Visualization of Earthquakes in the U.S. Using Dash Plotly

Aashutosh Aryal[#1], Sabin Thapa[*2]

Department of Computer Science and Engineering, Kathmandu University
Dhulikhel-4, Kavrepalanchok, Nepal

[1]aa02430718@student.ku.edu.np

[2]st02435818@student.ku.edu.np

***Abstract*— The data in the world is abundantly growing and it is not possible to understand such a huge amount of data without some graphical tools. Visualization plays a great role in helping understand complex data in a much simpler way even for non-technical people. In recent years, many strong and mild earthquakes have been occurring very frequently around the world. The United States of America is no exception. Since 1990, earthquakes have occurred in 39 U.S states. There has not been a major quake on the U.S. mainland since the 6.7 magnitude Northridge, California event in January 1994. However, minor earthquakes are frequent even today. The aim of this study is to find possible solutions to represent earthquake catalog data of the U.S. obtained from the U.S. Geological survey using visualization techniques. The dataset contains records for earthquakes all around the world, therefore, steps like data wrangling, Exploratory Data Analysis, data conversion and finally visualization are performed on the dataset inorder to obtain and visualize the earthquakes in the U.S. The visualization is performed using Python's Plotly Dash.**

***Keywords*— data visualization, data mining, Exploratory Data Analysis, data cleaning, big data**

I. Introduction

Natural calamities like the earthquake cannot be prevented but the effects can be significantly mitigated by identifying hazards and building safer structures in earthquake-prone regions. This work aims to give a visual information on the number of earthquakes occurring in different states in the U.S. in different months. With the help of visual elements like graphs, charts and maps, data visualization tools provide an accessible way to observe and understand trends, outliers and patterns in the data. Moreover, information can be presented to non-technical people much more efficiently using the visualization techniques.

The USA has invested a huge sum of money to mitigate the earthquake effect. However, the Federal Emergency Management Agency (FEMA) reported that earthquake losses in the United States add up to about $4.4 billion annually [1]. With the visualization tools, the concerned bodies can have a better understanding of the earthquakes taking place in different states and plan urbanization, industrialization and development accordingly. Therefore, understanding patterns, trends and outliers in the abundant data we have using visualization tools can help solve real world problems and help better the economy.

II. Technical Background

The earthquake data set is very interesting to work with as it contains geographical information and earthquake information which when utilized properly can help understand the data properly and hopefully help mitigate the effects of earthquakes.

The data was obtained from the U.S. Geological Survey. The USGS monitors, analyzes, and predicts current and evolving Earth-system interactions and delivers actionable information at scales and timeframes relevant to decision makers. The USGS provides science about natural hazards, natural resources, ecosystems and environmental health, and the effects of climate and land-use change.

The data set contains the records of earthquakes from all over the world, but a major portion of the dataset contained records for earthquakes in the U.S. Therefore, study of the patterns of earthquakes in the U.S. is performed in this work.

Using Python programming language and Plotly Dash framework of Python, which is an open source framework for building data visualization interfaces, visualization of the data from the dataset is performed.

A. Dash Plotly

Dash apps give a point-&-click interface to models written in Python, vastly expanding the notion of what's possible in a traditional "dashboard." With Dash apps, data scientists and

engineers put complex Python analytics in the hands of business decision-makers and operators [2].

B. *The USGS Data Set*

The USGS data set is maintained by the U.S. using the data obtained from the U.S. geological survey. The data set has its own API documentation [3] for the earthquake catalog where every method, queries and parameters are discussed. In addition, the data set is updated frequently, so the data used are up-to-date.

III. Related Work

There are several works related to earthquake data mining and data visualization. One of them is earthquake prediction using data mining [4]. This paper uses predictive statistical models that can be applied to areas such as seismic activity, the spreading of fire by visually presenting the data to the users. Our dashboard application also makes use of data mining in determining the region in the U.S. with the most earthquakes.

