PREDICTING BUILDING DAMAGE CAUSED BY EARTHQUAKE

PROJECT REPORT

SUBMITTED BY

MITHUN K K

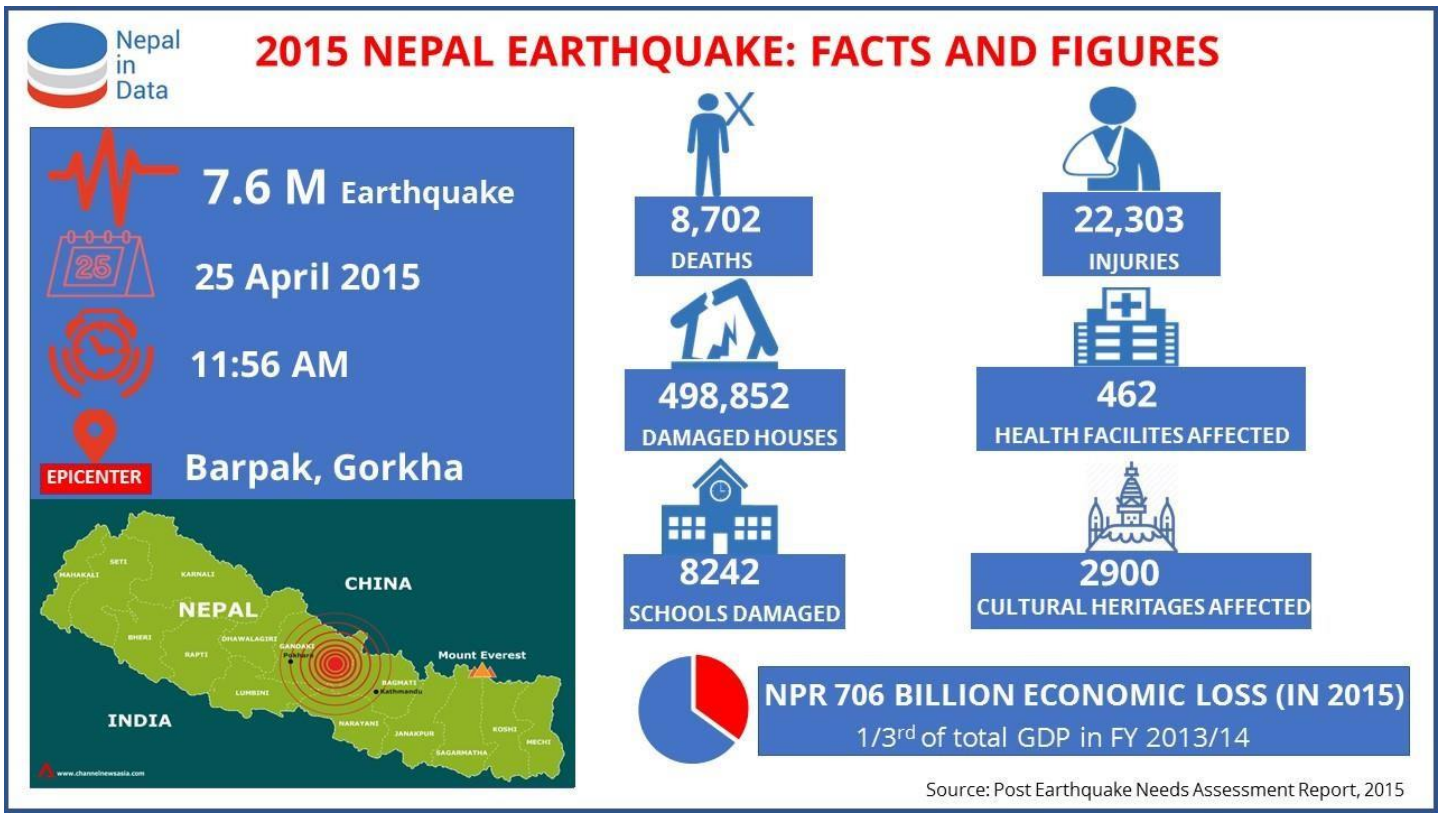
ABSTRACT

Earthquake is one of the most hazardous natural calamities. It occurs due to sudden shaking of ground which is caused by movement of seismic waves. The tectonic plates are always slowly moving, but they get stuck at their edges due to friction. When the stress on the edge overcomes the friction, there is an earthquake that releases energy in waves that travel through the earth's crust and cause the shaking that we feel. Earthquake can cause devastating loss of life and property. The assessment of damage level of post-earthquake structures through calculations is always complicated and time-consuming, and it is not suitable for situations when a large number of buildings need to be evaluated after an earthquake. One of the largest post-disaster datasets, which comes from the April 2015 Nepal earthquake, makes it possible to train models to predict damage levels based on building information. Here, we investigate the application of various machine learning methods to the problem of structural damage level prediction. The problem statement of this project is determining the degree of damage that is done to buildings, thus when an earthquake occur, we can help identify safe and unsafe buildings, thus avoiding death and injuries resulting from aftershocks. Leveraging the power of machine learning is one viable option that can potentially prevent massive loss of lives while simultaneously making rescue efforts easy and efficient

INTRODUCTION

Earthquake is one of the major natural disasters. Every year it causes a huge loss of life and property all around the world. An earthquake is the shaking of the surface of the earth due to the sudden release of energy in the earth's crust. As a result, seismic waves are created. The seismic activities in an area determine the type and intensity of the earthquake. Earthquakes are caused due to sudden tectonic movements in the earth's crust. When the tectonic plates slide over one another, there is a cause of orogeny which results in earthquakes and volcanoes. These disturbances cause vibrations that spread in all directions. As there is a relative motion of these plates, there is stress built up, which breaks by releasing the stored energy known as shock waves. The ground movements in earthquakes will cause huge damage to building structures. There are different levels of earthquake magnitude, but there is no systematic classification of the buildings damage. The dynamic analysis of a single structure usually takes a long time, which is not suitable for situations where a large number of buildings need to be evaluated after an earthquake. I hope that through the application of machine learning on this dataset, we can quickly classify the damage level of the structures in order to facilitate planning rescue and reconstruction and assessing the loss.

Our dataset is based on Nepal earthquake which happened on April 25th, 2015 in Gorkha After the April 2015 Nepal earthquake, which was the worst natural disaster to strike Nepal since 1934, the National Planning Commission, along with Kathmandu Living Labs and the Central Bureau of Statistics generated large post-disaster datasets, containing valuable information on earthquake impacts, household conditions, and socioeconomic- demographic statistics.



In 2015 the mortality rate due to earthquake was nearly 9000 and 22,000 people was got injured. Hundreds of thousands Nepalese were made houseless due to these natural calamities. World heritage sites in Kathmandu valley and some other regions get destroyed due to damage caused by an earthquake. We are able to categories the damage rate and intensity of that earthquake using different reports. Prediction keeps a set of input which decreases relative number of earthquakes from temporal distribution of part earthquake. Relative decrease in number of earthquakes is break in the normal seismic energy released from the region. Grades are used to represent the damage caused by an earthquake. Grades represent a damage to the building which was cause by the earthquake. There are three grades of the damage: Grade1, Grade 2 and Grade 3 are used to represent low, medium and complete destruction respectively [2]. The Nepal earthquakes of April and May 2015 killed around 9000 of people and injured around 20000. The Destruction caused was severe. Around 2 million were left homeless after the disaster.

On April 25, an earthquake of magnitude of 7.8Mw strikes followed by an earthquake of magnitude 7.5 which caused massive destruction resulting in damage of schools, public health centers, water system, power systems, roads and bridges along with homes of people [2]. In the surveys conducted by various organization, the results had some very unusual findings such as those villages seemed to observe a less amount of damage were the ones located near the epicentre and to the west of the fault. The earthquake showed very comparatively less magnitude of 5-6Mw as compared to northern region which had around earthquake of magnitude 7Mw. This showed that the intensity and magnitude of the earthquake was increasing in the North direction.

OBJECTIVES OF THE PROJECT

The main objective of the project is to classify the damage occurred to buildings in to three grades i.e. Grade 1, Grade 2 and Grade 3 representing low damage high damage and medium damage respectively

GENERAL BACKGROUND

Data Mining

Data Mining(DM) is a technique using which one could discover hidden patterns and relations among the attributes in a huge collection of data is discovered. Data Mining has its application in many fields, education, engineering, marketing, sports, finance, medical are few of them. Data Mining techniques also has the ability in making decision and problem solving in particular areas In this work, I have mainly concentrated on the area of education, where data mining applications could be applied for analysis and processing data. As the year passes the number of students enrolling in colleges and universities has increased so as their details including academics. The academic details of students are helpful in finding out how a particular student will perform in their university exams. A manual analysis is possible when the size of data is less, as the data get increased manual analysis will not be effective. In such cases, data mining will be helpful. In the field of education, data mining is mainly used to extract useful information of students such as their internal marks and other details from

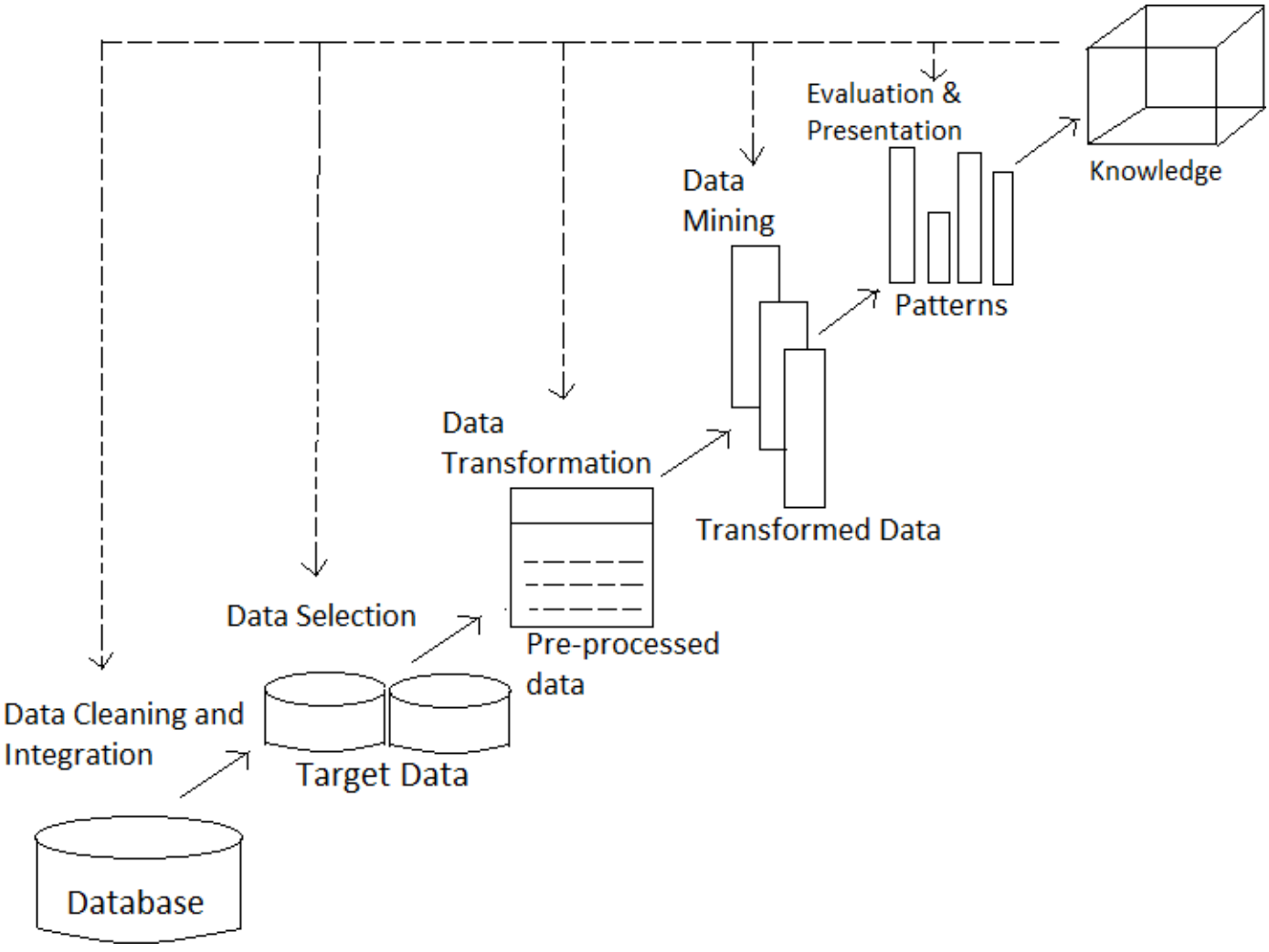


Fig 1- Steps in KDD process

the student data for further application like the prediction of performance of students. Data Mining is the suitable technique for managing these data to discover new knowledge and information about students. Data Mining also uses machine learning, visualization and statistical techniques to extract and find knowledge which is easily interpretable to others. Data mining and KDD(Knowledge discovery in Database) are often connected to each other. The process of KDD is used to form high-level knowledge from low level data. That is using KDD unknown, unwanted and implicit data could be removed from the large volume of data. The figure 1.1 shows the process of KDD and the steps involved in it. After collecting the raw data the following steps in KDD are performed.

