INTRODUCTION:

When two slabs of the ground abruptly move past one another, an earthquake occurs. The problem or faulty plane is the area where they slip. The hypocenter is the point beneath the land crust where the earthquake begins, while the epicentre is the location directly above it on the earth's surface. One such catastrophic event happened on April 25, 2015, at 11:56 a.m. Nepal Standard Time, the Gorkha earthquake struck with a magnitude of 7.8M. Its epicentre was in Barpak, Gorkha, east of Gorkha District, and its hypocenter was at a depth of around 8km. It is the deadliest natural disaster to hit Nepal since the earthquake in Nepal-Bihar in 1934. The earthquake generated an avalanche on Mount Everest, killing 21 people and making April 25, 2015 the deadliest day in the history of the world.

Nepal is an Asian country located on the Himalayas in the north ranges' southern slopes. It is a landlocked country bordered on the east, south, and west by India, and on the north by the Tibetan Autonomous Region of China. Its domain stretches from east to west for about 500 miles (800 kilometres) and from north to south for 90 to 150 miles. Kathmandu is the capital.

Nepal has some of the world's most rugged and severe alpine terrain. Mountains cover over 75 percent of the country. So another massive avalanche was generated by the earthquake in the Solang valley, with 250 people reported missing. Hundreds and thousands of Nepalese were forced to flee their homes, with entire villages levelled across the country. UNESCO World Heritage Sites in the Kathmandu Valley, including the Kathmandu Durbar Square, Patan Durbar Square, Bhaktapur Durbar Square, Changu Narayan Temple, Boudhanath stupa, and Swayambhunath Stupa, have had centuries-old structures demolished.

OBJECTIVE:

Millions of earthquakes occur each year all around the world. Since few earthquakes cause damage, they do produce a wealth of data that helps seismologists and engineers better understand the distribution, frequency, and severity of seismic hazards across the country. Seismograph networks provide critical earthquake parameter and waveform data in real time.

Lets devise a model that will assist constructing and related employees in constructing structures that will be able to resist future mega-quakes. Based on data gathered from prior earthquakes and the damages they produced, damage levels for a building can be calculated when a similar earthquake strikes again. Our objective is to anticipate the extent of damage to facilities caused by the 2015 Gorkha earthquake based on factors such as building location and structure.

Predictions of this nature have several advantages, including the ability to take preventative steps and the ability to anticipate the future.

- Greater catastrophe preparedness, such as developing a better building plan based on the location
- Prioritize renovation of old/damaged structures.
- Geographic classification of the location where the structure is located.
- Aids in the construction of building structures in any location.
- During construction, approving the number of storeys**.**

DATA:

Data path:

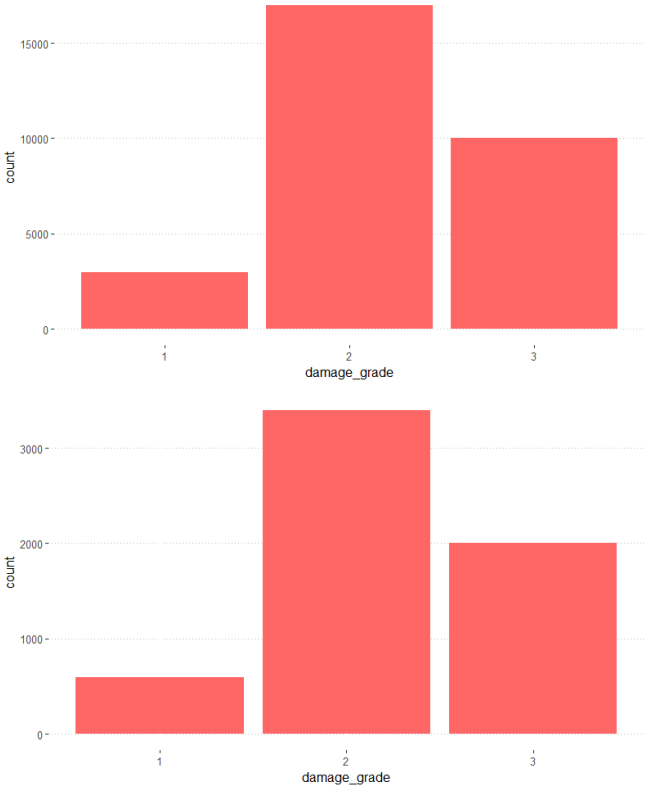
https://github.com/sushil-ds/Project-UOE.git

The majority of the data in the collection pertains to the structure of the buildings as well as their legal ownership. Each record in the information describes a particular building in the Gorkha earthquake-affected region. This dataset has 200k observations and 39 columns, with the building id column serving as a unique and random identification. The Data preparation section covers the remaining 38 characteristics.

