Seismic Insight

Data Science With Python Lab Project Report

Bachelor

in

Computer Science

By

Ch Narendhra Kumar and G Dinesh Karthik

S200061

S200270

Rajiv Gandhi University Of Knowledge And Technologies

S.M. Puram, Srikakulam -532410

Andhra Pradesh, India

Abstract

Seismic Insight is an Earthquake prediction project that utilizes datasets collected from kaggle to forecast seismic activities and tectonic movements. The project aims to enhance early warning systems by analyzing various geophysical parame- ters, historical seismic data, and present seismic data.Seismic Insight provides ac- curate predictions and reducing the impact of earthquakes on vulnerable regions. The project also focuses on developing a user-friendly interface to provide timely warnings and contribute to global efforts in minimizing earthquake-related risks due to which we can reduce the economical hazards by better infrastructure planning.

The main purpose of this project is to predict the magnitude earthquake for a region given by the user with the help of historical data.The parameters which are used in this project are Longitude,Latitude,depth error,magnitude error,etc. of the given reigion.

Contents

Chapter 1

Introduction

1.1 Introduction to Your Project

Now a days,most of the regions are effected by earthquakes and results in economic and human loss.To resolve this we came up with a project named Seismic insight.

Seismic Insight is a project that aims to forecast the occurence of earth- quakes by providing the data of their magnitude and several parameters.It wants to understand how earthquakes happen and find better ways to pre- dict them.By looking at seismic data and using some algorithms,the project gives us more insights about earthquakes.

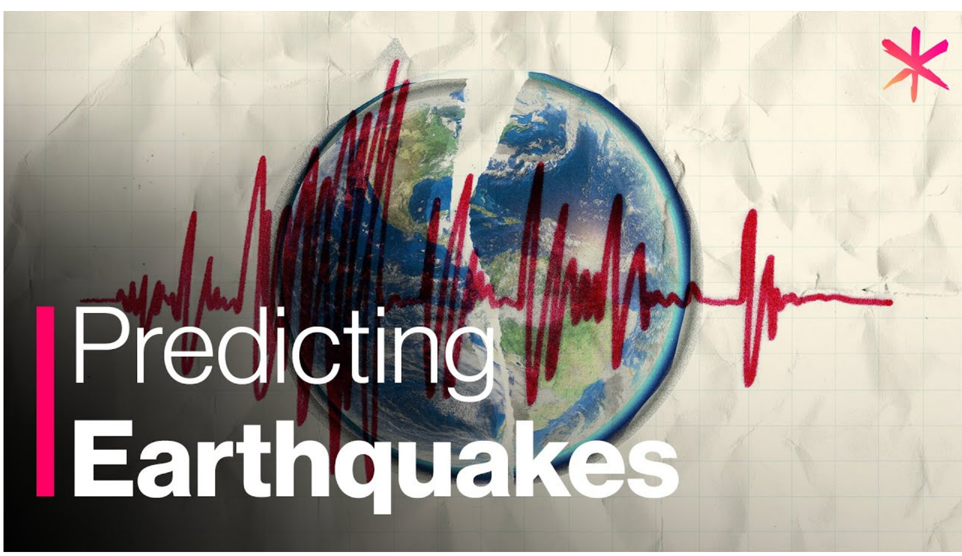


Figure 1.1: Histogram

1.2 Application

Seismic insight, derived from seismic data analysis, has various appli- cations in different fields. Seismic data is primarily used to study and understand the subsurface characteristics of the Earth. The following are some key applications of seismic insight:

Infrastructure Planning

Seismic insight helps in infrastructure planning by identifying the potential issues and ensuring the safety of Infrastructure.

Reduced Damages and Injuries

Early warning systems could significantly reduce the number of damages and injuries associated with earthquakes.

Faster Rescue Operations

By giving Early predictions we can ensure faster rescue and search operations by spread ing awareness in the locality.

1.3 Motivation Towards Your Project

As we noticed that most of places are effected by earthquakes which made human loss.It rises a thought of making this project as a solution. The main aim of this project is to provide accurate earthquake predictions to enhances public safety and confidence.The main motto is to reduce risk fac- tor due to earthquakes and to get awarness from earthquakes in people.It helps building healthier economy and better society.As earthquakes has an global impact in damage ,it drives to make an prediction model to mitigate impact.The project hopes to gives us making it easier to get ready and stay safe when earthquakes happen.

1.4 Problem Statement

Earthquakes has a significant threat to human safety and infrastructure worldwide.The dataset for this project is taken from Kaggle.The ability to predict earthquakes with precision is crucial for implementing timely and effective measures to minimize their impact. Despite advancements in seis- mology and geophysics, current earthquake prediction models are limited in their accuracy and reliability.This model aims to make the predictions with more accuracy.

Chapter 2

Approach To Your Project

2.1 Explain About Your Project

This project about predict earthquake magnitude by analysing historical data.This project helps in better architecture planning,as by getting magnitude predicted by our model they can have an idea how hard the construction should be built.We can use features present in dataset to predict that magnitude. Here we can even predict

2.2 Data Set

This Data set collected from kaggle for our project contains 22 columns explation is below:
time :Time when the event occurred. Times are reported in milliseconds.
latitude :Decimal degrees latitude. Negative values for southern latitudes.
longitude :Decimal degrees longitude. Negative values for western longitudes.
depth :Depth of the event in kilometers.
mag :Magnitude of event occured.
magType :The method used to calculate magnitude.
nst :The total number of seismic stations used to determine earthquake location.
gap :The largest azimuthal gap between azimuthally adjacent stations (in degrees).
dmin :Horizontal distance from the epicenter to the nearest station (in degrees).
rms :The root-mean-square (RMS) travel time residual, in sec, using all weights.

net :The ID of a data source contributor for event occured.

id :A unique identifier for the event.

types :A comma-separated list of product types associated to this event.

place :name of the region near to the event.

type :Type of seismic event.

locationSource :The network that originally authored the reported location of this event.

magSource :Network that originally authored the reported magnitude for this event.

horizontalError :Uncertainty of reported location of the event in kilometers.

depthError :The depth error, three principal errors on a vertical line.

magError :Uncertainty of reported magnitude of the event.

magNst :The total number of seismic stations to calculate the magnitude of earthquake.

status :Indicates whether the event has been reviewed by a human.

among these all columns we will drop the columns which are not useful for model development.

2.3 Prediction technique

As this project mainly aims to predict magnitude which is a numeriacal category it can be considered as a regression problem.Machine learning algorithms used to predict to solve a regression problem are as follows:

linear regression : A simple model that predicts a target variable by fitting a linear relationship.

polynomial regression:A regression model that uses linear regression but with higher degree transformation

random forest regression:predicts the target variable by averaging the outputs of multiple decision trees.

Among all we will show thebest technique by some metrics such as meansquarederror.

2.4 Graphs

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

Histogram plot for Magnitude

```
sns.histplot(df['mag'], bins=30, kde=True)
plt.title('Distribution of Earthquake Magnitudes')
plt.xlabel('Magnitude')
plt.ylabel('Frequency')
plt.show()
```

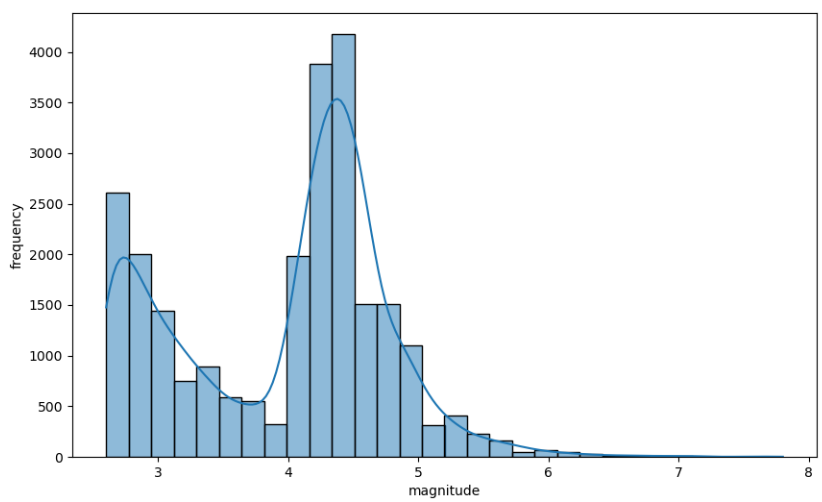


Figure 2.1: Histogram

By observing this above histogram we can say that magnitude is normally distributed as it is a nearly bell curve.

