CS506-02

Distributed Cloud Computing

Final Project

Name	Student Id
Naveen Sita Reddy Kovvuri	S02059263
Nithish Kumar Medabalimi	S02059513
Tanay Tammineni	S02065959
Sainath Reddy Mosali	S02061036
Lakshmi Kethineni	S02066249
Vidya Rani Vallala	S02066899

Spring 2024 CS506-01 22619

PROFESSOR: DR. ZIPPING LIU

Southeast Missouri State University

Project Name: Predicting the Effects of Earthquakes Using Cloud-Based Machine Learning

Project Description:

The goal of this project, "Predicting the Effects of Earthquakes Using Cloud-Based Machine Learning," is to create prediction models for earthquake features by utilizing PySpark and AWS SageMaker. Improving seismic event predicting and showcasing the effectiveness of cloud technology in handling and evaluating massive datasets are the main goals. The desired results include enhanced readiness for earthquakes and the capacity to react quickly to any seismic hazards by using reliable and accurate machine learning forecasts.

Scope:

The scope of this project includes building and implementing machine learning models with a focus on seismic event prediction using PySpark and Amazon SageMaker. Large seismic data sets are handled efficiently by the project by utilizing AWS cloud services, such as S3 for data storage and EC2 for processing power. Machine learning processes are made scalable and reliable by using SageMaker for model deployment and use of PySpark for data preparation and model training. Additionally, this project examines how to integrate AWS Lambda for possible real-time data processing, which might improve the system's ability to react to seismic activity in real time.

Requirements:

For this project, AWS's cloud computing services like EC2, S3, and Lambda, as well as PySpark for data processing and Amazon SageMaker for model deployment are required. For development and testing, Jupyter Lab is a crucial

tool, as are other Python libraries for data processing and show, such NumPy, pandas, and Matplotlib. High-performance computer resources are also required for the project to manage the massive amounts of data processing and analytics needed for earthquake prediction.

Architecture:

Data Management and Storage: Raw and processed seismic data are stored on Amazon S3, which serves as the main data repository.

Data Pre-Processing and Model Development:

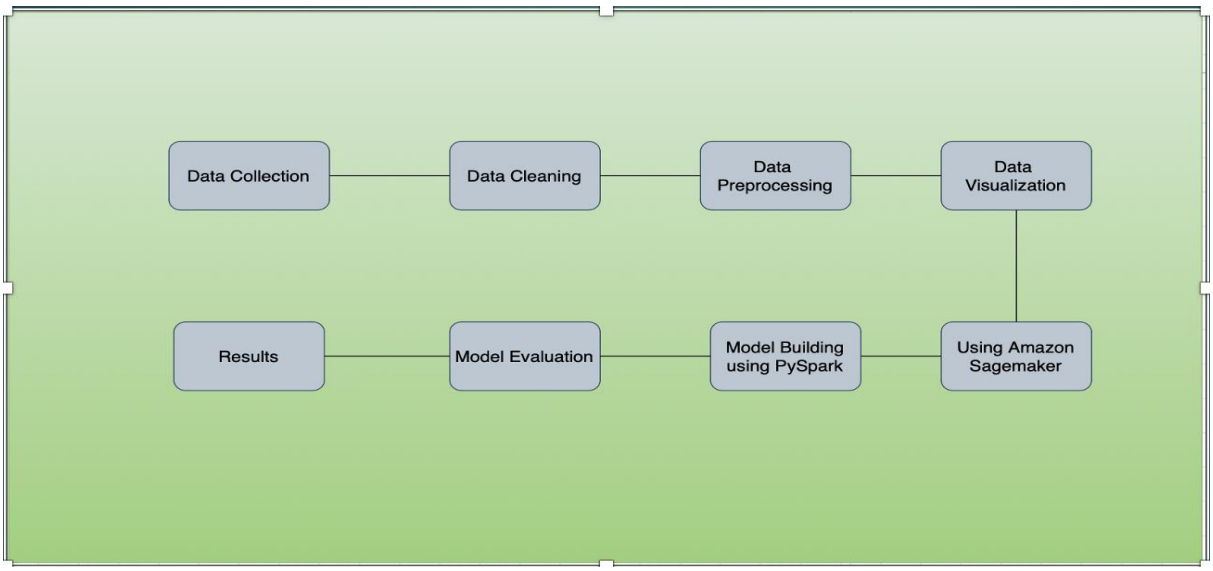
PySpark on AWS EC2: PySpark is run on EC2 instances to take use of its distributed computing resources for effective data processing and model training.

Model Management and Deployment: The machine learning models that have been trained go into production using AWS SageMaker.

Introduction:

The goal of this project is to accurately predict earthquake characteristics by utilizing PySpark and Amazon SageMaker. The project makes optimal use of cloud computing to collect and analyze massive amounts of seismic data by deploying these technologies on AWS. Enhancing earthquake prediction, improving disaster response systems, and showcasing the effectiveness and scalability of cloud-based data processing are the main goals. This project is significant because it shows how modern methods of machine learning may be used in real-world situations to reduce the risks connected with natural catastrophes, which has major implications for preparedness and safety.

Workflow Diagram:



Dataset Description:

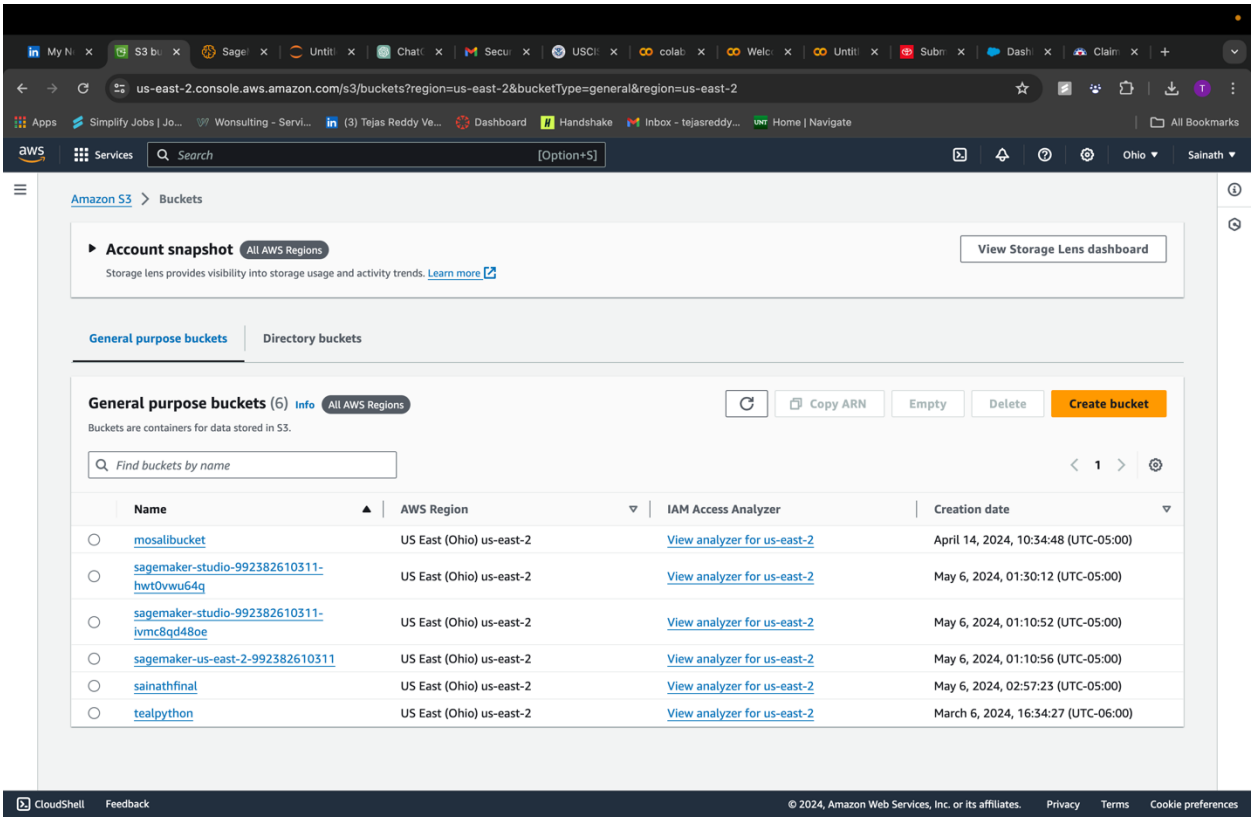
Variable Name	Description
magNst	Number of stations reporting the magnitude.
updated	last update of the record.
Depth	Depth of the earthquake in kilometers.
status	Event has been automatically reported or manually reviewed.
magType	The method or algorithm used to calculate the magnitude.
horizontalError	Errors in the measurement of the earthquake's horizontal position.
dmin	Horizontal distance from the epicenter to the nearest station (in degrees).
rms	Root Mean Square of the seismic wave, which is a measure of the accuracy of the network's earthquake location.
place	Descriptive location of the earthquake.
type	Type of seismic event.
gap	Gap between azimuthally adjacent stations.
status	Indicates if the event has been automatically reported or manually reviewed.
locationSource	The source that provided the location.
mag	Overall size or strength of the earthquake.
Time	Timestamp of the earthquake
Latitude, longitude	Coordinates of the earthquake epicenter.
nst	Number of seismic stations

Cloud Technologies and Services Used:

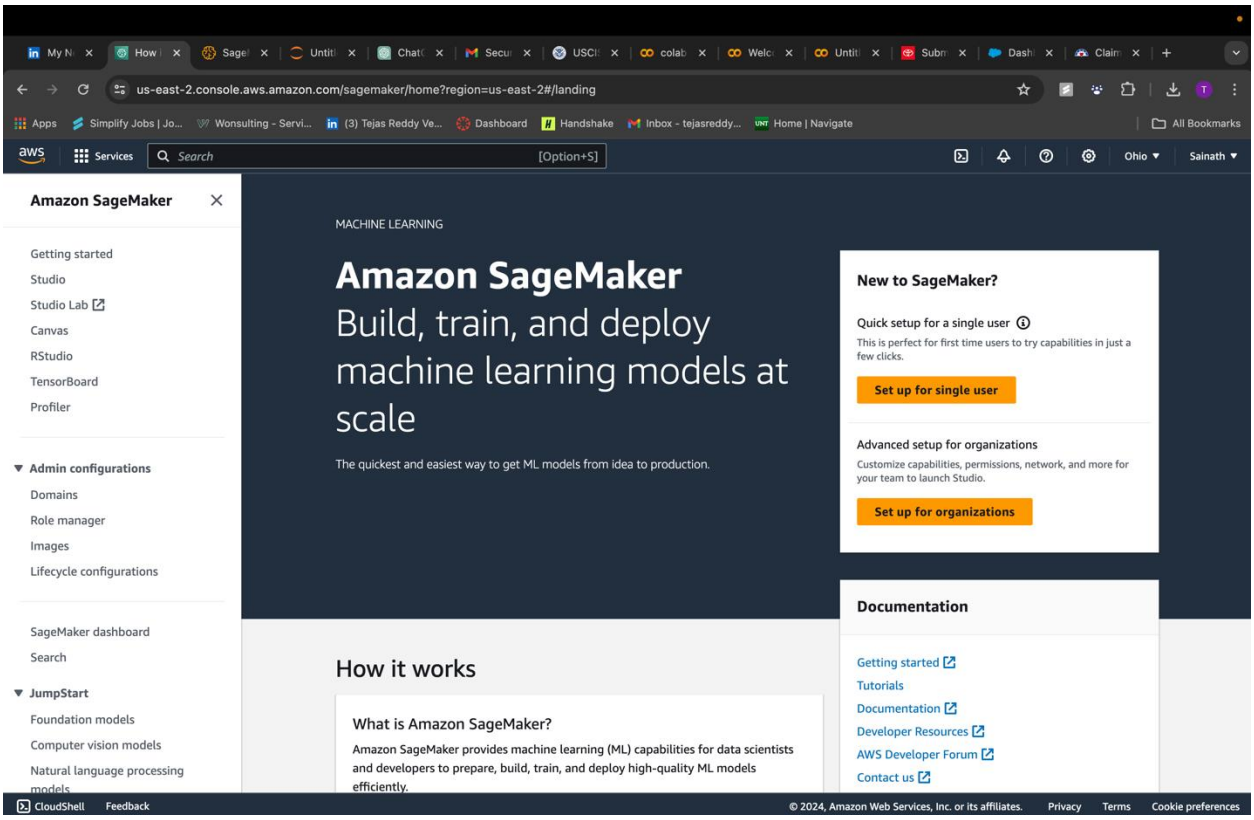
This project provides a full featured cloud computing environment by using a variety of Amazon Web Services (AWS). **PySpark** is used to run complex algorithms for machine learning in an effective manner because to the scalable computing facilities provided by Amazon EC2. Amazon S3, which provides reliable and safe data storage solutions, handles data management and storage. With its capabilities for simple scaling and real-time analytics, **Amazon SageMaker** is a vital resource for installing, tracking, and controlling machine learning models. The project's ability to handle seismic data as it comes in is further improved by the optional use of AWS Lambda, which enables the execution of code in response to real-time data triggers.

Project Implementation:

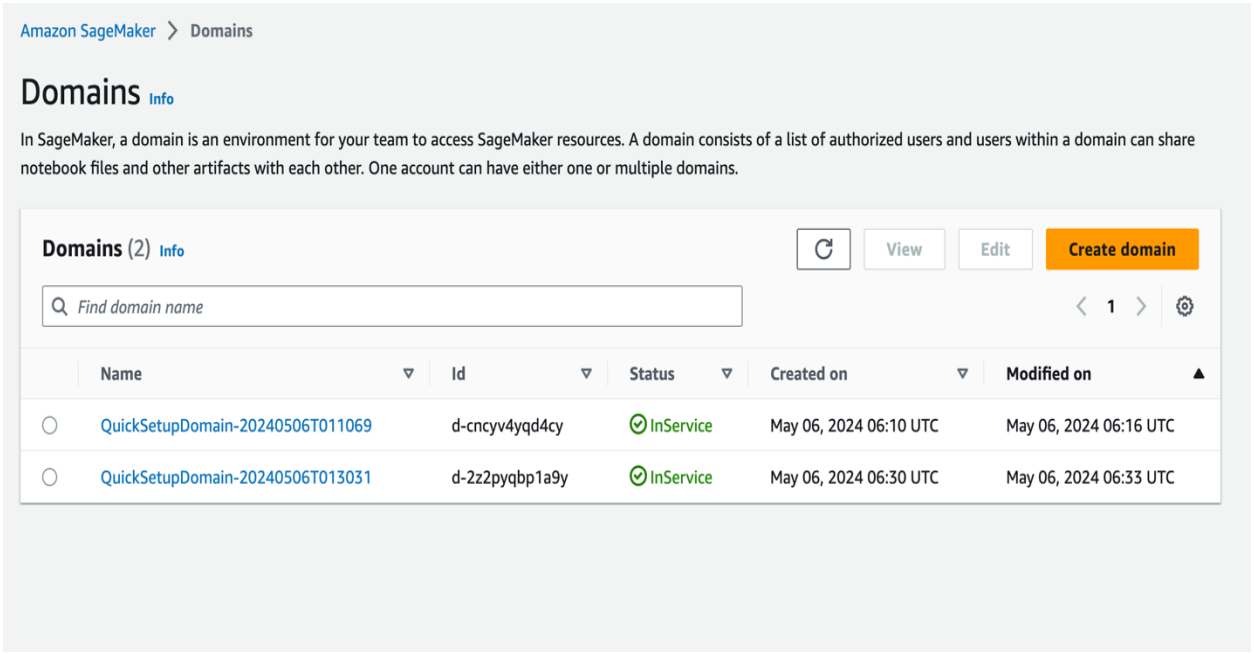
Creating an S3 bucket



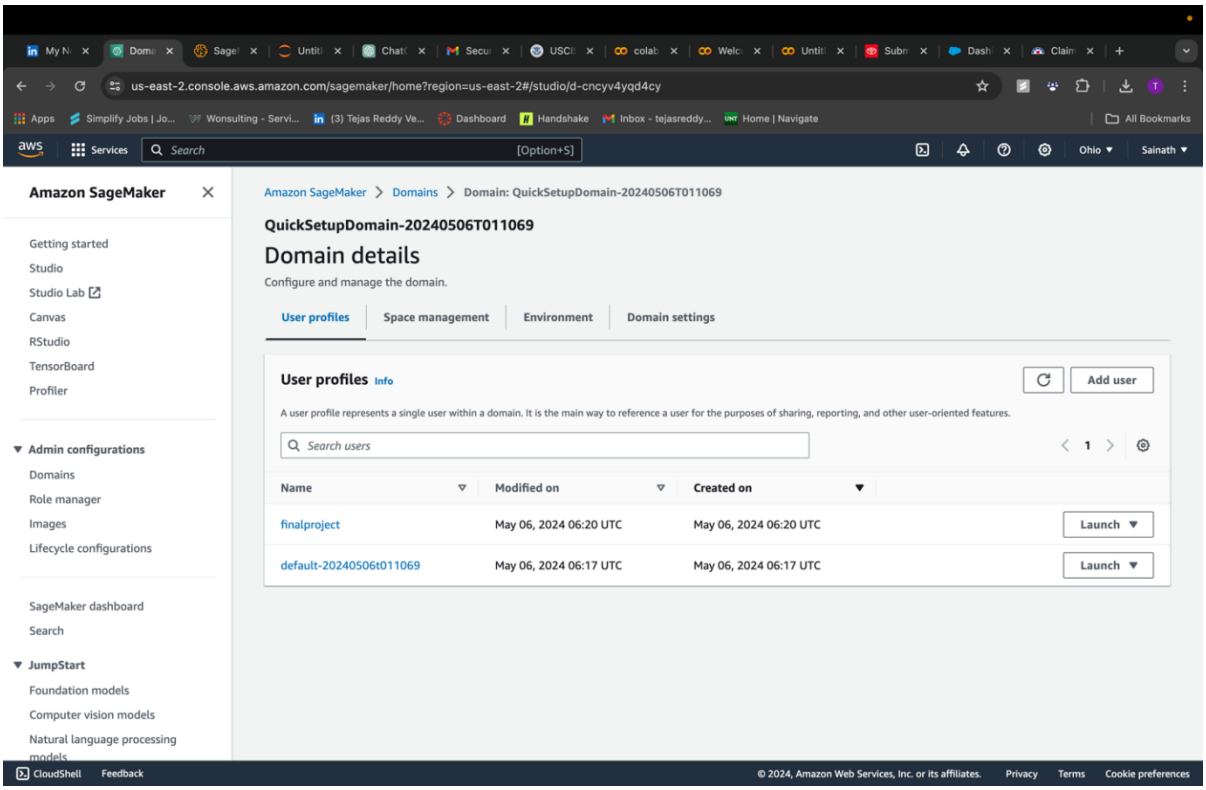
using amazon sagemaker to create an instance to deploy ML models