Kathmandu Living Labs and the Central Bureau of Statistics, which is part of the National Planning Commission Secretariat, gathered the data through household surveys utilising mobile techno**logies.Below are the feature column descriptions.**

- 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.**



ANALYSIS AND RESULT:

The primary goal is to estimate the ordinal variable "damage grade." This column displays the earthquake's impact on the damage grade. Damage measurement is divided into three categories: 1: Minimal damage 2: Amount of harm is moderate. 3: Destroyed to a large extent or entirely.

Based on the data, it appears that there may be a link between characteristics and targets. Let's come up with a few questions and see if we can prove them with the use of a hypothesis, which will help us develop a better model.

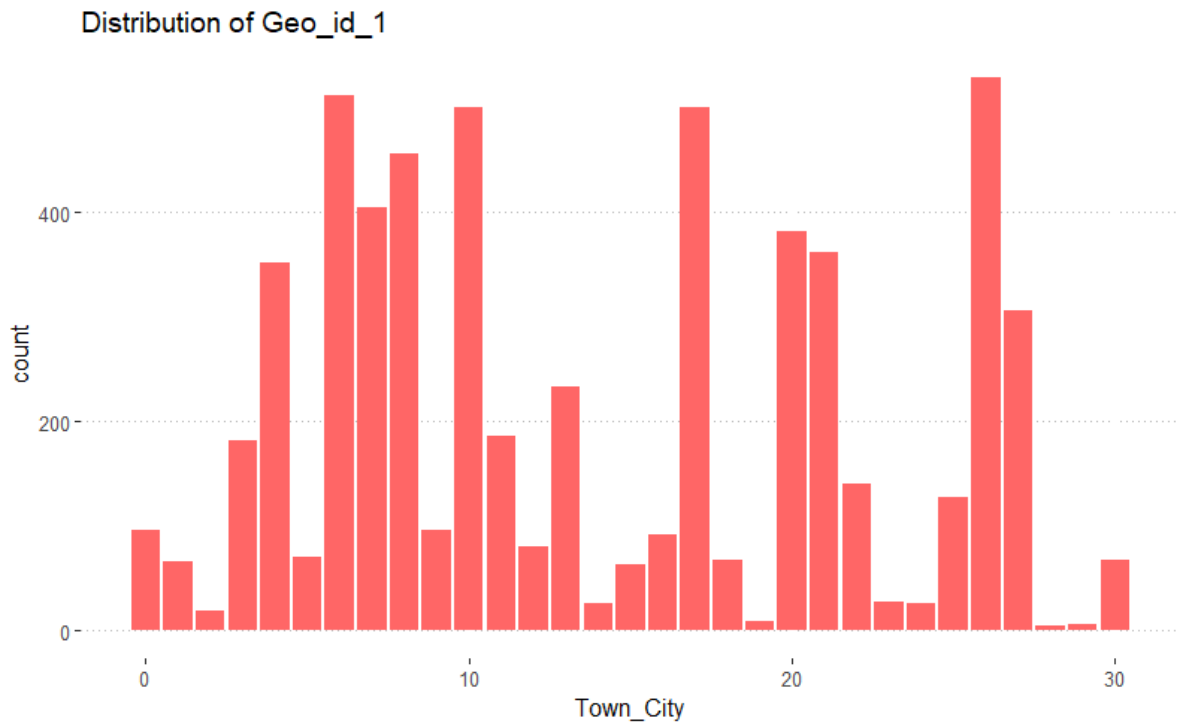
1.What are the sites that have an influence on the building's damage level?

2.What is the minimum age or age group of a severely damaged building?

3.What portion of families live in severely damaged structures?

4.Does the percentage of area have an effect on extreme damage?