PIE chart

```
plt.figure(figsize=(10,8))
count=seismic_data['magType'].value_counts()
plt.pie(count)
plt.legend(count.index)
plt.savefig("piechart")
plt.show()
```

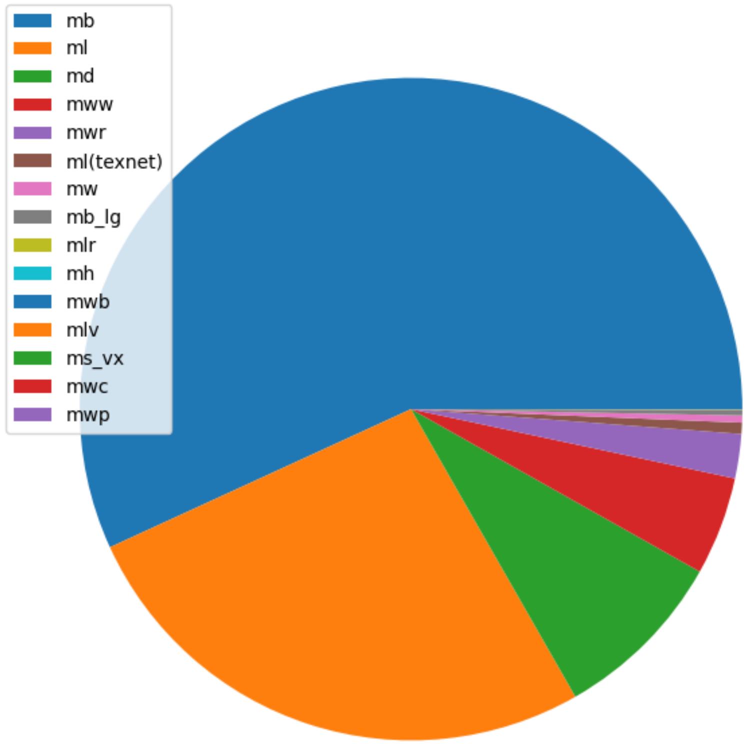


Figure 2.2: Pie chart

By observing we can say most used type of technique for magnitude prediction is mb.

2.5 Visualization

Data visualization is the process of using visual elements like charts,graphs,or maps to represent data.It translates complex,high-volume,or numerical data into a visual representation that is easier to process.

Box plot

```
df=seismic_data1.copy()
```

```
plt.figure(figsize=(10,6))
sns.boxplot(df.drop(columns="time"))
plt.xticks(rotation=45)
plt.savefig("boxplot")
plt.show()
```

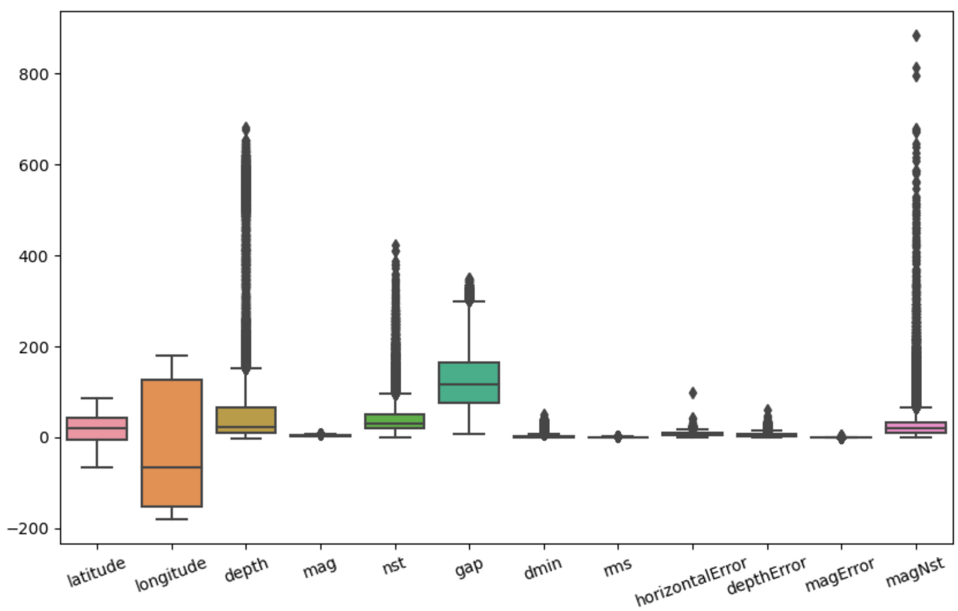


Figure 2.3: Box plot

By observing figure 2.3 ,there are outliers in most of the columns.we need to clear these outliers to increase accuracy.after clearing the graph is shown below

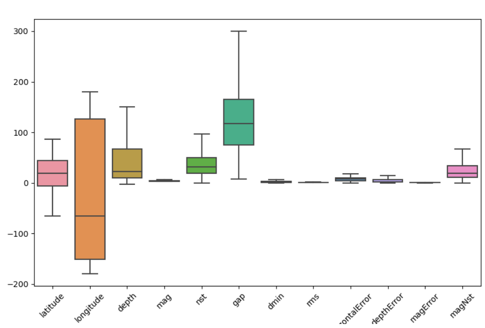


Figure 2.4: Box plot

Line Plot

```
sns.lineplot(x='date',y='mag',data=df,marker='o')
plt.xlabel('date')
plt.ylabel('magnitude')
plt.xticks(rotation=45)
plt.savefig("lineplot")
plt.show()
```