Another work is a review of application of data mining in earthquake prediction [5] which presents a review of application of data mining in the prediction of natural geological calamities. Our work can be expanded to help predict earthquakes in the earthquake-prone areas of the U.S. like Alaska using predictive statistical models.

Earthquakes are unpredictable natural phenomena that create a vast amount of damage, affecting communities and their environment. To reduce the effects of such hazards, frameworks like building resilience have emerged. These frameworks target increasing recovery after such a disaster, by introducing new designs, technologies, and components to the building as proposed by Kahandawa [6].

IV. Methods

Firstly, understanding the data properly is very crucial before actually visualizing the data. For this, various methods are performed to pre-process the data. The methods like Exploratory Data Analysis, data wrangling, data cleaning, noise reduction were performed in advance then when the data was understood properly and that the data was obtained in the correct format, data visualization was performed using the visualization tools through charts, maps and graphs.

A. *Exploratory Data Analysis*

The structure of the data was explored. Upon exploration, it was found that the data set contained a total of 35922 records (including headers) and 22 columns each.

```
dataviz (main) [?] $ wc -l datasets/{january.csv,february.csv,march.csv}
  12424 datasets/january.csv
  11184 datasets/february.csv
  12314 datasets/march.csv
  35922 total
dataviz (main) [?] $ awk -F ',' '{print NF; exit}' datasets/{january.csv,february.csv,march.csv}
22
dataviz (main) [?] $
```

The data is then loaded into the datasets. Here, we see 3 records less because the headers from the 3 files are not taken into consideration by pandas when calculating the shape of the data.

```
In [2]: def csv_to_df(file):
            return pd.read_csv(
                file,
                parse_dates=["time"]
            )

In [3]: jan = csv_to_df("datasets/january.csv")
        feb = csv_to_df("datasets/february.csv")
        mar = csv_to_df("datasets/march.csv")

In [4]: sum(size for (size, _) in (jan.shape, feb.shape, mar.shape))
Out[4]: 35919

In [5]: earthquakes = pd.concat([jan, feb, mar])

In [6]: earthquakes.shape
Out[6]: (35919, 22)
```

Of the various available columns and referring to their documentation, we deemed only a few columns to be sufficient for our exploration. So we got rid of the unnecessary attributes for the visualization.

```
In [7]: earthquakes.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 35919 entries, 0 to 12312
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   time             35919 non-null  datetime64[ns, UTC]
 1   latitude         35919 non-null  float64
 2   longitude        35919 non-null  float64
 3   depth            35919 non-null  float64
 4   mag              35916 non-null  float64
 5   magType          35916 non-null  object
 6   nst              21283 non-null  float64
 7   gap              26711 non-null  float64
 8   dmin             22113 non-null  float64
 9   rms              35919 non-null  float64
 10  net              35919 non-null  object
 11  id               35919 non-null  object
 12  updated          35919 non-null  object
 13  place            35784 non-null  object
 14  type             35919 non-null  object
 15  horizontalError  23893 non-null  float64
 16  depthError       35919 non-null  float64
 17  magError         25175 non-null  float64
 18  magNst           26684 non-null  float64
 19  status           35919 non-null  object
 20  locationSource   35919 non-null  object
 21  magSource        35919 non-null  object
dtypes: datetime64[ns, UTC](1), float64(12), object(9)
memory usage: 6.3+ MB

In [8]: # Dropping unnecessary columns
        earthquakes.drop(
            ["nst", "gap", "dmin", "rms", "updated", "horizontalError", "depthError", "magError", "magNst"],
            axis=1
        ).head()
```

B. Data Wrangling

To make the complex dataset more accessible, easier to analyze, and to remove errors, data wrangling was performed. Data wrangling, sometimes referred to as data munging, is the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics [7].