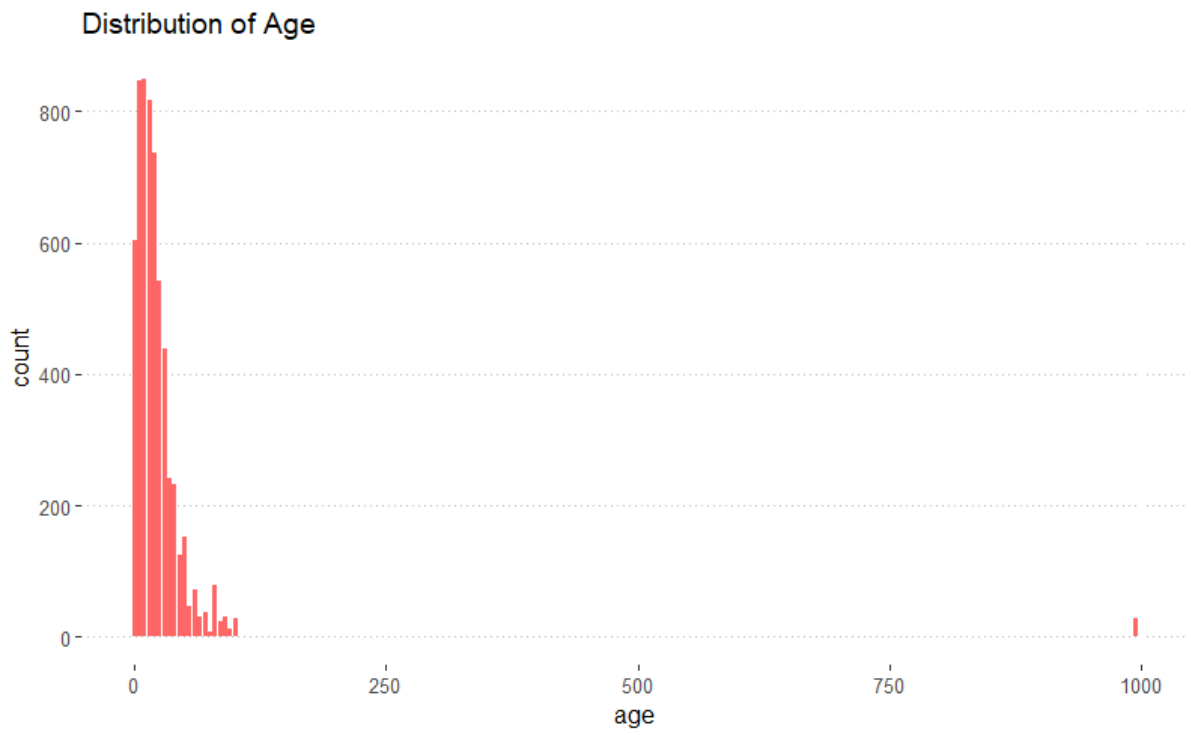
Towns/Cities, Villages, and Streets are all possible geo central ids. This leads to the first hypothesis, which asks if geolocation has an influence on damage grade. If yes, what are the ids that have an influence on severity? To minimise cardinality, consider this characteristic as a number value. Plot all three geo level ids against the damage grades in a distribution plot.



From figure its observed that the geo level ids have some sort of relation with damage grade. We can observe that the distribution of geo level 2 id and geo level 3 id is quite uniform.

2.We have structures that are roughly 995 years old, which appears to be a data collecting mistake at first sight. It was then discovered that buildings in Nepal may be that ancient after conducting a short Google search. We still have to deal with the outliers, though.

It is also self-evident that the age of a structure influences its harshness. However, we must determine the minimum age of any structure that falls into this mother nature's abyss. Plot a histogram to find out the relation between age and damage grade.