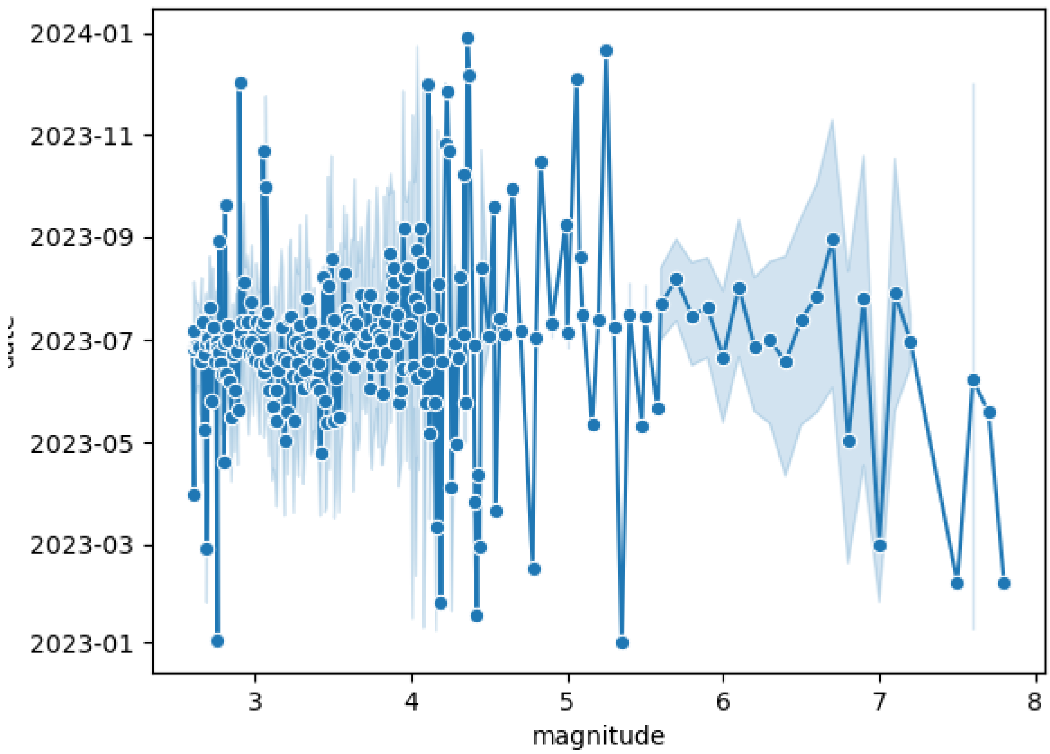


Figure 2.5: Line plot

By observing figure 2.4 we can say between 2023-01 and 2023-03 there is highest recorded magnitude.

Bar Plot

```
sns.barplot(x=df['type'],y=df['mag'])
plt.xlabel('type')
```

```
plt.ylabel('magnitude')
plt.xticks(rotation=45)
plt.savefig("bargraph")
plt.show()
```

By observing this below graph we can say that highest magnitude is recorded from volcanic eruption.

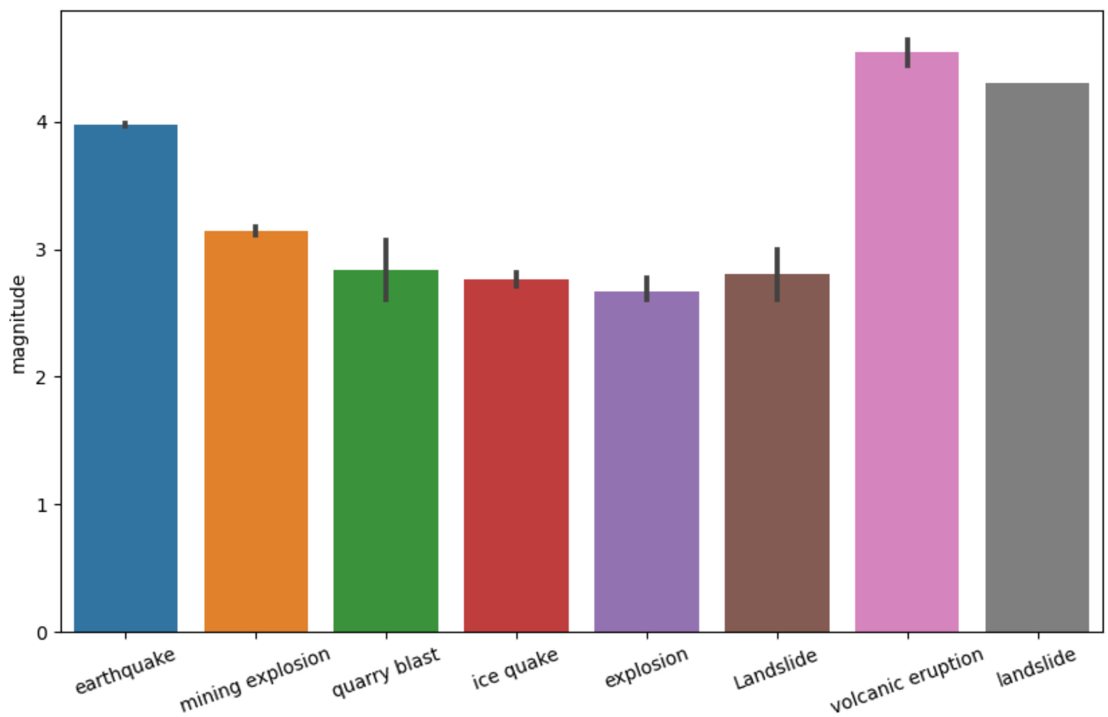


Figure 2.6: Bar graph

Bubble Plot

```
bubble = plt.scatter(x=df['mag'], y=df['depth'], s=df['nst']*5,
   alpha=0.5, c=df['nst'], cmap='viridis', edgecolors='w',
   linewidth=0.5)
plt.colorbar(bubble, label='Number of Reporting Stations')
plt.title('Bubble Plot of Seismic Events')
```

```
plt.xlabel('Magnitude')
plt.ylabel('Depth (km)')
plt.savefig("bubbleplot")
plt.show()
```

By observing this below graph we can say that . By observing fig 2.7, it represents relation

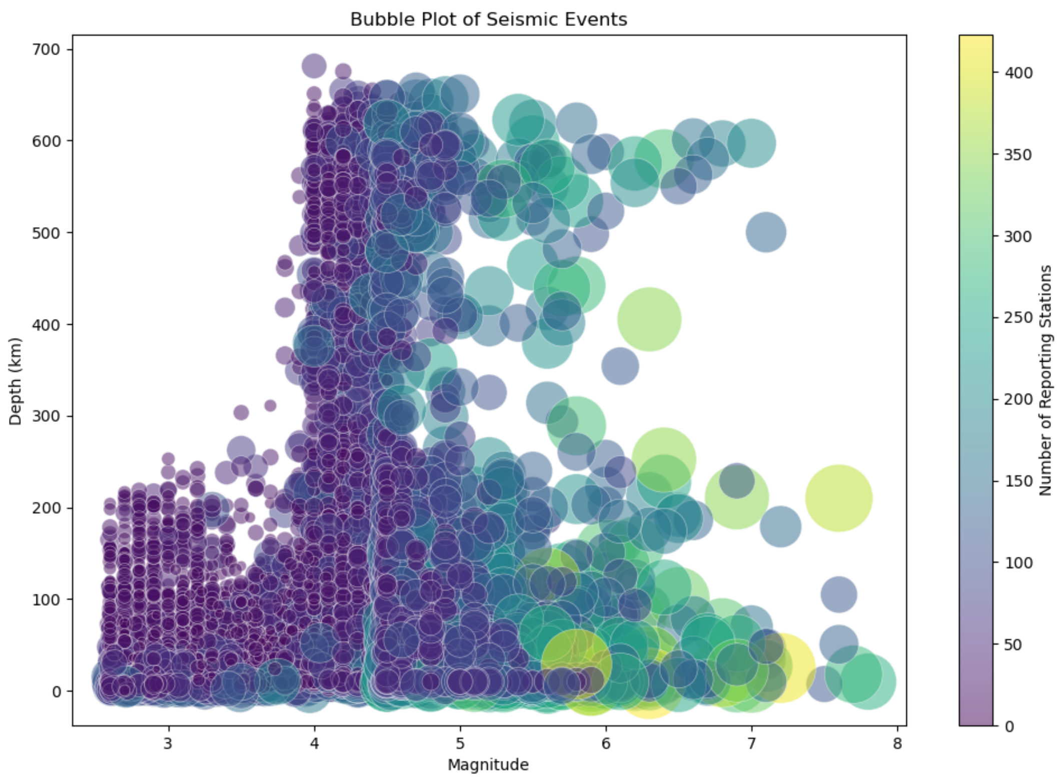


Figure 2.7: Bubble plot

between three factors and as the magnitude increases nst is also increased but by the scattering we can say most of the magnitudes are recorded in depth 0-200.

