EARTHQUAKE PREDICTION MODEL

A MAJORPROJECT REPORT

submitted

in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

KARNATI KALYAN 17B81A0585

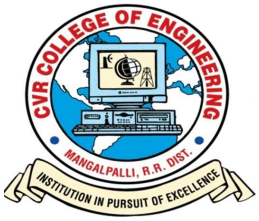
GODELA KARTHIK SAITEJA 17B81A0587

CHALLA KEERTHAN REDDY 17B81A0590

Under the guidance of

V.N.V.L.S. SWATHI

Assistant Professor, CSE Department



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CVR COLLEGE OF ENGINEERING

(***An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad***)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),

Rangareddy (D), Telangana- 501 510

MAY,2021

CVR COLLEGE OF ENGINEERING

(An UGC Autonomous Institute, Accredited by NAAC with 'A' Grade)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled **"EARTHQUAKE PREDICTION MODEL"** that is being submitted by **KARNATI KALYAN 17B81A0585, GODELA KARTHIK SAITEJA 17B81A0587, CHALLA KEERTHAN REDDY 17B81A0590** in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineeringto the CVR College of Engineering, is a record of bonafide work carried out by them under my guidance and supervision during the year 2020-2021.

The results embodied in this project work has not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of the project guide

V. N. V. L. S. Swathi

Assistant Professor, CSE

Signature of the HOD

Dr.A.Vani Vathsala

HOD, CSE

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

It is great pleasure to convey our profound sense of gratitude to the Principal **Dr. Nayanathara K Sattiraju**, Vice-Principal **Prof. L.C. Siva Reddy**, **Dr. A Vani Vathsala ,**Head of CSE Department, CVR College of Engineering, forbeing kind enough for arranging necessary facilities for executing the project in the college.

We shall remain grateful to **Ms. V. N. V. L. S Swathi,** Assistant Professor, CSE for providing us strong atmosphere by enforcing strict discipline to do the project with utmost concentration and dedication.

We deem it a pleasure to acknowledge our sense of gratitude to our internal project guide **Ms. V.N.V.L.S Swathi,** Assistant Professor, CSE under whom we have carried out the project work. His incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish a deep sense of gratitude and heartfelt thanks to management for providing excellent lab facilities and tools. Finally, we thank all those whose guidance helped us in this regard.

With Regards,

KARNATI KALYAN, 17B81A0585

GODELA KARTHIK SAITEJA, 17B81A0587

CHALLA KEERTHAN REDDY, 17B81A0590

ABSTRACT

An earthquake is a natural disaster known on account of the devastating effect it has on naturally occurring structures and manmade structures such as buildings, bungalows and residential locations to name a few. Earthquakes are measured using seismometers, that detect the vibrations due to seismic waves travelling through the earth's crust. In this work, the damage that is caused by an earthquake was classified into damage grades, ranging in values from one to five. A previously acquired data set was used, wherein a series of parameters were taken into consideration to predict the damage grade of a given building, which is associated with a Unique Identification String. The prediction was done using a survey of existing machine learning classifier algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Motivation

An earthquake is a calamitous occurrence that is detrimental to human interest and has an undesirable impact on the environment. Earthquakes have always caused in calculable damage to structures and properties and caused the deaths of millions of people throughout the world. In order to minimize the impact of such an event, several national, international and transnational organizations take various disaster detection and prevention measures. Time and quantity of the organization's resources are limiting factors, and organization managers face several difficulties when it comes to the distribution of the resources.Leveraging the power of machine learning is a viable option to predict the degree of damage that is done to buildings.

1.2 Problem statement

Leveraging the power of machine learning is a viable option to predict the degree of damage that is done to buildings. It can help identify safe and unsafe buildings which helps to predict damage prone areas and thus avoiding death and injuries resulting from an earthquake, while simultaneously making rescue efforts efficient.This is done by classifying these structures on a damage grade scale based on various factors like its age, foundation, number of floors, material used and several other parameters. Then the number of families and the probable casualties ward-by-ward in a district are taken into account.This enables distribution of relief forces proportionately ward-wise and its prioritization based on the extent ofdamage.

1.3 Project objectives

Earthquake are quite fatal and can cause quite a loss. Those that occur in the workplace can cause harm to employees, environment and damage to the equipment. Industrial related accidents, injuries and fatality data demonstrate that continued efforts and effective measures are necessary to reduce the number of industrial accidents, illnesses and and fatalities.This prediction can help identify safe and unsafe buildings which helps to predict damage prone areas and thus avoiding death and injuries resulting from an earthquake, while simultaneously making rescue efforts efficient.

1.4 Project organization

i. This report is divided into 7 chapters after this introductory chapter.
ii. Chapter 2 gives insights about proposed model, introduces characteristics of the problem and design challenges.
iii. Chapter 3 summarizes functional, non-functional requirements and system requirements along with software and hardware specifications.
iv. Chapter 4 deals with analysis and design of the proposed model which includes system architecture and technology description.
v. Chapter 5 encloses Implementation of the proposed model and testing with different scenarios.
vi. Chapter 6 includes conclusion and future work.
vii. Chapter 7 includes references.

2. LITERATURE SURVEY

2.1 Existing work

In [1], a quick assessment method for earthquake emergency is introduced. The method contains two different modes to obtain damage information from remote sensing images, one of which is based on damage index and the other adopts image classification. The damage index mode relies on traditional visual interpretation. After the damage index is given by experts, the ground intensity data can be gained, and then loss estimate parameters will be acquired from the experiential vulnerability matrix. The image classification mode is an application of digital image processing technique. Those loss estimate parameters can be calculated from the classification result which is sorted by the type of buildings and ranged by the damage degree. While the assessment models are introduced, the action of multi-resourced estimate data is explained to show how to find parameters in various data.

In [2], a probabilistic aspect of the earthquake prediction was given. A theoretical analysis of the earthquake prediction problem in space–time is presented. We find an explicit structure of the optimal strategy and its relation to the generalized error diagram. This study is an extension of the theoretical results for time prediction. The possibility and simplicity of this extension is due to the choice of the class of goal functions. The generalized error diagram allows us to suggest a natural measure of prediction efficiency at the research stage.

In [3], core idea of this work is to predict whereas an event is classified as negative or positive major earthquake by applying different machine learning algorithms. It is well known that there is no best algorithm or one solution that fits all the problems and datasets for machine learning since the performance of algorithms depends on many factors. Some algorithms are best for small data, while others perform better for a tons of data sample. Some algorithms require categorical inputs, while others need quantitative.

Another important criterion while choosing the algorithm is the complexity of the dataset and how many features the model needs to learn and predict. This is why, in this work, eight different algorithms have been applied on an earthquake dataset, namely: Random Forest, Naïve Bayes, Logistic Regression, MultiLayer Perceptron, AdaBoost, K-nearest neighbors, Support Vector Machine, and Classification and Regression Trees. For each selected model, various hyperparameters have been tested, and obtained prediction results have been fairly compared using various metrics, leading to a reliable prediction of major events for 3 of them.

In [4], it considers the damage prediction in each district in Kagoshima Prefecture by using a two-stages predictor. It consists of LRM (linear regression model) at the first stage and NN (neural networks) at the second stage. This predictor enables us to predict the number of damaged distribution poles and lines from weather forecasts of typhoon. Effectiveness of the approach is assured by applying it to the actual data.