1. Data cleaning: it is the process of removal of unwanted and noisy data from the huge volume of data, cleansing is the other name of this process.

2. Data integration: it is the process of combining certain data which are showing the same property.

3. Data selection: In this phase the data needed for the particular work is identified and fetched out from the huge volume of data.

4. Data transformation: The process of translating data from one format to another format which is useful for data analysis purpose, data consolidation is another name of data transformation.

5. Data mining: In the data mining step the whole dataset is traversed deeply to collect useful information for the work.

6. Pattern evaluation: It is the process of finding out the pattern and relations associated with each attribute and its instances.

7. Knowledge representation: The mining data is visually represented in this step and also the interpretation of results is being done.

MACHINE LEARNING

Machine Learning is simply the method of making the machine to learn certain things so that the machine will be capable of performing functions and tasks by its own. In order to do so first it understands the structure of the data and construct a model with the data which is easy to understand and could be used by the people. The machine learning algorithms uses statistical calculations in order to train the system and for obtaining required output. Machine learning has its application in facial recognition, optical character recognition, recommendation engines, self driving cars etc. Mainly there are three machine learning approaches: Supervised learning, Unsupervised learning and Reinforcement learning these classifications are based on how the system is made capable of understanding the problem scenario.

a. Supervised learning: In the supervised learning algorithms the training is done using example input data and the classification labels are created for output. The main purpose of this method is to make the algorithm learn by comparing its actual output with the obtained output to find out the errors, and modify the model accordingly. Therefore supervised learning algorithms uses pattern method to predict label values for the unlabeled data. A common case of supervised learning is the prediction of future events using the historical data.

b. Unsupervised learning: In unsupervised learning algorithm the data is provided without any labeling so that it could find the structure within the input data itself. The main goal of unsupervised learning is to unleash the hidden pattern within the dataset, and feature learning using which the machine automatically identifies the representation that are needed to classify the data. It is mainly used in case of transactional data. Unsupervised learning algorithm deals with complex data like the data which have no particular order and are unrelated to each other, this method will organize the data in a meaningful way. The applications include anomaly detection, detecting fraudulent credit card purchases etc.

c. Semi-supervised learning: In Semi-supervised learning a class of machine learning tasks and techniques that also make use of unlabeled data for training typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data).The application include semi-supervised learning is webpage classification and speech analysis c. Reinforcement learning: The working of Reinforcement Learning involves interaction with environment , this approach is said to posses true artificial intelligence. Reinforcement Learning is learning what to do and how to map situations to actions. The end result is to maximize the numerical reward signal. The learner is not told which action to take, but instead must discover which action will yield the maximum reward, its applications include autonomous car

SCOPE OF THE PROJECT

- This Project will help to predict the loss or damage caused to buildings during earthquakes.
- Government agencies- Government agencies will benefit from a closer and a faster approximation on the damage and havoc created by the earthquakes without any manual inspection. This can be conducive to effective post disaster recovery operations.
- Insurers- After a large scale and widespread disaster like the Gorkha earthquake, insurers' claims systems are accumulated with a huge number of new claims. It is incredibly tedious for claim handlers and inspectors to sift through all the data and determine the damage severity. Incorporating AI-based damage evaluation components with claim systems will provide the claim inspectors a single index (damage level), using which they can determine the severity of the damage. It will also quicken claims processing and aid in sending help to the people who are at-risk immediately.
- These predictions also help in understanding the effect of earthquakes and the seismic vulnerability of various buildings in different seismic zones. These are required for precise estimation of seismic disaster and also for planning risk reduction. It will save the human lives by predicting the grade of the buildings. Necessary efforts can be done on buildings after grade estimation to rescue it from earthquakes.

LITERATURE SURVEY

Machine Learning has been widely used for making earthquake predictions due to their ability to improve over time. With the huge amount of earthquake instrumental data, machine learning approaches are capable enough to improve efficiency and accuracy in earthquake prediction. Multiple machine learning methods including, Artificial Neural Network (ANN), Support Vector machine (SVM), K-nearest neighbor (KNN), Naive Bayes (NB) and random forest algorithms have been exercised for earthquake prediction. Many studies have been made on the damage rate of buildings in Japan.

- Mononobe [1970] was the first to investigate the overturning of tomb stones which is equivalent to ground seismic coefficient shows the relations obtained from the damage data of several large earthquakes in Japan and theoretical curves expressed by normal probability distribution function.
- Shiga [1976] discussed the damage rate of concrete buildings those which are reinforced based on his research on the earthquake resistance capacity of existing those which are reinforced buildings. He derived a simple formula to estimate roughly the resistance capacity of lower reinforced concrete buildings using the amount of column areas and wall areas through the analysis of damaged as well as undamaged buildings in the (1968) Tokachioki earthquake and the (1978) Miyagikenoki earthquake. He estimated the probability distribution of the earthquake resistance capacity of existing low-rise reinforced concrete buildings and modeled the probability distribution by Gamma distribution, based on which he gave the relation between the damage rate of reinforced concrete buildings and the ground motion intensity.
- The Gutenberg and Richter statistical model found a correlation between the magnitude of earthquake and frequency of earthquake. For structural design, this earthquake probability distribution model was used. In supervision of the California Geological Survey, Petersen conducted research and suggested a model that is time-independent. This time independent model demonstrating that chances of occurrence of earthquake follow the Poisson's distribution model.
- One of the first papers to explore the damage of earthquake was Rasita Kerdpoln (1787), who discussed vulnerability to natural disasters as an issue of power based on earthquake in Haiti. For more than one hundred years, scholars in the civil engineering field have been studying the influence of various factors on damage of post-earthquake structures.
- In 1984, A.C. Boissonnade et al. presented a consistent method for earthquake intensity classification based on the theory of statistical pattern recognition and developed a discriminative function for such identifications based on the Bayesian criterion.
- 2020, Gian P. Delacruz used Generative Adversarial Networks to classify structural damage caused by earthquakes, which is almost the earliest application of machine

learning in classifying damage of post earthquake structures. Samuel Roeslin et al. (2020) tried four algorithms to develop a damage prediction model from 340 post-earthquake buildings in the Mexico City and achieved more than 65% prediction accuracy. Similarly, Shohei Naito et al. (2020) used machine learning models and aerial photographs to classify buildings in the Kumamoto earthquake into four damage levels. In this project, we will be expanding on these works and applying more varied methods on a much larger dataset, trying to achieve higher prediction accuracy.

IMPLEMENTATION

DATA COLLECTION

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 socio-economic-demographic statistics.

We're trying to predict the ordinal variable damage_grade, which represents a level of damage to the building that was hit by the earthquake. There are 3 grades of the damage:

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

FEATURES

The dataset mainly consists of information on the buildings' structure and their legal ownership. Each row in the dataset represents a specific building in the region that was hit by Gorkha earthquake.

There are 39 columns in this dataset, where the building_id column is a unique and random identifier. The remaining 38 features are described in the section below. Categorical variables have been obfuscated random lowercase ascii characters. The appearance of the same character in distinct columns does **not** imply the same original value.

DESCRIPTION

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

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

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

EXPLORATORY DATA ANALYSIS

The chart below gives info regarding various columns in the dataset

```
Data columns (total 39 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   building_id                             260601 non-null  int64
 1   geo_level_1_id                          260601 non-null  int64
 2   geo_level_2_id                          260601 non-null  int64
 3   geo_level_3_id                          260601 non-null  int64
 4   count_floors_pre_eq                     260601 non-null  int64
 5   age                                     260601 non-null  int64
 6   area_percentage                         260601 non-null  int64
 7   height_percentage                       260601 non-null  int64
 8   land_surface_condition                  260601 non-null  object
 9   foundation_type                         260601 non-null  object
 10  roof_type                               260601 non-null  object
 11  ground_floor_type                       260601 non-null  object
 12  other_floor_type                        260601 non-null  object
 13  position                                260601 non-null  object
 14  plan_configuration                      260601 non-null  object
 15  has_superstructure_adobe_mud            260601 non-null  int64
 16  has_superstructure_mud_mortar_stone     260601 non-null  int64
 17  has_superstructure_stone_flag           260601 non-null  int64
 18  has_superstructure_cement_mortar_stone  260601 non-null  int64
 19  has_superstructure_mud_mortar_brick     260601 non-null  int64
 20  has_superstructure_cement_mortar_brick  260601 non-null  int64
 21  has_superstructure_timber               260601 non-null  int64
 22  has_superstructure_bamboo               260601 non-null  int64
 23  has_superstructure_rc_non_engineered    260601 non-null  int64
 24  has_superstructure_rc_engineered        260601 non-null  int64
 25  has_superstructure_other                260601 non-null  int64
 26  legal_ownership_status                  260601 non-null  object
 27  count_families                          260601 non-null  int64
 28  has_secondary_use                       260601 non-null  int64
 29  has_secondary_use_agriculture           260601 non-null  int64
 30  has_secondary_use_hotel                 260601 non-null  int64
```

```
 31  has_secondary_use_rental                260601 non-null  int64
 32  has_secondary_use_institution           260601 non-null  int64
 33  has_secondary_use_school                260601 non-null  int64
 34  has_secondary_use_industry              260601 non-null  int64
 35  has_secondary_use_health_post           260601 non-null  int64
 36  has_secondary_use_gov_office            260601 non-null  int64
 37  has_secondary_use_use_police            260601 non-null  int64
 38  has_secondary_use_other                 260601 non-null  int64
dtypes: int64(31), object(8)
memory usage: 77.5+ MB
```

CORRELATION

Checking the correlation between different independent variables and the output variable damage_grade