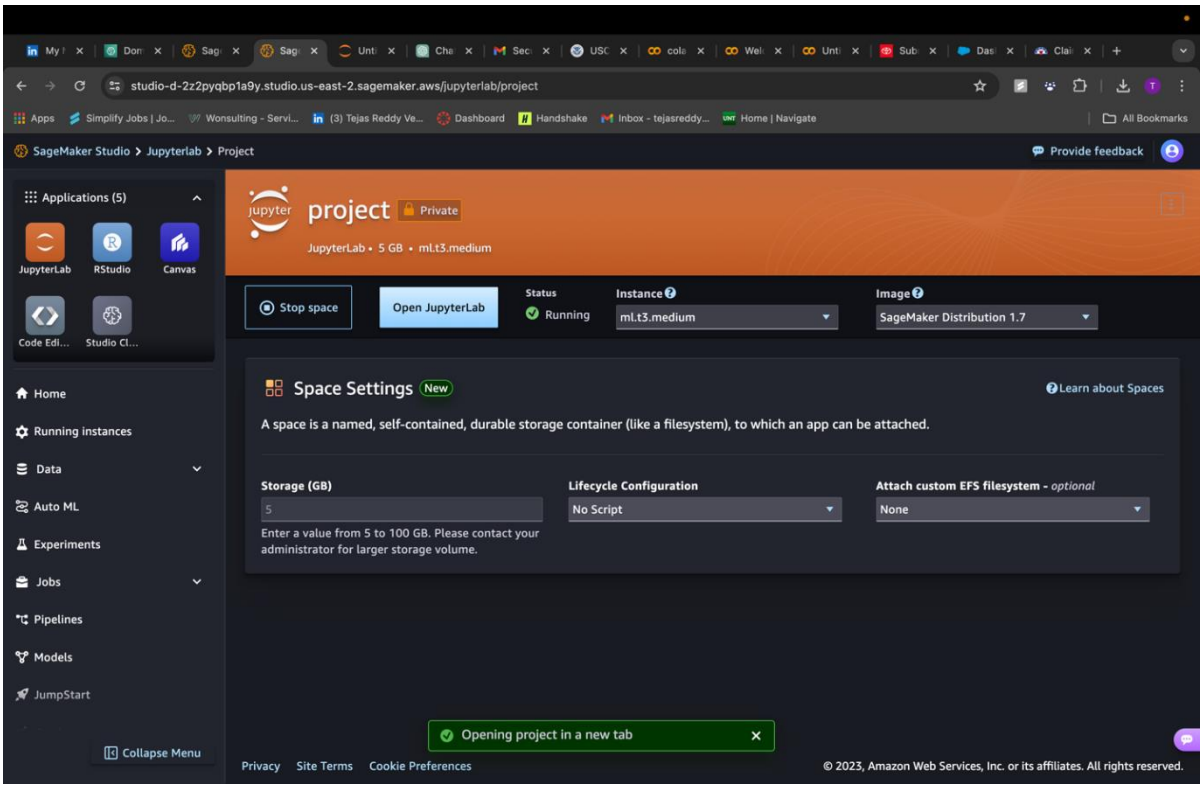
setting up the domain to work in jupyter hub



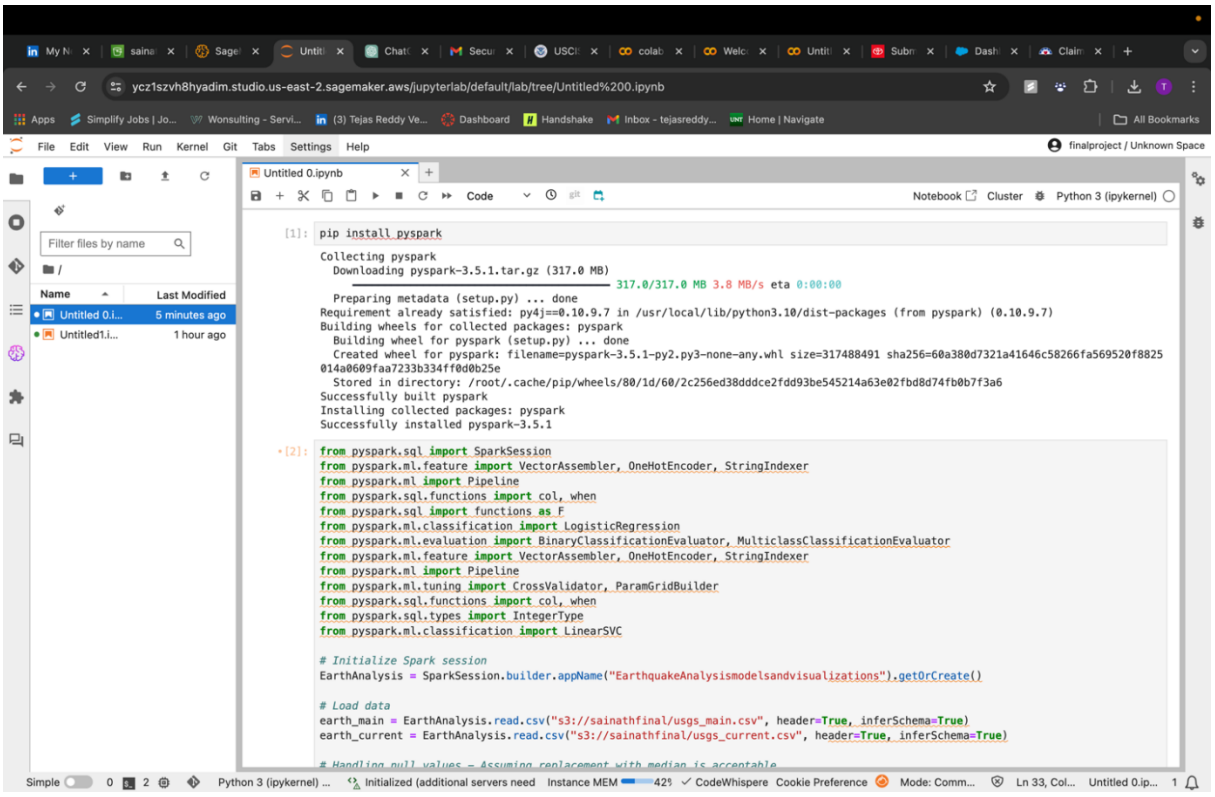
launching the studio



open jupyter lab



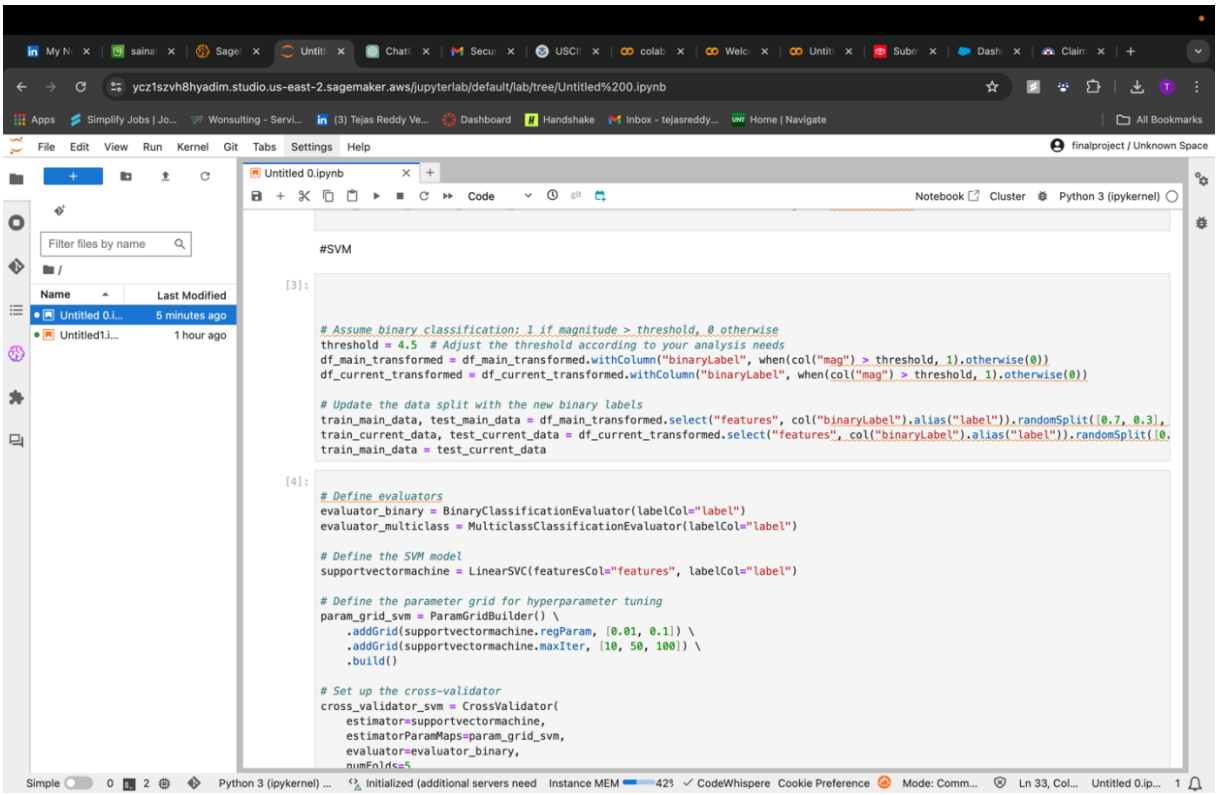
Importing and installing required packages



```
pip install pyspark

Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.0/317.0 MB 3.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488491 sha256=60a380d7321a41646c58266fa569520f8825
014a0609faa7233b334ff0d0b25e
  Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38dddce2fdd93be545214a63e02fbd8d74fb0b7f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1
```

