# Earthquake Damage Prediction Analysis

Data Analytics

A project reported submitted by,

19BCE7119 - G V Datta Adithya

19BCE7414 - Polu Venkata Sai Pavan Kalyan

19BCD7137 - Kaushik Karamsetty

19BCE7778 - Shiva Kumar Jalla

Under the guidance of,

Dr. Gopikrishnan S

Associate Professor, CSE,

Vellore Institute of Technology AP

# Executive Summary

Around the world, we face numerous natural calamities, and earthquakes are one of the most prominent calamities amongst them.

This project was developed as a means of estimating the damage that can be caused due to an incoming earthquake so that people residing in the danger zone can take steps to relocate and avoid the incoming disaster.

The data that we are utilizing in this project is collected based on the damage that was suffered by the various buildings located in the calamity zone, and this project was made to analyze this data to develop a prediction model to estimate the damage that could be done to buildings for future earthquakes.

# Introduction

This project was inspired from a Driven Data publication on the Earthquake Analysis that took place in Nepal in April 2015.

The scale of this earthquake reached a magnitude of 7.8Mw according to a US Geological Survey, and its epicenter was towards the east of Gorkha.

This earthquake took the lives of 9,000 people and injured nearly 22,000.

Data has been collected on the damage suffered by various buildings, identifying the degree of damage on a scale from one to three, where three represented the greatest damage.

The purpose of the project is to develop predictive models which help in identifying buildings prone to damage in other regions in the event of another possible earthquake.

This document analyses the elements that eventually affect the degree of damage that a natural disaster, such as an earthquake, can cause.

Predictive models have been built around this information to approximate a precise solution, which is measured with the performance metric required for the problem, finally, the conclusions of the study are raised.

# Body

The data was collected through surveys by Kathmandu Living Labs and the Central Bureau of Statistics, which works under the National Planning Commission Secretariat of Nepal.

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

In order to interact and retrieve the information that we require from the data, it is important that we understand the dataset that we are working with.

Judging by how the columns are related to one another, we can check the data out through graphs, plots, and numerous other mechanisms.

## Data Preprocessing

In order to do so, we use a few pre-processing techniques:

- Data Cleaning

- Data Reduction

- Data Transformation

Starting with the feature label which corresponds to the "damage_grade", we can observe that this has three possible values according to the degree of intensity of the destruction of the buildings int eh place where the earthquake took place.

These are divided into the following,

- 1 representing low damage.

- 2 representing medium damage.

- 3 representing complete destruction.

Throughout the analysis of the dataset, it is observed the consistency in the number of occurrences of the values is not balanced.

The value of 1 is much lower than the other two remain. In addition, it is worth mentioning that this feature is an ordinal type and has a specific order associated with the degree of intensity of the earthquake.

There are 39 features that have been analyzed differentiating between numerical and categorical attributes.

In the case of numerical features:

- count_floors_pre_eq
- age
- area_percentage
- height_percentage

In this case, an evident bias was identified.

We also face the issue of having existing outlier values, which can harm the prediction process.

In this case, processes were applied that reduce the intensity of this observed bias.

In the case of the feature count_families, it was not transformed and was used to perform the Feature Engineering.

In the case of the features related to the geographical region, it was identified that these features from their concept are of the categorical type since they represent regions in Nepal.

- geolevel_1_id
- geolevel_2_id
- geolevel_3_id

The problem for this case was that for each feature there is a very high number of categorical values, so in this case it was decided to transform these features by applying the conditional probability related to the target, which generated nine new features.

The data set has the following features that are categorical, they describe, in general, types of materials used in the type of land constructions, and some of the position tasks, among others. All these encoded,

- land_surface_condition
- foundation_type
- roof_type
- ground_floor_type
- other_floor_type
- position
- plan_configuration
- legal_ownership_status

We can observe that there is a coincidence between specific values of some of these features regarding the distribution in relation to the feature target.

This indicates that there is a set of common characteristics in various buildings.

In addition to the distribution in relation to the target, it is very similar in the values mentioned.

These following features are binary in the data, in this case it was determined to keep them as is, because they are binary valued:

- has_superstructure_adobe_mud
- has_superstructure_mud_mortar_stone
- has_superstructure_stone_flag
- has_superstructure_cement_mortar_stone
- has_superstructure_mud_mortar_brick
- has_superstructure_cement_mortar_brick
- has_superstructure_timber
- has_superstructure_adobe_bamboo

- has_superstructure_rc_non_engineered

- has_superstructure_rc_engineered

- has_superstructure_other

The same as the features before, the ones following are binary, in this case it was determined to keep them as is, except for the feature “has_secondary_use” which was redundant and was removed.

- has_secondary_use_agriculture

- has_secondary_use_hotel

- has_secondary_use_rental

- has_secondary_use_institution

- has_secondary_use_school

- has_secondary_use_industry

- has_secondary_use_health_post

- has_secondary_use_gov_office

- has_secondary_use_use_police

- has_secondary_use_other

## Prediction Models

According to the analytics that we have gathered, we then proceed to work on ML models that can predict the results.