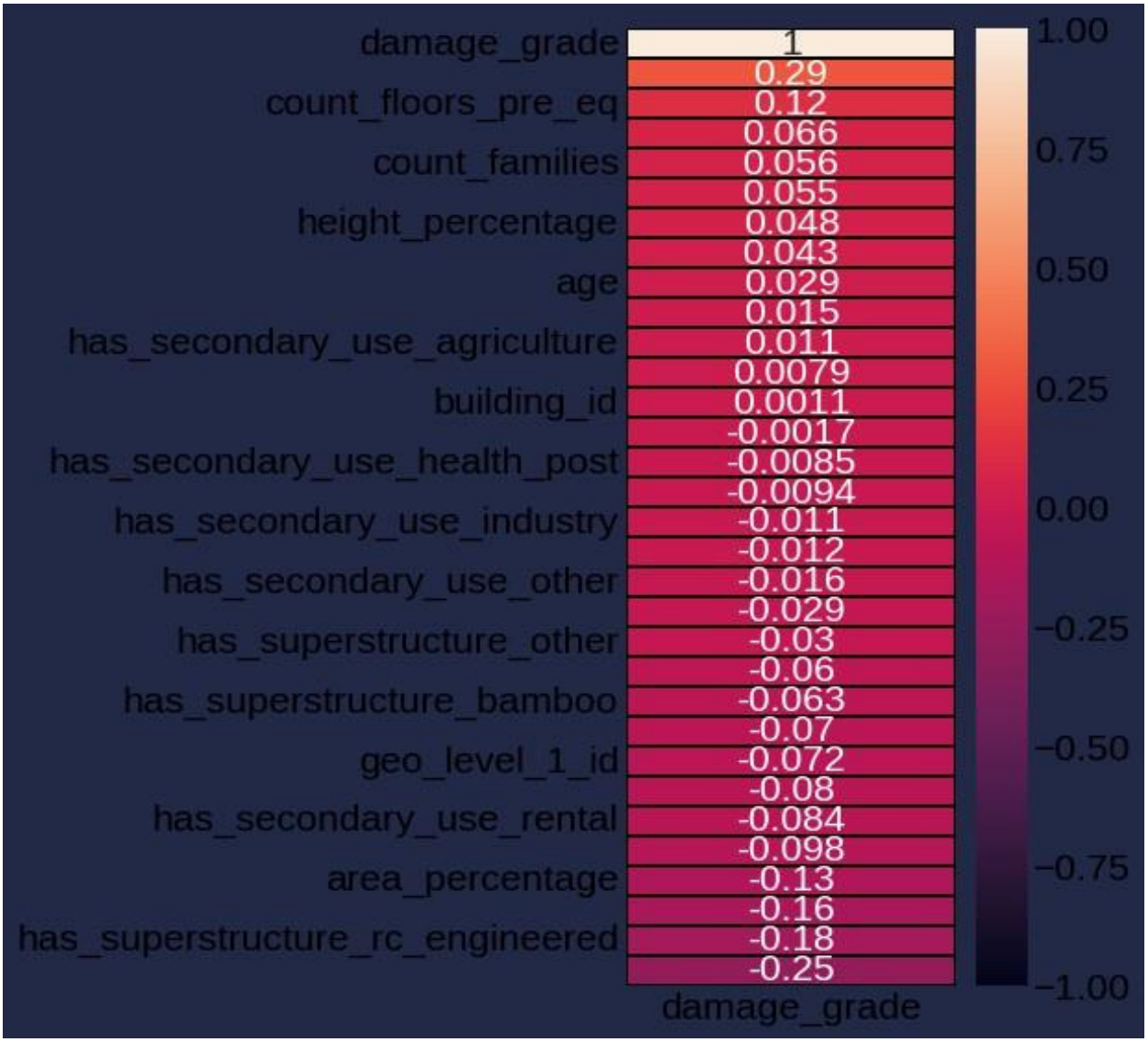


Fig 2 : Correlation of input columns with target column

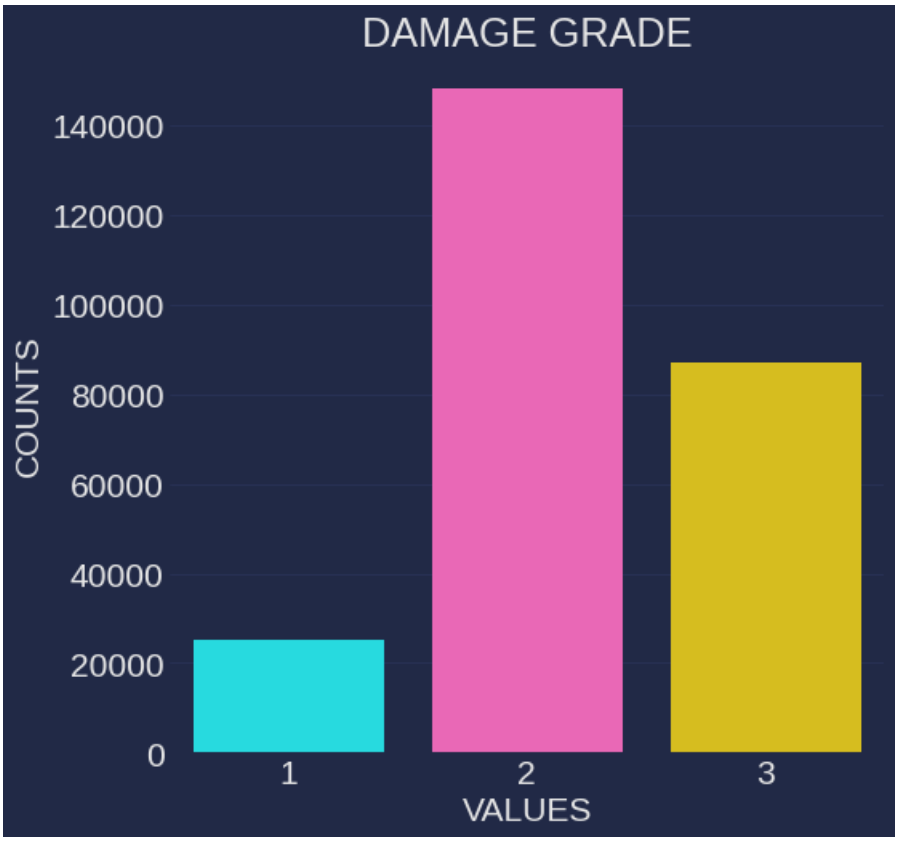


Fig 3 : EDA of distribution of damage grade

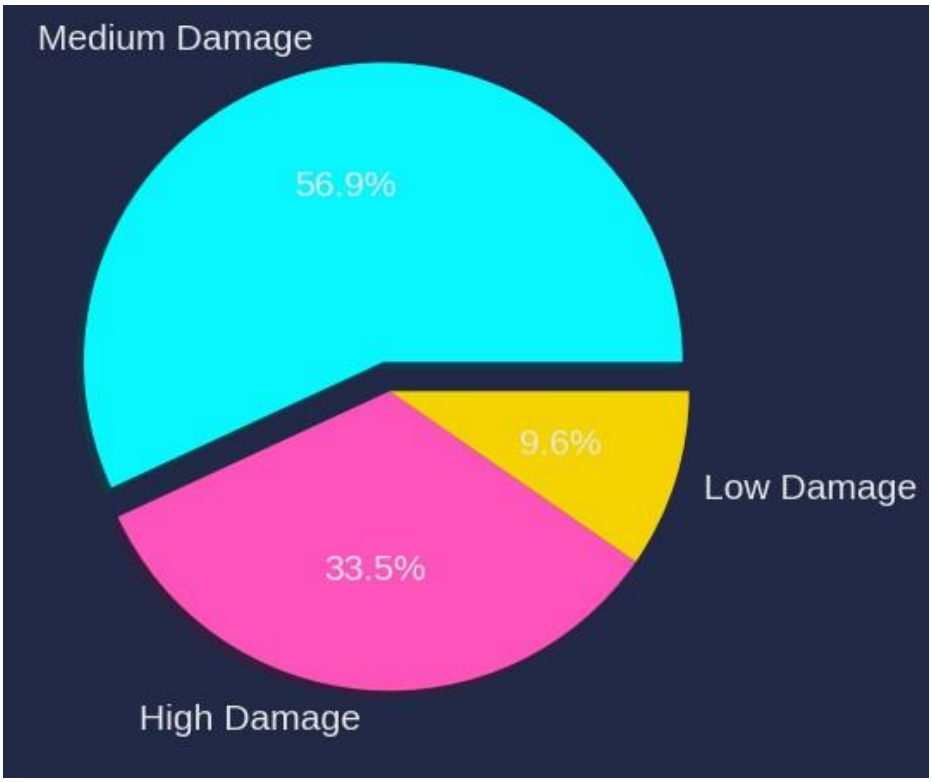


Fig 4 : Pie plot of distribution of damage grade

Plot a histplot to find the distribution of geo_level_1_id, geo_level_2_id, geo_level_3_id to find the distribution of the values in these 3 columns then plot 3 boxplots of the columns with respect to the target variable damage grade to check if there are any outliers and its effect on the target variable

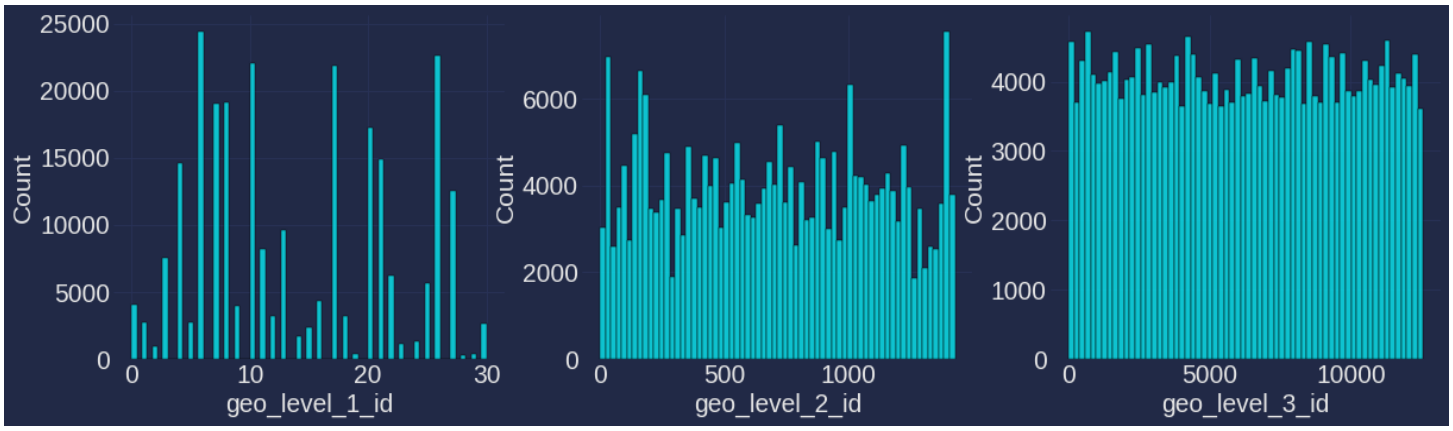


Fig 5 : Distribution of columns geo_level_(1,2,3)_id

Plot a boxplot to check if there are outliers in the 3 above mentioned columns

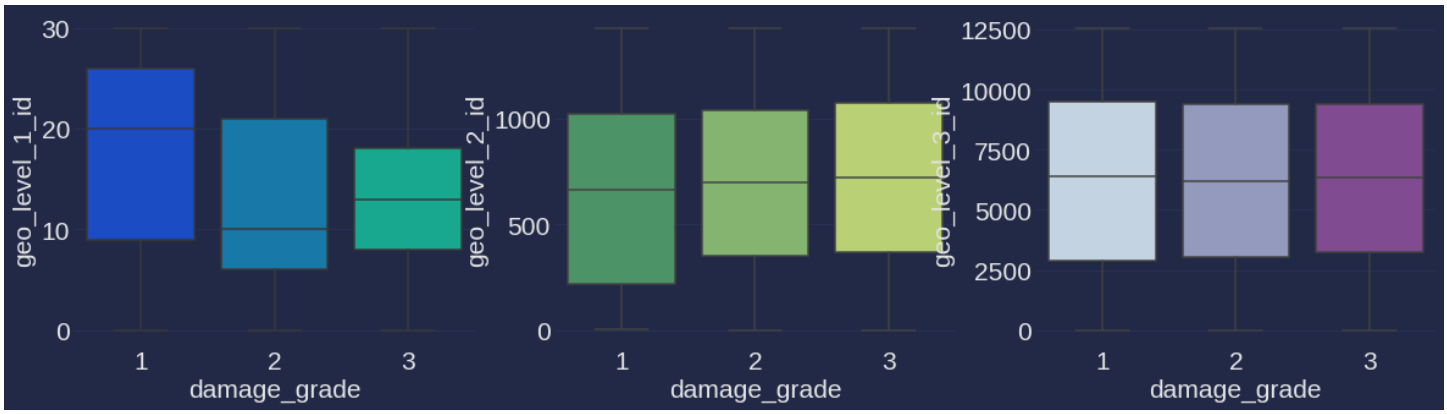


Fig 6 : Boxplot to check outliers in the 3 geo levels

Draw a two Kernel Distribution Estimation Plot first one depicting the probability density function of the columns geo level 1 id ,geo level 2 id ,geo level 3 id, age and area percentage, the second one depicting the probability density function of the columns geo level 1 id ,geo level 2 id ,geo level 3 id, age , area percentage and height percentage with respect to the 3 different grades of damage