We first started by extracting the locations where the earthquakes occurred. For this, using the 'place' attribute seemed like the simplest and the most efficient way to do so.

```
earthquakes["place"]
```

```
id
ok2022ccvb                    7 km WSW of Alex, Oklahoma
ak0221e05fgq                                Central Alaska
ak0221e051u3    34 km SE of Denali National Park, Alaska
ak0221e050xt                     35 km S of Adak, Alaska
ak0221e04enn    34 km SE of Denali National Park, Alaska
                                  ...
ak0222r8d3ze                76 km SW of Nanwalek, Alaska
nc73699171                     22km ENE of Alum Rock, CA
ak0222r8acmt                69 km NW of Skwentna, Alaska
uu60483552                        4 km S of Joseph, Utah
uw61819426                  6 km NNW of Six Mile, Canada
Name: place, Length: 35919, dtype: object
```

However, upon inspection, we found several flaws on the 'place' attribute which make it difficult to use for extracting location information that was originally intended. The flaws in the attribute were:

- **Missing values**

  135 earthquakes reported missing value for the place attribute.

```
earthquakes[earthquakes["place"].isnull()].shape
```

```
(135, 9)
```

- **Inconsistent data format**

  Some places have state codes, while others have the full state name and country names in some (last record in the image attached below has Canada). Some records did not have the place name and only the state. Even the full state names were inconsistent (Central Alaska vs Alaska).

```
states = earthquakes["place"].str.split(
    pat=",",
    n=1,
)
```

```
states
```

```
id
ok2022ccvb                      [7 km WSW of Alex,  Oklahoma]
ak0221e05fgq                                  [Central Alaska]
ak0221e051u3    [34 km SE of Denali National Park,  Alaska]
ak0221e050xt                       [35 km S of Adak,  Alaska]
ak0221e04enn    [34 km SE of Denali National Park,  Alaska]
                                      ...
ak0222r8d3ze                  [76 km SW of Nanwalek,  Alaska]
nc73699171                       [22km ENE of Alum Rock,  CA]
ak0222r8acmt                  [69 km NW of Skwentna,  Alaska]
uu60483552                          [4 km S of Joseph,  Utah]
uw61819426                    [6 km NNW of Six Mile,  Canada]
Name: place, Length: 35919, dtype: object
```

Moreover, the place above didn't seem to correspond with the actual region (state) where the earthquakes occurred. X km some direction of some place, Alaska did not necessarily indicate that the earthquake occured in Alaska. These also made it difficult to filter out US-regions for the non-US regions, or group earthquakes by regions for where the earthquake took place.

So we had to take another approach. Since the dataset also included the features: latitude and longitude where the earthquake occurred, we could use these attributes to look up the state where they occurred. For this we used a geocoder library.

```
import geocoder

def get_state(df):
    g = geocoder.osm([df["latitude"], df["longitude"]], method="reverse")
    if g.ok:
        geojson = g.geojson
        address = geojson["features"][0]["properties"]["raw"]["address"]
        return address.get("ISO3166-2-lvl4")
    return ""
```

The ISO 3166-2 is part of the ISO 3166 standard published by the International Organization for Standardization (ISO), and defines codes for identifying the principal subdivisions (e.g., provinces or states) of all countries coded in ISO 3166-1 [8].

So, this gives us a standard uniform representation of the country-province/state where the earthquake took place making it easier to group earthquakes on various levels (group by country, state, filter based on country or state, etc).

This method, however, had a downside - it was extremely slow. 'geocoder' internally makes API calls to open street maps to do the reverse lookup. Each API call took around 1-2 seconds. Applying this on a dataset with roughly 36,000 records took

around 24 hours with occasional connection interruption making it worse. This limited us in working with data of a large period (i.e., more months). The data from the month attribute was obtained from the 'time' attribute which was present in the dataset.