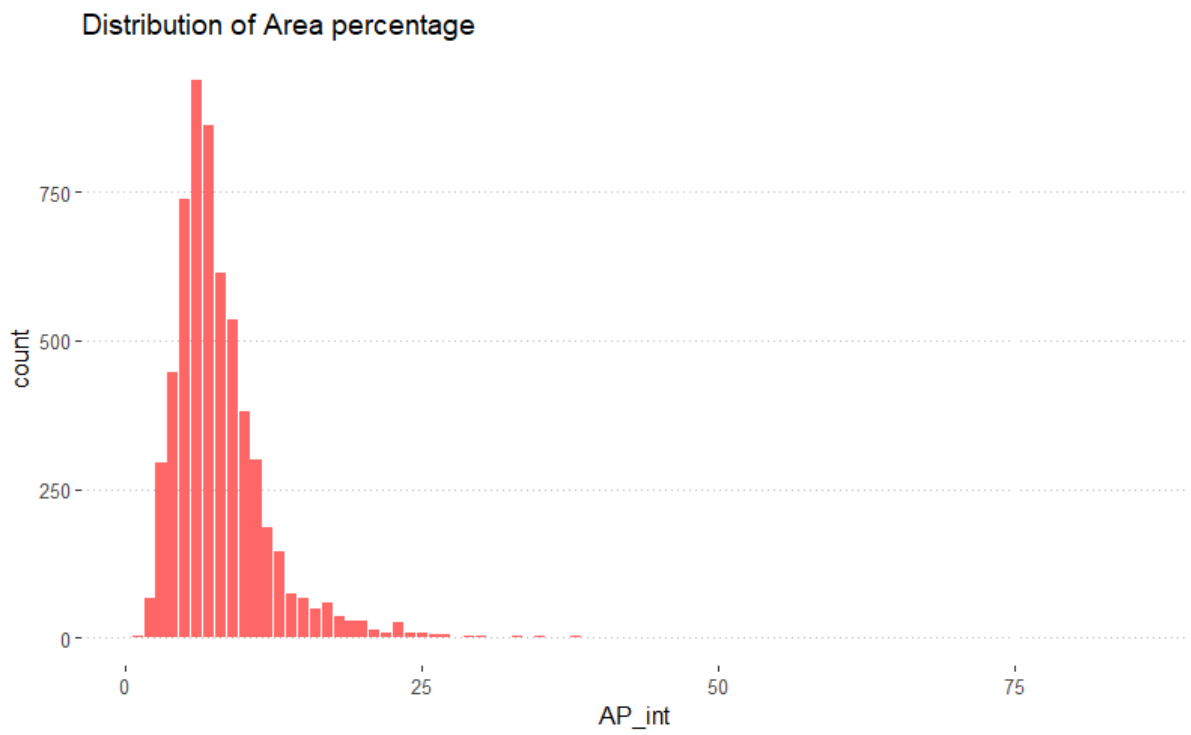


Since the age feature has a large number of numerical factors that make it difficult to make precise observations and determine the minimum age for serious damage. Let's build a new feature category for age by applying feature engineering techniques and generating bins.

around 90% of building ages fall under [0-50] range.2nd highest no. of bulidings are in the category high damage. Inferences, can be seen by pie chart below more easily

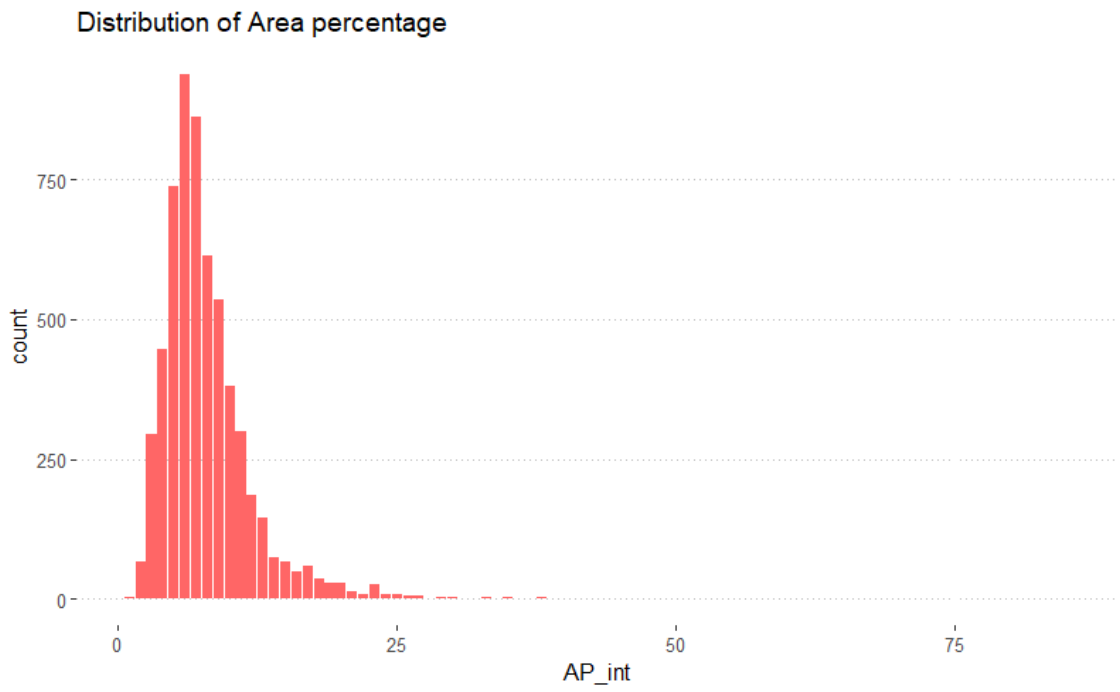
3.The number of families residing in the building is another essential aspect that determines the influence on human decline. In this scenario, we need to figure out how many families live in buildings that are about to be destroyed by quakes.