Area chart

```
plt.fill_between(df['date'],df['mag'],color='skyblue',alpha=0.4)
plt.plot(df['date'],df['mag'],color='slateblue',alpha=0.6)
plt.xlabel('date')
```

```
plt.ylabel('Magnitude')
plt.title('Area Chart: Magnitude Over Time')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig("areachart")
plt.show()
```

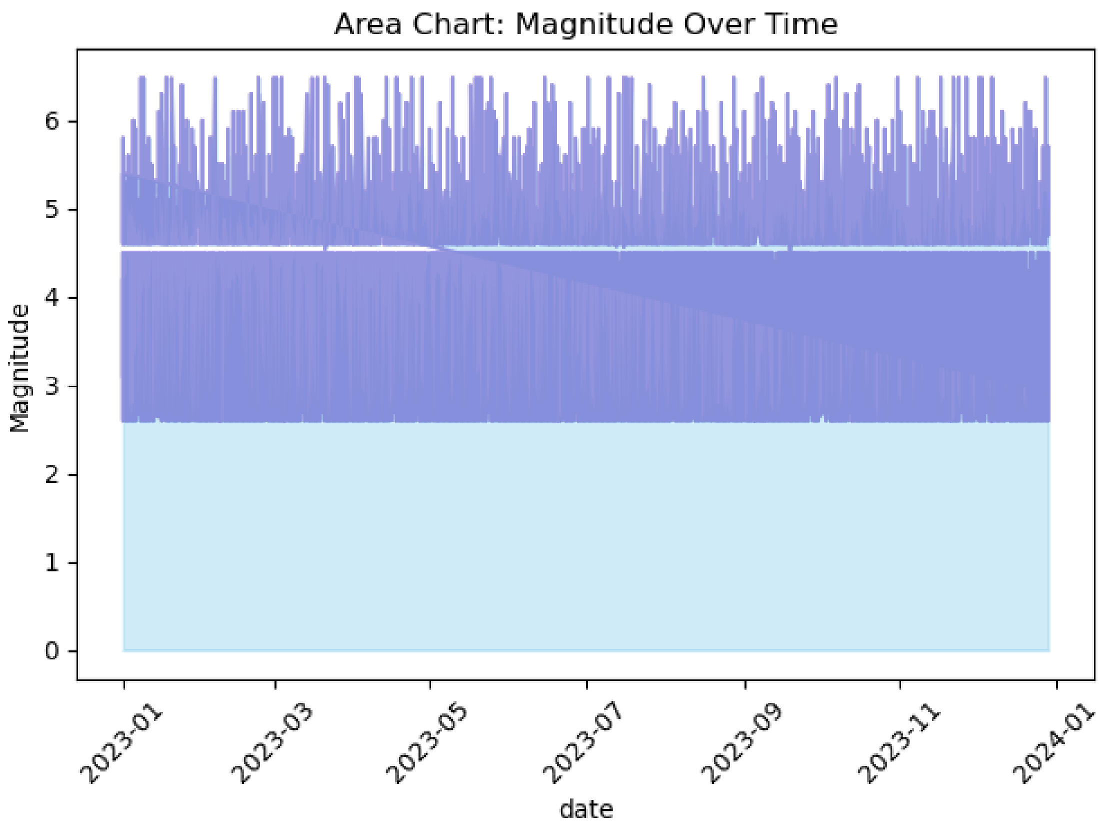


Figure 2.8: Area Chart

By observing fig 2.8, it is a combination of line plot at top and are areachart at down represents same as the line plot.

Waffle chart

```
from pywaffle import Waffle
data=df['mag_bin'].value_counts().to_dict()
```

```
plt.figure(
  FigureClass=Waffle,
  rows=5,
  columns=20,
  values=data,
  legend={'loc': 'upper left', 'bbox_to_anchor': (1.05, 1)},
)
plt.savefig("waffle chart")
plt.show()
```

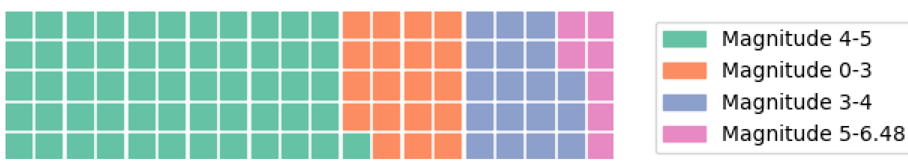


Figure 2.9: Waffle chart

By observing figure 2.9, we can say that most of themagnitudes ranged between magnitude 4-5

Word cloud

```
from wordcloud import WordCloud
text = ' '.join(df['place'])
```

```
wordcloud = WordCloud(width=800, height=400,
    background_color='white').generate(text)


plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Earthquake Locations')
plt.savefig("wordcloud")
plt.show()
```

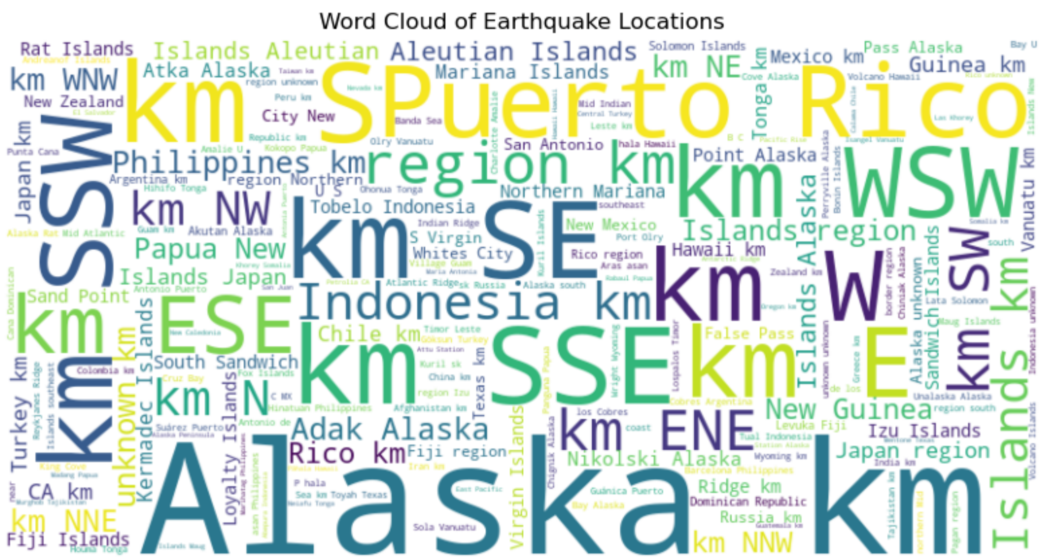


Figure 2.10: Word Cloud

By observing figure 2.10,we can say Alaska has more earthquake among all.

Heat map

```
correlation_matrix =
    df.drop(columns=['time','magType','place','type','date']).corr()


plt.figure(figsize=(10, 8))
```

```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
    fmt='.2f')
plt.title('Correlation Matrix')
plt.savefig("heatmap")
plt.show()
```

By observing fig 2.11 it is correlation matrix, here the highest absolute correlation factors will taken for model training and testing.