Finally, apply the suitable algorithms that are required for the Data Set and Predict the Result.

The two algorithms that we are going to utilize for the prediction model are,

### 1. Decision Tree Algorithm

The Decision Tree Algorithm is a general, predictive modelling tool with applications spanning several different areas.

In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on various conditions.

It is one of the most widely used and practical methods for supervised learning.

In this analysis, we have used the randomforest package to perform predictions.

### 2. Naïve Bayes Algorithm

The Naive Bayes Algorithm is a one of the popular classification machine learning algorithms that helps to classify the data based upon the conditional probability values computation.

It implements the Bayes theorem for the computation and used class levels represented as feature values or vectors of predictors for classification.

Naive Bayes Algorithm is a fast algorithm for classification problems.

## Conclusion

Going through the various observations that were provided in the dataset, it became very evident that there was a pattern of repetition in the details for a few buildings.

Studying and parsing through the datasets, the patterns grew more evident and as a result, we decided that it would be a good idea to work on forming a prediction model using effective prediction algorithms such as the Decision Tree Algorithm and the Naive Bayes Algorithm.

The models were developed to find accuracy from test datasets.

Overall, the prediction model was a success, and this project can be used as a means to estimate the impact of any future earthquake, to assess the damage that could be caused, and take precautionary measures further ahead of time.

## References

This project is currently being hosted at,

https://github.com/dat-adi/earthquake-analysis

Dataset for the earthquake analysis was taken from Kaggle,

https://www.kaggle.com/mullerismail/richters-predictor-modeling-earthquake-damage/activity

The link to the competition conducted for the analysis of this data,

https://www.drivendata.org/competitions/57/nepal-earthquake/

An article on how to predict the damage to a building in Python,

https://medium.com/swlh/predicting-damage-to-building-due-to-earthquake-using-data-science-e85a62adc0c0

# Appendices

## Appendix A

**G V Datta Adithya**

The team leader for this project.

Decided on the problem statement and the methodology to solve it, allocating work according to the role of each member in the team.

Contributed towards the documentation for the project, whilst also maintaining a check of the quality of the code that was being written in R.

Assisted in the data pre-processing section of the codebase and worked with test cases for the development of the prediction engine.

**Polu Venkata Sai Pavan Kalyan**

Actively worked on retrieval of resources and scoured for clean datasets that could be used for the problem statement.

Worked directly on the codebase and implemented the data cleaning and visualization steps.

Assisted with the documentation by contributing to the analysis of the output.

**Kaushik Karamsetty**

Actively worked on the prediction algorithms and assisted in the development of the prediction engine for the given problem statement.

Worked extensively on making the software usable and functional.

**Shiva Kumar Jalla**

Took a well refined glace at the discussions in the group.