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler, OneHotEncoder, StringIndexer
from pyspark.ml import Pipeline
from pyspark.sql.functions import col, when
from pyspark.sql import functions as F
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificationEvaluator
from pyspark.ml.feature import VectorAssembler, OneHotEncoder, StringIndexer
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.sql.functions import col, when
from pyspark.sql.types import IntegerType
from pyspark.ml.classification import LinearSVC

# Initialize Spark session
EarthAnalysis = SparkSession.builder.appName("EarthquakeAnalysismodelsandvisualizations").getOrCreate()

# Load data
earth_main = EarthAnalysis.read.csv("s3://sainathfinal/usgs_main.csv", header=True, inferSchema=True)
earth_current = EarthAnalysis.read.csv("s3://sainathfinal/usgs_current.csv", header=True, inferSchema=True)

# Handling null values - Assuming replacement with median is acceptable
```

retrieving data from

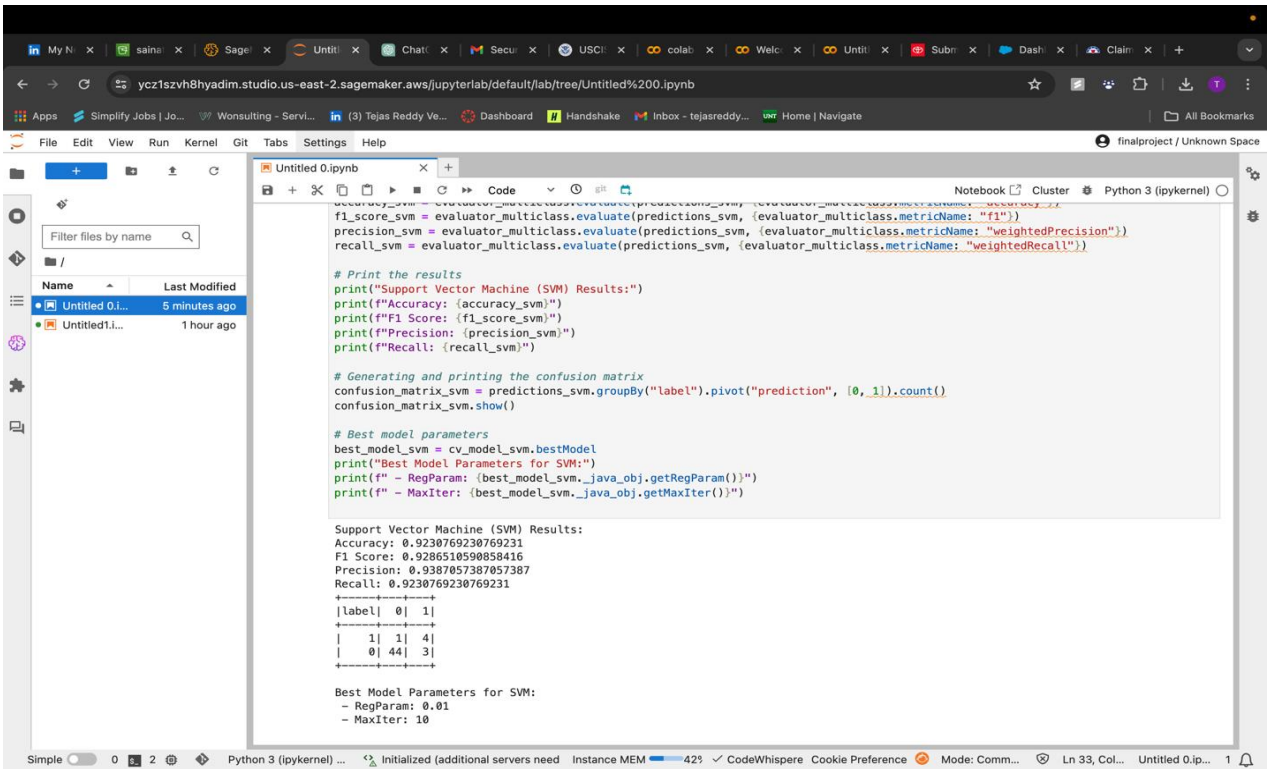


```
#SVM
```

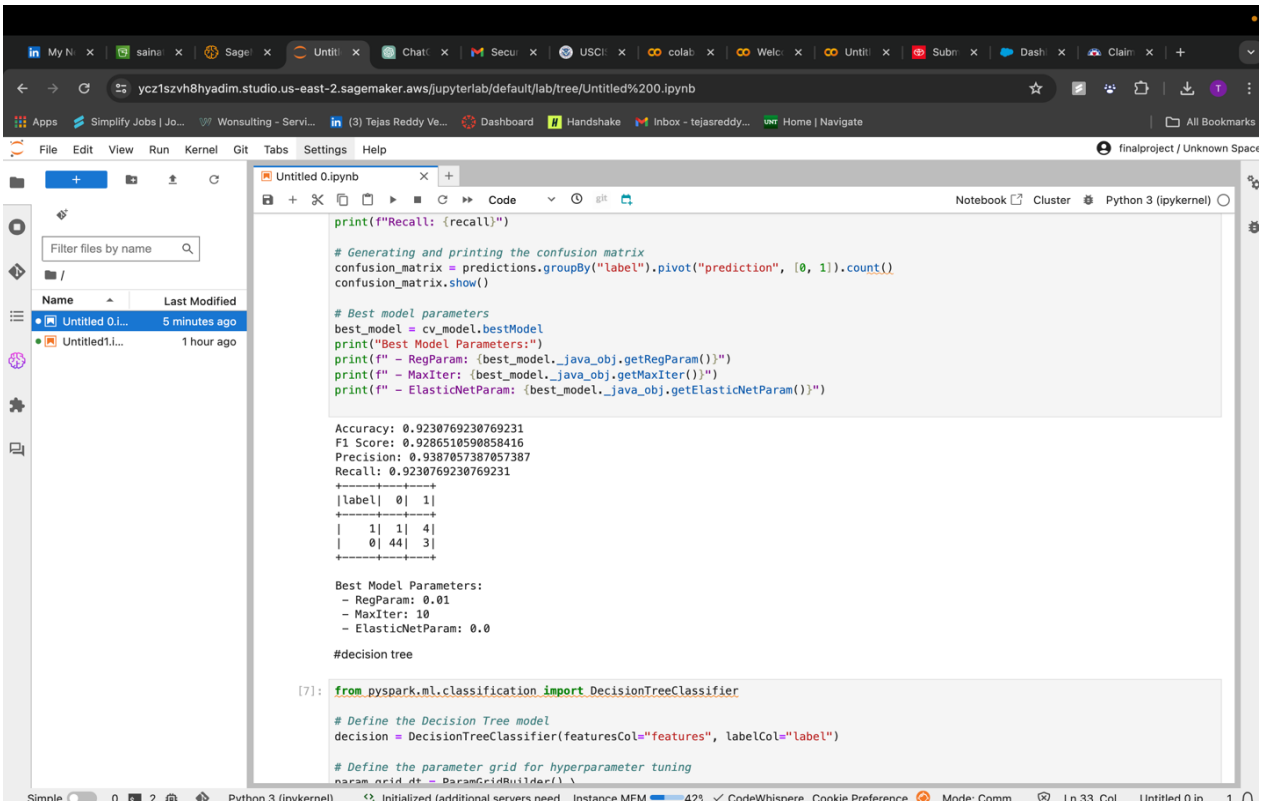
```
# Assume binary classification: 1 if magnitude > threshold, 0 otherwise
threshold = 4.5  # Adjust the threshold according to your analysis needs
df_main_transformed = df_main_transformed.withColumn("binaryLabel", when(col("mag") > threshold, 1).otherwise(0))
df_current_transformed = df_current_transformed.withColumn("binaryLabel", when(col("mag") > threshold, 1).otherwise(0))

# Update the data split with the new binary labels
train_main_data, test_main_data = df_main_transformed.select("features", col("binaryLabel").alias("label")).randomSplit([0.7, 0.3],
train_current_data, test_current_data = df_current_transformed.select("features", col("binaryLabel").alias("label")).randomSplit([0.
train_main_data = test_current_data
```

```
# Define evaluators
evaluator_binary = BinaryClassificationEvaluator(labelCol="label")
evaluator_multiclass = MulticlassClassificationEvaluator(labelCol="label")

# Define the SVM model
supportvectormachine = LinearSVC(featuresCol="features", labelCol="label")

# Define the parameter grid for hyperparameter tuning
param_grid_svm = ParamGridBuilder() \
    .addGrid(supportvectormachine.regParam, [0.01, 0.1]) \
    .addGrid(supportvectormachine.maxIter, [10, 50, 100]) \
    .build()