In [5] it proposes flood prediction modelling to overcome the nonlinearity problem and come out with advanced neural network technique for the prediction of flood water level 5 hours in advance. The input and output parameters used in this model are based on real-time data obtained from Department of Irrigation and Drainage Malaysia upon special request. Results showed that the Improved NARX model successfully predicted the flood water level 5 hours ahead of time and significant improvement can be observed from the original NNARX model.

In [6], core idea of this work is to major earthquake by applying fuzzy analysis. Fuzzy Logic is an approach to variable processing that allows for multiple values to be processed through the same variable. Fuzzy logic attempts to solve problems with an open, imprecise spectrum of data that makes it possible to obtain an array of accurate conclusions. Fuzzy logic is designed to solve problems by considering all available information and making the best possible decision given the input.

2.2 Limitations of Existing work

The prediction was only about occurance of earthquake but not the impact of the earthquake.There are less models which predict the magnitude of earthquake.Even though magnitude is predicted each individual building damage must be predicted for evacuation.Magnitude prediction may result in more error.Predicting whether the building is safe or unsafe would help in rescue and evacuation process.

3. SOFTWARE AND HARDWARE SPECIFICATIONS

3.1 Software requirements

Operating system	Windows
Language	Python 3.x
Libraries	i. Pandas ii. Numpy iii. Seaborn iv. Matpotlib v. Scipy vi. Sklearn
IDE	Jupyter or Google colab
Internet	Required

Table 3.1 Software Requirements

3.1.1 Functional Requirements

- The system should process the input given by user.
- System shall show the error message to the user when input given is not in the required format.
- System should preprocess the data given by user.
- System should able the predict the data required.

3.1.2 Non Functional Requirements

- **Availability:** This system will predict the damage grade of building.

- **Functionality:** This software will deliver on the functional requirements mentioned.
- **Reliability:** This software will work reliably for every kind of dataset.
- **Learning ability:** Easy to use.

3.2 Hardware requirements

Processor	i3 and above
RAM	1GB(min)
Hard Disk	5GB

Table 3.2 Hardware Requirements

4. DESIGN

4.1 Class diagram

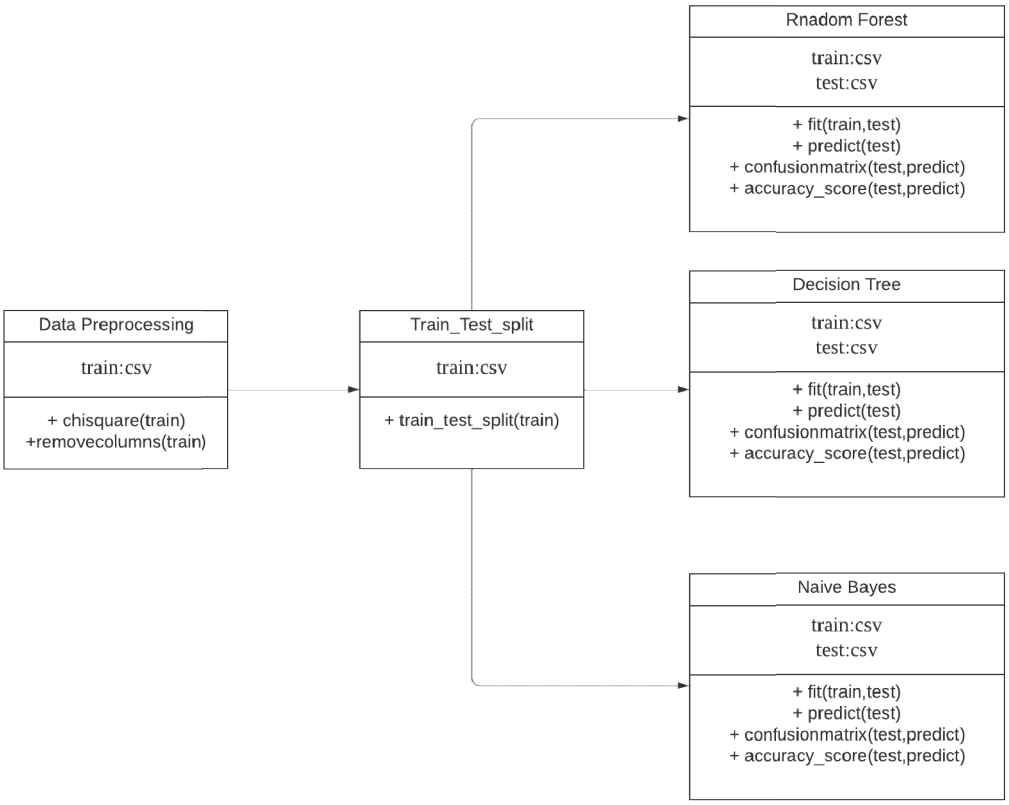


Fig 4.1 Class diagram

4.2 Usecase diagram

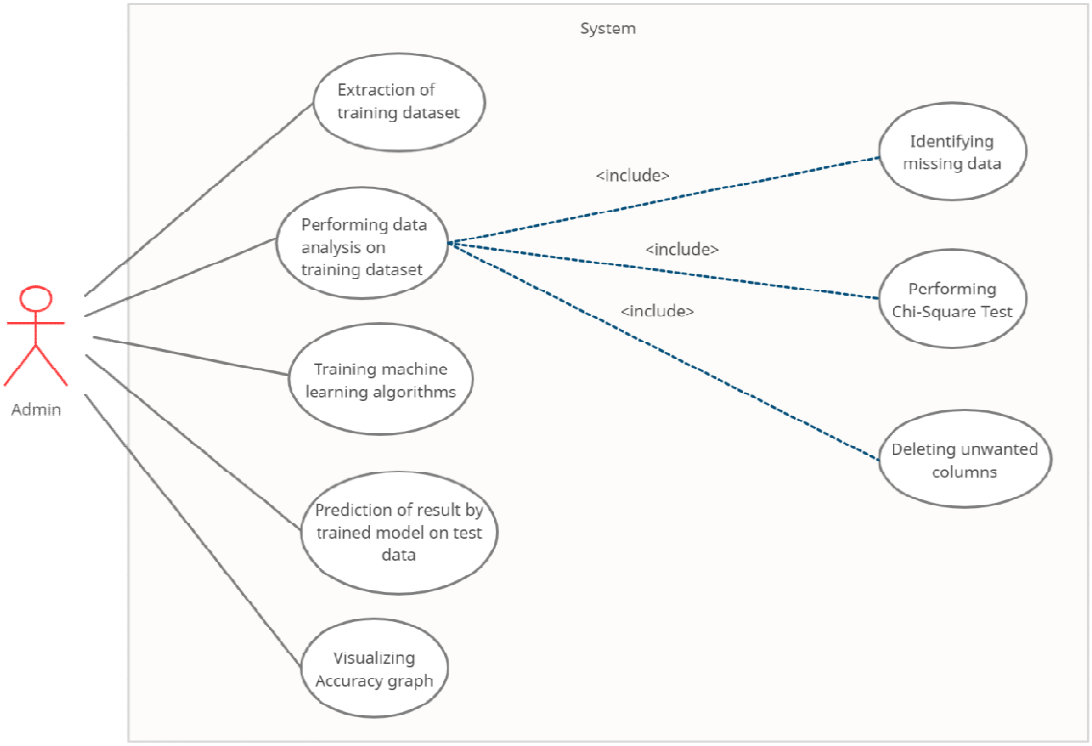


Fig 4.2 Use Case Diagram

The actor(admin) extracts the data performs exploratory data analysis on it. As the data is imbalanced,chi-square test is performed. In chi square test, we get the unwanted columns and we remove them.After that it is with machine learning algorithms and prediction of result on test data.Finally a graph which shows the accuracy of different models is displayed.