Let's use a bar plot to count the number of buildings that have a certain number of families residing in them.



Now let's figure out how many households live in those buildings that are on the verge of being destroyed if this happens again.

4.Area of building which is another major constraint which impacts severity in damage of buildings. The objective is to find out whether the limited evidence supports the thesis or not. Lets draw a histogram to find it out.



From the graph it is observed that for most structures, the normalised area of the building footprint is less than 20%. It's worth noting that some values can reach 100%, thus we need to be aware of the feature's outliers.

Its also implies area percentage has direct impact with increased damage .i.e. ,increase in area percentage increases damage building.

Missing Data Imputation

Lets begin work-around with EDA. For this project let us consider . Since our dataset has previously been cleaned, let's create a few random NA samples in numerical columns using the miss Forest library for testing reasons, so that we may apply missing value imputation approaches to stick to the actual observed values of features.

There are many techniques to handle missing values like,

MICE, AMELIA, HMISC and RANDOM FOREST etc.,

Lets inspect MICE In this project. Before that let us inspect the nature of missing values.

MCAR stands for "Missing Completely At Random." In the event of lacking data, this is the preferred scenario Too much missing data, assuming the data is MCAR, can also be an issue. For big datasets, a safe maximum threshold is usually 5% of the total. If more than 5% of a feature's or sample's data is missing, that feature should probably be removed.

MNAR stands for "missing not at random." Missing data that is not at random is a more significant problem, and in this instance, it may be prudent to investigate the data collection process further and try to figure out why the data is missing. For example, why

did the majority of individuals in a poll not respond to a given question? Was the question ambiguous?

Lets find out the missing values corresponding features and its percentage.

	feature	num_missing	pct_missing
	<fct>	<int>	<dbl>
1	CF	34	0.00567
2	A	30	0.00501
3	AP	29	0.00484
4	HP	26	0.00434

Now let us use MICE for imputing missing data with method as 'pmm',

m→number of imputations as 5.

method→pmm stands for predictive mean matching.

Atlast we arrive with clean data after imputation. We can see that the data is not shuffled by looking at the head and tail output. This is a major problem! When you split your data into a train set and a test set, we'll only choose damage grades from classes 1 and 2 (no damage grades from class 3 are in the top 80% of the observations), which means the algorithm will never encounter damage grade 3 features. This oversight will result in poor forecasting.

To overcome this issue we are shuffling the data. Let us Perform train and test split with 75 percentage data as train data.

Let's look at how to make multivariate graphs. We need to detect outliers and feature distribution with regard to goal variables for age, count of floor, area, and height percentages, as we described before. This was accomplished through the use of a box and density plot.

Build Multiple models:

Let us create multiple models with various classification algorithms with metric as accuracy.

We create models with LDA, CART, KNN, SVM and RF. Lets observe the Accuracy and K

appa scores for all the models. We see RF outperformed all other models. Even though the results are not that attractive, It seems the kappa score metric is acceptable to some extent. There is a room for improvement.

Because of the following specific categorization benefits, we are utilising Random Forest.

• It's a CART algorithm that deals with multiclass classification.

• It is a low-cost structure to create.

• It has a lightning-quick categorization time.

• Because trees are resistant to outliers because partitioning is based on the proportion of samples inside the split ranges rather than absolute values, it simplifies complicated connections between input variables and target variables and assigns variable significance to them.

Lets view Confusion matrix. Note that we are not balancing the target classes.

Introduction to SMOTE-A technique to over come imbalance.

SCUT is a library which is used to overcome imbalance in Target variables especially for multi classification. SCUT-SMOTE Cluster UpSampling Technique.

There is oversample function and undersample function. It helps in decrease or increase the majority or minority samples respectively, thereby to balance the target classes.

Finally we arrived with a balanced Multi Classification Model with71% ROC-value.

LIMITATIONS:

There is lot of room for improving model performance.

Below Are few modifications which could be done to improve model performance.