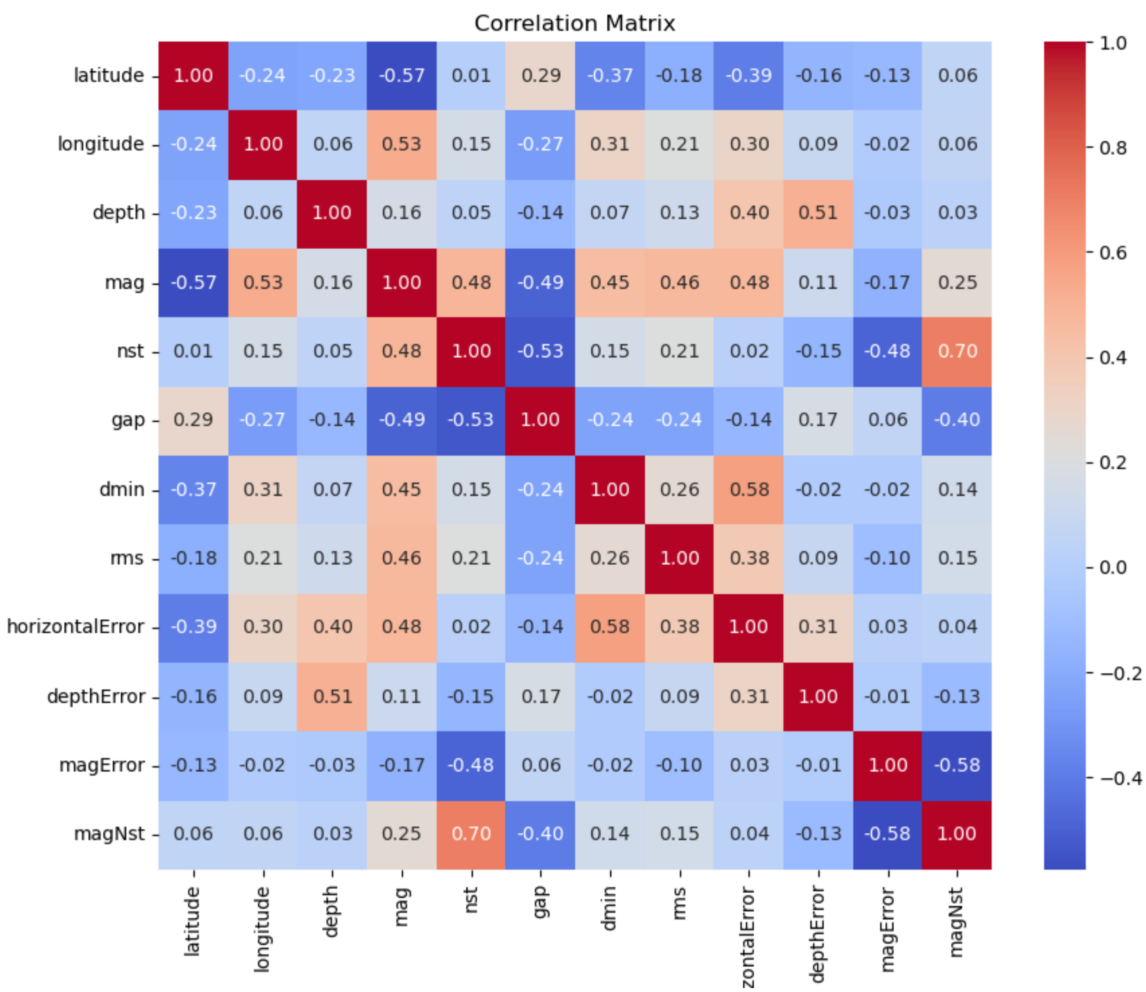


Figure 2.11: Heat Map

Scatter plot

```
sns.scatterplot(x=df1['mag'],y=df1['depth'])
```

```
plt.xlabel('magnitude')
plt.ylabel('gap')
plt.savefig('scatterplot')
plt.show()
```

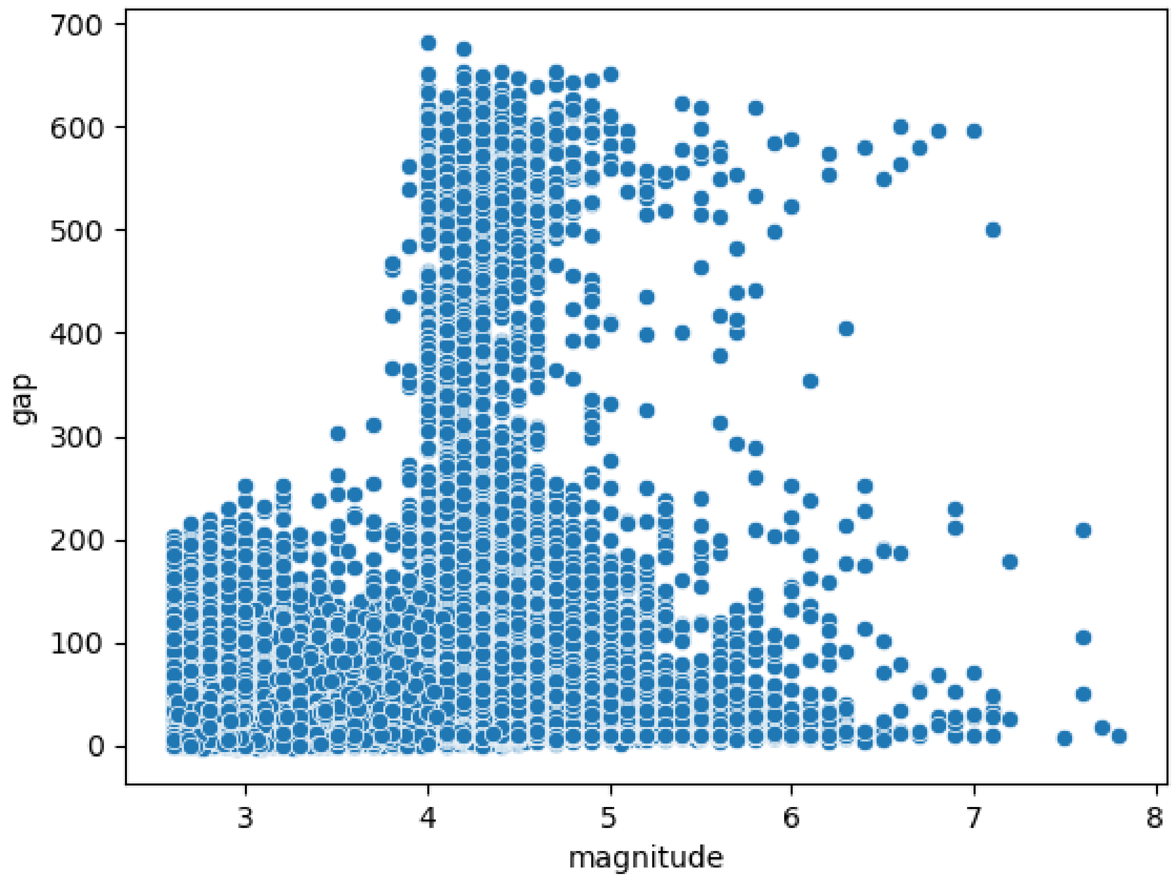


Figure 2.12: Scatter plot

By observing this graph 2.12 represents that as the gaps increases magnitude is decreasing,it was a inverse relation.