4.3 Activity diagram

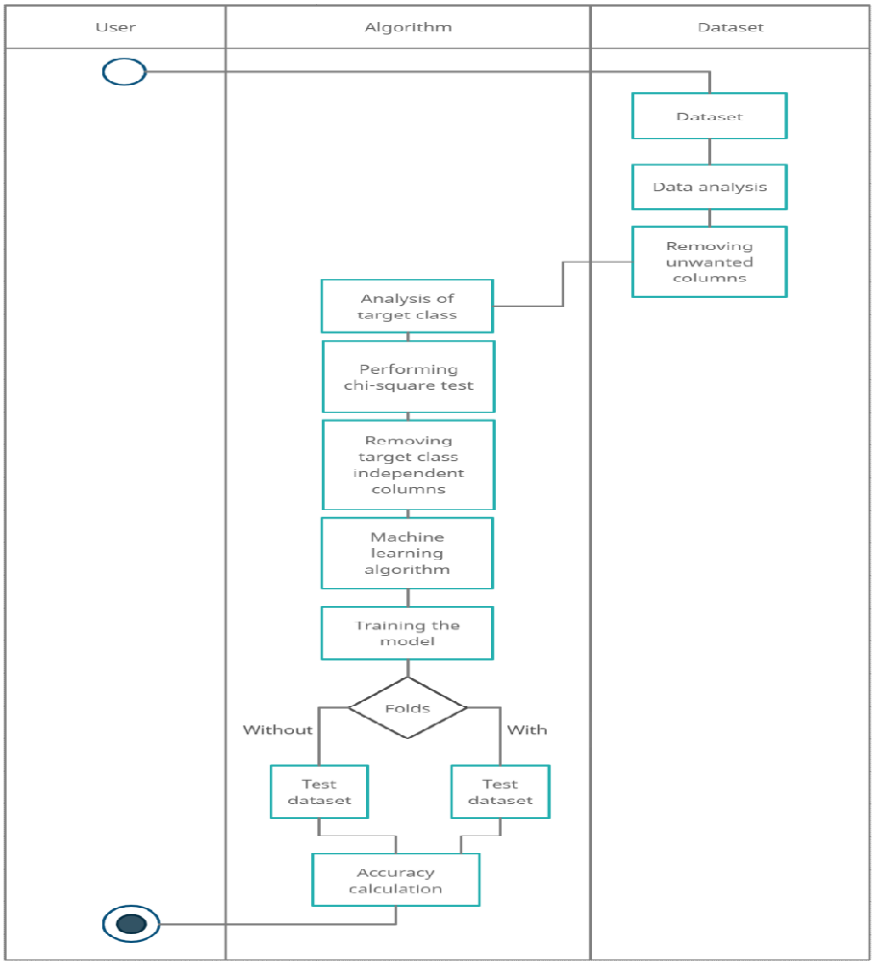


Fig 4.3 Activity Diagram

The above diagram represents the flow of the project where at the first the data is extracted from Kaggle, then preprocessing is done on the data. This data is given to the selected trained model to predict the damage grade of building.