## Appendix B

The following is the code used in the project and has been divided into sections for a better understanding of the codebase for the data analyst to utilize.

```
# Setting up the workspace

#--------------------------------------------------------------

# Setting up the working directory

setwd("~/DA/earthquake-analysis/src/datasets")

getwd()


# Reading data from the csv into the Data variable

Data <-

  read.csv("earthquake_dataset.csv",

           stringsAsFactors = FALSE,

           header = T)

Data


# Cleaning up NA values from the data

sum(is.na(Data))

Data <- na.omit(Data)

sum(is.na(Data))


# Viewing and checking the data
```

```
table(Data$age)

View(Data)

summary(Data)

str(Data)

colnames(Data)


#-------------------------------------------------------------

# Outlier Detection


# Importing the outliers detection library

library(outliers)


# Firstly, we will need to store outliers in a vector

outliers <- boxplot(Data$age)$out

outliers


# Then, we need to find out the rows where the outliers exist

Data[which(Data$age %in% outliers),]


# Now you can remove the rows containing the outliers

Data1 <- Data[-which(Data$age %in% outliers), ]


# If you check now with boxplot, you will notice that the outliers are gone

boxplot(x = Data1$age)


#-----------------------------------------------------------------


# We store the outliers in a vector

outliers <- boxplot(Data1$area_percentage)$out
```

```
# Then, we find where the outliers exist
Data1[which(Data1$area_percentage %in% outliers),]

# Now, we can remove the rows containing the outliers based on area percentage
Data2 <- Data1[-which(Data1$area_percentage %in% outliers),]

# If you check now with boxplot, you will notice that the outliers are gone
boxplot(x = Data2$area_percentage)

#------------------------------------------------------------------------

# Identifying outliers and forming a boxplot from the result
outliers <- boxplot(Data2$height_percentage)$out

# Then, we find where the outliers exist
Data2[which(Data2$height_percentage %in% outliers),]

# Now, we can remove the rows containing the outliers baseed on height percentage
Data3 <- Data2[-which(Data2$height_percentage %in% outliers),]

# Outliers have been removed at this point
boxplot(x = Data3$height_percentage)

#------------------------------------------------------------------

# Land check
# Allocating values to the Data based on the properties of the field
Data3$land_surface_condition[Data3$land_surface_condition == 'n'] <- 1
```

```
Data3$land_surface_condition[Data3$land_surface_condition == 'o'] <- 2

Data3$land_surface_condition[Data3$land_surface_condition == 't'] <- 3

Data3$land_surface_condition


land <- table(Data3$land_surface_condition)

land


# Roof check

# Allocating values to the Data based on the properties of the field

Data3$roof_type[Data3$roof_type == 'n'] <- 0

Data3$roof_type[Data3$roof_type == 'q'] <- 1

Data3$roof_type[Data3$roof_type == 'x'] <- 2

Data3$roof_type


roof <- table(Data3$roof_type)

roof


# Foundation check

# Allocating values to the Data based on the properties of the field

Data3$foundation_type[Data3$foundation_type == 'h'] <- 1

Data3$foundation_type[Data3$foundation_type == 'i'] <- 2

Data3$foundation_type[Data3$foundation_type == 'r'] <- 3

Data3$foundation_type[Data3$foundation_type == 'u'] <- 4

Data3$foundation_type[Data3$foundation_type == 'w'] <- 5

Data3$foundation_type


foundation <- table(Data3$foundation_type)

foundation
```

```r
# Ground Floor check
# Allocating values to the Data based on the properties of the field
Data3$ground_floor_type[Data3$ground_floor_type == 'f'] <- 1
Data3$ground_floor_type[Data3$ground_floor_type == 'm'] <- 2
Data3$ground_floor_type[Data3$ground_floor_type == 'v'] <- 3
Data3$ground_floor_type[Data3$ground_floor_type == 'x'] <- 4
Data3$ground_floor_type[Data3$ground_floor_type == 'z'] <- 5
Data3$ground_floor_type

ground <- table(Data3$ground_floor_type)
ground

# Other Floor check
# Allocating values to the Data based on the properties of the field
Data3$other_floor_type[Data3$other_floor_type == 'j'] <- 1
Data3$other_floor_type[Data3$other_floor_type == 'q'] <- 2
Data3$other_floor_type[Data3$other_floor_type == 's'] <- 3
Data3$other_floor_type[Data3$other_floor_type == 'x'] <- 4
otherfloor <- table(Data3$other_floor_type)
otherfloor

# Position check
# Allocating values to the Data based on the properties of the field
Data3$position[Data3$position == 'j'] <- 1
Data3$position[Data3$position == 'o'] <- 2
Data3$position[Data3$position == 's'] <- 3
Data3$position[Data3$position == 't'] <- 4
position <- table(Data3$position)
position
```

```r