# Set up the cross-validator
cross_validator_svm = CrossValidator(
    estimator=supportvectormachine,
    estimatorParamMaps=param_grid_svm,
    evaluator=evaluator_binary,
    numFolds=5
```

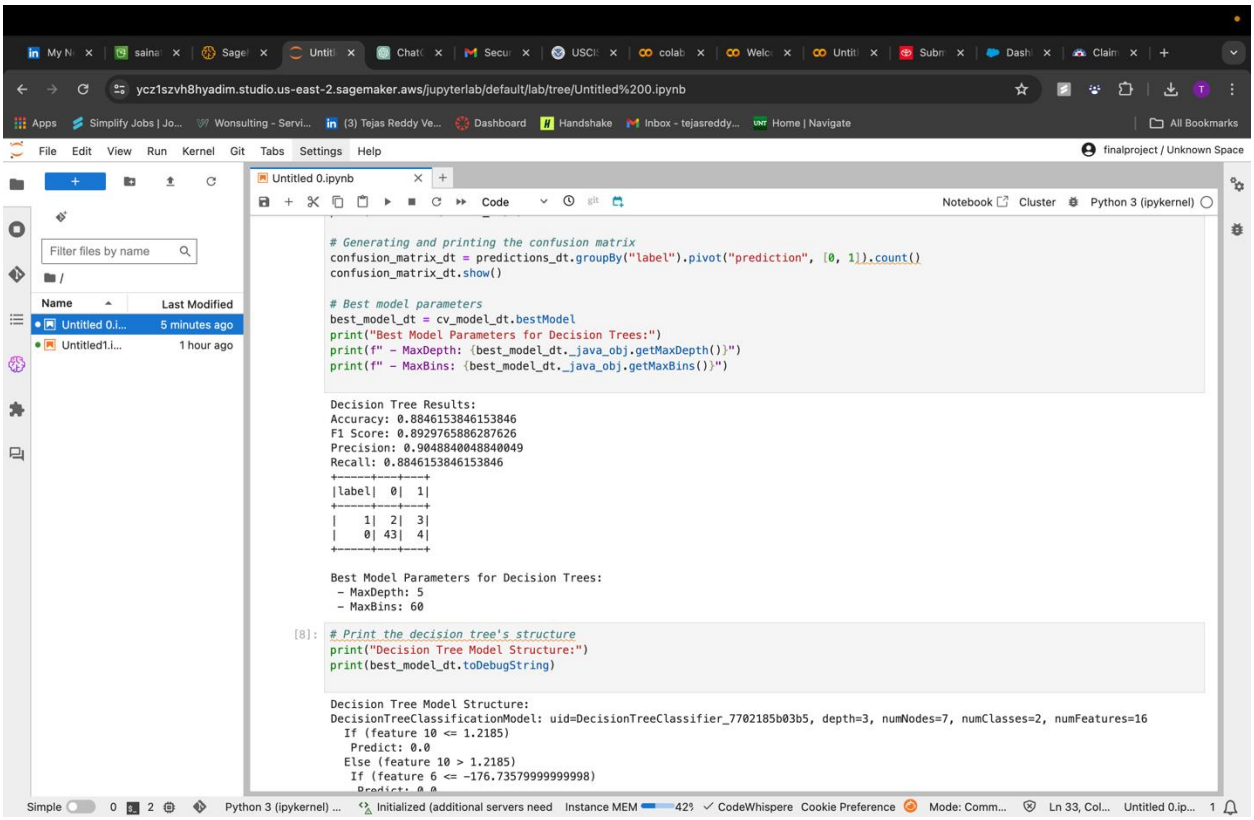
s3 bucket and creating a SVM (support vector machine) model



Results: SVM results



Desion tree results



decision tree structure

```
Decision Tree Model Structure:
DecisionTreeClassificationModel: uid=DecisionTreeClassifier_7702185b03b5, depth=3, numNodes=7, numClasses=2, numFeatures=16
  If (feature 10 <= 1.2185)
   Predict: 0.0
  Else (feature 10 > 1.2185)
   If (feature 6 <= -176.73579999999998)
    Predict: 0.0
   Else (feature 6 > -176.73579999999998)
    If (feature 8 <= 17.5)
     Predict: 0.0
    Else (feature 8 > 17.5)
     Predict: 1.0
```

random forest results

```
Random Forest Results:
Accuracy: 0.8846153846153846
F1 Score: 0.8929765886287626
Precision: 0.9048840048840049
Recall: 0.8846153846153846
+-----+---+---+
|label|  0|  1|
+-----+---+---+
|    1|  2|  3|
|    0| 43|  4|
+-----+---+---+

Best Model Parameters for Random Forest:
 - MaxDepth: 5
 - NumTrees: 20
```

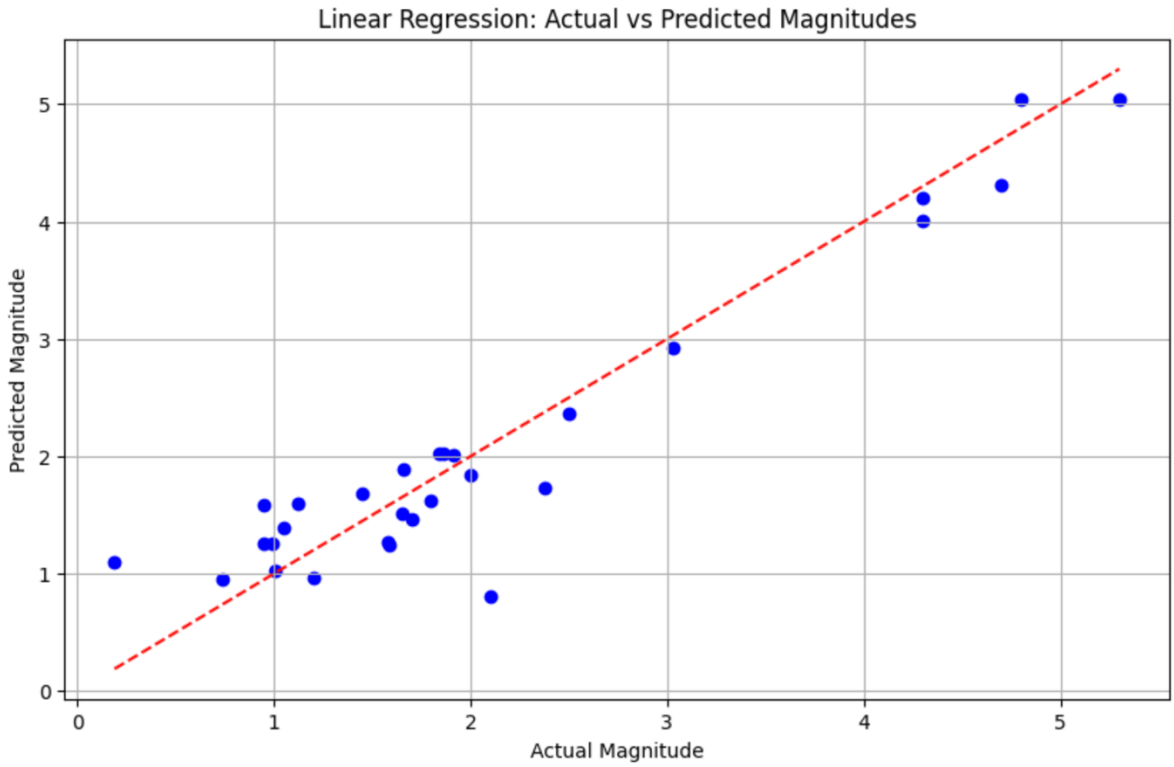
printing the importance of the features

```
# Print feature importances
importances = best_model_rf.featureImportances
print("Feature Importances:", importances)
# List of input columns to the VectorAssembler
input_columns = assembler_ins  # This list should have been defined when you set up your VectorAssembler

# Creating a dictionary to map feature vector indices back to feature names
feature_index = {i: col for i, col in enumerate(input_columns)}

# Optionally, convert feature importances to a more readable format
# Assuming you have a list of feature names corresponding to indices in the features vector
feature_names = [feature_index[i] for i in range(len(feature_index))]  # Reuse feature_index from previous examples
feature_importance_dict = dict(zip(feature_names, importances))
print("Readable Feature Importances:", feature_importance_dict)
```