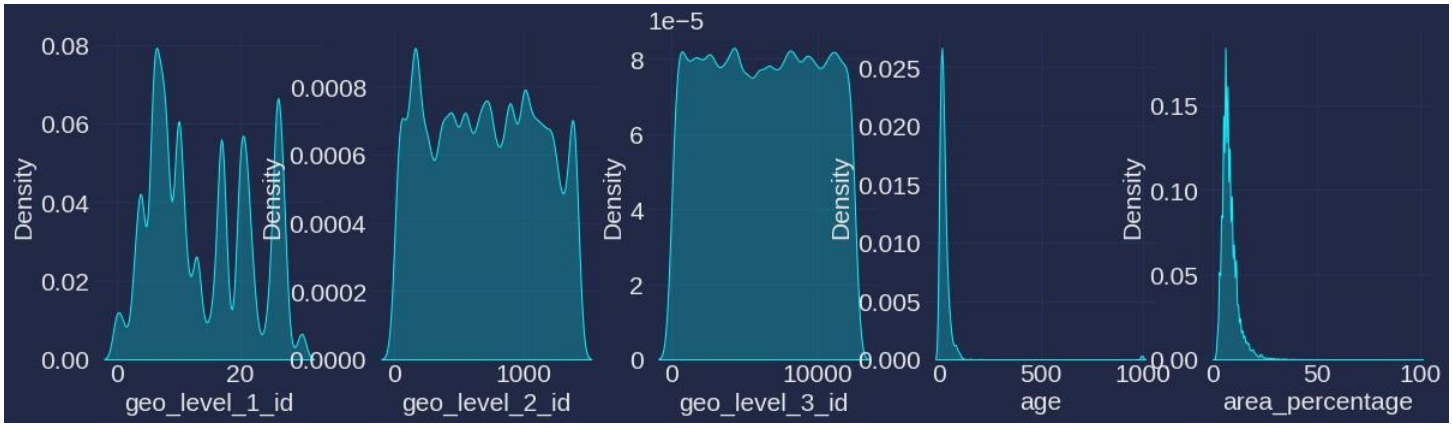


Fig 7: kdeplot of geo id, age, area percentage

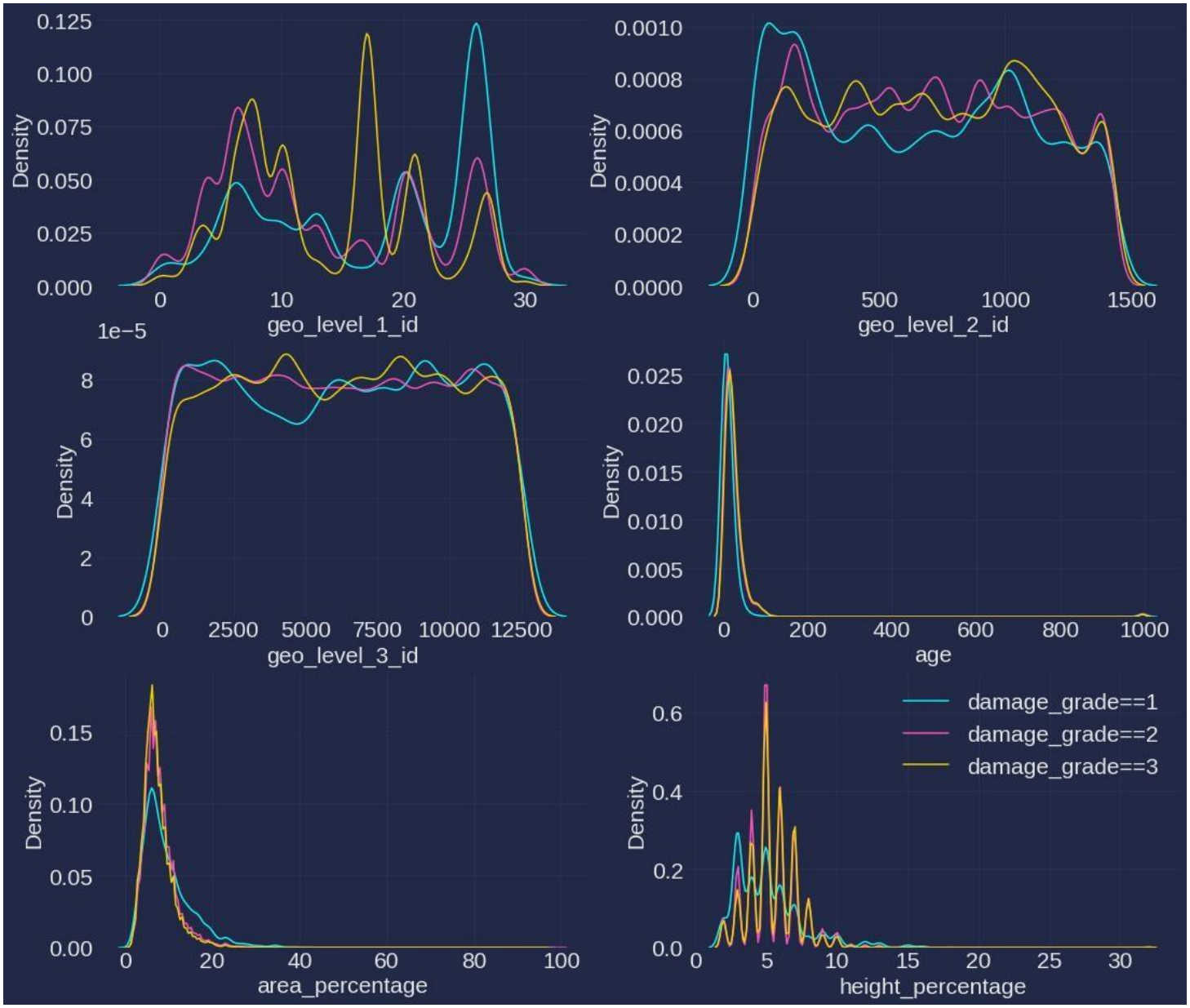


Fig 8: kdeplot of geo id, age and area percentage with respect to 3 different levels of damage

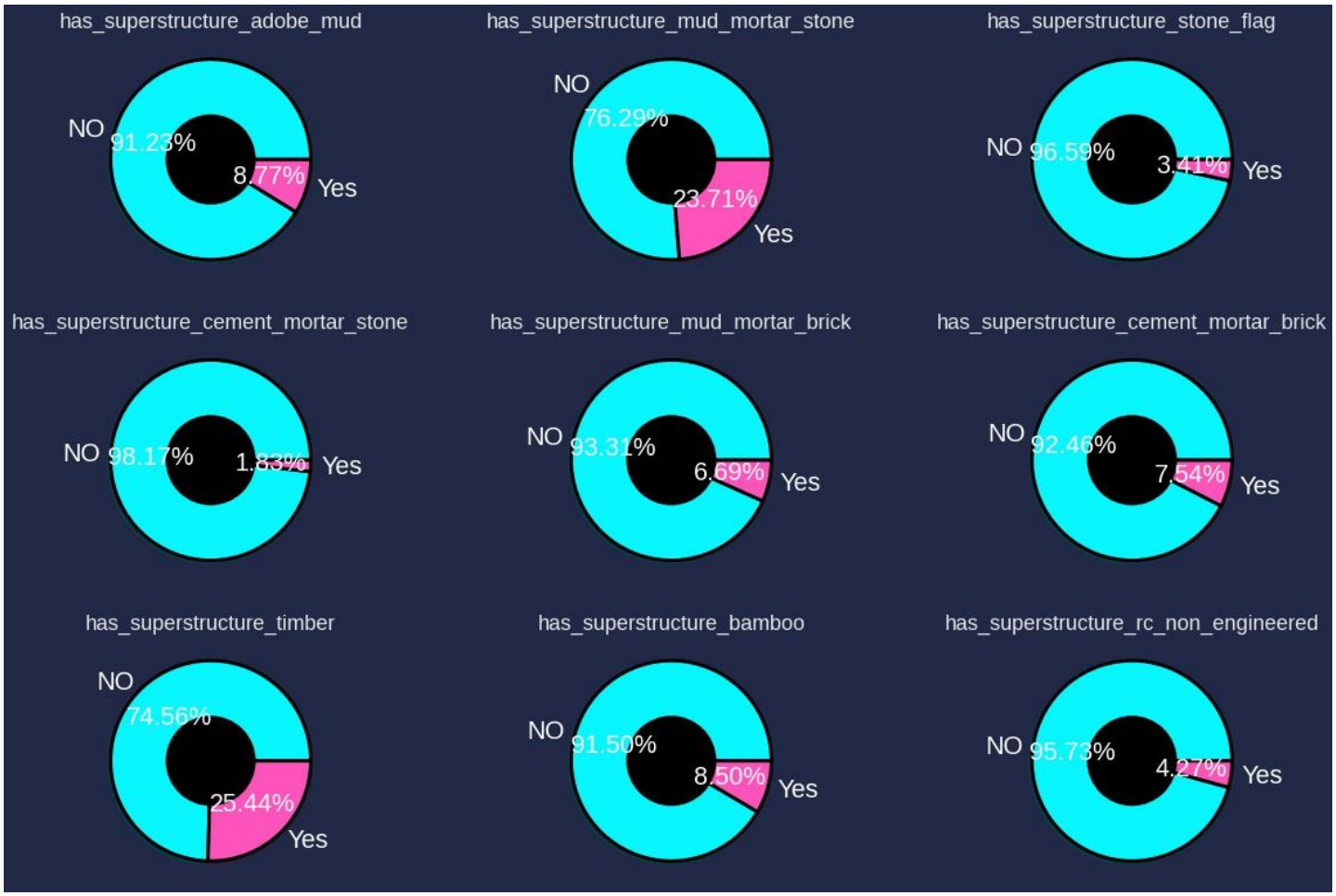


Fig 9 : Donut plot showing what percent of total buildings use different kinds of materials