Regression Plot

```
df_samp=df1.sample(frac=0.1)
plt.figure(figsize=(12, 6))
```

```
plt.subplot(1, 2, 1)
depth_reg_plot=sns.regplot(x="depth",y="mag",data=df_samp,marker='o',color='g
depth_reg_plot.set(xlabel='depth',ylabel='magnitude',title="Reg
   plot of Depth vs magnitude")

plt.subplot(1, 2, 2)
nst_reg_plot=sns.regplot(x="nst",y="mag",data=df_samp,marker='o',color='darkb
nst_reg_plot.set(xlabel='nst',ylabel='magnitude',title="Reg plot
   of nst vs magnitude")
plt.savefig('Regplot')
```

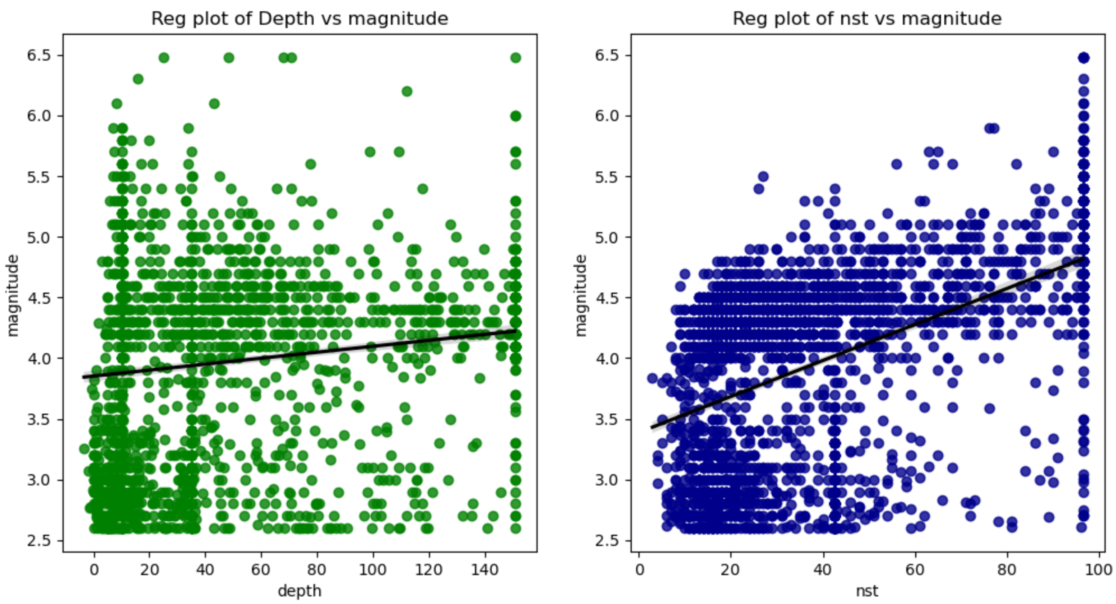


Figure 2.13: Regression plot

By observing this graph 2.12 represents that as the depth,nst increases magnitude increases.

Chapter 3

Code

Pandas

- Pandas is a free and widely used library in Python.
- It helps with handling and studying data.
- Pandas provides tools to check, tidy up, explore, and work with data.
- With Pandas, we can analyze large amounts of data and draw conclusions using statistical methods.
- It's great for cleaning up messy data, making it easier to read and use for analysis.
- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas is used for data manipulation and exploration
- We will further eliminate null values using this Pandas

3.1 Explain Our Code With Outputs

Importing Libraries

```python
import numpy as np
import pandas as pd
```

Libraries imported are pandas as we have discussed earlier and numpy it a python package used for numerical calculations.

Importing Datasets

Dataset Earthquakes is imported using readcsv syntax in pandas library

```python
dataFrame = pd.read_csv ( "
    /home/dk/Download/dspproject/earthquake.csv " )
df=dataFrame.copy()
```

Data Description and Exploration

The info method provides a summary of the dataframe, including information about the data types, non-null counts, and memory usage. It's particularly useful for un- derstanding the structure of the dataframe, identifying missing values, and assessing memory usage.

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26642 entries, 0 to 26641
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   time             26642 non-null  object
 1   latitude         26642 non-null  float64
 2   longitude        26642 non-null  float64
 3   depth            26642 non-null  float64
 4   mag              26642 non-null  float64
 5   magType          26642 non-null  object
 6   nst              25227 non-null  float64
 7   gap              25225 non-null  float64
 8   dmin             24776 non-null  float64
 9   rms              26642 non-null  float64
 10  net              26642 non-null  object
 11  id               26642 non-null  object
 12  updated          26642 non-null  object
 13  place            25034 non-null  object
 14  type             26642 non-null  object
 15  horizontalError  25093 non-null  float64
 16  depthError       26642 non-null  float64
 17  magError         24970 non-null  float64
 18  magNst           25065 non-null  float64
 19  status           26642 non-null  object
 20  locationSource   26642 non-null  object
 21  magSource        26642 non-null  object
dtypes: float64(12), object(10)
memory usage: 4.5+ MB
```

Here we will drop the unwanted columns they are net,updated,status,locationSource,magSource,id and we will print info for checking whether the columns are dropped or not.

```
df.drop(columns=["net","updated","status","locationSource","magSource","id"],
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26642 entries, 0 to 26641
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   time             26642 non-null  object
 1   latitude         26642 non-null  float64
 2   longitude        26642 non-null  float64
 3   depth            26642 non-null  float64
 4   mag              26642 non-null  float64
 5   magType          26642 non-null  object
 6   nst              25227 non-null  float64
 7   gap              25225 non-null  float64
 8   dmin             24776 non-null  float64
 9   rms              26642 non-null  float64
 10  place            25034 non-null  object
 11  type             26642 non-null  object
 12  horizontalError  25093 non-null  float64
 13  depthError       26642 non-null  float64
 14  magError         24970 non-null  float64
 15  magNst           25065 non-null  float64
dtypes: float64(12), object(4)
memory usage: 3.3+ MB
```

Head prints first five rows of the dataset.

```
df.head()
```

Out[48]:

	0	1	2	3	
time	2023-01-01T00:49:25.294Z	2023-01-01T01:41:43.755Z	2023-01-01T03:29:31.070Z	2023-01-01T04:09:32.814Z	2023 01T04:29:13.7
latitude	52.0999	7.1397	19.1631	-4.7803	53.3
longitude	178.5218	126.738	-66.5251	102.7675	-166.9
depth	82.77	79.194	24.0	63.787	
mag	3.1	4.5	3.93	4.3	
magType	ml	mb	md	mb	
nst	14.0	32.0	23.0	17.0	
gap	139.0	104.0	246.0	187.0	1
dmin	0.87	1.152	0.8479	0.457	
rms	0.18	0.47	0.22	0.51	
net	us	us	pr	us	
id	us7000j5a1	us7000j3xk	pr2023001000	us7000j3xm	us7000
updated	2023-03-11T22:51:52.040Z	2023-03-11T22:51:45.040Z	2023-03-11T22:51:29.040Z	2023-03-11T22:51:45.040Z	2023 11T22:51:38.0
place	Rat Islands, Aleutian Islands, Alaska	23 km ESE of Manay, Philippines	Puerto Rico region	99 km SSW of Pagar Alam, Indonesia	59 km SS Unalaska, Al
type	earthquake	earthquake	earthquake	earthquake	earthqu
horizontalError	8.46	5.51	0.91	10.25	
depthError	21.213	7.445	15.95	6.579	1
magError	0.097	0.083	0.09	0.238	0
magNst	14.0	43.0	16.0	5.0	
status	reviewed	reviewed	reviewed	reviewed	revie
locationSource	us	us	pr	us	
magSource	us	us	pr	us	

Tail prints last five rows of the dataset.