The unique states present in the dataset were:

```
earthquakes["state"].unique()
```

```
array(['US-OK', 'US-AK', 'US-CA', 'IR-18', 'ID-MA', 'US-HI', 'US-PR', nan,
       'US-WA', 'CA-YT', 'CA-BC', 'US-MO', 'US-NV', 'ID-SN', 'US-UT',
       'ID-NT', 'PG-WPD', 'US-ID', 'US-TX', 'AR-A', 'US-WY', 'FJ-E',
       'CL-AT', 'PG-ESW', 'ID-SA', 'AF-BDS', 'VU-PAM', 'ID-AC', 'JP-26',
       'DO-10', 'PG-EBR', 'ID-MU', 'US-MT', 'US-TN', 'US-KS', 'JP-02',
       'PE-UCA', 'TO-02', 'PE-HUC', 'ID-PA', 'BO-P', 'HT-NI', 'MX-CHP',
       'AR-K', 'PK-GB', 'US-OH', 'CN-SC', 'CL-RM', 'ID-BE', 'US-OR',
       'SB-TE', 'US-MN', 'AF-TAK', 'CL-AP', 'RU-SAK', 'JP-23', 'MX-BCN',
       'CL-AN', 'MX-JAL', 'JP-04', 'CL-TA', 'US-SC', 'PG-MPL', 'PG-WBK',
       'CN-XJ', 'US-CT', 'JP-30', 'CR-A', 'US-CO', 'GL-KU', 'CA-NB',
       'PG-NIK', 'AR-M', 'JP-08', 'PE-ICA', 'CN-QH', 'PH-BTG', 'DO-14',
       'CO-SAN', 'CD-TA', 'AR-J', 'US-NM', 'HT-OU', 'TW-ILA', 'PG-MRL',
       'CA-QC', 'PH-AKL', 'TO-03', 'CN-XZ', 'TJ-RA', 'AU-WA', 'ID-NB',
       'IN-UT', 'PE-ARE', 'US-MP', 'ID-JI', 'MA-03', 'DO-11', 'IR-03',
       'AF-BGL', 'PE-PUN', 'NZ-STL', 'MM-11', 'TO-04', 'HT-NO', 'IL-Z',
       'JP-15', 'IR-22', 'VU-TAE', 'PH-CAV', 'TR-10', 'PG-MPM', 'MM-01',
       'AR-Y', 'JP-44', 'JP-01', 'RU-KAM', 'IN-JK', 'PH-DAO', 'GT-JU',
       'ID-JA', 'JP-16', 'US-AR', 'MM-14', 'MX-GRO', 'US-GA', 'NZ-BOP',
       'PE-PAS', 'CN-LN', 'IR-10', 'ID-GO', 'VU-SAM', 'PG-SAN', 'VU-SEE',
       'ID-SG', 'VE-R', 'AF-BAL', 'IR-17', 'ID-SB', 'CO-CAS', 'EC-F',
       'US-AZ', 'PE-TAC', 'ZA-GP', 'IN-MN', 'PK-PB', 'TR-38', 'US-ME',
       'VU-MAP', 'ID-PB', 'MN-065', 'AF-BDG', 'RU-CHU', 'VU-TOB', 'ID-BT',
       'MX-OAX', 'US-SD', 'JP-46', 'PH-IFU', 'RO-VN', 'SB-MK', 'US-NH',
       'IR-08', 'IR-04', 'CN-YN', 'PH-DVO', 'RU-BU', 'CR-P', 'JP-12',
       'PL-02', 'JP-20', 'NZ-TKI', 'US-VI', 'PK-BA', 'US-NY', 'IQ-SU',
       'CL-CO', 'ID-ST', 'PG-NPP', 'PH-MAS', 'NI-LE', 'TJ-GB', 'PH-BTN',
       'TW-TXG', 'ID-JT', 'CN-GS', 'PE-LIM', 'BT-14', 'JP-28', 'ZA-NC',
       'PH-CAG', 'IR-05', 'TO-05', 'CL-ML', 'IN-AN', 'PG-NSB', 'AR-F',
       'CL-VS', 'TZ-23', 'PE-PIU', 'US-MA', 'GT-SM', 'BO-S', 'RU-SA',
       'TL-AN', 'CL-LI', 'SB-WE', 'IR-07', 'NZ-HKB', 'TN-HB', 'SS-EE',
```