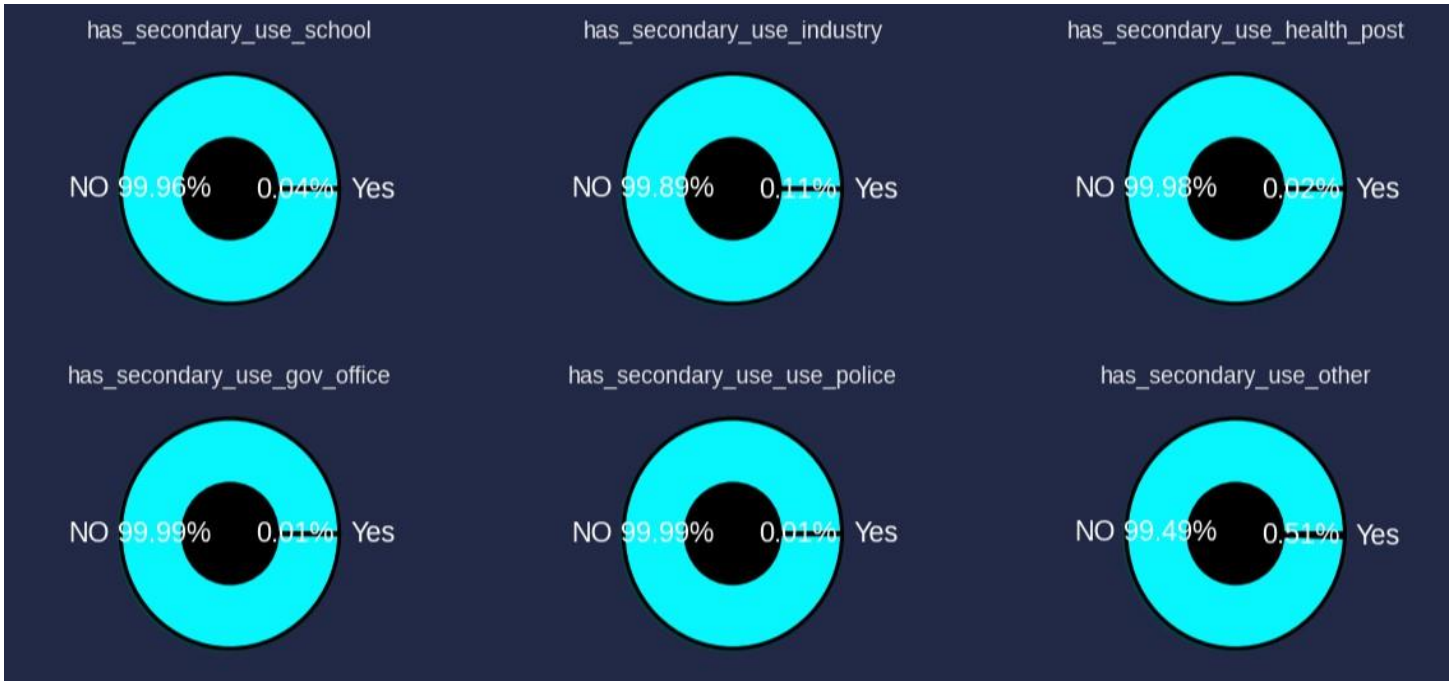


Fig 10 : Donut plot showing what percent of total buildings have secondary uses

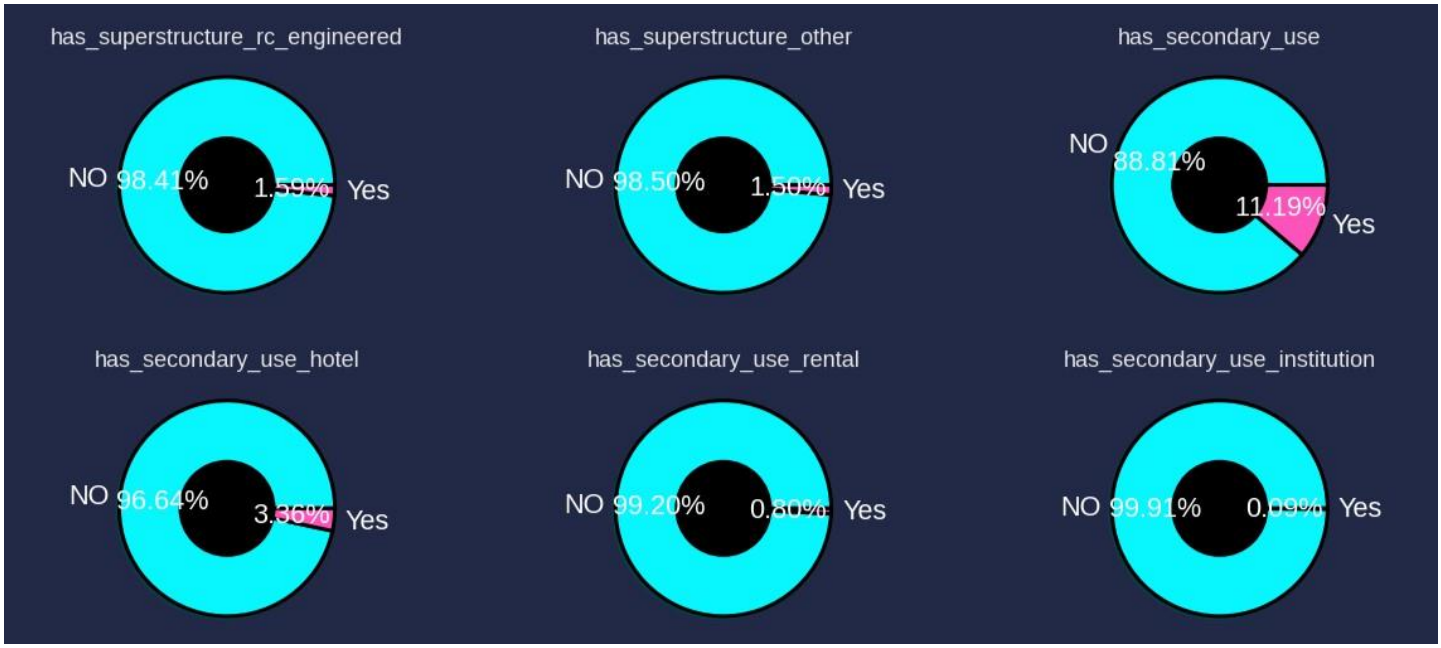


Fig 11 : Donut plot showing what percent of total buildings have secondary uses

Plot damage grade distribution with respect to materials used in construction

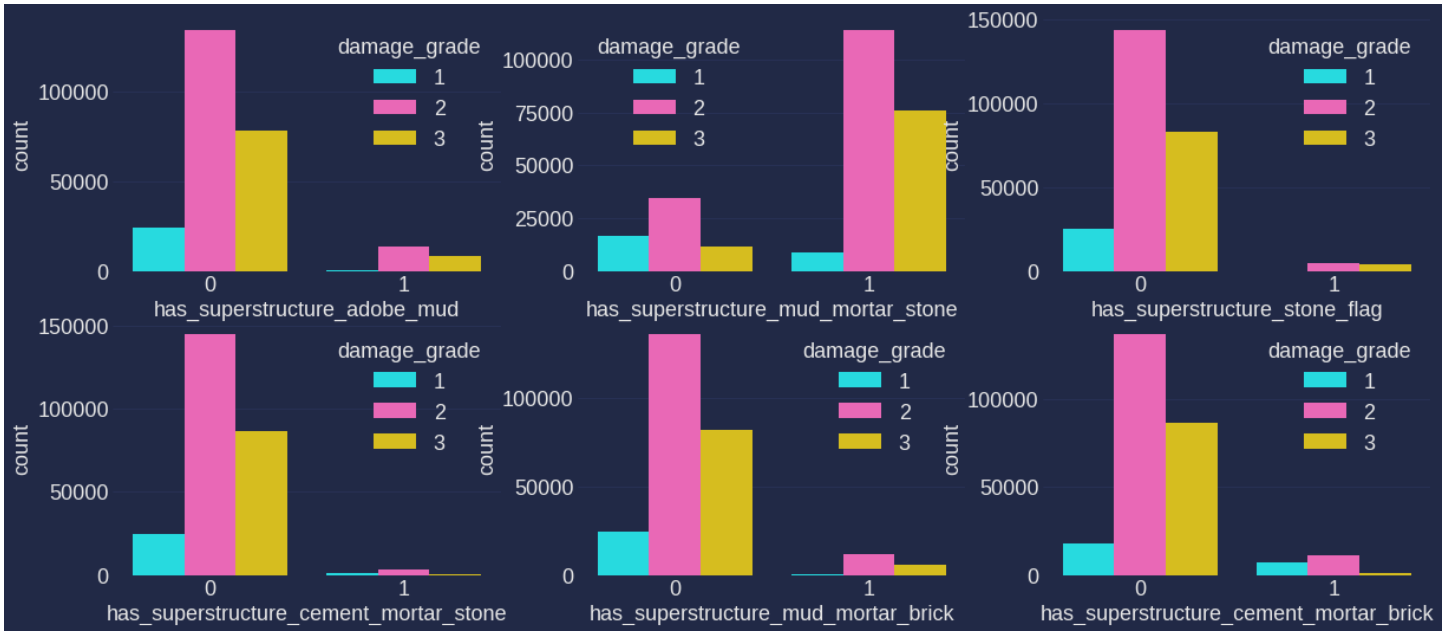


Fig 12 : plot showing relationship between damage grade and materials used

The plot suggest that buildings that used bricks, stones and cement faired better in comparison to buildings that use only mud , mortar and brick

Fig 13 : plot damage grade distribution with respect to secondary usage and materials used

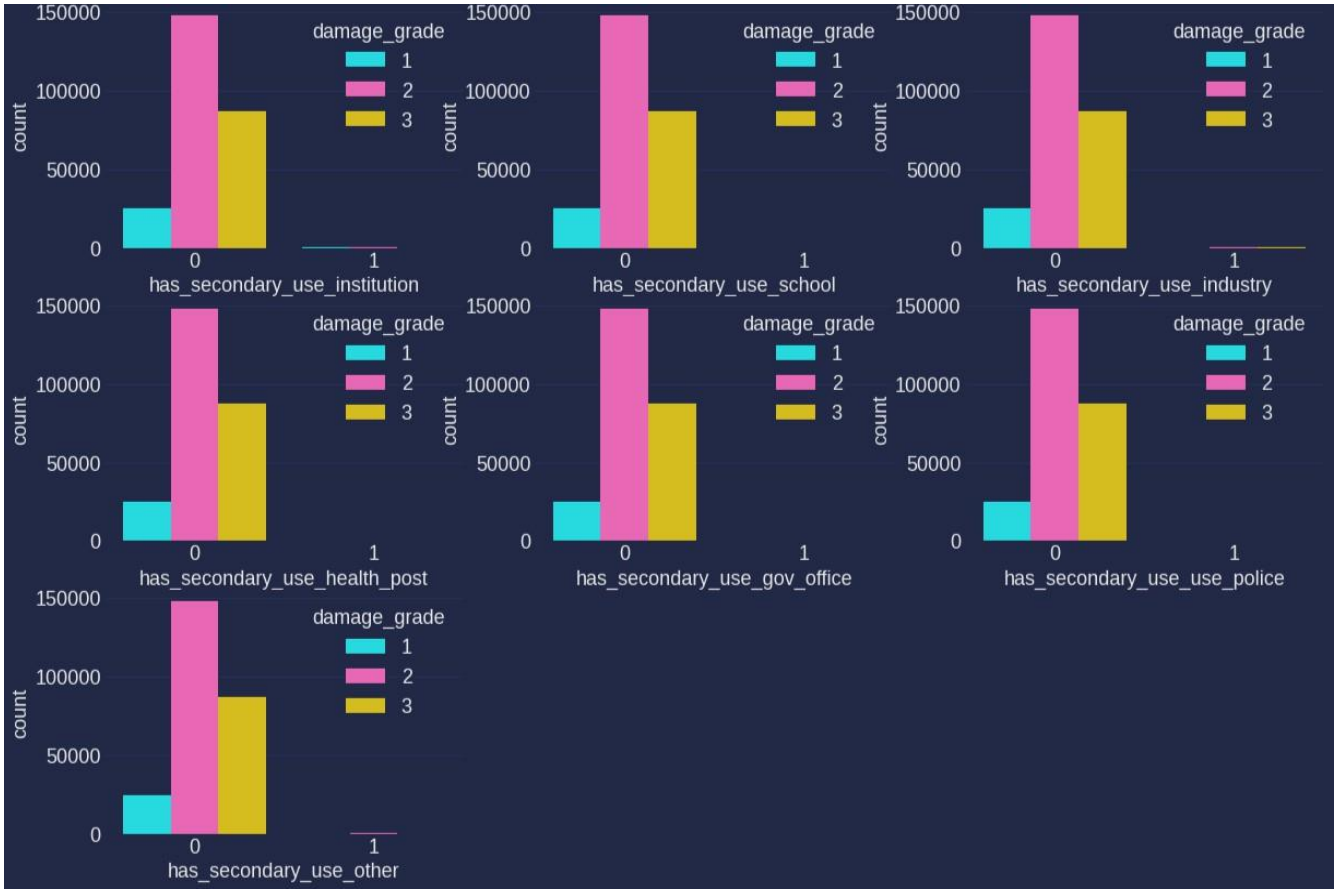


Fig 14 : plot damage grade distribution with respect to secondary usage and materials used

Category columns =land_surface_condition , foundation_type , roof_type , ground_floor_type , other_floor_type Position,plan_configuration and legal_ownership_status