# Plan configuration check
# Allocating values to the Data based on the properties of the field
Data3$plan_configuration[Data3$plan_configuration == 'a'] <- 1
Data3$plan_configuration[Data3$plan_configuration == 'c'] <- 2
Data3$plan_configuration[Data3$plan_configuration == 'd'] <- 3
Data3$plan_configuration[Data3$plan_configuration == 'f'] <- 4
Data3$plan_configuration[Data3$plan_configuration == 'm'] <- 5
Data3$plan_configuration[Data3$plan_configuration == 'n'] <- 6
Data3$plan_configuration[Data3$plan_configuration == 'o'] <- 7
Data3$plan_configuration[Data3$plan_configuration == 'q'] <- 8
Data3$plan_configuration[Data3$plan_configuration == 's'] <- 9
Data3$plan_configuration[Data3$plan_configuration == 'u'] <- 10
plan <- table(Data3$plan_configuration)
plan

# Legal Ownership status check
# Allocating values to the Data based on the properties of the field
Data3$legal_ownership_status[Data3$legal_ownership_status == 'a'] <- 1
Data3$legal_ownership_status[Data3$legal_ownership_status == 'r'] <- 2
Data3$legal_ownership_status[Data3$legal_ownership_status == 'v'] <- 3
Data3$legal_ownership_status[Data3$legal_ownership_status == 'w'] <- 4

legal <- table(Data3$legal_ownership_status)
legal

# Finding a mean from the given data
mean <- mean(Data3$age, na.rm = TRUE)
mean
```

```
sum(is.na(Data3))

# Allocating to the age column
Data3$age[Data3$age == '0'] <- mean
Data3$age

age <- table(Data3$age)
age

#------------------Data Visualization------------------------------

# Importing the plot library
library(ggplot2)

# Set up factors.
Data3$damage_grade <- as.factor(Data3$damage_grade) #CONVERT TO FACTORS
Data3$land_surface_condition <- as.factor(Data3$land_surface_condition) #CONVERT TO FACTORS
ggplot(Data3, aes(x = damage_grade)) + geom_bar()

#Individual performance of damage_grade which is categorical column
graph1 <- ggplot(Data3, aes(x = damage_grade))
graph1 + geom_bar(fill = "blue") + geom_text(stat = 'count', aes(label = ..count..))

#For Continuous column age
graph2 <- ggplot(Data3, aes(x = age))
graph2 + geom_dotplot(dotsize = 0.5)

#for continuous area percentage
graph3 <- ggplot(Data3, aes(x = area_percentage))
```

```
graph3 + geom_dotplot(dotsize = 0.5)

png(file = "areaplot")

dev.off()


# Identification of the the Disribution of each attribute

graph4<-ggplot(Data3,aes(x=damage_grade))

graph4 +
geom_qq(mapping=NULL,data=Data3,geom="point",position="identity",na.rm=TRUE,distribution=stats::
qnorm,dparams =TRUE,show.legend=NA,inherit.aes=TRUE)

png = (file="damage_grade_qqplot")

dev.off()


# For categorical column Lan_surface_condition

graph4 <- ggplot(Data3, aes(x = land_surface_condition))

graph4 + geom_density(fill = "#FFBCDE")

png(file = "land disribution plot")

dev.off()


# Relation between two attributes

# For 2 Continous Columns Age and Area Percentage

graph5 <- ggplot(Data3, aes(x = age, y = area_percentage))

graph5 + geom_point(size = 1, shape = 22, color = "blue")

png(file = "scatter plot1")

dev.off()


# violin plot

# age and damage grade

graph6 <- ggplot(Data3, aes(x = age, y = damage_grade))

graph6 + geom_violin(color = "green", fill = "pink")
```

```
png(file = "violin plot1")

dev.off()


# violin plot

# area and damage grade

graph8 <- ggplot(Data3, aes(x = area_percentage, y = damage_grade))

graph8 + geom_violin(color = "green", fill = "pink")

png(file = "violin plot 2")

dev.off()


# violin plot

# height and damage grade

graph9 <- ggplot(Data3, aes(x = height_percentage, y = damage_grade))

graph9 + geom_violin(color = "green", fill = "yellow")

png(file = "violin plot3")

dev.off()



# Jitterplot

# for land suraface condition and damage grade

graph10 <- ggplot(Data3, aes(x = land_surface_condition, y = damage_grade)) + geom_jitter(position =
position_jitter(0.2))

graph10

png(file = "jitterplot")

dev.off()


#jitterplot

# for land suraface condition and damage grade
```

```
u <- ggplot(Data3, aes(x = land_surface_condition, y = damage_grade)) + geom_jitter(position =
position_jitter(0.2))

u

png(file = "jitterplot")

dev.off()

n


#-------------------- Applying ML Algorithms -----------------

#------------------- Decision Trees -------------------------------------------

# Importing libraries for the ML Algorithms

library(caTools)

library(caret)

library(party)

library(randomForest)

library(e1071)


# Cleaning and setting data

Data3$damage <- factor(Data3$damage_grade)

Data3$damage <- Data3$damage_grade1


# Partitioning Data into training and Testing

set.seed(1234)

pd <- sample(2, nrow(Data3), replace = TRUE, prob = c(0.8, 0.2))

train <- Data3[pd == 1, ]

validate <- Data3[pd == 2, ]

library(party)


# Creating a tree based on the data

tree <- ctree(
```