```
Feature Importances: (16,[0,2,5,6,7,8,9,10,11,12,13,14,15],[0.0060206460206460215,0.17621730073258485,0.12445300689048641,0.0638252
9262035082,0.06187677779534876,0.03896364929171424,0.06349353430548997,0.1596512358758316,0.0545426105365245,0.11474855478793751,0.
029275085232850585,0.008660130718954244,0.09827217519128048])
Readable Feature Importances: {'magTypeOHE': 0.0060206460206460215, 'typeOHE': 0.0, 'latitude': 0.17621730073258485, 'longitude':
0.0, 'depth': 0.0, 'nst': 0.12445300689048641, 'gap': 0.06382529262035082, 'dmin': 0.06187677779534876, 'rms': 0.03896364929171424,
'horizontalError': 0.06349353430548997, 'depthError': 0.1596512358758316, 'magError': 0.0545426105365245, 'magNst': 0.1147485547879
3751}
```

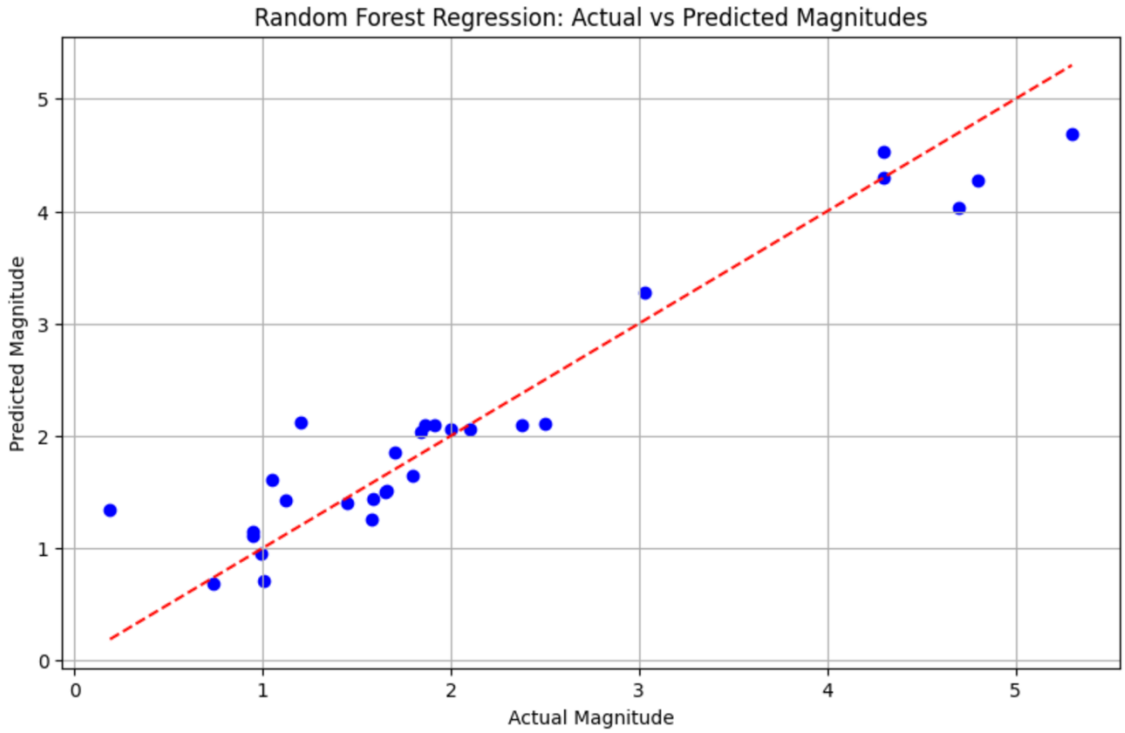
Linear Regression plots

```
+------------------+-----+--------------------+
|        prediction|label|            features|
+------------------+-----+--------------------+
|  5.04192489342752|  4.8|[0.0,0.0,1.0,0.0,...|
| 4.009377612272793|  4.3|[0.0,0.0,1.0,0.0,...|
| 4.316257388620885|  4.7|[0.0,0.0,1.0,0.0,...|
| 5.038339507460163|  5.3|[0.0,0.0,1.0,0.0,...|
|4.2042868440534935|  4.3|[0.0,0.0,1.0,0.0,...|
+------------------+-----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 0.410369
```

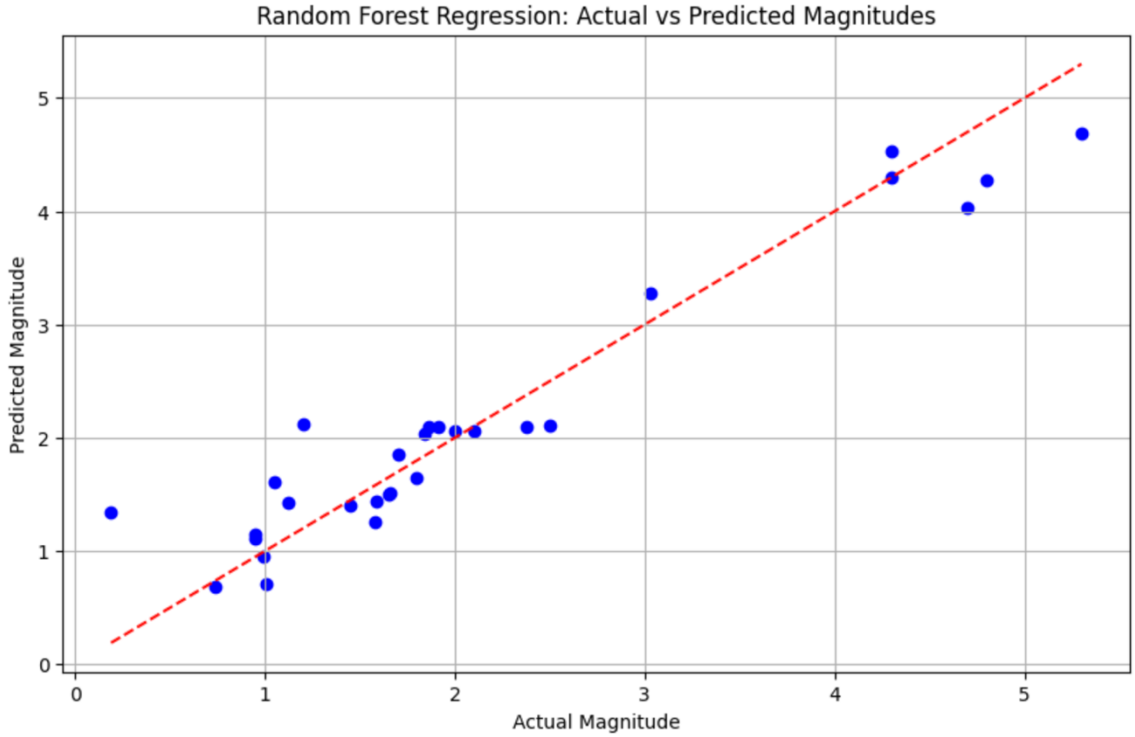


Random forest plots