4.4 Sequence diagram

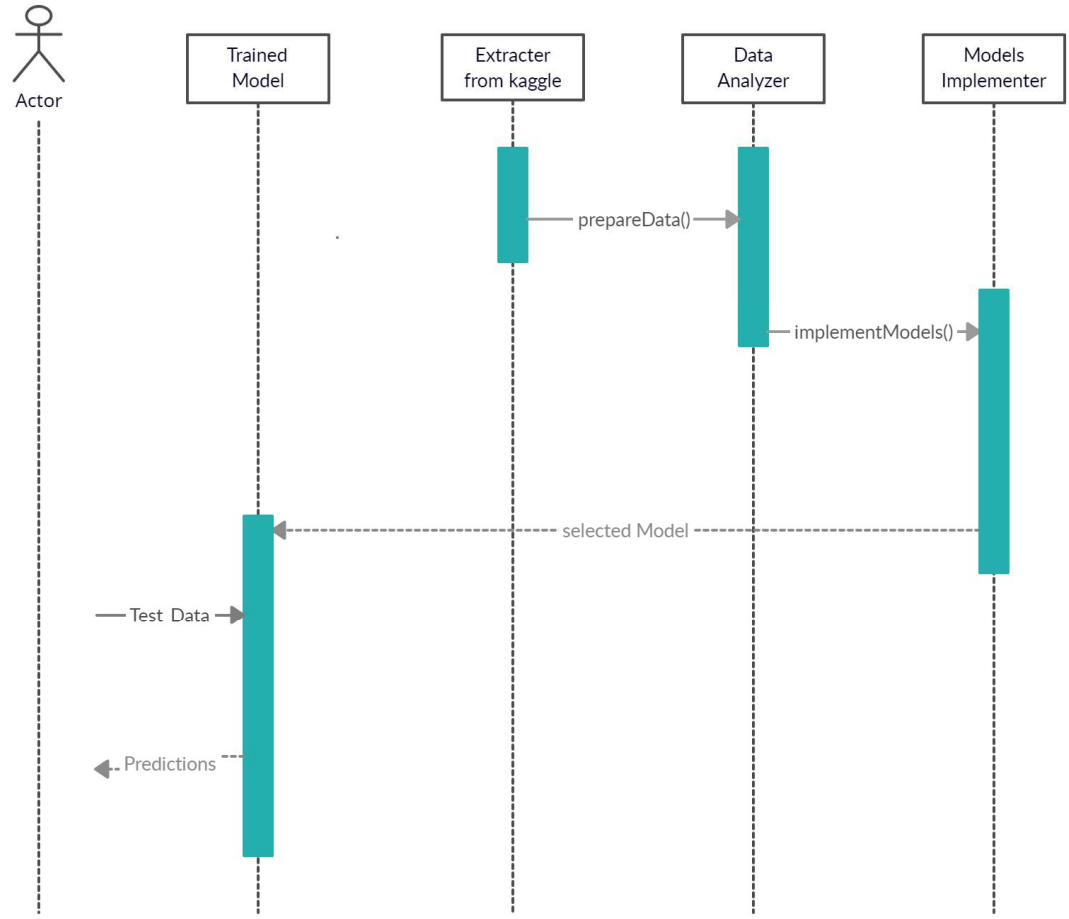


Fig 4.4 Sequence Diagram

4.5 System architecture

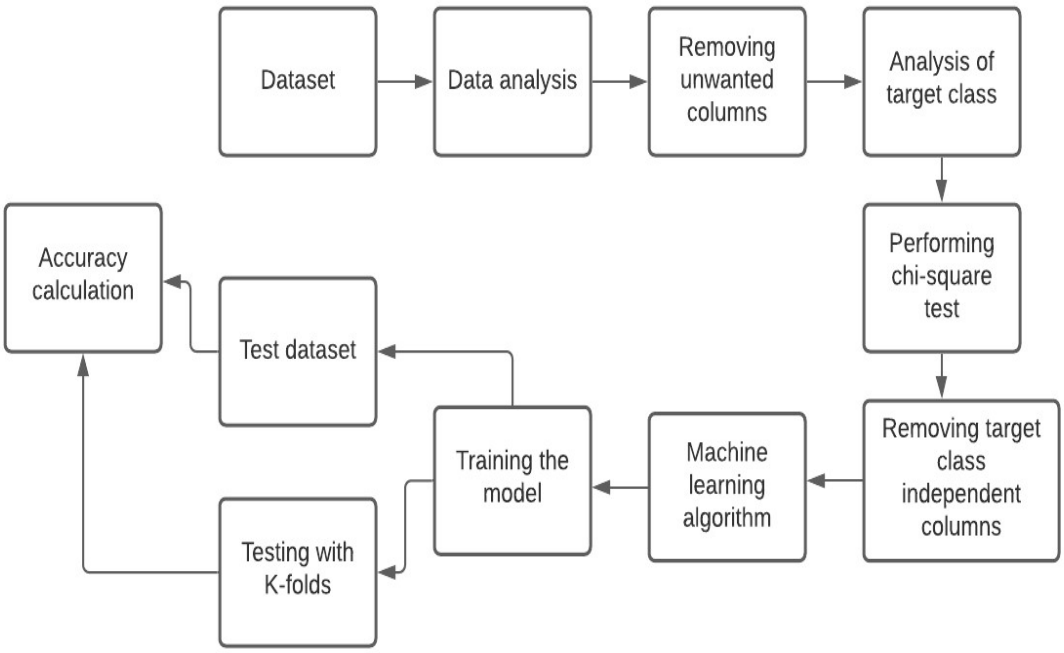


Fig 4.5 Architecture Diagram

The architecture diagram describes the overall view of the project right from its components to its final stages. At the first the data is extracted then EDA is performed. Removed unwanted columns by performing chi-square.Then are passed to machine learning models with and without folds.

4.6 Technology description

4.6.1 Python

Python is widely used in the software development industry. There are many of reasons for this.

i) **High-level object-oriented programming language:** Python includes effective symbolism.
ii) **Rapid application development:** Because of its concise code and literal syntax, the development of applications gets accelerated. The reason for its wide usability is its simple and easy-to-master syntax. The simplicity of the code helps reduce the time and cost of development.
iii) **Dynamic typescript:** Python has high-level incorporated data structures blended with dynamic typescript and powerful binding.

Some of the unique features that make Python the most ubiquitous language among the developer community are:

a) Python supports code reusability and modularity.
b) It has a quick edit-inspect-debug cycle.
c) Debugging is straightforward in Python programs.
d) It has its own debugger written in Python itself, declaring to Python's reflective power.
e) Python includes a plethora of third-party components present in the Python Package Index (PyPI).

4.6.2 Jupyter

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data.

JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning.

JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

We can run Jupyter on Windows, Linux, or Mac OS. Additionally, it contains modules and packages that help programmers develop software using Python in less time and with minimal effort. Further, it can also be customized according to the requirements of developers.

Features of Jupyter:

i) Intelligent Code Editor

ii) Code Navigation

iii) Refactoring

iv) Assistance for Many Other Web Technologies

v) Support for Popular Python Web Frameworks

vi) Assistance for Python Scientific Libraries

4.6.3 Google colab

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science. It may also have a long term perspective of building a customer base for Google Cloud APIs which are sold per-use basis.

4.6.4 CSV

CSV (Comma Separated Values) is a simple **file format** used to store tabular data, such as a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

5 IMPLEMENTATION AND TESTING

5.1 Implementation

Implementation is in 7 phases

1) Importing required Modules.
2) Extracting the data from Kaggle.
3) Analyzing the data using Exploratory Data Analysis.
4) Removal of unwanted columns .
5) Splitting train and test data.
6) Implementation of different models.
7) Visulazing confusion matrix.

5.1.1 Modules

What is pandas?

Pandas stands for “Python Data Analysis Library”, Pandas is an open-source library that is built on top of NumPy library. It is a python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier and mainly used for data manipulation and analysis.it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software.

What is numpy?

NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. It is an open-source project and you can use it freely. NumPy stands for Numerical Python.

Why use numpy?

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important

What is matplotlib?

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and Ipython shells, Jupyter notebook and web application servers also.

What is warnings?

Warningmessages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn't warrant raising an exception and terminating the program. For example, one might want to issue a warning when a program uses an obsolete module. Python programmers issue warnings by calling the warn() function defined in this module.

What is sklearn?

Scikit-learn (formerly scikits. Learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

What is seaborn?

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn provides an API on top of Matplotlib that offers sane choices for plot style and color defaults, defines simple high-level functions for common statistical plot types, and integrates with the functionality provided by pandas data frames**.**

What is Scipy?

SciPy in Python is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands. SciPy is built on the Python NumPy extention. SciPy is also pronounced as “Sigh Pi.”

Why use Scipy?

- SciPy contains varieties of sub packages which help to solve the most common issue related to Scientific Computation.
- SciPy package in Python is the most used Scientific library only second to GNU Scientific Library for C/C++ or Matlab’s.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of NumPy library.

5.1.2 Dataset

The dataset used in this project is downloaded from Kaggle[7]. It is a dataset containing 4 csv files:

a.)train.csv

b.)Building_structure.csv

d.)Building_ownership_use.csv

Table 5.1 train.csv

Variable	Description
area_assesed	Indicates the nature of the damage assessment in terms of the areas of the building that were assessed.
Damage_grade	Damage grade assigned to the building after assessment (Target Variable)
building_id	A unique ID that identifies every individual building
district_id	District where the building is located
has_geotechnical_risk	Indicates if building has geotechnical risks
has_geotechnical_risk_fault_crack	Indicates if building has geotechnical risks related to fault cracking
has_geotechnical_risk_flood	Indicates if building has geotechnical risks related to

	flood
has_geotechnical_risk_land_settlement	Indicates if building has geotechnical risks related to land settlement
has_geotechnical_risk_landslide	Indicates if building has geotechnical risks related to landslide
has_geotechnical_risk_liquefaction	Indicates if building has geotechnical risks related to liquefaction
has_geotechnical_risk_other	Indicates if building has any other geotechnical risks
has_geotechnical_risk_rock_fall	Indicates if building has geotechnical risks related to rock fall
has_repair_started	Indicates if the repair work had started
vdcmun_id	Municipality where the building is located

Table 5.2 Building_structure.csv

Variable	**Description**
building_id	A unique ID that identifies every individual building
district_id	District where the building is located
vdcmun_id	Municipality where the building is located
ward_id	Ward number in which the building is located
count_floord_pre_eq	Number of floors that the building had before the earthquake
count_floors_post_eq	Number of floors that the building had after the earthquake
age_building	Age of building(in years)
plinth_area_sq_ft	Plinth area of the building(in square feet)
height_ft_pre_eq	Height of the building before the earthquake(in feet)

height_ft_post_eq	Height of the earthquake after the earthquake(in feet)
land_surface_condition	Surface condition of the land in which the building is built
foundation_type	Type of foundation used in the building
roof_type	Type of roof used in the building
ground_floor_type	Type of construction used in other floors
other_floor_type	Type of construction used in other floors
postion	Postion of the building
plan_configuration	Building plan configurationm
has_superstructure_adobe_mud	Indicates if the superstructure of the building is made of Abode/Mud
has_superstructure_mud_mortar_stone	Indicates if the superstructure of the building is made of mud mortar
has_superstructure_cement_mortar_brick	Indicates if the superstructure of the building is made of Mud Mortar- Brick
has_superstructure_timber	Indicates if the superstructure of the building is made of Timber
has_superstructure_bamboo	Indicates if the superstructure of the building is made of Bamboo
has_superstructure_rc_non_engineered	Indicates if the superstructure of the building is made of RC(Non Engineered)
has_superstructure_rc_engineered	Indicates if the superstructure of the building is made of RC(Engineered)
has_superstructure_other	Indicates if the superstructure of the building is made of any other material
condition_post_eq	Actual condition of the building after the earthquake