```
df.tail()
```

Out[49]:

	26637	26638	26639	26640	26
time	2023-12-29T03:37:19.334Z	2023-12-29T04:38:54.109Z	2023-12-29T08:42:05.747Z	2023-12-29T11:02:48.679Z	2023 29T16:31:16.6
latitude	-6.9527	32.3262	-7.2411	-19.1602	25
longitude	154.9829	141.7386	68.0663	169.0428	96.5
depth	10.0	10.0	10.0	153.264	
mag	5.2	5.1	5.1	4.7	
magType	mb	mb	mb	mb	
nst	72.0	74.0	60.0	40.0	
gap	60.0	121.0	54.0	61.0	
dmin	3.924	1.803	12.776	3.746	4
rms	0.93	0.7	0.57	0.82	
net	us	us	us	us	
id	us6000m0c5	us6000m0ch	us6000m0dr	us6000m0e5	us6000r
updated	2023-12-29T04:05:57.040Z	2023-12-29T10:59:44.533Z	2023-12-29T08:57:05.040Z	2023-12-29T11:22:46.040Z	2023 29T16:45:27.0
place	89 km SW of Panguna, Papua New Guinea	Izu Islands, Japan region	Chagos Archipelago region	49 km NNW of Isangel, Vanuatu	92 km WS Myitky Myar
type	earthquake	earthquake	earthquake	earthquake	earthqu
horizontalError	10.07	9.17	8.02	8.52	
depthError	1.765	1.87	1.792	7.433	1
magError	0.048	0.042	0.09	0.081	0
magNst	141.0	187.0	40.0	46.0	
status	reviewed	reviewed	reviewed	reviewed	revie
locationSource	us	us	us	us	
magSource	us	us	us	us	

Describe prints the count,mean and desciptive statistics of the columns, which helps in filling null values.

```
df.describe()
```

Out[53]:

	count	mean	std	min	25%	50%	75%	
latitude	26642.0	16.852798	30.389200	-65.8497	-6.415275	18.884167	41.82795	86
longitude	26642.0	-11.487497	130.053399	-179.9987	-149.608650	-64.811833	126.96510	179
depth	26642.0	67.491224	116.762456	-3.3700	10.000000	21.998000	66.83300	681
mag	26642.0	4.007395	0.794423	2.6000	3.220000	4.300000	4.50000	7
nst	25227.0	42.571332	37.662352	0.0000	19.000000	30.000000	52.00000	423
gap	25225.0	124.930971	67.430145	8.0000	73.000000	111.000000	165.00000	350
dmin	24776.0	2.692908	4.043568	0.0000	0.612000	1.579000	3.17200	50
rms	26642.0	0.581575	0.256276	0.0100	0.410000	0.590000	0.75000	1
horizontalError	25093.0	7.017267	4.072365	0.0000	4.140000	7.060000	9.73000	99
depthError	26642.0	4.475056	4.451649	0.0000	1.848000	2.019000	6.66900	60
magError	24970.0	0.122735	0.102271	0.0000	0.080000	0.111000	0.15000	4
magNst	25065.0	33.315939	48.022567	0.0000	10.000000	18.000000	36.00000	884

```python
df['magType'].value_counts()
```

```
Out[29]: magType
mb            15906
ml             6522
md             2117
mww            1185
mwr             616
ml(texnet)      134
mw               87
mb_lg            60
mlr               4
mh                3
mwb               3
mlv               2
ms_vx             1
mwc               1
mwp               1
Name: count, dtype: int64
```

Handling Null values

To handle null values we need to first find them for that we use below code:

```python
df.isnull().sum()
```

```
Out[54]: time                  0
latitude              0
longitude             0
depth                 0
mag                   0
magType               0
nst                1415
gap                1417
dmin               1866
rms                   0
place              1608
type                  0
horizontalError    1549
depthError            0
magError           1672
magNst             1577
dtype: int64
```

Now we need fill those null columns with its resepctive technique, for numerical columns we will fill with its mean and for categorical columns we fill it with mode or related word.

```python
null_cols=["nst","gap","dmin","horizontalError","magError","magNst"]
for i in range(0,6):
    df1[null_cols[i]].fillna(df1[null_cols[i]].mean(),inplace=True)
df1["place"].fillna("unknown",inplace=True)
df.isnull.sum()
```

```
Out[59]: time              0
         latitude          0
         longitude         0
         depth             0
         mag               0
         magType           0
         nst               0
         gap               0
         dmin              0
         rms               0
         place             0
         type              0
         horizontalError   0
         depthError        0
         magError          0
         magNst            0
         dtype: int64
```

Handling Duplicates

Duplicates should be removed for reducing redundancy in the dataset.To remove we need to find them for that we need to excute the below code

```
duplicates = df1.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 1960
```

To know whether deleted or not we will see the shape of the dataframe. Before deleting duplicates:

```
dataFrame.shape()
```

```
[6]: (26642, 22)
```

After deleting duplicates

```
df.shape()
```

```
Number of rows after removing duplicates: 24682
```

Date Time Splitting

We need to split date and time from time column,so that it will be used for data visualization.

```
df['time']=pd.to_datetime(df['time'])
df['date']=df['time'].dt.date
df['time']=df['time'].dt.time
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 24682 entries, 0 to 26641
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   time             24682 non-null  object
 1   latitude         24682 non-null  float64
 2   longitude        24682 non-null  float64
 3   depth            24682 non-null  float64
 4   mag              24682 non-null  float64
 5   magType          24682 non-null  object
 6   nst              24682 non-null  float64
 7   gap              24682 non-null  float64
 8   dmin             24682 non-null  float64
 9   rms              24682 non-null  float64
 10  place            24682 non-null  object
 11  type             24682 non-null  object
 12  horizontalError  24682 non-null  float64
 13  depthError       24682 non-null  float64
 14  magError         24682 non-null  float64
 15  magNst           24682 non-null  float64
 16  date             24682 non-null  object
dtypes: float64(12), object(5)
memory usage: 3.4+ MB
```

Binning

Binning should be done for better data visualisation.

```
bins = [0, 3, 4, 5, 6.48]
labels = ['Magnitude 0-3', 'Magnitude 3-4', 'Magnitude 4-5',
   f'Magnitude 5-6.48']
df['mag_bin'] = pd.cut(df['mag'], bins=bins, labels=labels,
   right=False, include_lowest=True)
```

```
df.columns
```

```
Index(['time', 'latitude', 'longitude', 'depth', 'mag', 'magType', 'nst',
       'gap', 'dmin', 'rms', 'place', 'type', 'horizontalError', 'depthError',
       'magError', 'magNst', 'date', 'mag_bin'],
      dtype='object')
```

We will see which magnitude range has most of the values.

```
df['mag_bin'].value_counts()
```

```
Out[96]: mag_bin
         Magnitude 4-5       13742
         Magnitude 0-3        4725
         Magnitude 3-4        4440
         Magnitude 5-6.48     1775
         Name: count, dtype: int64
```

Handling Outliers

We need to handle outliers ,if not it may lead to overfitting, handling is done as below:

```
cols=['depth','mag','nst','gap','dmin','rms','horizontalError','depthError','
for col in cols:
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df1[col] = df1[col].clip(lower_bound, upper_bound)
```

As this seismic data is important for model prediction and training ,capping of outliers is good over removing.

One Hot Encoding(Converting categorical values to numerical values

```
df = pd.get_dummies(df, columns=['type','magType'])
```

```
df.columns
```

```
Index(['time', 'latitude', 'longitude', 'depth', 'mag', 'nst', 'gap', 'dmin',
       'rms', 'place', 'horizontalError', 'depthError', 'magError', 'magNst',
       'date', 'mag_bin', 'type_Landslide', 'type_earthquake',
       'type_explosion', 'type_ice quake', 'type_landslide',
       'type_mining explosion', 'type_quarry blast', 'type_volcanic eruption',
       'magType_mb', 'magType_mb_lg', 'magType_md', 'magType_mh', 'magType_ml',
       'magType_ml(texnet)', 'magType_mlr', 'magType_mlv', 'magType_ms_vx',
       'magType_mw', 'magType_mwb', 'magType_mwc', 'magType_mwp',
       'magType_mwr', 'magType_mww'],
      dtype='object')
```

Data Splitting

Here the data is shuffled for better model performance and the data is divided into train and test dataframes with 0.2 of 1 for testing and 0.8 of 1 for training.X is the datframe formed after removing uncorrelated columns and target variable and y is the target variable.