```
Root Mean Squared Error (RMSE) on test data with Random Forest = 0.396129
```



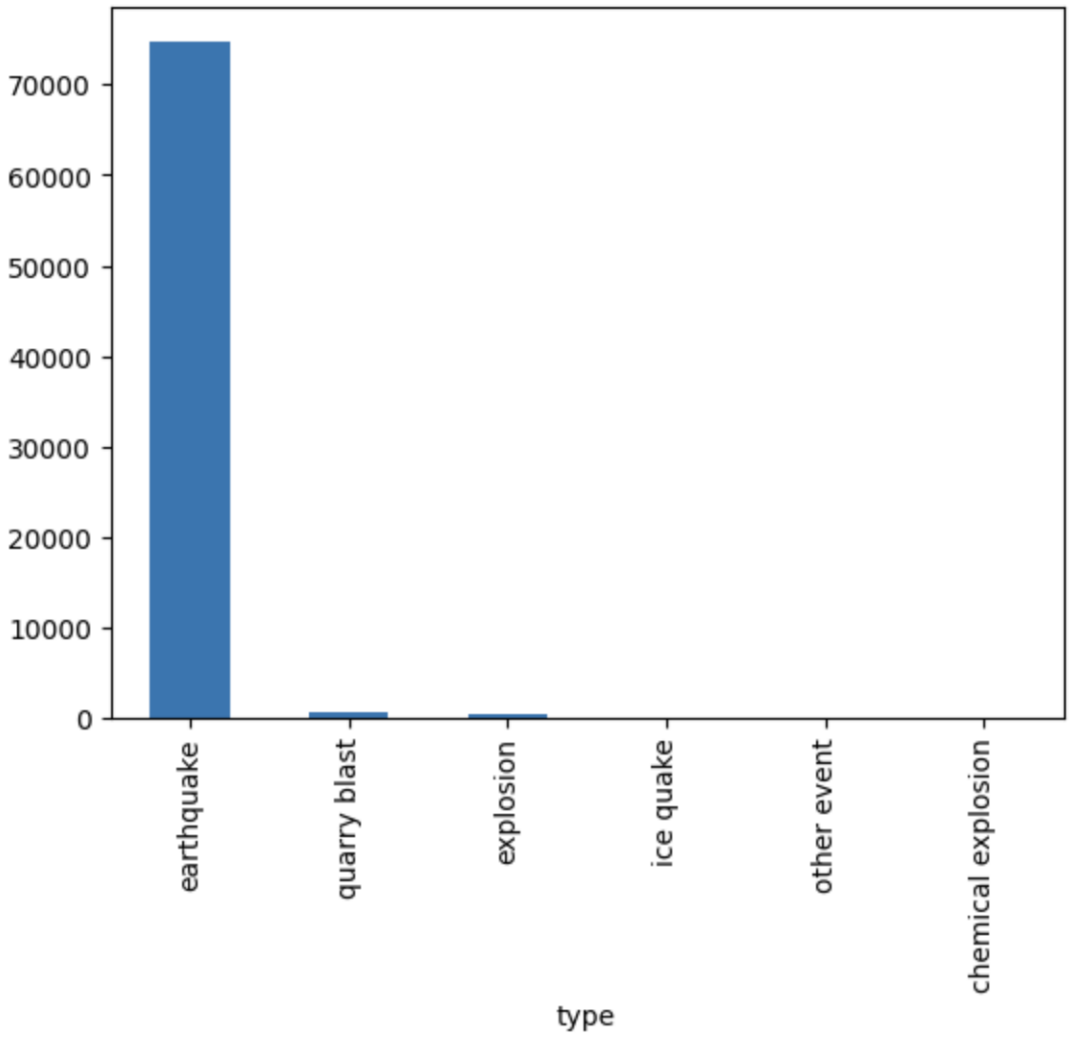
Root Mean Squared Error (RMSE) on test data with Random Forest = 0.396129



Data Visualization:

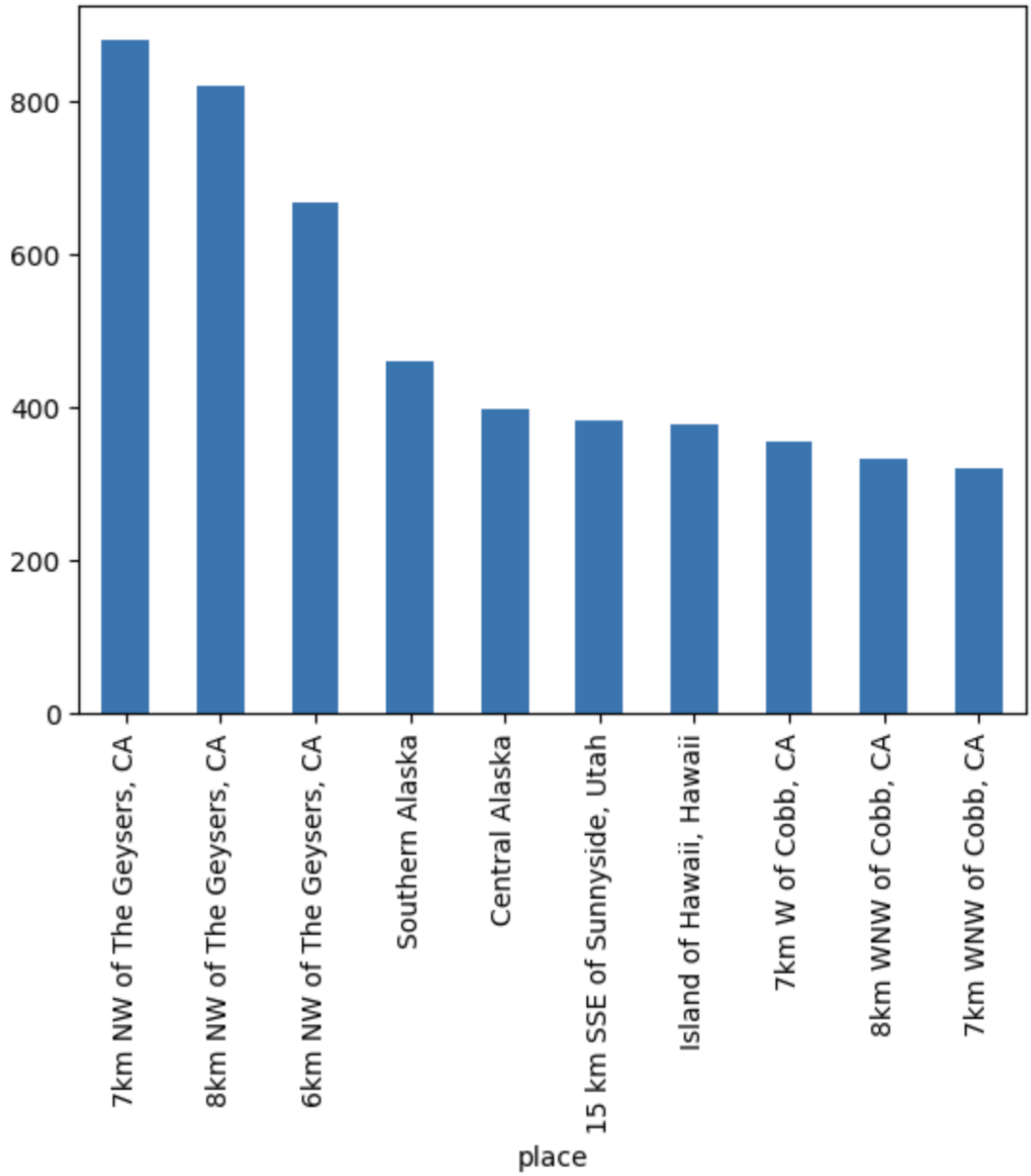
type of event

```
<Axes: xlabel='type'>
```

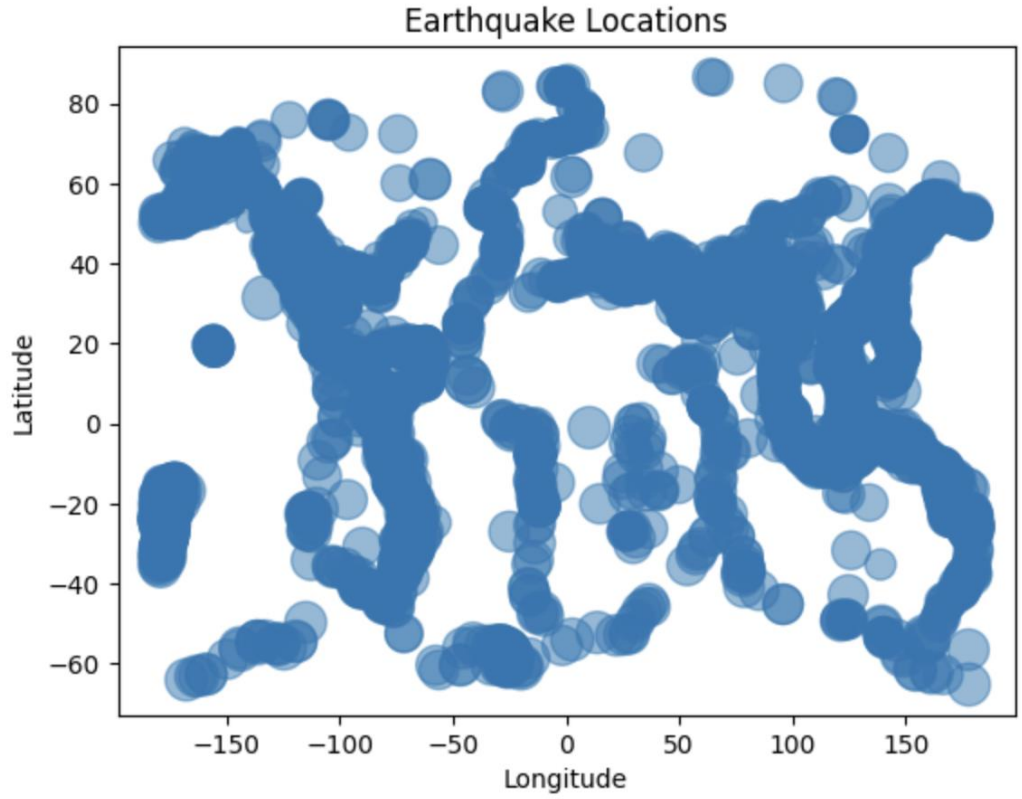


place vs event

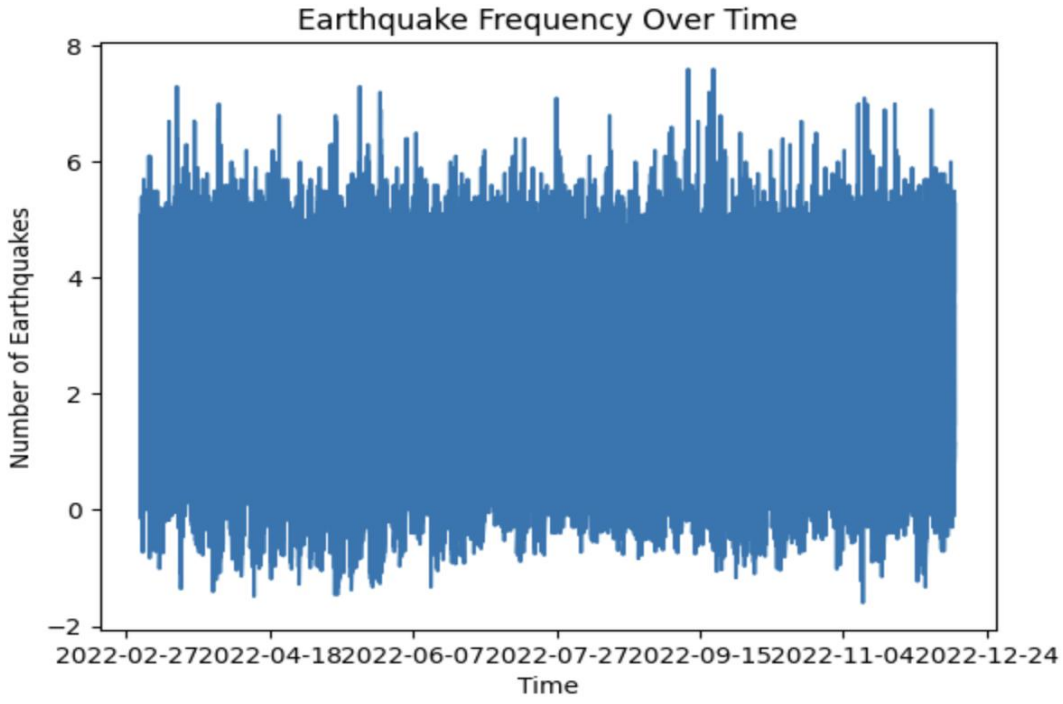
```
<Axes: xlabel='place'>
```



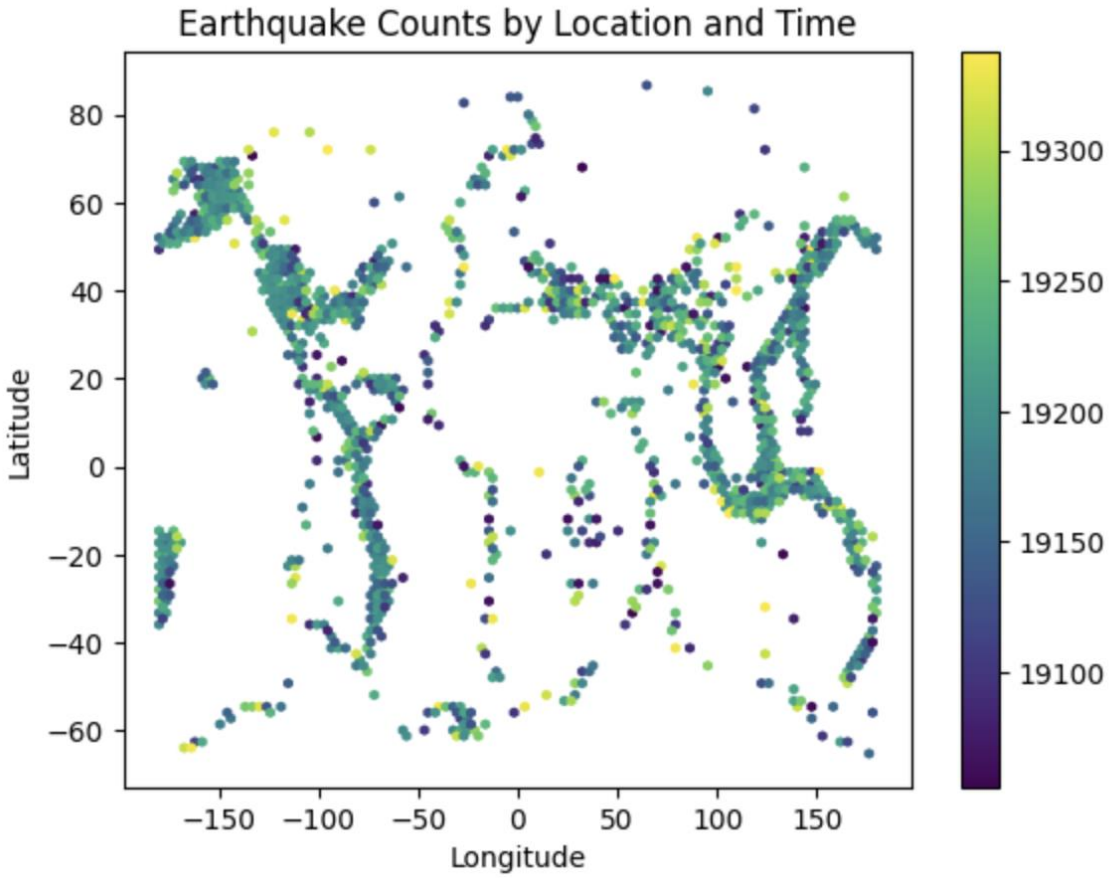
longitude vs latitude to show location worldwide



Time vs number of earthquakes



Earthquake counts by location and time



Dealing with null values