Table 5.3 Building_ownership_use.csv

Variable	Description
building_id	A unique ID that identifies every individual building
district_id	Distrcit where the building is located
vdcum_id	Municipality where the building is located
ward_id	Ward Number in which the building is located
legal_ownership_status	Legal ownership status of the land in which the building was built
count_families	Number of families in the building
has_secondary_use	Indicates if the building is used for any secondary purpose
has_secondary_use_agriculture	Indicates if the building is secondarily used for agricultural purpose
has_secondary_use_hotel	Indicates if the building is secondarily used as hotel
has_secondary_use_rental	Indicates if the building is secondarily used for rental purpose
has_secondary_use_institution	Indicates if the building is secondarily used for institutional purpose
has_secondary_use_school	Indicates if the building is secondarily used as school
has_secondary_use_industry	Indicates if the building is secondarily used as industrial purpose
has_secondary_use_health_post	Indicates if the building is secondarily used as health post
has_secondary_use_office	Indicates if the building is secondarily used as government office
has_secondary_use_police	Indicates if the building is secondarily used as police station
has_secondary_use_other	Indicates if the building is secondarily used as other purpose

5.1.3 Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, to spot anomalies. The main anomalies found in the data is missing values and independency of target variable on other variables.

5.1.3.1 Missing Values

Missing is filled by method fillna( ).

5.1.3.2 Chi-Square test

A **chi-squared test**, also written as χ^2 **test**, is a statistical hypothesis test that is valid to perform when the test statistic is chi-squared distributed under the null hypothesis, specifically Pearson's chi-squared test and variants thereof. Pearson's chi-squared test is used to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies in one or more categories of a contingency table.

```
def ChiSquareTest(cat,res_train):

    for c in cat:
        print(c)
        tab = pd.crosstab(res_train['damage_grade'], res_train[c])
        stat, p, dof, expected = chi2_contingency(tab)
        print('dof=%d' % dof)
        prob = 0.95
```

```
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))
 if abs(stat) >= critical:
        print('Dependent (reject H0)')
else:
        print('Independent (fail to reject H0)')


# H0(target variable is independent on given attribute)
```

5.1.4 Removing Unwanted Columns

After performing chi-square we get to know the unwanted columns by rejecting the H0 or failed to reject H0.

5.1.5 Splitting train and test data.

We split the dataset into two different datasets they are test data and train data.

X_train, X_test,y_train, y_test = train_test_split(data, z_train, test_size=0.2)

- X_train-the data used to train the model.
- y_train-the train data of model.
- X_test- the data used to test the model.
- y_test-the test data of model.

5.1.6 Supervised Learning

The dataset has been cleaned, pre-processed and analyzed for understanding the dataset. After such a process, and yet before coming to modeling, the dataset has to split up into two parts: Train and Test dataset. The training dataset is used to train the algorithm used to prepare an algorithm to comprehend. To learn and deliver results. It incorporates both input data and the desired output. The test datacollection is utilized to assess how well the algorithm was prepared with the traineddataset.

By using the Supervised learning Algorithms to train the dataset and also to test,predictions were made as to the desired outcome. The system was able to split, trainand test the dataset. Along with that, the feature importance was also given as theoutput where it had the percentage of these possibilities in occurring in the merefuture datasets that would be added.

The aim for us is to predict the Damage grade assigned to the building after assessment of Earth Quake.

The following are Supervised Algorithms which we used for prediction

- Decision Tree,
- Naïve Bayes'.
- Neural Networks.

5.1.7 Naïve Bayes

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems.

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation

p(cj | d) = p(d | cj) *p(cj)/ p(d)

• p(cj | d) = probability of instance d being in class cj , This is what we are trying to compute

• p(d | cj) = probability of generating instance d given class cj , We can imagine that being in class cj , causes you to have feature d with some probability

• p(cj) = probability of occurrence of class cj , This is just how frequent the class cj , is in our database

• p(d) = probability of instance d occurring This can actually be ignored, since it is the same for all classes

Since our target variable has 5 different values our classification is multi-class classification.

```
naive_bayes = GaussianNB()
naive_bayes.fit(X_train,y_train)
prediction= naive_bayes.predict(X_test)
```

5.1.8 Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining

multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

```
random_forest = RandomForestClassifier(
criterion='entropy',n_estimators=50,max_features='log2', max_depth=4,n_jobs=-1,
random_state=0)
random_forest.fit(X_train,y_train)
prediction= random_forest.predict(X_test)
```

5.1.9 Neural Networks

The Model is a Neural network with 307 inputs and 3 hidden layers and 8 categories/bins:

- The neural Network Architechture is created using torch nn module, which has 8 output. Log_softmax is performed to get the label.
- nn.dropout(.02) is used so that the network does not over fit.
- nn.NLLLoss is used to calcuate the loss. since the output is a softmax which is in the form of exponential, Natural Log loss will be best to get the loss
- loss.backward() performs the backward pass
- optimizer.step() updates the new weight
- optimizer.zero_grad() sets the grdient to zero for next backword propagation

- model.eval() freezes the gradient and removes the dropout for evaluation/validation during training

Parameter tuning for the model:

- Epochs
- Learning rate
- Mini Batch size
- Number of Hidden layers
- Number of Hidden nodes

5.2 Testing

5.2.1 Black BoxTesting

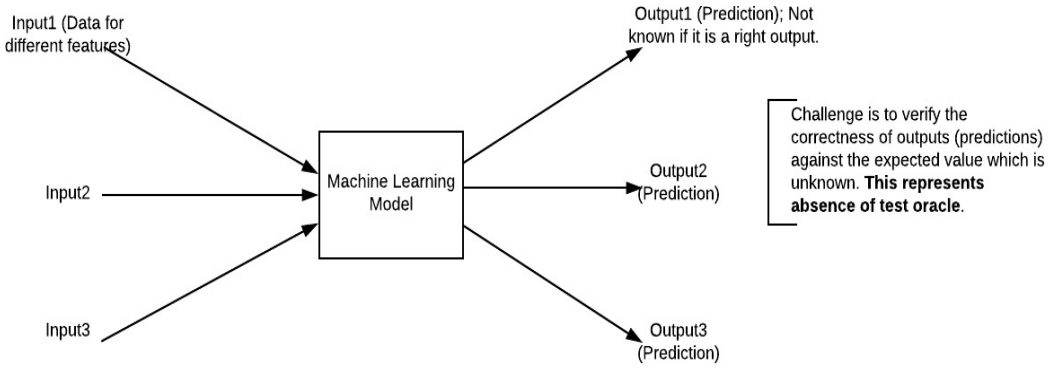


Fig 5.1 Black BoxTesting

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.