Since we were concerned with the earthquakes in the US, we drop values based on the 'state' attribute which don't correspond to the US region ie. don't start with 'US-'.

```
# We see states for countries other than the USA. Also, the states field reports null for islands and stuff.
#So dropping countries other than the USA.
us_earthquakes = earthquakes[earthquakes["state"].str.startswith("US", na=False)]
```

We also see some earthquakes with missing magnitude. Since no information about these earthquakes wer found for manual insertion and since magnitude couldn't be extracted off of another attribute, these were dropped as well.

```
us_earthquakes[us_earthquakes["mag"].isnull()]
```

	id	time	latitude	longitude	mag	magType	place	type	status	locationSource	state
5013	nc71127084	2022-01-18 14:50:55.480000+00:00	35.975333	-120.371000	NaN	NaN	10km NNE of Parkfield, CA	earthquake	reviewed	nc	US-CA
10631	nc73674341	2022-01-06 06:11:48.790000+00:00	38.798500	-122.701500	NaN	NaN	3km NNW of Anderson Springs, CA	earthquake	reviewed	nc	US-CA
2422	pr71336423	2022-02-21 21:01:10.020000+00:00	17.956500	-66.928167	NaN	NaN	2 km SW of Guánica, Puerto Rico	earthquake	reviewed	pr	US-PR

```
# Dropping earthquakes with missing magnitude values
us_earthquakes = us_earthquakes.dropna(subset=["mag"])
```

C. Data Conversion

Upon inspecting the data, we can see that a lot of the columns have only a few unique values. These correspond to categorical values where the data can have one of a number of available values.

```
us_earthquakes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29357 entries, 0 to 12311
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              29357 non-null  object
 1   time            29357 non-null  datetime64[ns, UTC]
 2   latitude        29357 non-null  float64
 3   longitude       29357 non-null  float64
 4   mag             29357 non-null  float64
 5   magType         29357 non-null  object
 6   place           29281 non-null  object
 7   type            29357 non-null  object
 8   status          29357 non-null  object
 9   locationSource  29357 non-null  object
 10  state           29357 non-null  object
 11  state_code      29357 non-null  object
 12  month           29357 non-null  object
dtypes: datetime64[ns, UTC](1), float64(3), object(9)
memory usage: 3.1+ MB
```

```
# Converting values to appropriate type
```

```
us_earthquakes.nunique()
```

```
id                29357
time              29352
latitude          21907
longitude         23048
mag                 578
magType               8
place             12011
type                  6
status                2
locationSource       15
state                36
state_code           36
month                 3
dtype: int64
```

So, we convert these attributes to hold categorical values.

```
us_earthquakes["magType"] = us_earthquakes["magType"].astype("category")
us_earthquakes["type"] = us_earthquakes["type"].astype("category")
us_earthquakes["status"] = us_earthquakes["status"].astype("category")
us_earthquakes["locationSource"] = us_earthquakes["locationSource"].astype("category")
us_earthquakes["state"] = us_earthquakes["state"].astype("category")
us_earthquakes["state_code"] = us_earthquakes["state_code"].astype("category")
us_earthquakes["month"] = us_earthquakes["month"].astype("category")
```

```
us_earthquakes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29357 entries, 0 to 12311
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              29357 non-null  object
 1   time            29357 non-null  datetime64[ns, UTC]
 2   latitude        29357 non-null  float64
 3   longitude       29357 non-null  float64
 4   mag             29357 non-null  float64
 5   magType         29357 non-null  category
 6   place           29281 non-null  object
 7   type            29357 non-null  category
 8   status          29357 non-null  category
 9   locationSource  29357 non-null  category
 10  state           29357 non-null  category
 11  state_code      29357 non-null  category
 12  month           29357 non-null  category
dtypes: category(7), datetime64[ns, UTC](1), float64(3), object(2)
memory usage: 1.8+ MB
```

In doing so, we see the memory usage drop from 3.1Mb to 1.8Mb. That is a 41% decrease in memory consumption. The data now seems good enough to be used for visualization.

D. Data Visualization

After performing the operations like EDA, data cleaning and error removal on the data set, the data was then ready to be visualized. Visualization was done using a Python framework, Plotly Dash.

We are visualizing earthquakes occurrence by states for each month i.e., we want to see how many earthquakes occurred in a particular month in each of the states in the U.S. For this, we group the data by the month and the state code and then count the number of occurrences of earthquakes for each state. The data is then plotted using a choropleth map [9].

```
df = df.groupby(["month", "state_code"]).agg(earthquakes=("id", "count")).reset_inde
```

```
df.sample(10)
```

	month	state_code	earthquakes
2	February	AZ	8
12	February	MA	0
79	March	HI	635
6	February	GA	0
89	March	MT	146
81	March	IL	1
87	March	MO	14
61	January	OR	25
14	February	MN	3
30	February	TX	114

We then also plot the top 10 earthquakes, by magnitude, for each month. From these two visualizations, we can see that Alaska is the most seismically-active state in the U.S.

"The landmass beneath the Pacific Ocean is one of a few dozen tectonic plates that make up the earth's crust. Each year, the Pacific Plate pushes a couple of inches towards Alaska, which is generally considered to be part of the North American Plate. Where these two plates meet, the dense oceanic rocks of the Pacific thrust under the more buoyant continental rocks of Alaska." [10]

V. Results

The main purpose of this study was to better understand the earthquakes in the U.S. using the data USGS provides through visualization techniques. Upon exploring the data and performing all the preliminary analysis and data cleaning on it, the data was finally ready to be visualized. The number of occurrences of the earthquake for each month in the states was then plotted. Along with this, the top 10 earthquakes by magnitude for each month was also plotted.

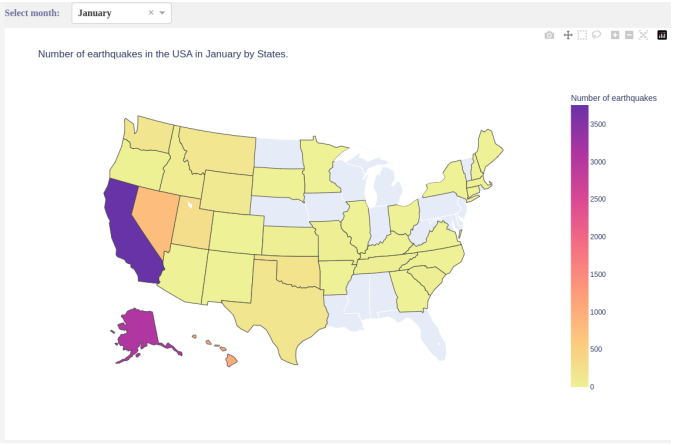

Fig. 1 Map showing the number earthquakes in the U.S

The figure shows the plot of the number of earthquakes in the month January in the U.S. The month can be picked using the dropdown provided and the color scale on the side reveals how frequent the earthquakes are in a particular state.

Upon hovering over the state, we can see the name of the state and the number of earthquakes that occurred in that state in a particular month. Clearly, Alaska is the earthquake-prone state and the number of earthquakes occurred in January was 3082.

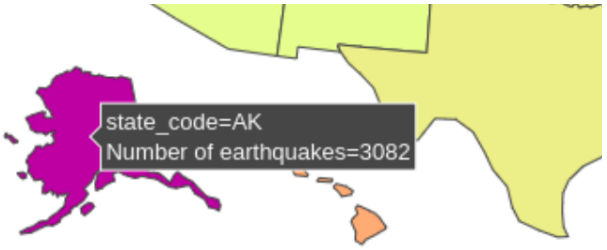

Fig. 2 Alaska with 3082 earthquakes in January

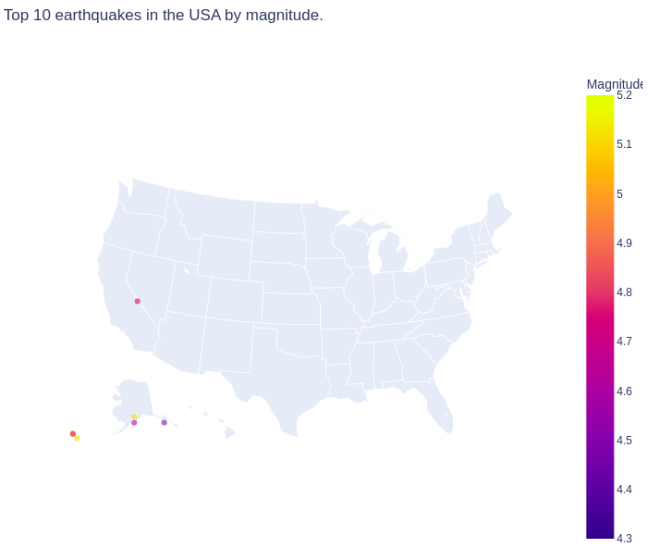

Fig. 3 Plot showing top 10 earthquakes in January by magnitude

The top 10 earthquakes by magnitude are also plotted to observe the states that are prone to high magnitude earthquakes. Visualization such as this can be very beneficial to help mitigate the earthquake effects in the future.

Further visualization is done with the help of bar graphs and pie charts. For instance, earthquakes reviewed by status and the earthquakes by type are plotted against their count using a bar graph and a pie chart. Users have an option to view the result in a bar chart or a pie chart as per their requirements.

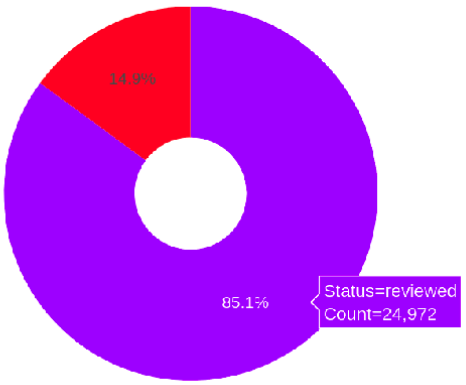

Fig. 4 Pie chart showing earthquakes by review status

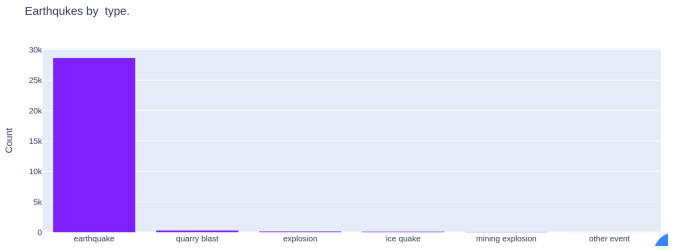

Fig. 5 Bar chart showing number of earthquakes by type

VI. Discussion

The USGS data set had thousands of records and upon cleaning the data and converting them to meet our requirement, we were able to visually represent it. The visual map showed that Alaska is the state with the highest number of earthquakes with 3082 earthquakes in the month of January alone. Upon further research, the information turned out to be true - Alaska is indeed the state with the highest number of earthquakes [11].

With the help of the plot that visualizes the top 10 earthquakes by magnitude, we can again see that the quakes of higher magnitude are again occurring in Alaska or the seas near Alaska. The January 11, 2022, M 6.6 Nikolski, Alaska (Aleutian Islands) earthquake occurred as a result of thrust faulting as the Pacific plate subducts beneath the North America plate at the Aleutian Trench, in the subduction zone extending to the southwest from Alaska [12].