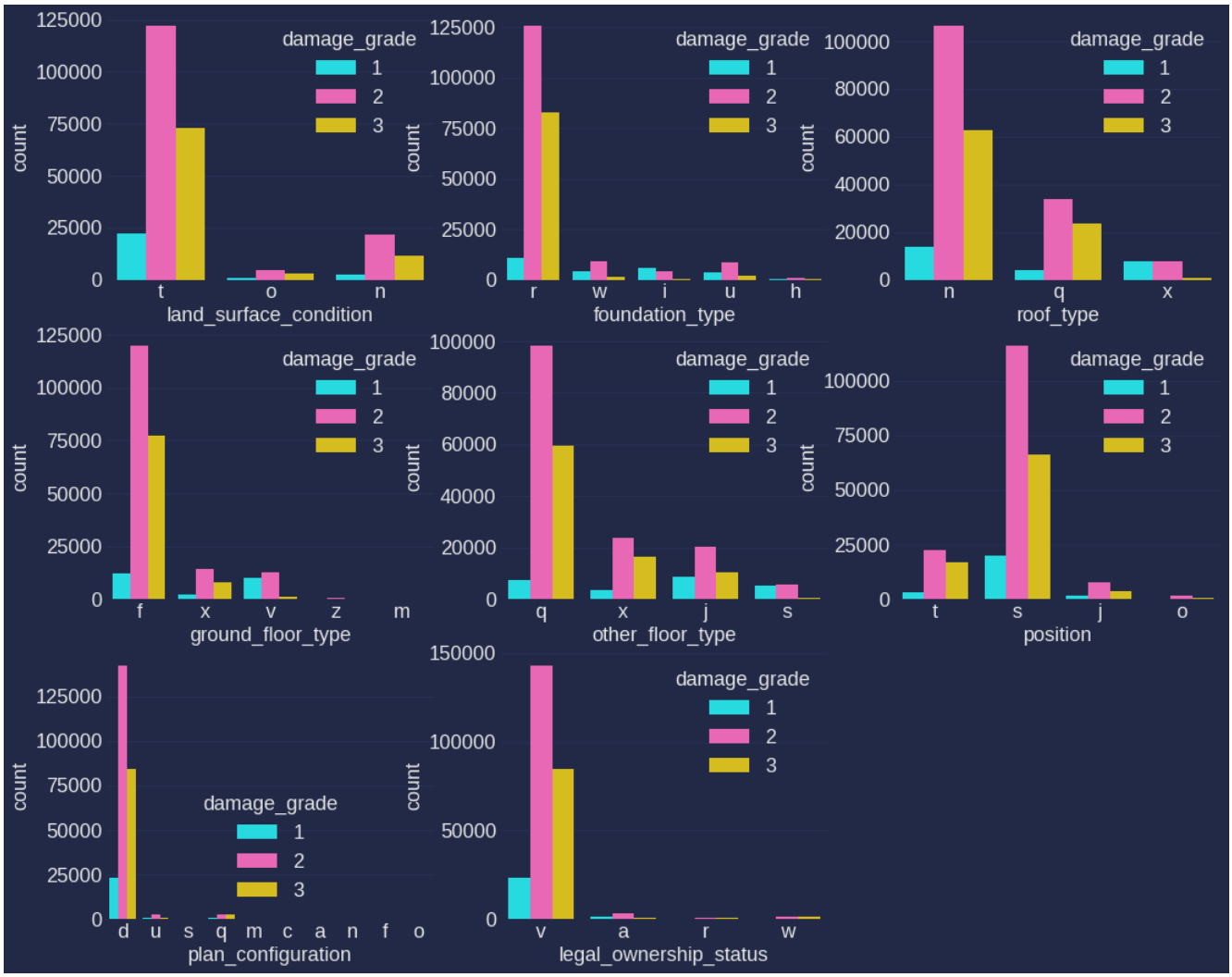


Fig 15: Plot damage grade distribution with respect to columns in category columns

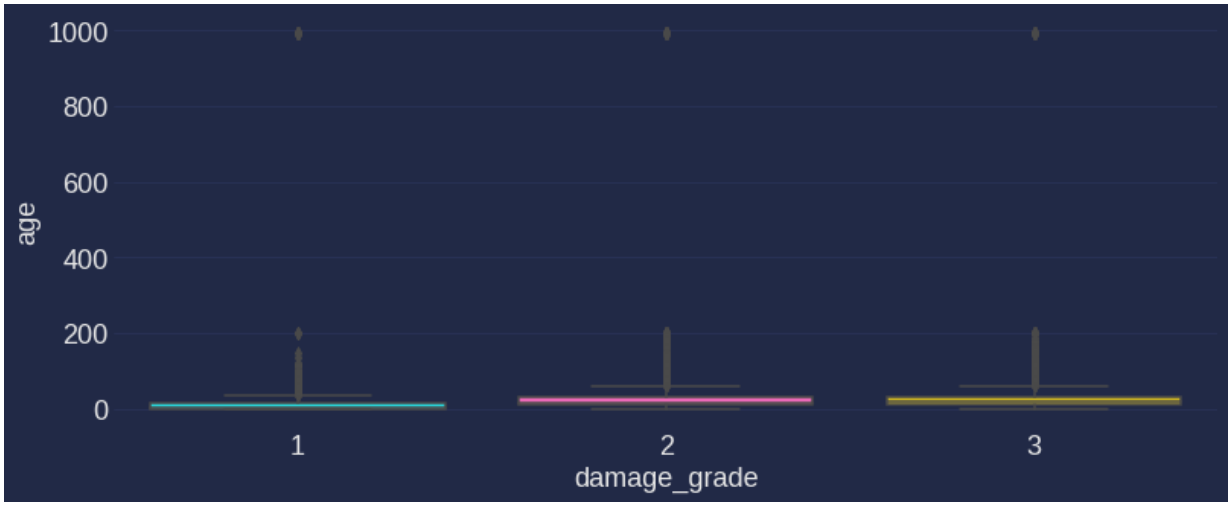


Fig 16: Box plot to find outliers in age column

The boxplot clearly indicates the presence of outliers in the age columns. so all buildings with age above 100 are removed from the data. After dropping the distribution of damage grade with respect to age is plotted

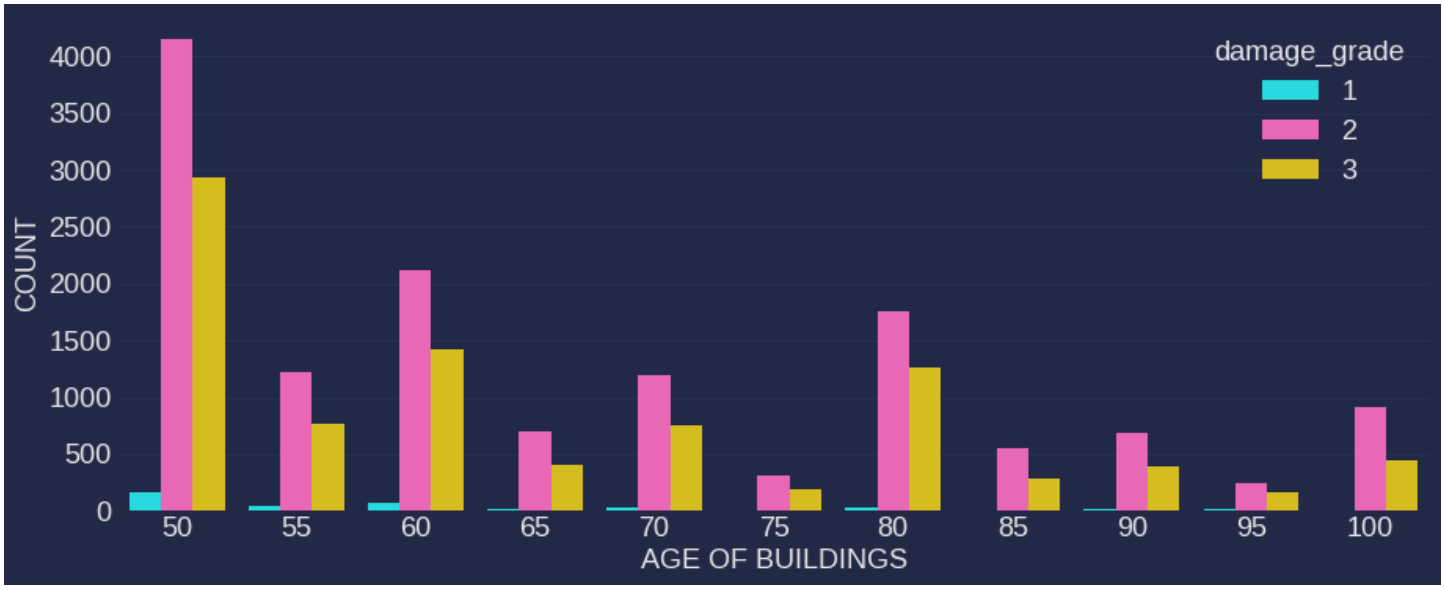


Fig 17: Distribution of damage grade with respect to age of buildings

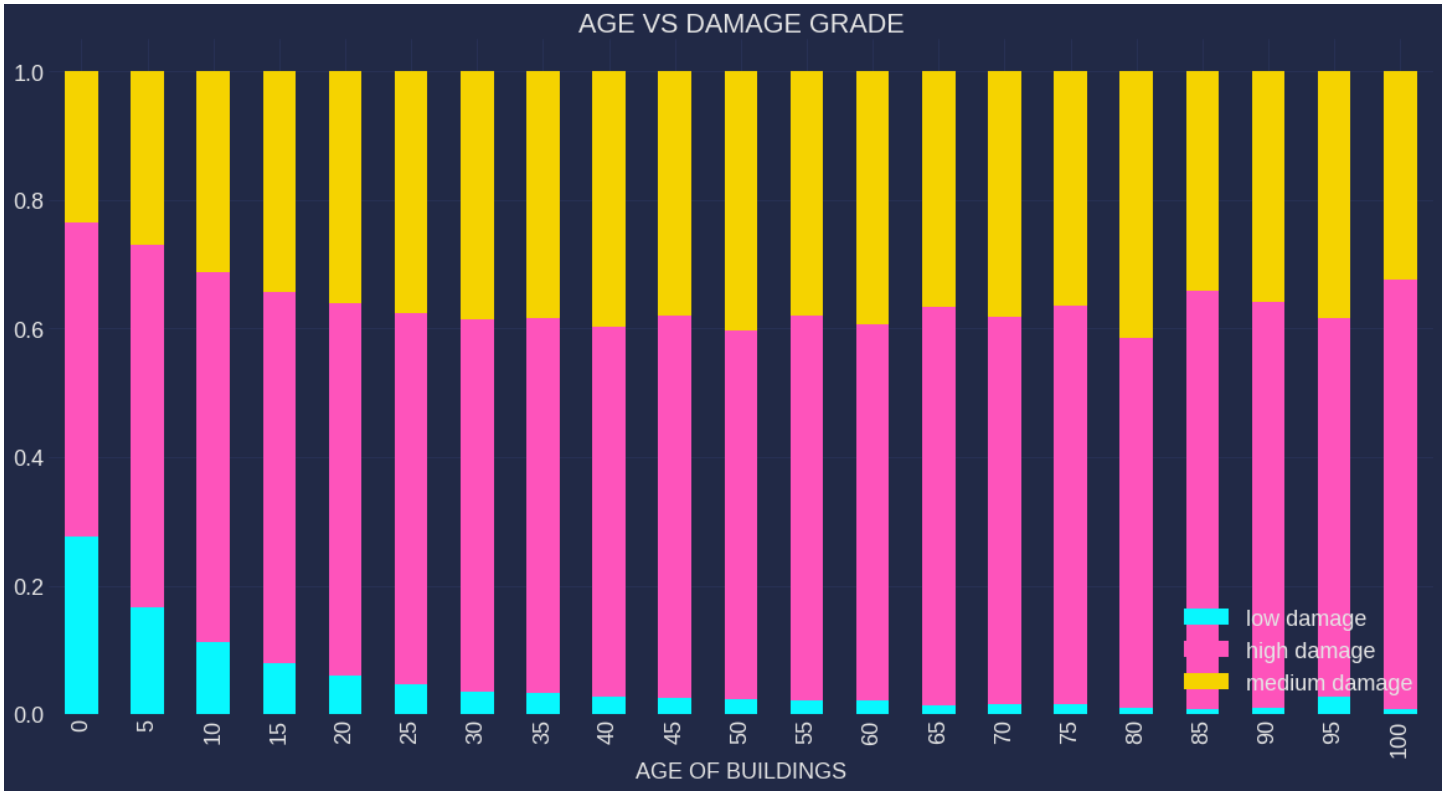


Fig 18: Distribution of damage grade with respect to age of buildings

The plot suggest damage increases in proportion with the age of buildings

FEATURE SELECTION

In feature selection first a function is created with data and threshold as the parameters to check for multicollinearity among the input variables. Then the function is called with the training dataset as data and 0.7 as threshold.the result is then stored in to the variable cf.the output of the function suggests that the columns has_secondary_usage_agriculture and height_percentage are highly correlated and fits our set parameters.the two columns are then

dropped from the training data. Further the columns building_id does not correlate with our output and would not have any significant effect on our training model, so it is also dropped from our dataset. After feature selection the categorical columns are label encoded and the dataset is split in test data and train data using train test split with train data having 80% and test data having 20%.after splitting data we move on to model creation