Input	Actual Output	Predicted Output
[16,6,324,0,0,0,22,0,0,0,0,0,0]	0	0
[16,7,263,7,0,2,700,9,10,1153,832,9,2]	1	1

Table 5.4 Example Black Box Testing

The model gives out the correct output when different inputs are given which are mentioned in Table 5.4. Therefore the program is said to be executed as expected or correct program.

5.2.2 Confusion Matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm.Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

Confusion matrix is sqaure matrix of size n where n is the number of values for targeted variable.

Let us consider a confusion matrix A nxn

A[i,j] indicates the number of times i is predicted as j

In our case n=5

0th row - grade1,1st row - grade 2,2nd row - grade 3,3rd row - grade 4,4th row - grade 5

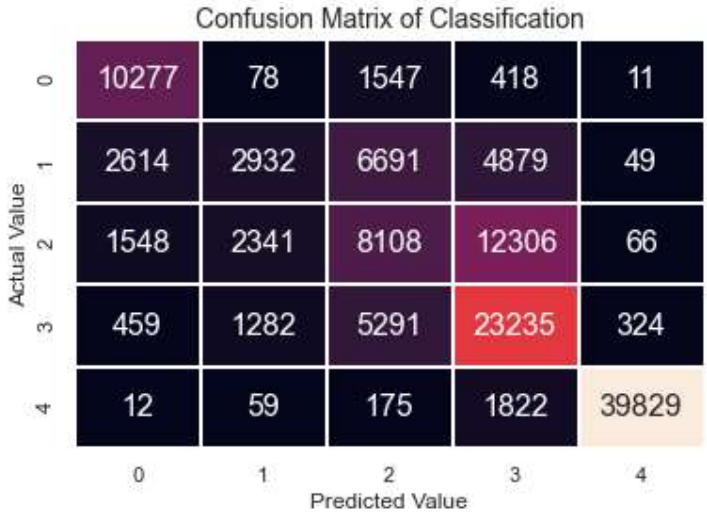


Fig 5.2 Confusion Matrix of Naïve Bayes

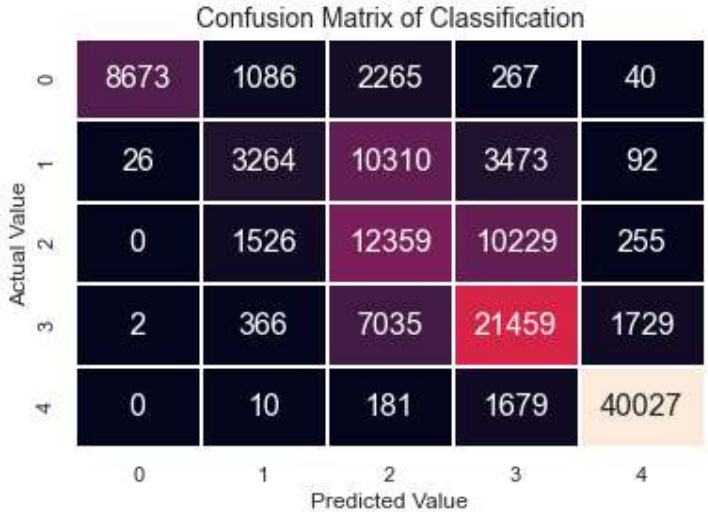


Fig 5.3 Confusion Matrix of Random Forest

6 CONCLUSION& FUTURE SCOPE

6.1 Conclusion

Thus, the aim of this project is to predict the damage grade of buildings. The Random forest algorithm is high accurate in predicting compared to Naive bayes and Decision Tree.The use K-folds the accuracy of prediction is increasing the accuracy is directly proportion to number of folds.It is also seen that damage grade 1,5 have high accuracy in prediction.

Table 6.1 Accuracy comparision

Algorithm	Accuracy
Decision Tree	66.19
Naive Bayes	67.01
Neural Networks	70.59

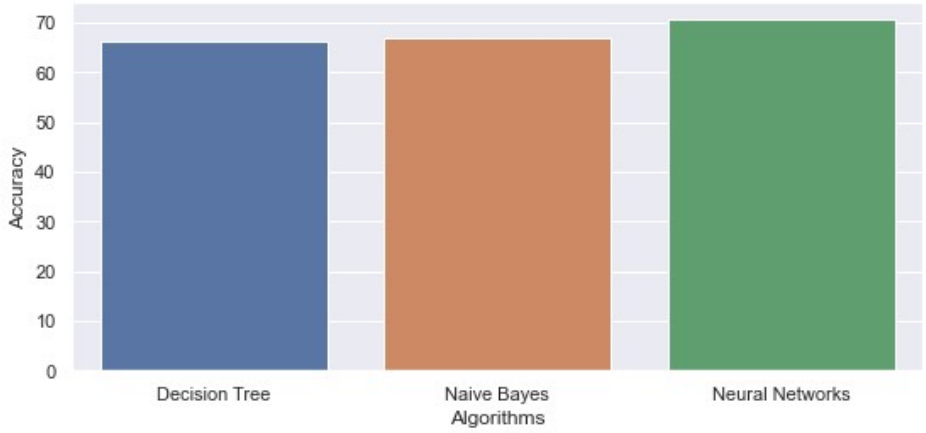


Fig 6.1 Accuracy comparision

6.2 Future scope

In future we plan to enable maps and show colors for different grades for different building.And also to host our application on web.Also, in future we would like to collect data from real time database and a dataset with more variation and a higher quality can really boost the accuracy of our current models. Also we think that using more complex models like artificial neural networks, or applying deep learning.

Maintaing a realtime data of building structure would help in better prediction and

Using hybrid models for increasing accuracy.

Due to time and knowledge constraints we could not develop great UI/UX. As an improvement to this model, we will give priority to use progressive web app that uses better rendering tools such as angularJS that improves client side UI/UX experience.

REFERENCES

[1]Long Wang, Xiaoqing Wang, Aixia Dou, Dongliang Wang "Study on construction seismic damage loss assessment using RS and GIS" International Symposium on Electromagnetic compatibility, 2014.

[2]Roxane Mallouhy, Chady Abou Jaoude,Christophe Guyeux ,Abdallah Makhoul,"Earthquake event prediction using various machine learning algorithms".

[3]G. Molchan and V. Keilis-Borok,"Earthquake prediction: probabilistic aspect".

[4]H Takata, H. Nakamura, T Hachino,"On prediction of electric power damage by typhoons in each district in Kagoshima Prefecture via LRM and NN".

[5]Ramli Adnan. Abd Manan Samad, Zainazlan Md Zain, Fazlina Ahmat Ruslan,"5 hours flood prediction modeling using improved NNARX structure: case study Kuala Lumpur"

[6]Dezhang Sun, Baitao Sun,"Rapid prediction of earthquake damage to buildings based on fuzzy analysis"2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, vol. 3, p. 1332-1335, 2010.