- **We have not performed modelling with all the features in the original Data set.**
- **We can experiment with other missing data imputation Methods.**
- **We can do several hyper parameter tuning.**
- **We can perform Feature Importance with most significant features.**
- **We can do other modelling techniques.**
- **We can use other metrics for evaluating using Cross validation.**
- **We can improvise the method of other SMOTE techniques like ADASYN,DENSITY-SMOTE etc.**

CONCLUSION:

For this project, We devised several hypothesis and we managed to prove to an extent. we also managed to address the imbalance to some extent. But it is important to note that we used only 17 features out of total 39 features. We generated random NA values and used MICE technique to address the missing value imputation. Due to that we observed some variance in the prediction output.

The Above information and plots helps us to understand the research questions formulated and the proof of evidence derived from the limited evidence. It is also keen to note that we observed ~65% accuracy as Balanced Class Predictions. Since this is a Multi classification Model, ROC helps us to get better insights on model metrics.

Appendix:

```
#Clear Environment

rm(list = ls())


# library(pacman) install pacman if not installed


sessionInfo() #This tells you all the whereabouts regarding your environment and installed version.

options(max.print=100000) #Displays full output


##Setting up the working Directory

setwd("C:/Users/44777/Documents/Earthquake_dmg_Prediction/")

getwd()


# install.packages(c("zoo","xts","quantmod"))

# install.packages( "C:/Users/44777/Downloads/DMwR_0.4.1.tar.gz", repos=NULL, type="source")


#Install required libraries:

pacman::p_load(pacman, dplyr, GGally, ggplot2, ggthemes,
```

```
        ggvis, httr, lubridate, plotly, rio, rmarkdown, shiny,
        stringr, tidyr, ggpubr, missForest, Amelia, DataExplorer,
        caret,intervals,gstat,UBL,smotefamily,rpart,nnet,funModeling,
        xgboost,Ckmeans.1d.dp,scutr,ranger,MASS,randomForest,kernlab,mlbench)


#Read csv file from source
df<- read.csv("Earthquake_Damage.csv")

# Details regarding the dataset
head(df)
summary(df)
str(df)

# Data Wrangling
df_new <- rename (df,
            "S_id" = X,
            "Town_City" = geo_level_1_id,
            "Village" = geo_level_2_id,
            "Streets" = geo_level_3_id,
            )


# Frequency of Target variable in Population
table(df_new$damage_grade)
#Plot Histogram to determine the distribution Frequency of Target variable in Population
ggplot(df_new, aes(damage_grade)) +
  geom_bar(fill = "#FF6666") +
  theme_pubclean()
```

```
# obtain stratified sample with 20 % of entire population.
strat_sample <- df_new %>%
  group_by(damage_grade) %>%
  sample_frac(size=.20)
# Frequency of Target variable in sample
table(strat_sample$damage_grade)
#Plot Histogram to determine the distribution Frequency of Target variable in sample
ggplot(strat_sample, aes(damage_grade)) +
  geom_bar(fill = "#FF6666") +
  theme_pubclean()

#Smoothed Density estimate plot of each class variable in sample
strat_sample$damage_grade_fac <- as.factor(strat_sample$damage_grade)

ggplot(strat_sample, aes(x = damage_grade_fac, colour = damage_grade_fac)) +
  geom_density(aes(group = damage_grade_fac, fill = damage_grade_fac), alpha = 0.3) +
  labs(title = "Distribution of each Type")


# Lets factorize categorical variable

strat_sample$land_surface_condition  <- as.factor(strat_sample$land_surface_condition)
strat_sample$foundation_type  <- as.factor(strat_sample$foundation_type)
strat_sample$roof_type  <- as.factor(strat_sample$roof_type)
strat_sample$ground_floor_type  <- as.factor(strat_sample$ground_floor_type)
strat_sample$other_floor_type  <- as.factor(strat_sample$other_floor_type)
strat_sample$position  <- as.factor(strat_sample$position)
```

```
strat_sample$plan_configuration  <- as.factor(strat_sample$plan_configuration)

# Lets work with hypothesis:

# 1.What are the sites that have an influence on the building's damage level?

#Plot Histogram to determine the distribution Frequency of Target variable in sample

