOR RANDOM FOREST, with mtry=', mtry_, 'ntree=', ntree_, '\n \n')

    predictionsVal <- predict(model.RF, valFull)
    accuracy <- mean(predictionsVal == valFull$damage_grade)
    cat('The accuracy for this random forest was',accuracy, '\n')

    cat('\n Confusion Matrix \n')
    print(model.RF$confusion)

    cat('\n Variables sorted in importance (decreasing)\n')
    imp <- as.data.frame(importance(model.RF))
    print(subset ( imp[order(-imp$MeanDecreaseGini),] ,select=MeanDecreaseGini))
}
```

To tune the parameter values, 5 different pairs of mtry and ntree values are created (this is all that computational resources permitted). Due to limited computational resources, the following code is not run here in the report.

```
mtry <- c(5,10,15,20,25)
ntree <- c(80,70,60,50,40)
cat('Try all 5 pairs, inspect the console for the model that gives the best accuracy.')
createRandomForest(mtry[1],ntree[1], trainFull, valFull)
createRandomForest(mtry[2],ntree[2], trainFull, valFull)
createRandomForest(mtry[3],ntree[3], trainFull, valFull)
createRandomForest(mtry[4],ntree[4], trainFull, valFull)
createRandomForest(mtry[5],ntree[5], trainFull, valFull)
```

The best model using the original dataset (based on accuracy) is reconstruced, but this time trained on the training and the validation set. Afterward the prediction helper function is used to store the predictions of this model.

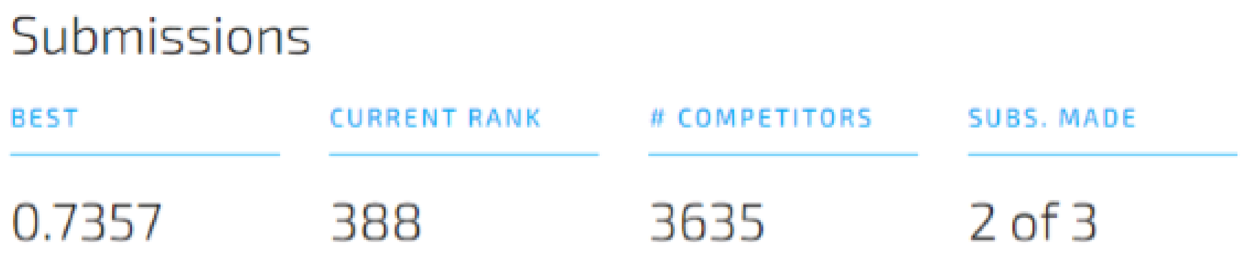
```
cat('The best model for the original dataset was the first parmeter pair')
model.RF <- randomForest(as.factor(damage_grade) ~ ., data=rbind(trainFull,valFull), ntree=ntree[2], mt
savePredictions(model.RF)
```

Results

The metric used for model evaluation was accuracy, which is calculated as the proportion of times the model predicted correctly. The accuracy was displayed for each model (in the code cell outputs). Visually inspecting, the greatest accuracy was acheived by the random forests construsted from the filtered dataset. That model is reconstructed in the code below. Again the code is not run due to the time it takes. Looking at the variable importance given from the random forest, it is clear that all of the geographic variables were helpful. Also building use as predicted was not especially helpful. Surprisingly superstructure material was not very helpful either, it was however better then building use features.

```r
loadFin<- modules::use('Helpers/Load_Final_Data.R')
train <- loadFin$filtered()[[1]]
trainLab <- loadFin$filtered()[[2]]
val <- loadFin$filtered()[[3]]
valLab <- loadFin$filtered()[[4]]
test <- loadFin$filtered()[[5]]
#when R reads the csv, it thinks damage_grade is an integer, so convert it back to a factor
trainLab$damage_grade <- as.factor(trainLab$damage_grade)
valLab$damage_grade <- as.factor(valLab$damage_grade)
manipulate<- modules::use('Helpers/Manipulate.R')
trainFull <- manipulate$combineLab(train, trainLab)
valFull <- manipulate$combineLab(val, valLab)
model.RF <- randomForest(as.factor(damage_grade) ~ ., data=rbind(trainFull,valFull), ntree=70, mtry=10,
```

The predictions of the final model on the test dataset were submitted to the driven data, the accuracy and ranking this model acheived is shown below. The model was successful because it ranked the top 11% (388th position out of over 3635) of all models created for this problem, which is a high ranking.



Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. MADE
0.7357	388	3635	2 of 3

Figure 1: Final model ranking in the world.

Conclusion

With the urban sprawl occurring throughout the world, institutions need a great infrastructure planning. One such way is to plan for the damage on buildings from earthquakes. Hence several models from two different model classes (linear and tree based) were constructed to predict effect on building damage of an earthquake the size of Gorkha.

Although the dataset was already in a format that facilitated the creation of good models, further datasets were created. One of them being a filtered dataset (with fewer features then the original) and the other being a dataset with more features then the original. The same models were developed for all three dataset, and the final model was selected based on the accuracy of the validation set. The final model was a random forest trained on the filtered datasets. The model was successful, because it had great performance (high accuracy) on the testset (model ranked in the top 11%).

This model has the potential to impact institutions around the world, helping them predict the impact of earthquake damage on buildings. In the future, modelling with greater computational resources, and using more powerful models, such as neural networks and support vector machines. Furthermore, the results of variable importance from tree models, such as random forests, could be used to remove certain unimportant features from the dataset. The limitations of this work is certainly the final accuracy, which is not significantly greater then the accuracy is a base model, that constantly predicted level 2 building damage. Another limitation of this project is the lack of cross validation, which should have been used instead of using an isolated holdout validation set.