[7]https://www.kaggle.com/arpitr07/predict-building-damage-grade/output

APPENDIX: SOURCE CODE

```
#Importing essential libraries


import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import scipy.stats as ss

from scipy.stats import chi2_contingency

from scipy.stats import chi2

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

from sklearn.metrics import homogeneity_score

from sklearn.metrics import silhouette_score

from sklearn.metrics import classification_report,f1_score,precision_score,recall_score,accuracy_score

from sklearn.model_selection import StratifiedKFold


#To Ignore Warnings

import warnings

warnings.filterwarnings('ignore')


# Importing and understanding our dataset

train=pd.read_csv('D:\Major Project\Dataset/train.csv')
```

```
# Shape of dataset
train.shape

# Printing out a few columns
train.head()
train.columns
len(train.columns)
#To find null cells
train.isnull().sum()

# Using of other datasets
owner = pd.read_csv('D:\Major Project\Dataset/Building_Ownership_Use.csv')
structure = pd.read_csv('D:\Major Project\Dataset/Building_Structure.csv')

# Combining of other datasets for preprocessing
combine = pd.merge(owner,structure, on='building_id')
trainfinal = pd.merge(combine,train, on = 'building_id')

# Train Data before preprocessing
trainfinal.columns,len(trainfinal.columns)

trainfinal.info()

trainfinal.isnull().sum()

#filling null values
```

```
trainfinal['has_repair_started'].fillna(trainfinal['has_repair_started'].mode()[0],
inplace=True)

trainfinal.drop((['vdcmun_id_y','district_id_y','ward_id_y','vdcmun_id','district_id'])  ,
axis = 1,inplace=True)

features = list(trainfinal.columns)

# Let's understand our columns better:

trainfinal.info()

# Converting Nominal to Numeric for the purpose of classification(Target Variable)

conversion = {'damage_grade' : {"Grade 1" : 1, "Grade 2" : 2, "Grade 3" : 3,"Grade
4" : 4,"Grade 5" : 5}}

train_temp = pd.DataFrame()

train_temp['damage_grade'] = trainfinal['damage_grade']

train_temp.replace(conversion, inplace = True)

trainfinal['damage_grade'] = train_temp['damage_grade']

#Understanding Other Variables

trainfinal['legal_ownership_status'].value_counts().plot.bar()

sns.distplot(trainfinal['age_building'])

sns.set(font_scale=0.7)

sns.countplot(trainfinal['area_assesed'])
```

```
cat = [c for c in trainfinal if trainfinal[c].dtypes == "object"]
cat.remove('building_id')
print(cat)

# Chi-Square test on Each Types for Understanding dependency of Target with each type

def ChiSquareTest(cat,res_train):

 for c in cat:
   print(c)
   tab = pd.crosstab(res_train['damage_grade'], res_train[c])
   stat, p, dof, expected = chi2_contingency(tab)
   print('dof=%d' % dof)
   prob = 0.95
   critical = chi2.ppf(prob, dof)
   print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))
   if abs(stat) >= critical:
     print('Dependent (reject H0)')
   else:
     print('Independent (fail to reject H0)')

   # H0(target variable is independent on given attribute)

   alpha = 1.0 - prob
   print('significance=%.3f, p=%.3f' % (alpha, p))
```

```
    if p <= alpha:
      print('Dependent (reject H0)')
    else:
      print('Independent (fail to reject H0)')

    print(" ")
```

 # Performing Chi-Square test on Object Types for Understanding dependency of Target with each object type

```
ChiSquareTest(cat,trainfinal)

cat_binary = [c for c in trainfinal if len(trainfinal[c].unique()) == 2]
cat_binary
```

 # Performing Chi-Square test on Binary Types for Understanding dependency of Target with each Binary type

```
ChiSquareTest(cat_binary,trainfinal)
```

#Removal columns on which Target is not dependent(By Chi-Square Test)

```
trainfinal.drop(['has_secondary_use_use_police','building_id'], axis = 1,inplace=True)
```

 # Converting Nominal to Numeric in preprocessed data for the purpose of classification

```
cont = [c for c in trainfinal if len(trainfinal[c].unique()) > 15]
indices = 0,1,2,3
```

```
cont = [i for j, i in enumerate(cont) if j not in indices]


train_copy = trainfinal


train_copy['IsPrivate'] = (train_copy["legal_ownership_status"] == "Private") * 1

train_copy['IsFlat'] = (train_copy["land_surface_condition"] == "Flat") * 1

train_copy['IsMudFoundation'] = (train_copy["foundation_type"] == "Mud mortar-Stone/Brick") * 1

train_copy['IsBambooRoofLight'] = (train_copy["roof_type"] == "Bamboo/Timber-Light roof") * 1

train_copy['IsFloorTypeMud'] = (train_copy["ground_floor_type"] == "Mud") * 1

train_copy['OtherFloorTypeMud'] = (train_copy["other_floor_type"] == "TImber/Bamboo-Mud") * 1

train_copy['IsNotAttached'] = (train_copy["position"] == "Not attached") * 1

train_copy['IsPlanConfigRectangular'] = (train_copy["plan_configuration"] == "Rectangular") * 1

train_copy['count_floors_change'] = (train_copy['count_floors_post_eq'] - train_copy['count_floors_pre_eq'])

train_copy['height_ft_change'] = (train_copy['height_ft_post_eq'] - train_copy['height_ft_pre_eq'])


train_copy.drop(['count_floors_pre_eq', 'height_ft_pre_eq'], axis=1, inplace=True)


remove_columns = ["legal_ownership_status","land_surface_condition","foundation_type","roof_type","ground_floor_type","other_floor_type","position","plan_configuration","count_floors_post_eq","height_ft_post_eq"]

def dropColumns(res_train_copy,remove_columns):
```

```
  for i in remove_columns:
    res_train_copy.drop([i],axis = 1, inplace = True)
  return res_train_copy

train_copy = dropColumns(train_copy,remove_columns)

train_copy.shape

train_copy.isnull().sum()

train_copy['count_families'].fillna(train_copy['count_families'].mode()[0],inplace=True
)

traindone = pd.get_dummies(train_copy)
traindone.drop(["district_id_x","vdcmun_id_x","ward_id_x"],axis = 1, inplace = True)

traindone.isnull().sum()

# Target Variable Statistics
sns.set(font_scale=1)
sns.countplot(traindone['damage_grade'])
```