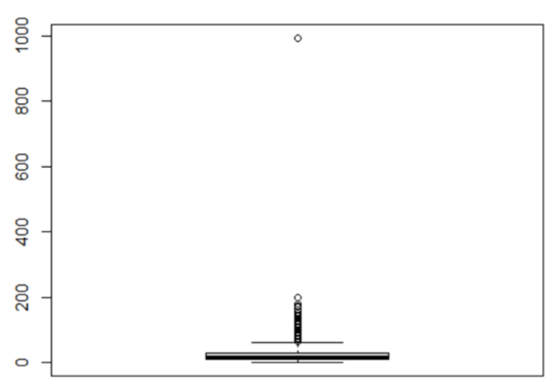
```
  damage_grade ~ age +
    area_percentage +
    height_percentage +
    count_floors_pre_eq +
    has_superstructure_adobe_mud + has_superstructure_mud_mortar_stone +
    has_superstructure_stone_flag +
    has_superstructure_cement_mortar_stone +
    has_superstructure_mud_mortar_brick +
    has_superstructure_cement_mortar_brick +
    has_superstructure_timber +
    has_superstructure_bamboo +
    has_superstructure_rc_non_engineered +
    has_superstructure_rc_engineered +
    has_superstructure_other +
    count_families +
    has_secondary_use +
    has_secondary_use_agriculture +
    has_secondary_use_hotel +
    has_secondary_use_rental +
    has_secondary_use_institution +
    has_secondary_use_school +
    has_secondary_use_industry +
    has_secondary_use_health_post +
    has_secondary_use_gov_office +
    has_secondary_use_use_police +
    has_secondary_use_other,
  data = train,
  controls = ctree_control(mincriterion = 0.9, minsplit = 20000)
)
```