BUILDING THE MODEL

In the intial stage of building the model, models are created using Logistic regression , Decision tree classifier , random forest classifier ,knn and XGBclassifer using default parameters.after fitting the train data and prediction the test data using the created models ,accuracy and f1scoe of the models are compared

The classification report of the above mentioned models are as follows

Logistic regression :

	precision	recall	f1-score	support
1	0.44	0.07	0.13	4902
2	0.57	0.99	0.72	29430
3	0.00	0.00	0.00	17364
accuracy			0.57	51696
macro avg	0.34	0.35	0.28	51696
weighted avg	0.37	0.57	0.42	51696

Decision tree classifier:

	precision	recall	f1-score	support
1	0.49	0.52	0.50	4902
2	0.71	0.71	0.71	29430
3	0.62	0.62	0.62	17364
accuracy			0.66	51696
macro avg	0.61	0.61	0.61	51696
weighted avg	0.66	0.66	0.66	51696

Random forest classifier:

	precision	recall	f1-score	support
1	0.65	0.49	0.56	4902
2	0.73	0.83	0.77	29430
3	0.72	0.61	0.66	17364
accuracy			0.72	51696
macro avg	0.70	0.64	0.66	51696
weighted avg	0.72	0.72	0.71	51696

Knn:

	precision	recall	f1-score	support
1	0.58	0.52	0.55	4902
2	0.73	0.79	0.76	29430
3	0.69	0.62	0.65	17364
accuracy			0.71	51696
macro avg	0.67	0.64	0.65	51696
weighted avg	0.70	0.71	0.70	51696

XGBclassifier :

	precision	recall	f1-score	support
1	0.63	0.36	0.46	4902
2	0.66	0.88	0.76	29430
3	0.73	0.42	0.54	17364
accuracy			0.68	51696
macro avg	0.67	0.55	0.58	51696
weighted avg	0.68	0.68	0.65	51696

Two functions are created to compare the f1 score and accuracies of the models

```
Basic Logicistic Regression Model With Default Parameters:  0.5691155988857939
Decision Tree Classifier Model With Default Parameters:  0.6583488084184463
Basic Random Forest Classifier Model With Default Parameters:  0.7202104611575365
KNeighbours Model With Default Parameters:  0.7078690807799443
XGB Model With Default Parameters:  0.6750038687712783
```

Fig 19:F1 score comparison

```
Basic Logicistic Regression Model With Default Parameters:  0.5691155988857939
Decision Tree Classifier Model With Default Parameters:  0.6572268647477562
Basic Random Forest Classifier Model With Default Parameters:  0.7203845558650572
KNeighbours Model With Default Parameters:  0.7078690807799443
XGB Model With Default Parameters:  0.6750038687712783
```

Fig 20:F1 score comparison

For giving a graphical comparison of the above results two functions are created for comparing the accuracies and f1scores of the various models

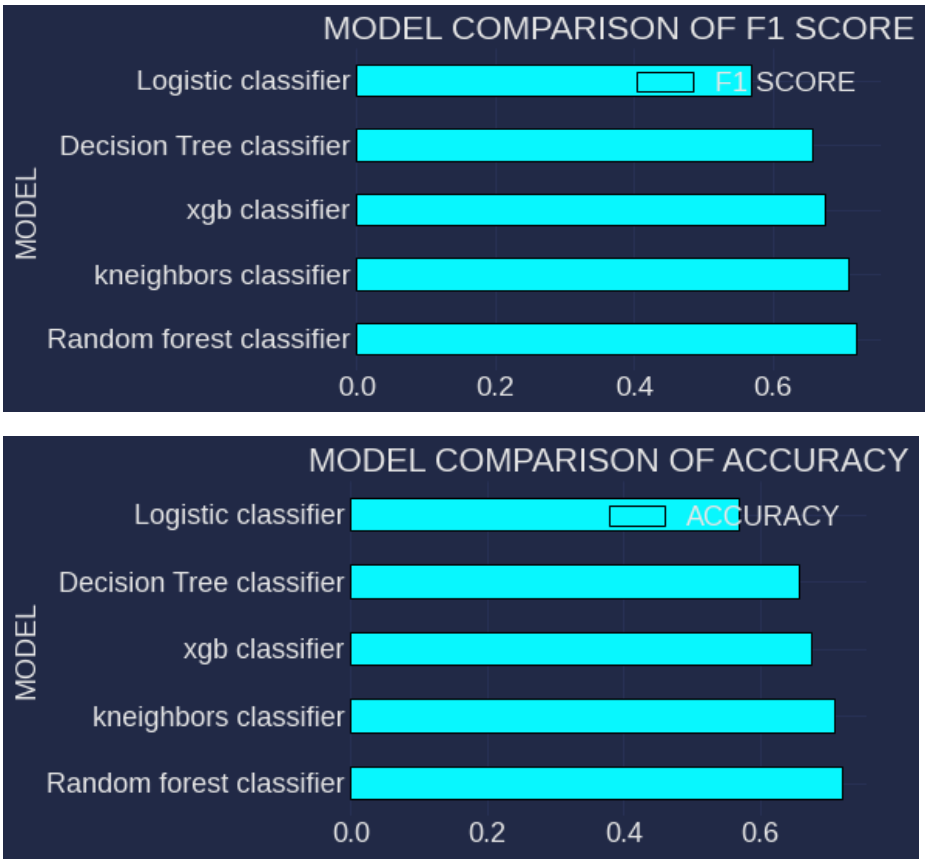


Fig 21:F1 score and accuracy comparison

From the comparison it is clear that random forest classifier model, knn and XGBclassifier models are giving the highest accuracies with default parameters. Now these three models are again created with tuned parameters .Random forest model is created with n_estimator as 500 and the model is used for prediction with test data. The classification report and confusion matrix is as follows

	precision	recall	f1-score	support
1	0.66	0.49	0.56	4902
2	0.73	0.83	0.78	29430
3	0.73	0.61	0.66	17364
accuracy			0.72	51696
macro avg	0.71	0.64	0.67	51696
weighted avg	0.72	0.72	0.72	51696

Fig 22: classification report of random classifier model

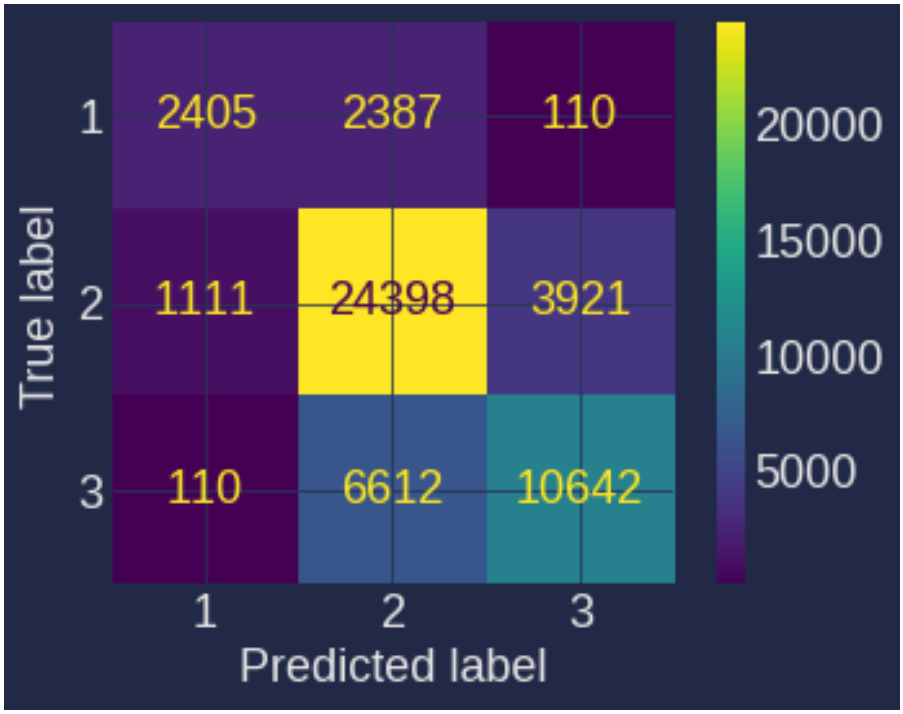


Fig 23: confusion matrix of random classifier model

Even after using tuned parameters the random forest classifier model is not significantly improving the performance of the model .Now XGBclassifier model is created with tuned parameters and performance is evaluated. XGBclassifier model is created with maximum number of concurrently running workers as 1 , max depth of 10 and learning rate of 0.1.

Classification report and confusion matrix of the prediction is as follows:

	precision	recall	f1-score	support
1	0.69	0.54	0.60	4902
2	0.75	0.85	0.79	29430
3	0.76	0.64	0.69	17364
accuracy			0.75	51696
macro avg	0.73	0.67	0.70	51696
weighted avg	0.75	0.75	0.74	51696

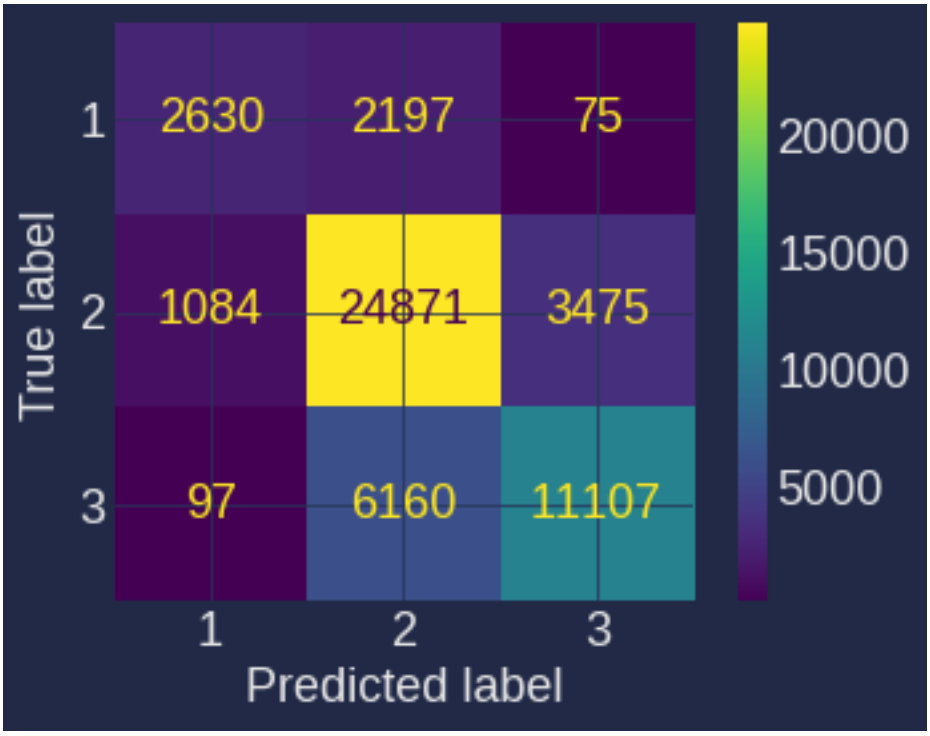


Fig 24:Classification report and confusion matrix of XGBclassifier

From the output it is clear that XGBclassifer model is showing significant improvement after using the hyper tuned parameters

CONCLUSION

The project successfully predicted different grades of damage to buildings due to earthquakes . Out of the various machine learning models used in the project random forest classification.knn and xgbclassifier were the most accurate in the initial portion of the project using default parameters. After using hyper tuned parameters accuracies of knn and random forest model did not show much improvement where as xgb model showed significant improvement in accuracy and f1score.xgb model predicted building damage with an accuracy of 75%

This model may assist stakeholders and decision-makers in rapid seismic risk assessment in order to formulate and implement new plans and policies in earthquake disaster risk reduction. Further investigation should be carried out for a better understanding of the applicability of the machine learning model in earthquake-induced rapid building damage prediction based on the need and interests of the decision-makers and stakeholders

CODE

Importing necessary modules

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split,RandomizedSearchCV,GridSe
archCV
from sklearn.metrics import f1_score,accuracy_score,confusion_matrix,classific
ation report
```

Setting the fontsyle