ggplot(strat_sample, aes(Town_City)) +
  geom_bar(fill = "#FF6666") + labs(title = "Distribution of Geo_id_1")+
  theme_pubclean()
ggplot(strat_sample, aes(Village)) +
  geom_bar(fill = "#FF6666") + labs(title = "Distribution of Geo_id_2")+
  theme_pubclean()
ggplot(strat_sample, aes(Streets)) +
  geom_bar(fill = "#FF6666") + labs(title = "Distribution of Geo_id_3")+
  theme_pubclean()

# 2.What is the minimum age or age group of a severely damaged building?

ggplot(strat_sample, aes(age)) +
  geom_bar(fill = "#FF6666") + labs(title = "Distribution of Age")+
  theme_pubclean()

# 3.What portion of families live in severely damaged structures?

ggplot(strat_sample, aes(count_families)) +
  geom_bar(fill = "#FF6666") + labs(title = "Distribution of families")+
  theme_pubclean()
```

```
# 4.Does the percentage of area have an effect on extreme damage?


ggplot(strat_sample, aes(AP_int)) +
  geom_bar(fill = "#FF6666") + labs(title = "Distribution of Area percentage")+
  theme_pubclean()


# Check for Missing Values
nrow(strat_sample[!complete.cases(strat_sample),])


#Generate random missing values in columns count of floors, age, area percentage and height percentage in sample.
ss_mis_c_a_ap_hp_org <- prodNA(strat_sample[,6:9], noNA = 0.005)
ss_mis_c_a_ap_hp<-rename(ss_mis_c_a_ap_hp_org,
                        "CF" = count_floors_pre_eq,
                        "A" = age,
                        "AP" = area_percentage,
                        "HP" = height_percentage,
                        )


# Convert the datatype to int
CF_int<-as.integer(ss_mis_c_a_ap_hp_org$count_floors_pre_eq)
A_int<-as.integer(ss_mis_c_a_ap_hp_org$age)
AP_int<-as.integer(ss_mis_c_a_ap_hp_org$area_percentage)
HP_int<-as.integer(ss_mis_c_a_ap_hp_org$height_percentage)


#determine the percentage of missing values in missing values generated column in sample
DataExplorer::profile_missing(ss_mis_c_a_ap_hp)
```

```
#Create a new dataframe with needed features.

impdata<-
data.frame(strat_sample$Town_City,strat_sample$Village,strat_sample$Streets,
           strat_sample$foundation_type,strat_sample$roof_type,
           strat_sample$ground_floor_type,strat_sample$other_floor_type,
           strat_sample$position,strat_sample$plan_configuration,
           strat_sample$count_families,CF_int,A_int,AP_int,HP_int,
           strat_sample$building_id,strat_sample$land_surface_condition,
           strat_sample$damage_grade_fac)


data_to_imp<-rename(impdata,
           "Town_City" = "strat_sample.Town_City",
           "Village" = "strat_sample.Village",
           "Streets" = "strat_sample.Streets",
           "Roof_type" = "strat_sample.roof_type",
           "Ground_floor" = "strat_sample.ground_floor_type",
           "Foundation" = "strat_sample.foundation_type",
           "Other_floor" = "strat_sample.other_floor_type",
           "Position" = "strat_sample.position",
           "Plan" = "strat_sample.plan_configuration",
           "Land_surface_condition" = "strat_sample.land_surface_condition",
           "count_families" = "strat_sample.count_families",
           "building_id" = "strat_sample.building_id",
           "damage_grade" = "strat_sample.damage_grade_fac"
           )



#determine the percentage of missing values in missing values generated column in
sample

DataExplorer::profile_missing(data_to_imp)
```

```
#impute values with MICE
set.seed(12345)
library(mice)

#Get the pattern of Data
md.pattern(data_to_imp)

#Plot to get missing values in percentages.
library(VIM)
mice_plot <- aggr(data_to_imp, col=c('navyblue','yellow'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(data_to_imp), cex.axis=.7,
                  gap=3, ylab=c("Missing data","Pattern"))

#Use MICE to impute missing values.
miceData <- mice(data_to_imp,m=5,maxit=50,meth='cart',seed=500)
summary(miceData)

#check mice imputation on CF_int column.
miceData$imp$CF_int

#Find the method of imputation algorithm
miceData$meth

#Obtain the completed dataset after imputation.
EQ_Data <- complete(miceData,1)

#Check for percentage of missing values in complete dataset after miceimputation
```

```