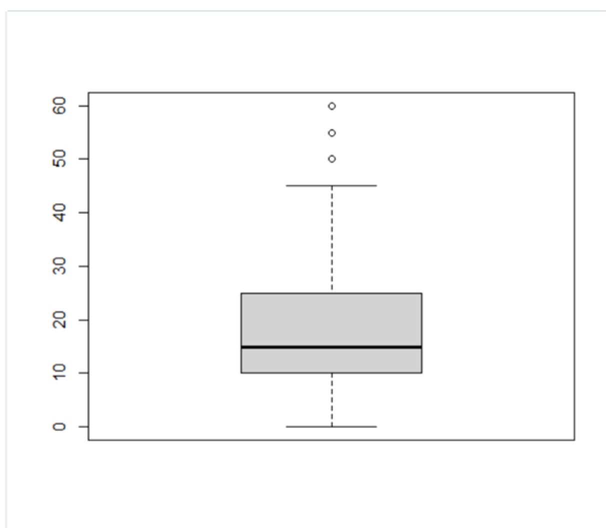
```
tree

plot(tree)
predict(tree, validate)

tab <- table(predict(tree), train$damage)
print(tab)

accuracy <- 1 - sum(diag(tab)) / sum(tab)
print(accuracy * 10)

# ------------------- Validate set---------------------
# Testing predictions and providing the accuracy report
testPred <- predict(tree, newdata = validate)
tab1 <- table(testPred, validate$damage)
print(tab1)

accuracy1 <- 1 - (sum(diag(tab)) / sum(tab))
print('Accuracy:')
print(accuracy1 * 100)

length(testPred)
testPred <- as.numeric(testPred)
building_id = testPred[1:45965]
building_id

submit1 = data.frame(building_id, testPred)
write.csv(submit1, "submit1.csv", row.names = FALSE)
data2 <- read.csv("submit1.csv")
```

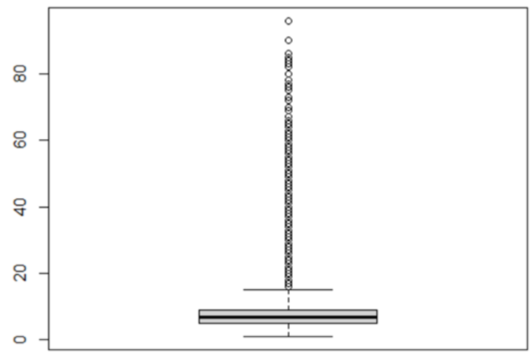
```
data2

data2$testPred <-
  cut(data2$testPred,
      seq(0, 3, 1),
      right = FALSE,
      labels = c(1:3))
data2$testPred
graph1 <- ggplot(data2, aes(x = testPred))
graph1 + geom_bar(fill = "blue") + geom_text(stat = 'count', aes(label = ..count..))

# -------------------------- Naive Bayes ------------------------------------
library(e1071)
library(caTools)
library(caret)

split <- sample.split(Data3, SplitRatio = 0.7)
train_cl <- subset(Data3, split == "TRUE")
test_cl <- subset(Data3, split == "FALSE")
set.seed(1234)  # Setting Seed
classifier_cl <- naiveBayes(damage_grade ~ ., data = train_cl)
classifier_cl
y_pred <- predict(classifier_cl, newdata = test_cl)
y_pred

# ----------------------- Finding Accuracy ------------------------------------
tab1 <- table(y_pred, test_cl$damage_grade)
print(tab1)
```

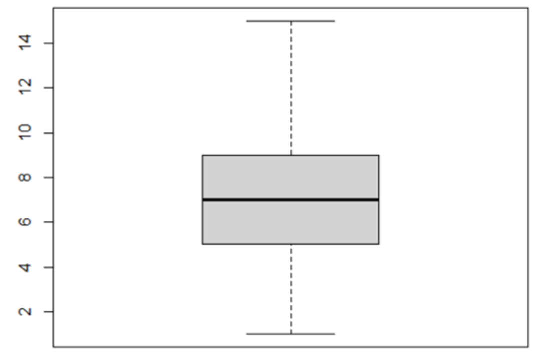
```
accuracy1 <- 1 - (sum(diag(tab1)) / sum(tab1))
print('Accuracy:')
print(accuracy1 * 100)

length(y_pred)
building_id = testPred[1:69186]
building_id

submit = data.frame(building_id, y_pred)

write.csv(submit, "submit2.csv", row.names = FALSE)
data3 <- read.csv("submit2.csv")

data3
graph1 <- ggplot(data3, aes(x = y_pred))
graph1 + geom_bar(fill = "blue") + geom_text(stat = 'count', aes(label = ..count..))
```

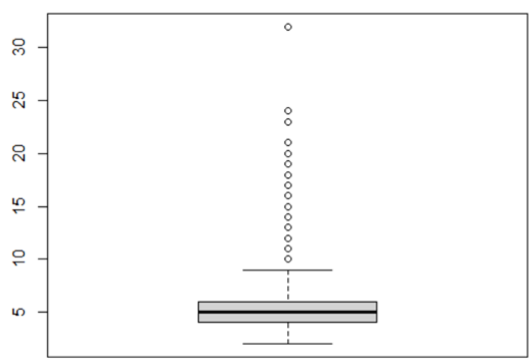
## Snapshots of the outputs

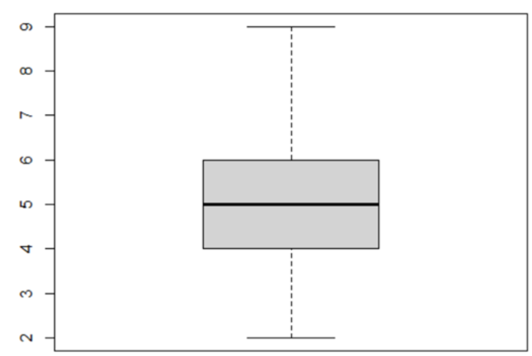


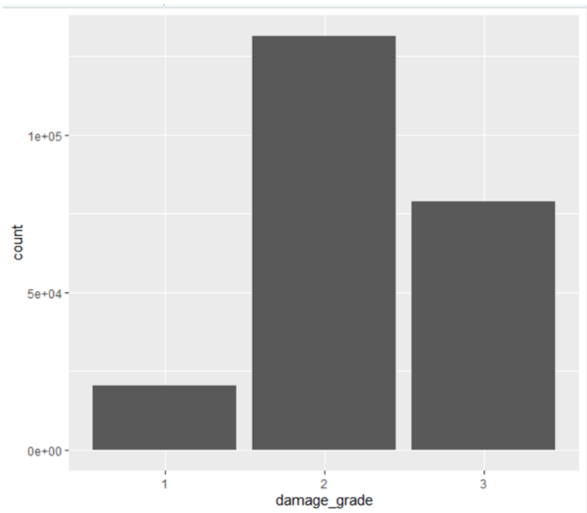



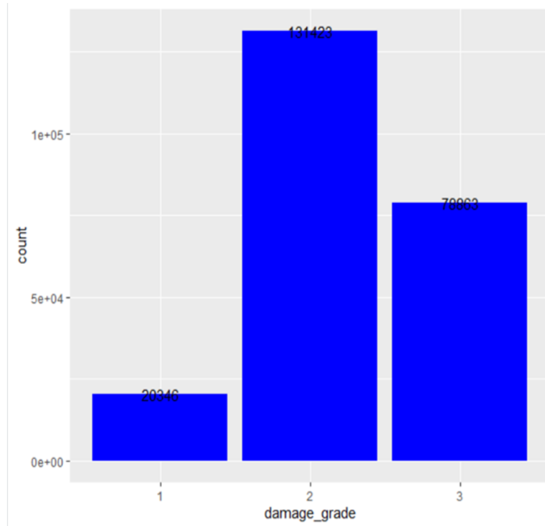

0
20
40
60
80

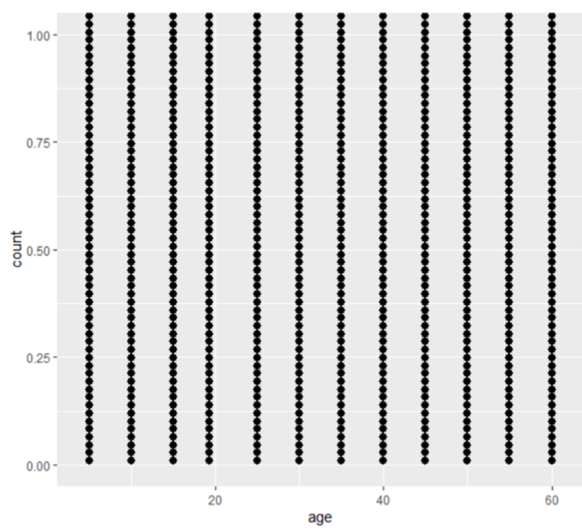

2
4
6
8
10
12
14

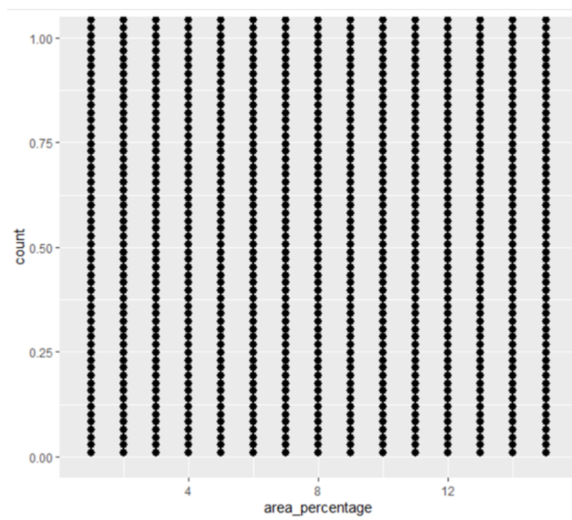

5
10
15
20
25
30

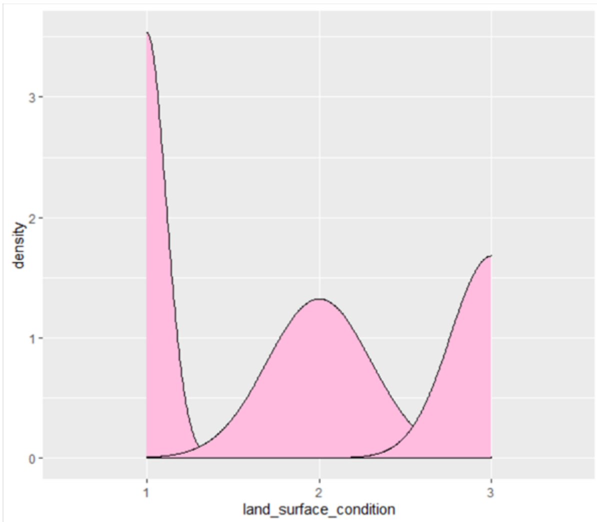

2
3
4
5
6
7
8
9


1e+05
5e+04
0e+00
count
1
2
3
damage_grade

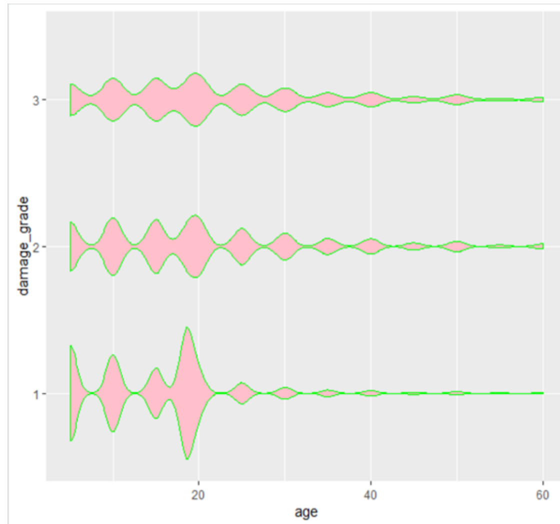

131423
78863
20346
1e+05
5e+04
0e+00
count
1
2
3
damage_grade


1.00
0.75
0.50
0.25
0.00
count
20
40
60
age

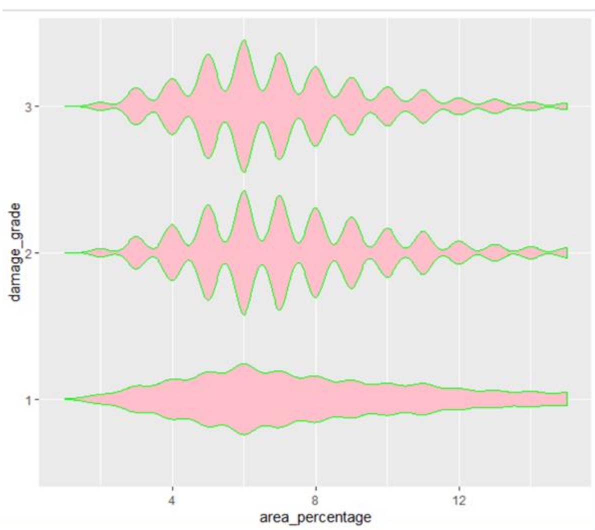

1.00
0.75
0.50
0.25
0.00
count
4
8
12
area_percentage



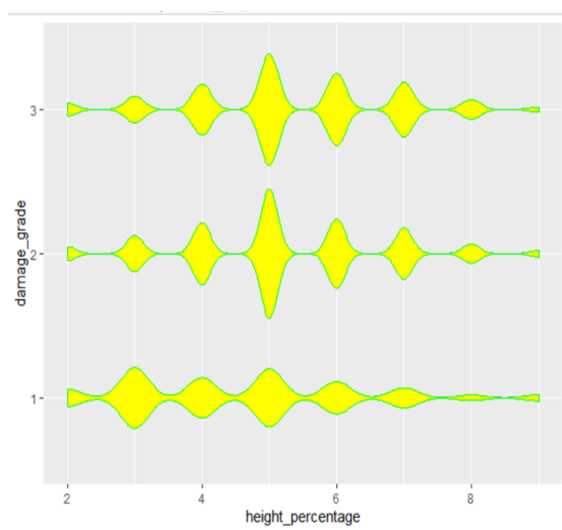
density
land_surface_condition
0
1
2
3
1
2
3

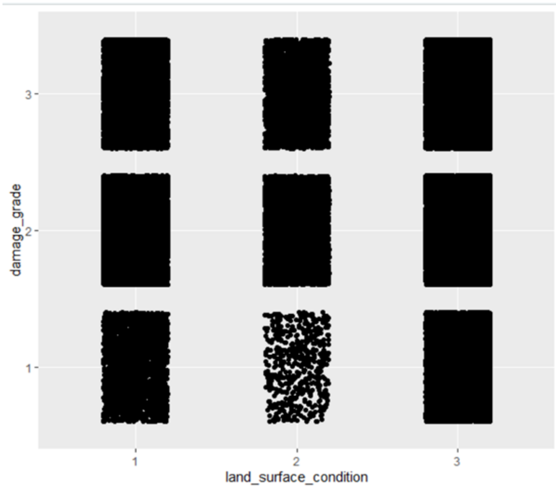
damage_grade
age
1
2
3
20
40
60

damage_grade
area_percentage
1
2
3
4
8
12

damage_grade
height_percentage
1
2
3
2
4
6
8




PREDICTION AND ACCURACY OUTPUTS

1. Decision Tree Algorithm

```
> testPred <- predict(tree, newdata = validate)
> tab1 <- table(testPred, validate$damage)
> print(tab1)

testPred     1     2     3
       1   213   124     3
       2  3760 24597 13838
       3    52  1550  1828
> accuracy1 <- 1 - (sum(diag(tab)) / sum(tab))
> print('Accuracy:')
[1] "Accuracy:"
> print(accuracy1 * 100)
[1] 42.34541
```

## 2. Naïve Bayes Algorithm

```
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
         1          2          3 
0.08845063 0.56960841 0.34194096 

Conditional probabilities:
   building_id
Y       [,1]     [,2]
  1 528510.9 303352.6
  2 524985.9 304760.8
  3 527450.2 304163.2

   geo_level_1_id
Y       [,1]     [,2]
```

```
  3 716.8311 407.9166

   geo_level_3_id
Y       [,1]     [,2]
  1 6299.175 3749.124
  2 6233.227 3662.836
  3 6325.302 3603.476

   count_floors_pre_eq
Y       [,1]      [,2]
  1 1.697619 0.6077631
  2 2.053708 0.5946307
  3 2.153827 0.6298165

   age
Y       [,1]      [,2]
```

```
> y_pred <- predict(classifier_cl, newdata = test_cl)
> y_pred
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [31] 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [61] 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 2 3 3 3 3 1 1 3 1 3 3 3 3 1 3
 [91] 3 3 3 3 3 1 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3
[121] 3 3 3 3 3 1 3 3 3 2 3 3 3 3 3 3 3 3 3 3 1 2 3 3 3 1 3 1 3 3
[151] 1 3 3 3 3 3 3 3 3 3 3 3 1 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1
[181] 3 3 2 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3
[211] 3 3 1 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3
[241] 3 3 3 3 3 3 1 3 1 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3
[271] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3
[301] 3 1 1 3 3 3 3 1 1 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 1 3 1
[331] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 3
[361] 3 3 3 3 1 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[391] 3 3 3 3 3 3 3 1 3 3 3 3 1 1 3 3 3 1 3 1 1 3 1 3 3 1 3 1 3 3
[421] 3 3 3 3 2 3 3 1 3 3 1 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 1 1 3 3
[451] 3 1 3 3 1 3 3 3 3 3 3 2 3 3 3 1 3 3 1 3 3 3 2 3 3 3 3 3 3 1
[481] 1 3 3 3 3 3 2 3 2 3 3 3 1 1 2 3 3 3 3 1 3 3 3 2 3 3 3 1 3 3
[511] 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 1 3 3 3 3 3 3 3 3 2
[541] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3
[571] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 1 3 3 3 3 1 3 3 3
[601] 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[631] 3 3 3 1 1 3 3 3 3 3 3 3 3 2 1 3 3 1 3 3 3 3 3 1 2 3 3 3 3 3
[661] 3 1 1 3 3 3 3 1 1 3 1 3 1 3 3 1 3 3 3 3 3 3 1 1 3 2 3 3 3 3
[691] 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[721] 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3
[751] 3 3 3 3 2 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[781] 3 3 1 3 3 3 3 3 3 1 1 3 3 3 3 2 3 3 3 3 1 3 3 3 3 3 3 3 3 3
[811] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 1 3 3
[841] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 1 3 3 3 3 3 3 1
[871] 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3
```

```
> tab1 <- table(y_pred, test_cl$damage_grade)
> print(tab1)

y_pred     1     2     3
     1  2661  4449   927
     2   147  1297   459
     3  3258 33716 22272
>
> accuracy1 <- 1 - (sum(diag(tab1)) / sum(tab1))
> print('Accuracy:')
[1] "Accuracy:"
> print(accuracy1 * 100)
[1] 62.08771
>
> length(y_pred)
[1] 69186
```