```
COLOR = 'black'
plt.rcParams['font.size'] = 21
plt.rcParams['text.color'] = COLOR
plt.rcParams['axes.labelcolor'] = COLOR
plt.rcParams['xtick.color'] = COLOR
plt.rcParams['ytick.color'] = COLOR
import mplcyberpunk
```

Importing datasets

```
train=pd.read_csv('train_values.csv')
labels=pd.read_csv('train_labels.csv')
```

Analysing the data

```
print(f"Train shape : {train.shape}")
print(f"Labels shape : {labels.shape}")
train.head()
labels.head()
train.info()
```

```
# Check if building id in labels matches building id in train
labels.building_id.equals(train.building_id)

# assign damage_grade in labels to train
train['damage_grade']=labels['damage_grade']

train.columns

print(f"Train shape : {train.shape}")

train.describe().T.style.background_gradient(cmap='plasma')
train.describe(include='object')
```

Code for checking correlation between dependant and independent variables

```
corr=train.corr()
plt.subplots(figsize=(15,15))
sns.heatmap(corr, xticklabels=True,yticklabels=True,  vmin=0, vmax=1,
            cmap='viridis',square=True)

fig, ax = plt.subplots(figsize=(4,10))
sns.heatmap(train.corr()[['damage_grade']].sort_values(by='damage_grade', asce
nding=False),vmin=-1, vmax=1, linewidths=1, linecolor='black', annot=True);


Output in fig 2
```

Check for null values in train dataset

```
train.isnull().sum()
```

Find out Unique values in all columns

```
train.nunique()
```

Code for EDA analysis

```
plt.figure(figsize=(8,8))
plt.style.use('cyberpunk')
sns.countplot(train['damage_grade'])
plt.xlabel('VALUES')
plt.ylabel('COUNTS')
plt.title('DAMAGE GRADE')
```

output in fig 3

```
labels=['Medium Damage','High Damage','Low Damage']
fig,ax1 = plt.subplots(figsize=(8,8))
plt.style.use('cyberpunk')
ax1.pie(list(train['damage_grade'].value_counts()/len(train['damage_grade'])*1
00), explode = (0.1,0,0), labels=labels, autopct='%1.1f%%', shadow=True)

plt.show()
```

output in fig 4

```
fig, axes = plt.subplots(ncols = 3, figsize = (20, 5))
plt.style.use('cyberpunk')
sns.histplot(train['geo_level_1_id'],  ax = axes[0])
sns.histplot(train['geo_level_2_id'],  ax = axes[1])
sns.histplot(train['geo_level_3_id'],  ax = axes[2])
```

output in fig 5

```
fig, axes = plt.subplots(ncols = 3, figsize = (20, 5))
plt.style.use('cyberpunk')
sns.boxplot(y=train['geo_level_1_id'], x=train['damage_grade'], ax = axes[0],p
alette='winter')
sns.boxplot(y=train['geo_level_2_id'], x=train['damage_grade'], ax = axes[1],p
alette='summer')
sns.boxplot(y=train['geo_level_3_id'], x=train['damage_grade'], ax = axes[2],p
alette='BuPu')
```

output in fig 6

```
continuous = ['geo_level_1_id','geo_level_2_id','geo_level_3_id','age','area_p
ercentage','height_percentage']
fig, axs = plt.subplots(1, 5, figsize=(20, 5))
for ax, col in zip(axs, continuous):
    sns.kdeplot(train[col], ax=ax, shade=True);
```

output in fig 7

```
continuous = ['geo_level_1_id','geo_level_2_id','geo_level_3_id','age','area_percentage',
'height_percentage']

def continuous_plot(continuous):
  fig = plt.figure(figsize=(18,16))
```

```
  plt.style.use('cyberpunk')
  for i,j in enumerate(continuous):
    ax = fig.add_subplot(3,2,i+1)
    sns.kdeplot(train.loc[train['damage_grade'] == 1, j], ax=ax, label='damage_grade==1')
    sns.kdeplot(train.loc[train['damage_grade'] == 2, j], ax=ax, label='damage_grade==2')
    sns.kdeplot(train.loc[train['damage_grade'] == 3, j], ax=ax, label='damage_grade==3')
  plt.legend()
  plt.show()
continuous_plot(continuous)
```

output in fig 8

```
colmns=[]
val=[]
binary = train.columns[train.columns.str.startswith('has')]
for bcol in binary:
  colmns.append(bcol)
  val.append(train[bcol].value_counts().sort_index().values)
pd.DataFrame(val, index=colmns)


binary = train.columns[train.columns.str.startswith('has')]

def binary_plot(binary):
  plt.rcParams['font.size'] = 15
  plt.style.use('cyberpunk')
  fig = plt.figure(figsize=(20,37))
  for i,j in enumerate(binary):
    ax = fig.add_subplot(8,3,i+1)
    labels=('NO','Yes')
    plt.pie(train[j].value_counts(),labels=labels,autopct="%.2f%%",shadow=True
,wedgeprops={'linewidth': 3.0, 'edgecolor': 'black'},
            textprops={'size': 'x-large','color':'white'})
    centre=plt.Circle((0, 0), 0.45,fc='black')
    plt.title(f'{j}')
    plt.gcf().gca().add_artist(centre)
  plt.show()
binary_plot(binary)
```

output in fig 9, fig10 and fig 11

```


binary = train.columns[train.columns.str.startswith('has')]

def binary_plot(binary):
  plt.rcParams['font.size'] = 18
  plt.style.use('cyberpunk')
```

```
  fig = plt.figure(figsize=(20,37))
  for i,j in enumerate(binary):
    ax = fig.add_subplot(8,3,i+1)
    sns.countplot(x=train[j], ax=ax, hue=train['damage_grade'])
  plt.show()
binary_plot(binary)
```

output in fig 12 , fig 13 and fig 14

```
categorical = train.select_dtypes(include=np.object).columns

def categorical_Plot(categorical):
  plt.rcParams['font.size'] = 18
  plt.style.use('cyberpunk')
  fig = plt.figure(figsize=(18,15))
  for i,j in enumerate(categorical):
    ax = fig.add_subplot(3,3,i+1)
    sns.countplot(x=train[j], ax=ax, hue=train['damage_grade'])
  plt.show()
categorical_Plot(categorical)
```

output in fig 15

```
plt.figure(figsize=(13,5))
sns.boxplot(y=train['age'], x=train['damage_grade'])
plt.figure(figsize=(15,4))
filt=train['age']<=100
train=train[filt]
data1=pd.crosstab(train.age,train.damage_grade)
data1.div(data1.sum(1).astype(float),axis=0).plot(kind='bar',stacked=True,figs
ize=(20,10))
plt.title('AGE VS DAMAGE GRADE')
plt.xlabel('AGE OF BUILDINGS')
plt.legend(['low damage','high damage','medium damage'],loc='lower right')
plt.show()
```

output in fig 16 , fig 17 and fig 18

FEATURE SELECTION

Defining a function for implementing feature selection

```
def correlation(data,thresh):
    col_corr=set()
```

```
    corr=data.corr()
    for i in range(len(corr.columns)):
        for j in range(i):
            if abs(corr.iloc[i,j])>thresh:
                col=corr.columns[i]
                col_corr.add(col)
    return col_corr
```

Setting the parameter, calling function and dropping the selected columns

```
cf=correlation(train,0.7)
train.drop(cf,axis=1,inplace=True)
test.drop(cf,axis=1,inplace=True)
X=train
X=X.drop(["damage_grade","building_id"],axis=1)
y=train["damage_grade"]
```

LABEL ENCODING

```
label_encoding_columns=['land_surface_condition', 'foundation_type', 'roof_type',
       'ground_floor_type', 'other_floor_type', 'position',
       'plan_configuration', 'legal_ownership_status']

for i in label_encoding_columns:
    X[i]=X[i].astype("category")
    X[i]=X[i].cat.codes

from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test = train_test_split(X,y,test_size = 0.20,random
 state = 42)
```

Logistic Regression :

```
lr = LogisticRegression(multi_class='multinomial')
lr.fit(x_train,y_train)
lr_pred=lr.predict(x_test)
print(classification_report(y_test, lr_pred))
```

Decision tree Classifier :

```
dc = DecisionTreeClassifier()
dc.fit(x_train,y_train)
dc_pred=dc.predict(x_test)
print(classification_report(y_test, dc_pred))
```

Randomforest Classifier:

```
rc = RandomForestClassifier()
rc.fit(x_train,y_train)
rc_pred=rc.predict(x_test)
print(classification_report(y_test, rc_pred))

Output:
```

KNeighborsClassifer:

```
knc = KNeighborsClassifier()
knc.fit(x_train,y_train)
knc_pred=knc.predict(x_test)
print(classification_report(y_test, knc_pred))
```

XGBClassifier :

```
xg = XGBClassifier()
xg.fit(x_train,y_train)
xg_pred=xg.predict(x_test)
print(classification_report(y_test, xg_pred))
```

Define a function to compare f1 score of different models

```
def check_model_f1_score(model):
  model.fit(x_train, y_train)
  test_y_pred = model.predict(x_test)
  return f1_score(y_test, test_y_pred, average='micro')

lr_score = check_model_f1_score(lr)
print('Basic Logicistic Regression Model With Default Parameters: ', lr_score)
dc_score = check_model_f1_score(dc)
print('Decision Tree Classifier Model With Default Parameters: ', dc_score)
rc_score = check_model_f1_score(rc)
print('Basic Random Forest Classifier Model With Default Parameters: ', rc_sco
re)
knc_score = check_model_f1_score(knc)
print('KNeighbours Model With Default Parameters: ', knc_score)
xg_score = check_model_f1_score(xg)
print('XGB Model With Default Parameters: ', xg_score)
```

```
plt.figure(figsize=(12,8))
model_comparison = pd.DataFrame({'MODEL':['Logistic classifier','Decision Tree classifier
', 'Random forest classifier', 'kneighbors classifier','xgb classifier'], 'F1 SCORE':[lr_
score,dc_score,rc_score,knc_score,xg_score]})
plt.style.use('cyberpunk')
model_comparison.sort_values('F1 SCORE', ascending = False).plot(x = 'MODEL', y = 'F1 SCO
RE', kind = 'barh', edgecolor = 'black')
plt.title('MODEL COMPARISON OF F1 SCORE')
```

Define a function to compare accuracy of different

```
def check_model_accuracy(model):
  model.fit(x_train, y_train)
  test_y_pred = model.predict(x_test)
  return accuracy_score(y_test, test_y_pred)

lr_score = check_model_accuracy(lr)
print('Basic Logicistic Regression Model With Default Parameters: ', lr_score)
dc_score = check_model_accuracy(dc)
print('Decision Tree Classifier Model With Default Parameters: ', dc_score)
rc_score = check_model_accuracy(rc)
print('Basic Random Forest Classifier Model With Default Parameters: ', rc_score)
knc_score = check_model_accuracy(knc)
print('KNeighbours Model With Default Parameters: ', knc_score)
xg_score = check_model_accuracy(xg)
print('XGB Model With Default Parameters: ', xg_score)


plt.figure(figsize=(12,8))
model_comparison = pd.DataFrame({'MODEL':['Logistic classifier','Decision Tree classifier',
                    'Random forest classifier', 'kneighbors classifier',
                    'xgb classifier'], 'ACCURACY':[lr_score,dc_score,rc_score,
                                                   knc_score,xg_score]})
plt.style.use('cyberpunk')
model_comparison.sort_values('ACCURACY', ascending = False).plot(x = 'MODEL', y = 'ACCURACY', kind
= 'barh', edgecolor = 'black')
plt.title('MODEL COMPARISON OF ACCURACY')
```

Model creation with parameters

```
rf = RandomForestClassifier(n_estimators = 500,random_state = 1, max_depth=None,n_jobs=1)

rf.fit(x_train,y_train)
```

```
rf_pred=rf.predict(x_test)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
cm=confusion_matrix(y_test,rf_pred)
print(classification_report(y_test, rf_pred))

plot_confusion_matrix(rf, x_test, y_test);
```

Output in fig 23

```
clf=XGBClassifier(n_jobs=-
1,n_estimators= 600, max_depth= 10,learning_rate= 0.1)

clf.fit(x_train,y_train)
xgb_pred=clf.predict(x_test)

plot_confusion_matrix(clf, x_test, y_test);

print(classification_report(y_test,xgb_pred))
```

Output in fig 24: