# Earthquake Prediction Using Machine Learning

Lasta Maharjan
Data 3421 - Data Mining

# Introduction

Earthquakes can be described sudden shaking of the ground caused by seismic activity that happens between the movement of tectonic plates, and when these movements accumulates energy in the form of rock stress, then it is suddenly released. As recently Turkey was hit by high magnitude (7.8) earthquake causing 60,000 casualties including deaths, injured and missing. Forecasting earthquakes is one of the most important problems as the consequences of earthquakes are devastating.

# Why Earthquake Prediction?


- Nepal earthquake (April 25, 2015)
- Magnitude: 7.8 Mw or 8.1
- Aftershocks: 7.2, 7.0, 6.9, 6.2 and more till date
- Epicenter was identified of 80 km to the northwest of Kathmandu, the capital of Nepal.
- Death toll: 10,000+
- Missing cases: 15,000

# Why is it Important?

- Because earthquakes do not happen on regular intervals it is difficult to predict when the next one will occur
- Methods for predicting earthquakes on those faults vary; one of them being 100% accurate
- Predictions are given for a time frame instead of an exact date

Geologists from various nations have had predicted to have a massive earthquake within 10 years in Nepal before the actual earthquake hit on April 25, 2015.

# Dataset

- The earthquake dataset has been collected from United States Geological Survey (USGS):

https://www.usgs.gov/programs/earthquake-hazards/data

- The website is updated every 15 minutes, with the real time monitoring, station, and other seismic data.
- Size: 2.27 MB in CSV format
- 22 attributes

# Dataset

```
df.head()
```

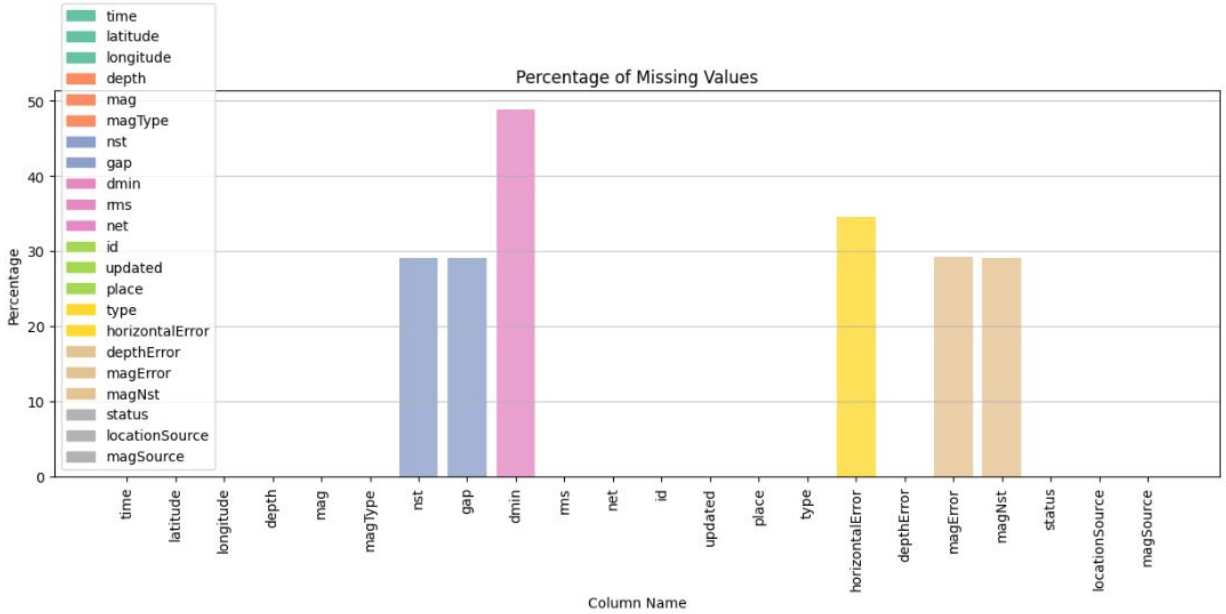
| | time | latitude | longitude | depth | mag | magType | nst | gap | dmin | rms | ... | updated | place | type | horizontalError |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2023-04-27T22:16:18.888Z | 57.103400 | -155.239500 | 12.300000 | 2.000000 | ml | NaN | NaN | NaN | 0.47 | ... | 2023-04-27T22:19:04.369Z | 67 km WNW of Akhiok, Alaska | earthquake | NaN |
| 1 | 2023-04-27T22:12:59.780Z | 19.244167 | -155.376495 | 28.650000 | 1.800000 | md | 30.0 | 144.0 | NaN | 0.21 | ... | 2023-04-27T22:16:07.780Z | 11 km ENE of Pāhala, Hawaii | earthquake | 0.800000 |
| 2 | 2023-04-27T21:57:03.753Z | 59.822700 | -153.348300 | 131.400000 | 1.800000 | ml | NaN | NaN | NaN | 0.50 | ... | 2023-04-27T22:00:03.269Z | 42 km E of Pedro Bay, Alaska | earthquake | NaN |
| 3 | 2023-04-27T21:56:56.150Z | 19.395166 | -155.273498 | 1.080000 | 2.110000 | ml | 22.0 | 60.0 | NaN | 0.23 | ... | 2023-04-27T22:02:44.730Z | 6 km SW of Volcano, Hawaii | earthquake | 0.360000 |
| 4 | 2023-04-27T21:55:45.756Z | 32.865601 | -100.934880 | 5.428955 | 2.639135 | mlv | 21.0 | 64.0 | 0.020049 | 0.20 | ... | 2023-04-27T22:09:44.913Z | 16 km N of Snyder, Texas | earthquake | 0.694442 |

5 rows × 22 columns

# Dataset Attributes

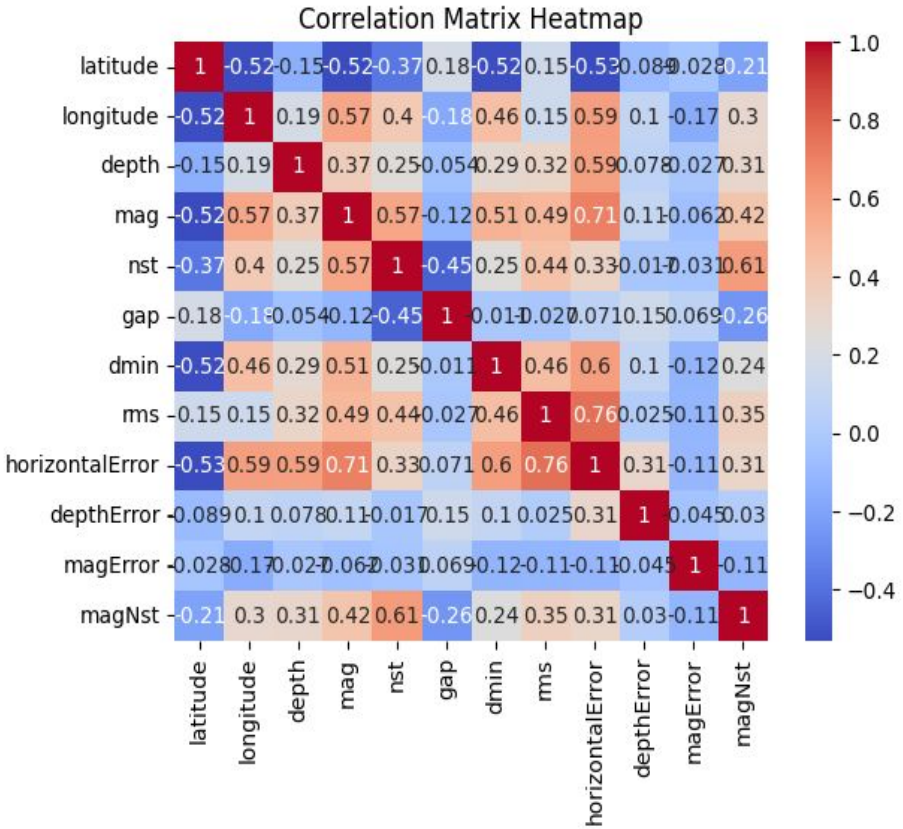
1. **time:** Date & Time when earthquake exactly took place
2. **latitude**: The latitude is the number of degrees north (N) or south (S) of the equator and varies from 0 at the equator to 90 at the poles.
3. **longitude:** The longitude is the number of degrees east (E) or west (W) of the prime meridian which runs through Greenwich, England.
4. **depth:** The depth where the earthquake begins to rupture.
5. **Mag**: Earthquake magnitude is a measure of the size of an earthquake at its source. It is a logarithmic measure.
6. **magType:** The method or algorithm used to calculate the preferred magnitude for the event.
7. **Nst:** Number of seismic stations which reported P- and S-arrival times for this earthquake.
8. **Gap:** The largest azimuthal gap between azimuthally adjacent stations (in degrees).
9. **Dmin:** Horizontal distance from the epicenter to the nearest station (in degrees).
10. **Rms:** The root-mean-square (RMS) travel time residual, in sec, using all weights.
11. **Net:** The ID of a data contributor. Identifies the network considered to be the preferred source of information for this event.
12. **Id:** A unique identifier for the event.
13. **Updated:** Time when the event was most recently updated.
14. **Place:** Textual description of named geographic region near to the event.
15. **Type:** Type of seismic event.
16. **locationSource:** The network that originally authored the reported location of this event.
17. **magSource:** Network that originally authored the reported magnitude for this event.
18. **horizontalError:** The horizontal location error, in km, defined as the length of the largest projection of the three principal errors on a horizontal plane.
19. **depthError:** The depth error, in km, defined as the largest projection of the three principal errors on a vertical line.
20. **magError:** Uncertainty of reported magnitude of the event. The estimated standard error of the magnitude.
21. **magNst:** The total number of seismic stations used to calculate the magnitude for this earthquake.
22. **status:** Indicates whether the event has been reviewed by a human. Status is either automatic or reviewed. Automatic events are directly posted by automatic processing systems and have not been verified or altered by a human.

# Data Preparation (Missing Values)

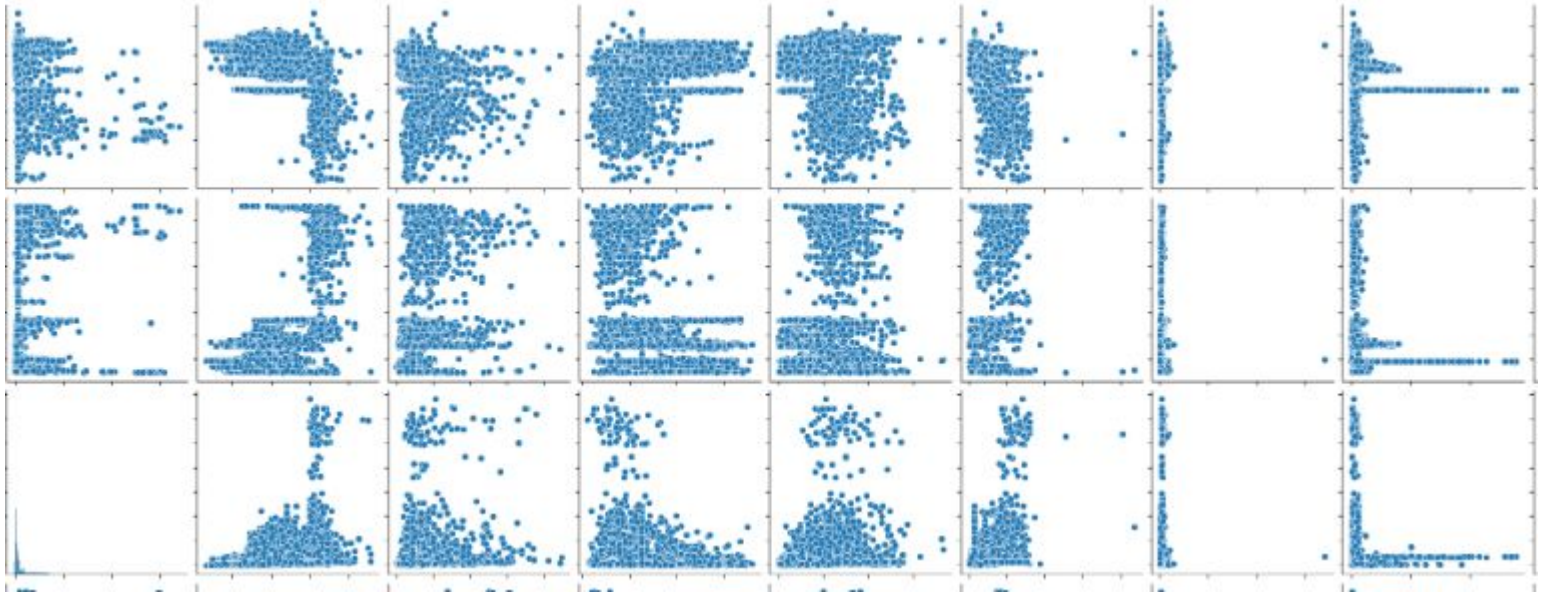


# Correlation

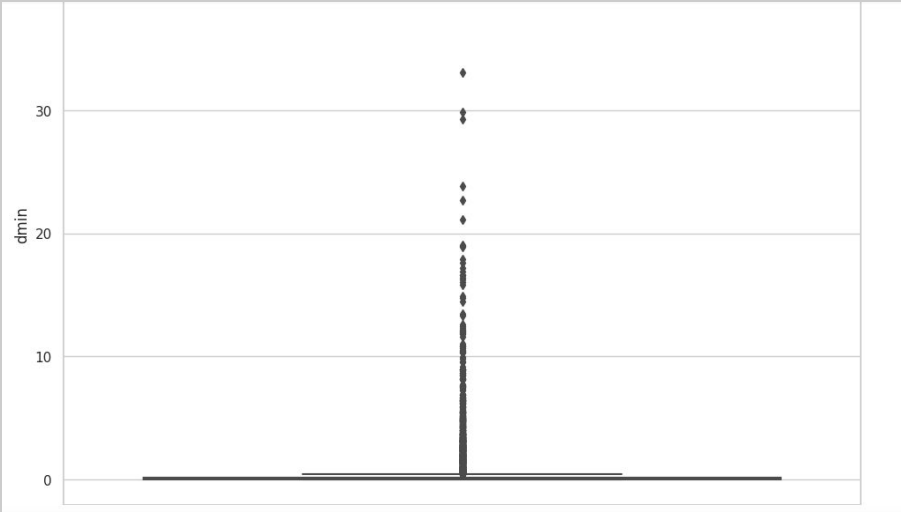
- Horizontal Error has highest correlation with Magnitude and rms The root-mean-square (RMS) travel time residual, in sec, using all weights.



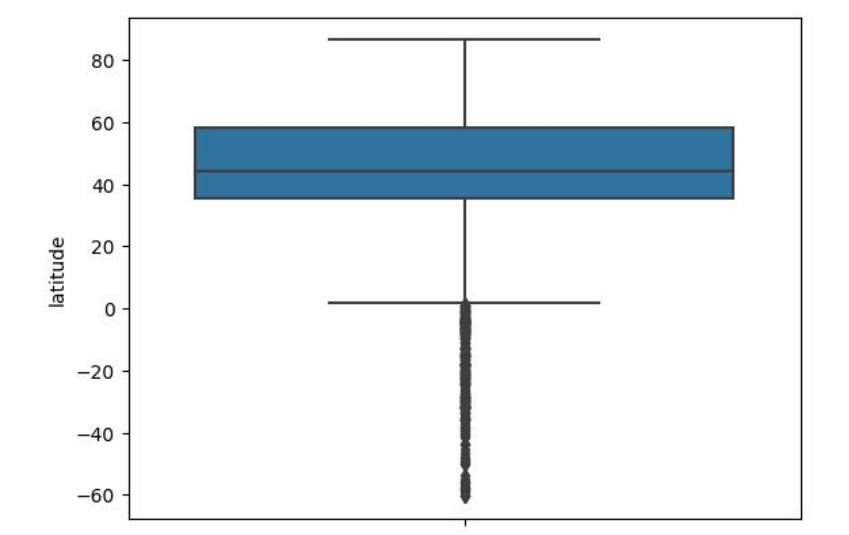
# Pairplots

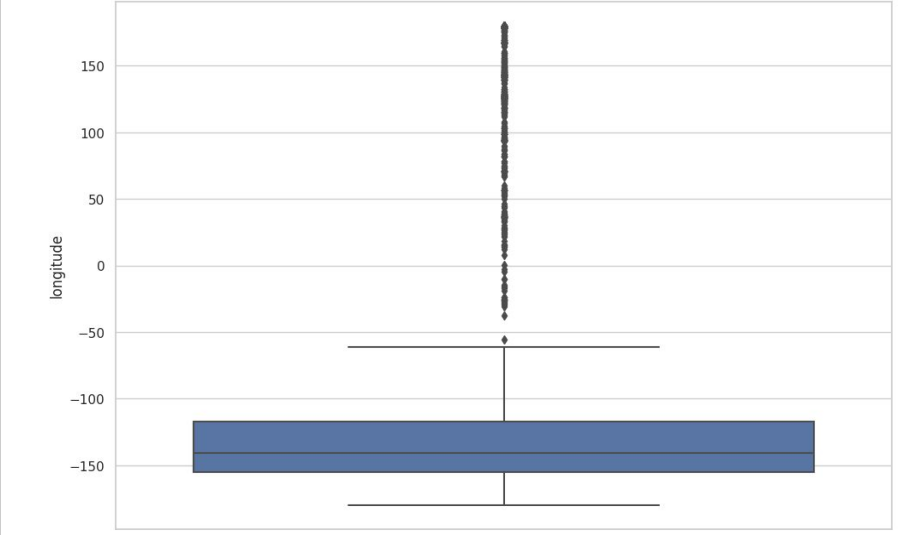


# Outliers

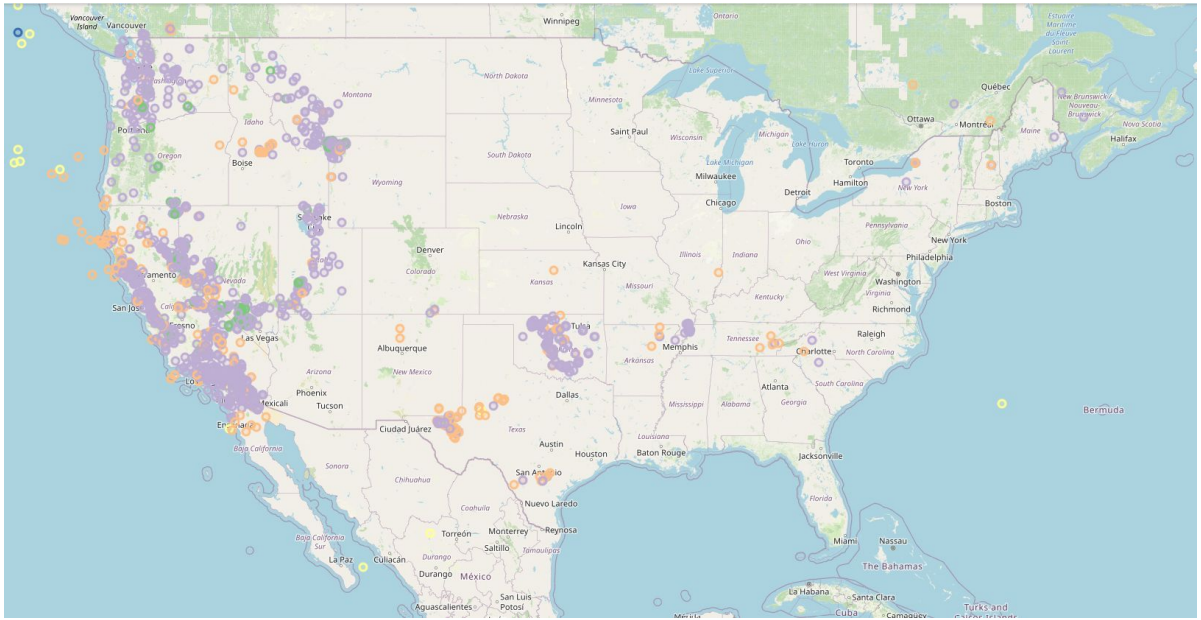


# Outliers

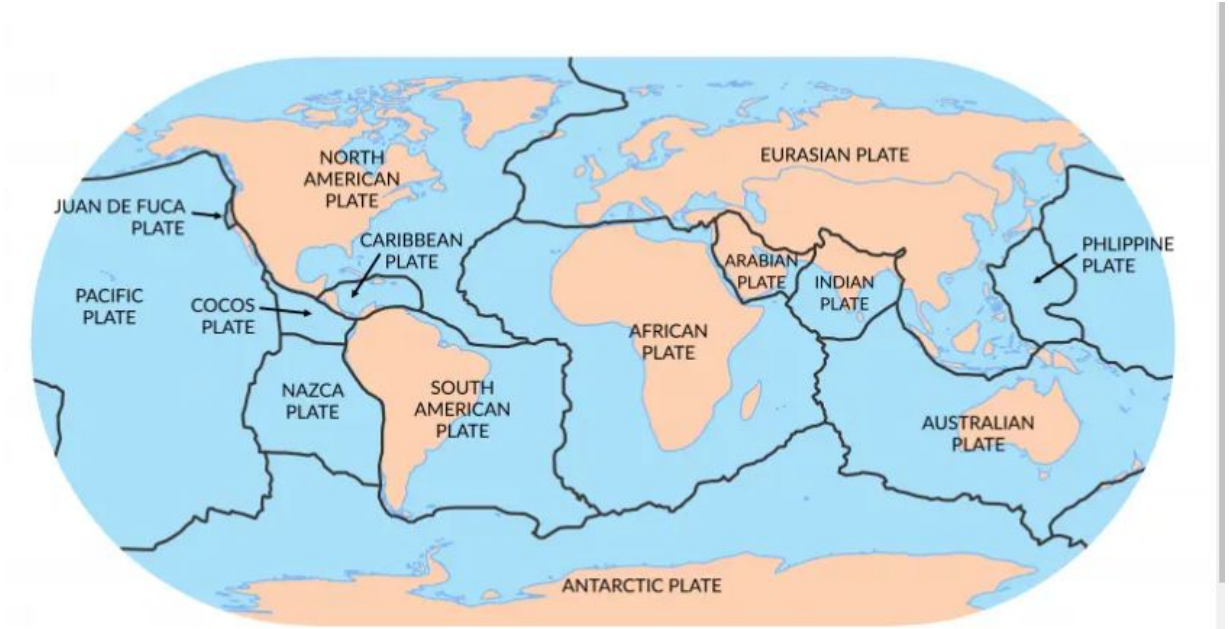


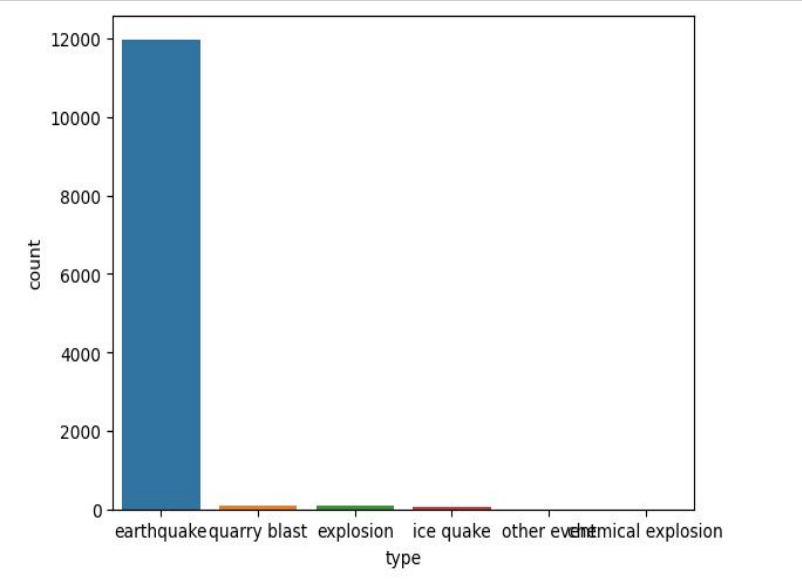
# Earthquake on USA Map

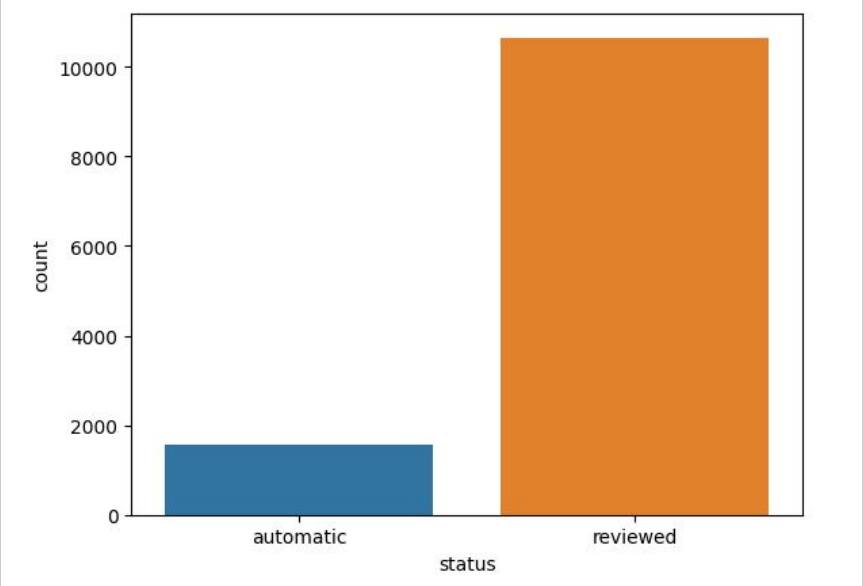


# Tectonic Plates

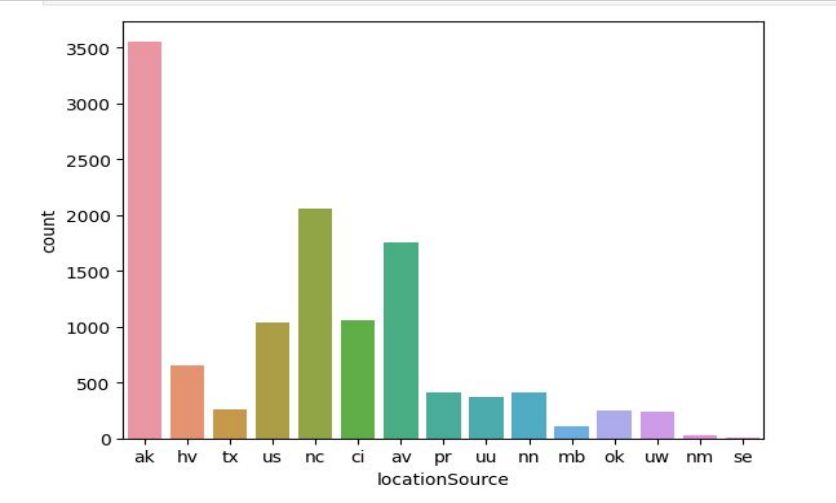


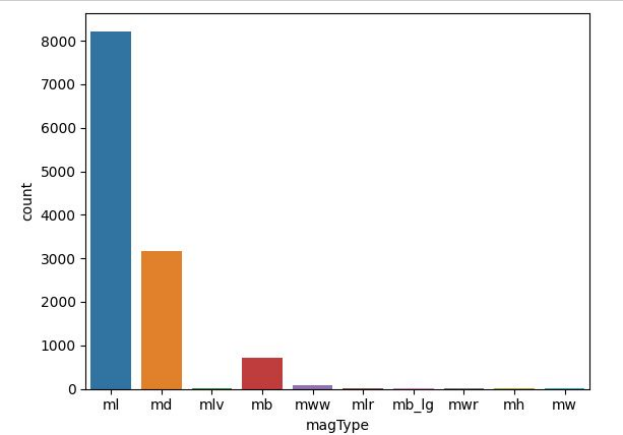
# Categorical Data

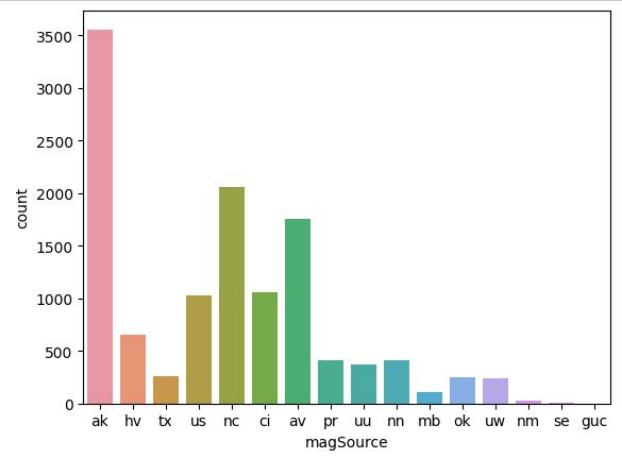


# Categorical Data



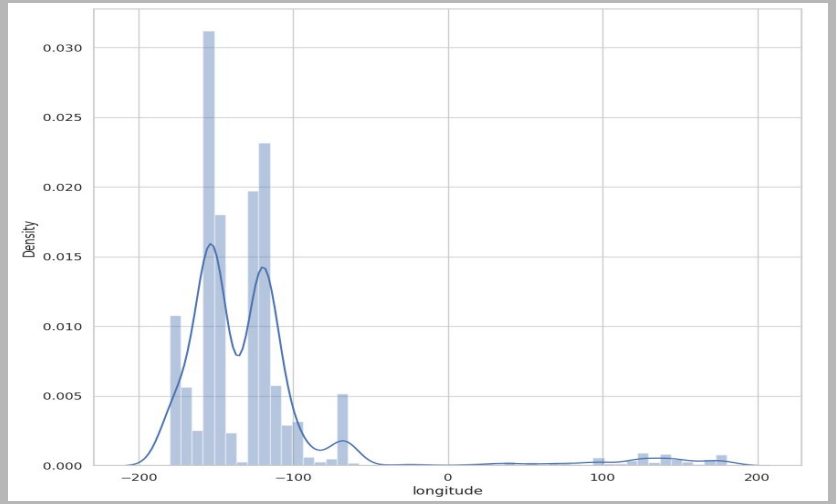
# LabelEncoding

- assigning numerical labels to categorical data earthquake type, location Source, Magnitude Source, Magnitude Type, Status, etc.
- Unique numerical labels to each categories in each features

```
encoder = LabelEncoder()
df['magType_encoded'] = encoder.fit_transform(df['magType'])
df['net_encoded'] = encoder.fit_transform(df['net'])
df['type_encoded'] = encoder.fit_transform(df['type'])
df['status_encoded'] = encoder.fit_transform(df['status'])
df['locationSource_encoded'] = encoder.fit_transform(df['locationSource'])
df['magSource_encoded'] = encoder.fit_transform(df['magSource'])
```

# Normalizing the Data

- Used MinMaxScaler to Normalize the data, a lot of them were right or left skewed.



# TimeStamp

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

```
         time   latitude   longitude        depth       mag         nst  \
0  2023-04-27  57.103400 -155.239500   12.300000  2.000000  21.045276
1  2023-04-27  19.244167 -155.376495   28.650000  1.800000  30.000000
2  2023-04-27  59.822700 -153.348300  131.400000  1.800000  21.045276
3  2023-04-27  19.395166 -155.273498    1.080000  2.110000  22.000000
4  2023-04-27  32.865601 -100.934880    5.428955  2.639135  21.000000

         gap    rms                              place  horizontalError  depthError  \
0  124.79864   0.47    67 km WNW of Akhiok, Alaska          1.463664    0.300000
1  144.00000   0.21    11 km ENE of Pāhala, Hawaii          0.800000    1.230000
2  124.79864   0.50  42 km E of Pedro Bay, Alaska           1.463664    0.500000
3   60.00000   0.23     6 km SW of Volcano, Hawaii          0.360000    0.260000
4   64.00000   0.20       16 km N of Snyder, Texas          0.694442    0.929844

   magError      magNst  magType_encoded  net_encoded  type_encoded  \
0  0.240858   15.840573                4            0             1
1  0.890000    3.000000                2            3             1
2  0.240858   15.840573                4            0             1
3  0.240000    8.000000                4            3             1
4  0.181092   12.000000                6           11             1

   status_encoded  locationSource_encoded  magSource_encoded
0               0                       0                  0
1               0                       3                  4
2               0                       0                  0
3               0                       3                  4
4               1                      11                 12
```

# Machine Learning Models

- **Split the data into training and testing sets (80% for training, 20% for testing)**
- Used models:
- 1. Support Vector Machine
- 2. RandomForest
- 3. K- nearest neighbors

## Support Vector Machine

```
svm = SVR()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
```

Mean Score: 0.281

## RandomForest Regressor

```
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)
rf_reg.fit(X_train, y_train)
y_pred_rf = rf_reg.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
print("Mean Squared Error:", mse_rf)
```

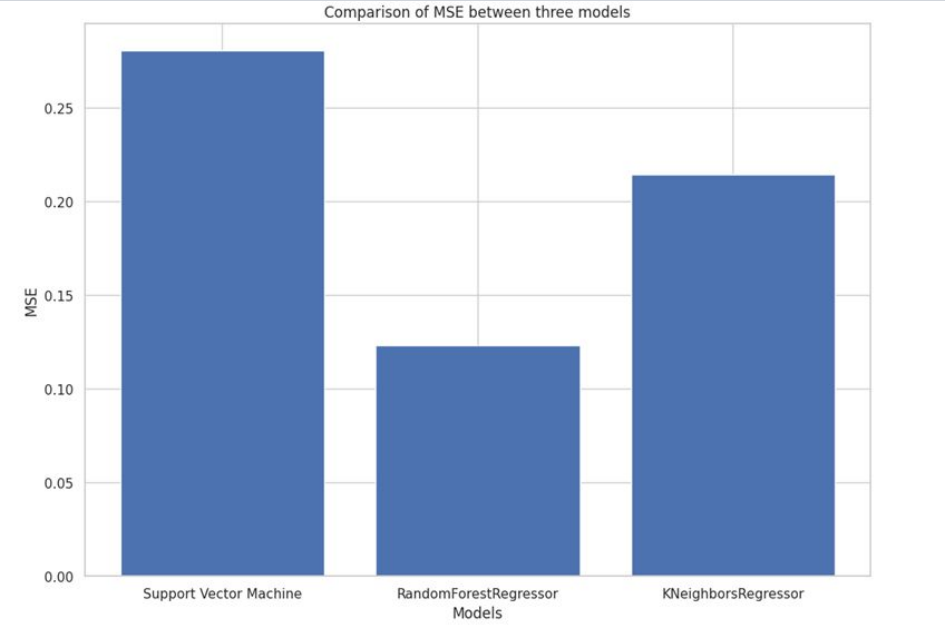
Mean Score: 0.123

# KNN Regressor

```
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)
knn_preds = knn_model.predict(X_test)
mse_knn = mean_squared_error(y_test, knn_preds)
print("Mean Squared Error (MSE):", mse_knn)
```

Mean Score: 0.2141

# Model Evaluation Comparison



# Conclusion

The earthquake magnitude prediction project has demonstrated the usefulness of machine learning algorithms in predicting the severity of earthquakes. By analyzing seismic data and using various machine learning techniques, we were able to accurately predict earthquake magnitudes within a certain range.

# Challenges

- Finding a good dataset, for some parts of the world, due to lack of seismic stations, the data was not evenly collected.
- Some datasets were missing a lot of features, only had longitude, latitude and magnitude

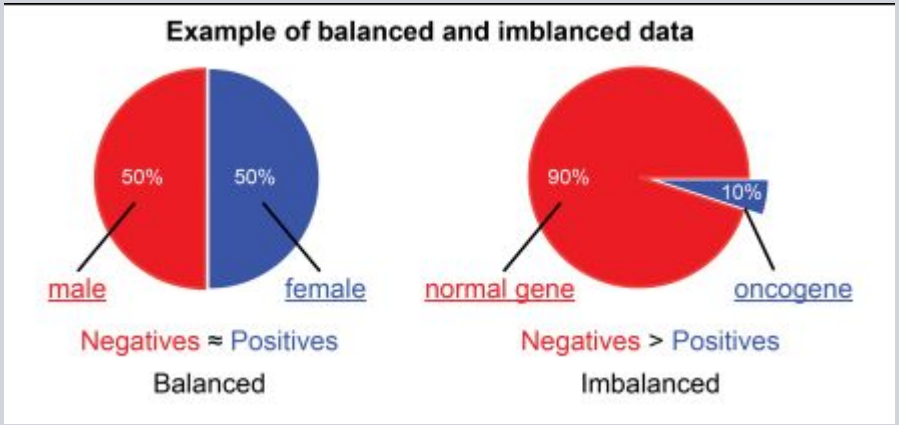
# Future Work

- Create a artificial neural network to better predict the magnitude of the earthquake
- And, in addition to predicting the magnitude of earthquake in future trying to predict the time stamp, would be my next few steps for this project.

# How to handle Imbalanced Data?

# Imbalanced Data

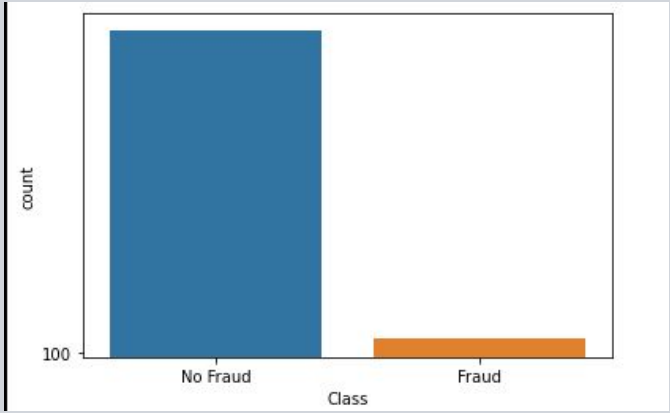
Imbalanced data refers to a situation in which the distribution of classes in a dataset is not equal, resulting in one or more classes being underrepresented or overrepresented. In other words, there is a significant difference in the number of instances of each class in the dataset.



# Example

For example, in a dataset of credit card fraud detection, the majority of transactions may be legitimate, while only a small fraction of transactions are fraudulent. This creates an imbalanced dataset, where the fraudulent transactions are underrepresented compared to the legitimate transactions.

Imbalanced data can be problematic for machine learning algorithms because they may have a tendency to favor the majority class and ignore the minority class. This can lead to biased or inaccurate models that perform poorly on the minority class.



# How to Handle it?

- Resampling the datasets to balance the classes
- Adjusting the algorithm's parameters, or using specialized algorithms designed for imbalanced data. There are also specialized machine learning algorithms that are designed specifically for imbalanced data, such as Random Forest with Balanced Random Forest algorithm or cost-sensitive learning algorithms, optimized to handle class imbalance and may outperform traditional algorithms on imbalanced datasets.
- Ensemble methods can be used to combine the results of multiple models trained on different samples of the data. Bagging, boosting and stacking methods can be used to handle imbalanced data.
- In some cases, collecting more data can help to balance the classes in the dataset, especially for rare events.

# References:


- https://en.wikipedia.org/wiki/April_2015_Nepal_earthquake
- https://en.wikipedia.org/wiki/2023_Turkey%E2%80%93Syria_earthquake
- https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php