From the bar chart or the pie chart, we can see the the maximum type quakes are due to the natural earthquakes(over 28,000) and some are due to quarry blasts(327), explosion, ice quakes and mining explosion.

VII. Conclusions

Using the Data Mining methodologies like Exploratory Data Analysis, Data Cleaning, Data Wrangling and Data Visualization, this work has helped us get better understanding on working with huge datasets and accurately determining the features or the attributes of our interest. The earthquake dataset initially was very huge and contained so much information that was not all of our interest. After performing all the Data Mining tasks, we were able to extract only the useful information that we needed and using visualization techniques, the information was presented graphically so that even non-technical people can have a better understanding of what's going on.

Upon review and research of other scientific and technical data, we can thus conclude that the visualization indeed corresponds to the real information and the information displayed is accurate.

To sum up, this work on visualization of the earthquakes in the U.S. is not only limited to the U.S. for just the three months(January, February and March), this can be extended to visualize the earthquake of any country in the world during any month provided there is enough data to visually represent.

Acknowledgment

This work was done for the partial fulfillment of the requirements of the course Data Mining under the guidance and supervision of Dr. Rajani Chulyadyo, Assistant Professor, Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Nepal. We are highly indebted to Dr. Chulyadyo for her continuous guidance and supervision as well as for providing us with necessary information regarding the project.

References

[1] "Annual U.S. earthquake losses," *GovCon*, 2022. [Online]. Available: https://www.govcon.com/doc/annual-us-earthquake-losses-estimated-at-44b-0001. [Accessed: 10- Nov- 2022].

[2] "Dash Plotly: The Premier Data App Platform for Python," *Plotly*, 2013. [Online]. Available: https://plotly.com/dash/. [Accessed: 01- Nov- 2022].

[3] "API Documentation - Earthquake Catalog," *USGS*, 2022. [Online]. Available: https://earthquake.usgs.gov/fdsnws/event/1/. [Accessed: 01- Nov- 2022].

[4] Kulkarni, Kulkarni, "Earthquake Prediction Using Data Mining," *International Journal of Science and Research, vol. 6, pp. 2319-7064, 2015*

[5] V. Otari, R. V. Kulkarni, "A review of application of data mining in earthquake prediction," *International Journal of Computer Science and Information, vol. 3, no. 2, pp. 3570-3574, 2012*

[6] Kahandawa K.A.R.V.D, Domingo N.D, Park K.S, Uma S.R, (2018), *'Earthquake damage estimation systems: Literature review,' Procedia Engineering, 212, pp.622-628.*

[7] "Data Wrangling," *Wikipedia, 2022*. [Online]. Available: https://en.wikipedia.org/wiki/Data_wrangling. [Accessed: 02- Nov- 2022].

[8] "International Organization for Standardization," *Wikipedia*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/International_Organization_for_Standardization/. [Accessed: 04- Nov- 2022].

[9] "Choropleth Maps in Python," *Plotly*, n.d. [Online]. Available: https://plotly.com/python/choropleth-maps/. [Accessed: 09- Nov- 2022].

[10] "Why earthquakes happen in Alaska," *University of Alaska Fairbanks*, 2022. [Online]. Available: https://earthquake.alaska.edu/earthquakes/about. [Accessed: 15- Nov- 2022].

[11] "Alaska experiencing earthquakes every 10 minutes," *New York Post*, 2022. [Online]. Available: https://nypost.com/2022/07/20/alaska-experiences-an-earthquake-every-10-minutes-scientists-say/. [Accessed: 19- Nov- 2022].

[12] "A 6.6 Magnitude Earthquake in Alaska in January," *USGS*, 2022. [Online]. Available: https://earthquake.usgs.gov/earthquakes/eventpage/us7000gawk/executive. [Accessed: 19- Nov- 2022].