```
df_sample=df.sample(frac=1)
df_sample_copy=df_sample.copy()
from sklearn.model_selection import train_test_split

X = data_sample_copy.drop(columns=['time', 'date', 'place',
    'mag','mag_bin'])
y = data_sample_copy['mag']

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2)
```

Model Building

There are so many techniques in model building let us use 3 techniques out of it those are linear regression,polynomial regression,random forest regression

Linear Regression

```
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
linear_predictions = linear_model.predict(X_test)
```

Polynomial Regression

In order to train the model using polynomial regression first we need to change its degree as per our dataset.We choose degree 2,and there we will transform those train and test sets of X and then we will fit train sets into the model.

```
poly_features = PolynomialFeatures(degree=2)
X_poly_train = poly_features.fit_transform(X_train)
X_poly_test = poly_features.transform(X_test)

poly_model = LinearRegression()
poly_model.fit(X_poly_train, y_train)
poly_predictions = poly_model.predict(X_poly_test)
```

Random Forest Regression

Here while making an model we pass some hyperparameters for tuning there are many like maxdepth,etc.We passed nestimators with a value of 100.nestimator=100 means that 100 weak decision trees are group to form a strong regressor.

```
rf_model = RandomForestRegressor(n_estimators=100)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

Metrics(Model Evaluation)

As in the above steps we have trained models now we need to check the performance of each and decide which technique is best for this dataset.Before checking the accuracy we need to take a base prediction and base Root mean square error for making a simple model with which we cannot decide our model is working well or not.if the rmse of the models is less than the base rmse then the model is performing well or else not.we choose mean as our data is symmetrically distributed.

```
base_pred=np.mean(y_test)
print("base prediction is",base_pred)
```

```
base prediction is 3.9658598339173583
```

```
base_pred=np.repeat(base_pred,len(y_test))
print("length of base predictions is",len(base_pred))
```

```
length of base prediction is 4937
```

```
base_root_mean_squared_error=np.sqrt(mean_squared_error(y_test,base_pred))
print("base rmse is :',base_root_mean_squared_error)
```

```
base mse is  0.8095708351859834
```

Linear Regression

```
linear_predictions
```

```
array([4.07268514, 4.28841   , 4.72816686, ..., 4.18346733, 4.42264933,
       5.55716754])
```

```
lin_ms=mean_squared_error(y_test,linear_predictions)
lin_rms=np.sqrt(lin_ms)
print("Linear RMSE is:",lin_rms)
```

```
Linear RMSE is: 0.28268572205779446
```

As the value of rmse of linear model is less than the base rmse it its produces less errors.

```
train_score=linear_model.score(X_train,y_train)
test_score=linear_model.score(X_test,y_test)
train_score,test_score
```

```
(0.8819172946106953, 0.8780735192525182)
```

Train and test score represents the variance of the model as it was about 0.88 for both train and test the relationships are clearly trained by the model.

Polynomial Regression

```
poly_predictions
```

```
array([4.59034768, 4.86434824, 4.45032945, ..., 4.48167254, 2.83989969,
       4.21042027])
```

```
poly_ms=mean_squared_error(y_test,poly_predictions)
poly_rms=np.sqrt(poly_ms)
print(poly_rms)
```

As the value of rmse of polynomial model is less than the base rmse it its produces less errors and better performance than linearmodel.

```
train_score=poly_model.score(X_poly_train,y_train)
test_score=poly_model.score(X_poly_test,y_test)
train_score,test_score
```

```
Polynomial Regression RMSE: 0.24934456504225688
```

```
(0.912296450222331, 0.9057654930540987)
```

Train and test score represents the variance of the model as it was about 0.90 for both train and test the relationships are clearly trained by the model.

Random Forest Regression

```
rf_predictions
```

```
array([4.267 , 4.204 , 4.717 , ..., 4.169 , 4.651 , 5.8946])
```

```
rf_ms=mean_squared_error(y_test,rf_predictions)
rf_rms=np.sqrt(rf_ms)
print(rf_rms)
```

```
Random Forest RMSE is: 0.22162277999090899
```

As the value of rmse of random forest model is less than the base rmse it its produces less errors and better performance than linearmodel and polymodel.

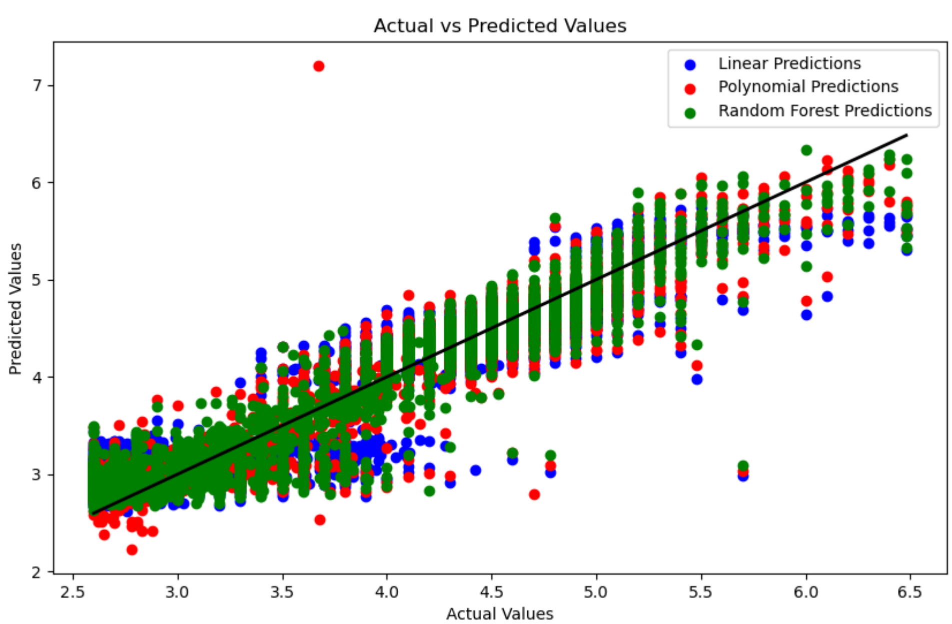
```
train_score=rf_model.score(X_train,y_train)
test_score=rf_model.score(X_test,y_test)
train_score,test_score
```

```
(0.9893288712050212, 0.925059068334223)
```

Train and test score represents the variance of the model as it was about 0.98,0.92 for train and test respectively, the relationships are clearly trained by the model and We can say that base rmse is less for random forest regressor and even train and test scores are very high,so we can say that the predictions are good among all.

Regression plot for models

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, linear_predictions, color='blue',
    label='Linear Predictions')
plt.scatter(y_test, poly_predictions, color='red',
    label='Polynomial Predictions')
plt.scatter(y_test, rf_predictions, color='green', label='Random
    Forest Predictions')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
    color='black', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.legend()
plt.savefig('reggplot')
plt.show()
```



Three models regression plots are shown in this above plot, We have added this in code section for better understanding which model performance is good. By observing the plot the model which is less deviated form diagonal line that model performance is good we can say by observing random forest has good accuracy than all.

Chapter 4

Conclusion and Future Work

In conclusion, our project aimed to predict earthquake magnitudes using machine learning models like linear regression, polynomial regression, and random forest regression. We trained these models with our data and found that random forest regression performed the best, achieving the highest scores on both training and test datasets. This suggests that its predictions are quite accurate.

Looking ahead, we plan to explore more advanced techniques to further improve prediction accuracy. Additionally, we believe our dataset could also be used to predict the occurrence of earthquakes over time, making it a valuable resource for future research and disaster preparedness efforts.