DataExplorer::profile_missing(EQ_Data)


#Determine levels of target variable

levels(EQ_Data$damage_grade)


#Exploratory data analysis, data preparation and model performance of mice imputed EQ dataset

funModeling::df_status(EQ_Data)


shuffle_index <- sample(1:nrow(EQ_Data))

head(shuffle_index)


dv <- caret::dummyVars("
~Foundation+Roof_type+Ground_floor+Other_floor+Position+Plan+Land_surface_condition ", data = EQ)

new_df <- data.frame(predict(dv, newdata = EQ))

# head(new_df[, 3:9])

dim(new_df)

str(new_df)


EQ_end<-cbind(EQ$Town_City,EQ$Village,EQ$Streets,new_df,

        EQ$damage_grade,EQ$CF_int,EQ$A_int,EQ$count_families,EQ$AP_int,

        EQ$HP_int)


EQ_ML<-rename(EQ_end,

              "Town_City" = "EQ$Town_City",

              "Village" = "EQ$Village",

              "Streets" = "EQ$Streets",

              "CF_int" = "EQ$CF_int",

              "A_int" = "EQ$A_int",
```

```
                "AP_int" = "EQ$AP_int",
                "HP_int" = "EQ$HP_int",
                "count_families" = "EQ$count_families",
                "damage_grade" = "EQ$damage_grade"
               )


EQ <- EQ_ML[shuffle_index, ]

trainIndex <- createDataPartition(EQ$damage_grade, p = .75, list = FALSE, times = 1)
train <- EQ[ trainIndex,]
val  <- EQ[-trainIndex,]

#Correlation for all numerical Variables
GGally::ggcorr(EQ %>%
          select_if(is.numeric),label = "Correlation_Matrix")



x <- EQ[,-10]
y <- EQ[,10]

# Multivariate plots
featurePlot(x=x, y=y, plot="box")

#Density Plots by class value
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)

#Test Harness:
```

```
#Run algorithms using 5-fold cross validation

control <- trainControl(method="cv", number=5)

metric <- "Accuracy"

metric


# Build Models


# linear Discrimant Analysis

set.seed(1234)

lda_fit <- train(damage_grade~., data=EQ, method="lda", metric=metric,
trControl=control)


# CART

set.seed(1234)

cart_fit <- train(damage_grade~., data=EQ, method="rpart", metric=metric,
trControl=control)


# KNN

set.seed(1234)

knn_fit <- train(damage_grade~., data=EQ, method="knn", metric=metric,
trControl=control)


# SVM

set.seed(1234)

svm_fit <- train(damage_grade~., data=EQ, method="svmRadial", metric=metric,
trControl=control)


# RF

set.seed(1234)

rf_fit <- train(damage_grade~., data=EQ, method="rf", metric=metric, trControl=control)
```

```
# summarize accuracy of models
results <- resamples(list(lda=lda_fit, cart=cart_fit, knn=knn_fit, svm=svm_fit, rf=rf_fit))
summary(results)

# compare accuracy of models
dotplot(results)

# estimate skill of RF on the validation dataset
predictions <- predict(rf_fit, val)
confusionMatrix(predictions, val$damage)

# NEED FOR SMOTE:
table(train$damage_grade)
train_EQ_dmg_grade_imb <- train[train$damage_grade %in% c(1,2,3), ]
str(train_EQ_dmg_grade_imb)
train_EQ_dmg_grade_imb$damage_grade <-
as.numeric(train_EQ_dmg_grade_imb$damage_grade)
str(train_EQ_dmg_grade_imb)
table(train_EQ_dmg_grade_imb$damage_grade)

train_scutted <- SCUT(train_EQ_dmg_grade_imb, "damage_grade", undersample =
undersample_kmeans,
                      usamp_opts = list(k=5))
str(train_scutted)
table(train_scutted$damage_grade)

# SMOTED Build model
train_scutted$damage_grade<-as.factor(train_scutted$damage_grade)
val$damage_grade<-as.factor(val$damage_grade)
```

```
rf_fit_sm<-train(damage_grade~.,data = train_scutted,method="rf",metric = metric
,trControl=control,summaryFunction=multiClassSummary)

predictors_sm<-names(train_scutted)[names(train_scutted)!='damage_grade']

# val$damage_grade <- as.numeric(val$damage_grade)


# estimate skill of RF after smote on the validation dataset

predictions_sm <- predict(rf_fit_sm, val)

str(predictions_sm)


confusionMatrix(predictions_sm, val$damage_grade, mode = "everything")



# To find ROC

library(pROC)


predictions_sm<-as.numeric(predictions_sm)

val$damage_grade<-as.numeric(val$damage_grade)

auc<-multiclass.roc(predictions_sm, val$damage_grade)

auc


rs <- auc[['rocs']]

plot.roc(rs[[1]])

sapply(2:length(rs),function(i) lines.roc(rs[[i]],col=i))


# Clear packages

# p_unload(all)  # Remove all add-ons

# detach("package:", unload = TRUE)
```