#Supervised Learning

#Modeling,Training and Prediction

```
z_train = traindone['damage_grade']
traindone.drop(['damage_grade'], axis = 1, inplace = True)

from sklearn.model_selection import train_test_split
X_train, X_test,y_train, y_test = train_test_split(traindone, z_train, test_size=0.2, random_state=101)



# Naive Bayes

from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)

# predicting the data using test data
model_predicted = model.predict(X_test)
probs = model.predict_proba(X_test)



#finding the confusion matrix
conf_mat = confusion_matrix(y_true=y_test, y_pred=model_predicted)

# visualizing confusion matrix
sns.heatmap(conf_mat, annot=True, annot_kws={"size":16}, fmt="d", cbar=False, linewidths=0.1,cmap="rocket")
plt.title("Confusion Matrix of Classification", fontsize=14)
```

```
plt.ylabel("Actual Value", fontsize=12)

plt.xlabel("Predicted Value", fontsize=12)

plt.show()


# printing the classifcation report

print("Understanding Classifciation:")

rint(classification_report(y_test, model_predicted))


#accuracy

score_nb=accuracy_score(y_test,model_predicted)

print("Accuracy=",score*100)


print("Mean Absolute Error:", metrics.mean_absolute_error(y_test,model_predicted))

print("Mean Squared Error:", metrics.mean_squared_error(y_test,model_predicted))

print("Root Mean Square Error:", np.sqrt(metrics.mean_squared_error(y_test, model_predicted)))




# RandomForest

#importing required module

from sklearn.ensemble import RandomForestClassifier

random_forest = RandomForestClassifier( criterion='entropy',n_estimators=50, max_features='log2', max_depth=4,n_jobs=-1, random_state=0)

model.fit(X_train, y_train)
```

```
# predicting the data using test data
model_predicted = model.predict(X_test)
probs = model.predict_proba(X_test)


#finding the confusion matrix
conf_mat = confusion_matrix(y_true=y_test, y_pred=model_predicted)

# visualizing confusion matrix
sns.heatmap(conf_mat, annot=True, annot_kws={"size":16}, fmt="d", cbar=False, linewidths=0.1,cmap="rocket")
plt.title("Confusion Matrix of Classification", fontsize=14)
plt.ylabel("Actual Value", fontsize=12)
plt.xlabel("Predicted Value", fontsize=12)
plt.show()

# printing the classifcation report
print("Understanding Classifciation:")
rint(classification_report(y_test, model_predicted))

#accuracy
score_rf=accuracy_score(y_test,model_predicted)
print("Accuracy=",score*100)

print("Mean Absolute Error:", metrics.mean_absolute_error(y_test,model_predicted))
print("Mean Squared Error:", metrics.mean_squared_error(y_test,model_predicted))
```

```
print("Root Mean Square Error:", np.sqrt(metrics.mean_squared_error(y_test, model_predicted)))
```

```
#Librabries for creating and training neural Networks
from torch import nn, optim
import torch.nn.functional as F
import torch

def iterate_minibatches(X, y, mini_batchsize, indices):
    '''
    Function yields minibatches for features andtargets for shuffled indices at each epoch and returns ir

    Args:
    ---
    X(tensor)- tensor Features for training
    y(tensor) - tensor Targets for training
    mini_batch(int) - Rows to be returned for each mini batch
    indices(np.array) - shuffled array of indicies

    Returns:
    -------
    Mini batch features and targets
    '''
```

```
    for start_ind in range(0, X.shape[0], mini_batchsize):
        end_ind = min(start_ind + mini_batchsize, X.shape[0])
        #print("start:end",start_idx,end_ind)
        new_in = indices[start_ind:end_ind]
        #print("new_in",new_in)
        yield X[new_in], y[new_in]



def create_train_model(X_train,y_train):
    '''
    creates,trains a NN model and returns the model
    Args:
    ---
    X_train(DataFrame)-  Features for training
    y(DataFrame) - Targets for training

    Returns:
    -------
    optimized neural network model
    '''
    #number input layer noders = total number of columns in X_train
    inputs_ = X_train.shape[1]
    ## using torch nn module to define the structure of nueral network
    class NeuralNet(nn.Module):
```

```
    def __init__(self):
        #initializing using nn __init__()
        super().__init__()
        self.fc1 = nn.Linear(inputs_, 307)
        self.fc2 = nn.Linear(307, 256)
        self.fc3 = nn.Linear(256, 8)
        self.dropout = nn.Dropout(p=0.2)
    def forward(self, x):
        #print(x.shape)
        x = self.dropout(F.relu(self.fc1(x)))
        x = self.dropout(F.relu(self.fc2(x)))
        x = F.log_softmax(self.fc3(x), dim=1)

        return x



model1 = NeuralNet()
# Defining the loss
criterion = nn.NLLLoss()
optimizer = optim.Adam(model1.parameters(), lr=0.005)
epochs = 25 #number of times the whole training data is feeded in the neural network for training
val_print = 4 #validate after every 4rth time and print training, validation score and validation accuracy
batch_size = 16000 #for this 1/4rth of the X_train rows i.e 63761
step = 0
```

```
#converting to torch values for training
X_train_t = torch.tensor(X_train.values).type(torch.float)
y_train_t = torch.tensor(_train.values).type(torch.long)

#indices in X_train
indices = np.arange(X_train_t.shape[0])

#stores respective losses
train_losses, val_losses = [], []
loss_counts = []
loss_count = 0
#for epochs from 0 to n-1 epochs
for e in range(epochs):
   #shuffle indices after every epoch to create new shuffled mini batches
   np.random.shuffle(indices)
   #setting running loss as 0
   if step == 0:
      running_loss = 0

   #yield mini batch till the last of the row is reached
   for batch in iterate_minibatches(X_train_t, y_train_t, batch_size,indices):
      step += 1
      X_batch, y_batch = batch
      #if first 3 mini batch, train the model
      if step%val_print != 0:
```

```
    #print("train")
    optimizer.zero_grad()

    log_ps = model1(X_batch)
    #calculate loss
    loss = criterion(log_ps, y_batch)
    #print(loss)
    loss.backward()
    optimizer.step()

    running_loss += loss.item()
    #print(running_loss)

# for the 4rth mini batch validate
if step%val_print == 0:
    #print("validate")
    loss_count +=1
    val_loss = 0
    accuracy = 0

    # gradients turned off for validation
    with torch.no_grad():
        model1.eval()
        log_ps = model1(X_batch)
        val_loss += criterion(log_ps, y_batch)
```

```
            ps = torch.exp(log_ps)
            #gets the top predicted class
            top_p, top_class = ps.topk(1, dim=1)
            #binaries into true and false by comparing y_test and predicted list
            equals = top_class == y_batch.view(*top_class.shape)
            #calcuates the accuracy
            accuracy += torch.mean(equals.type(torch.FloatTensor))

        #next forward feed
        model1.train()

        # calculating average training loss of the three trains in the epochs
        #calculating validation loss for the 4rth mini batch
        train_losses.append(running_loss/(val_print- 1))
        val_losses.append(val_loss)
        loss_counts.append(loss_count)
        running_loss = 0

        #print losses,accuracy after every 4rth epoch
        if e%5 == 1:
            print("Epoch: {}/{}.. ".format(e+1, epochs),
                  "Training Loss: {:.3f}.. ".format(train_losses[-1]),
                  "validation Loss: {:.3f}.. ".format(val_losses[-1]),
                  "validation Accuracy: {:.3f}".format(accuracy))

#Print the learning curve Validation score  and training score
```

```
    plt.plot(loss_counts[10:],train_losses[10:],'-b', label='train')
    plt.plot(loss_counts[10:],val_losses[10:],'-g', label='validation')

    #return the trained model
    return model1

model =tr.create_train_model(X_train,y_train)
X_test_t = torch.tensor(X_test.values).type(torch.float)
with torch.no_grad():
    model.eval()
    prediction = model(X_test_t)
ps = torch.exp(prediction)
ps.shape
top_p, top_class = ps.topk(1, dim=1)
#converting tensor type to Series
top_class = pd.Series(top_class.numpy().ravel())
score_nn =accuracy_score(y_test, top_class)

# Accuracy Calculation

scores = [score_dt,score_nb,score_nn]
percentscore=[round(i*100,2) for i in scores]
foldscore=[round(i,2) for i in scores1]
algorithms = ["Decision Tree","Naive Bayes","Neural Network"]
```

```
# Accuracy Table
from tabulate import tabulate
data=[]
for i in range(len(algorithms)):
   list=[algorithms[i],percentscore[i],foldscore[i]]
   data.append(list)
print(tabulate(data, headers=["Algorithm","Accuracy"]))

# Accuracy Graph

sns.set(rc={'figure.figsize':(9,4)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy")

sns.barplot(algorithms,percentscore)
```