```
US_df.isnull().sum()
```

```
time                0
latitude            0
longitude           0
depth               0
mag                14
magType            14
place               0
type                0
state               0
severity            0
depth category      0
dtype: int64
```

```
null_mag = US_df[US_df['mag'].isnull()].index
US_df.drop(null_mag, inplace = True)
US_df.isnull().sum()
```

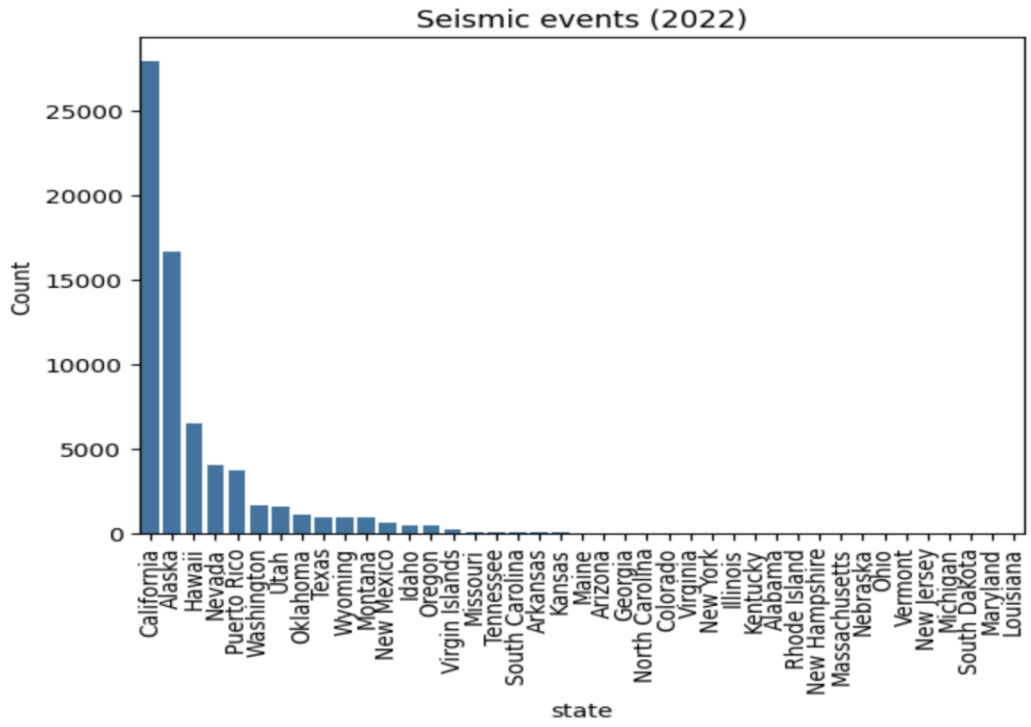
```
time               0
latitude           0
longitude          0
depth              0
mag                0
magType            0
place              0
type               0
state              0
severity           0
depth category     0
dtype: int64
```

Original and changed values

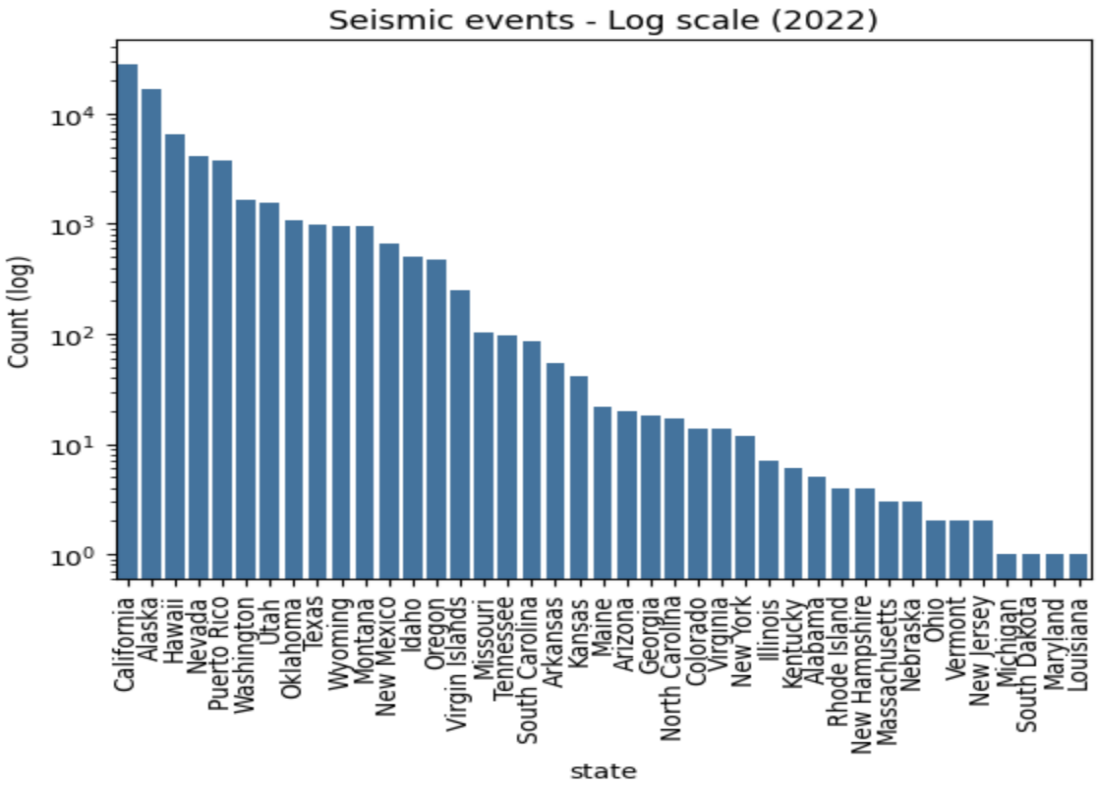
```
print('Original value counts for magnitude types:')
print(US_df['magType'].value_counts())
US_df['magType'].replace(['Ml', 'mww', 'Mb'],['ml', 'mw', 'mb'], inplace = True)
print('Corrected value counts for magnitude types:')
print(US_df['magType'].value_counts())
```

```
Original value counts for magnitude types:
magType
ml       43344
md       24598
mb         244
mb_lg      125
mh          52
mw          32
mlv         25
mww         17
mwr         16
Ml          12
mlr         11
Mi           3
Mb           2
Name: count, dtype: int64
Corrected value counts for magnitude types:
magType
ml       43356
md       24598
mb         246
mb_lg      125
mh          52
mw          49
mlv         25
mwr         16
mlr         11
Mi           3
Name: count, dtype: int64
```

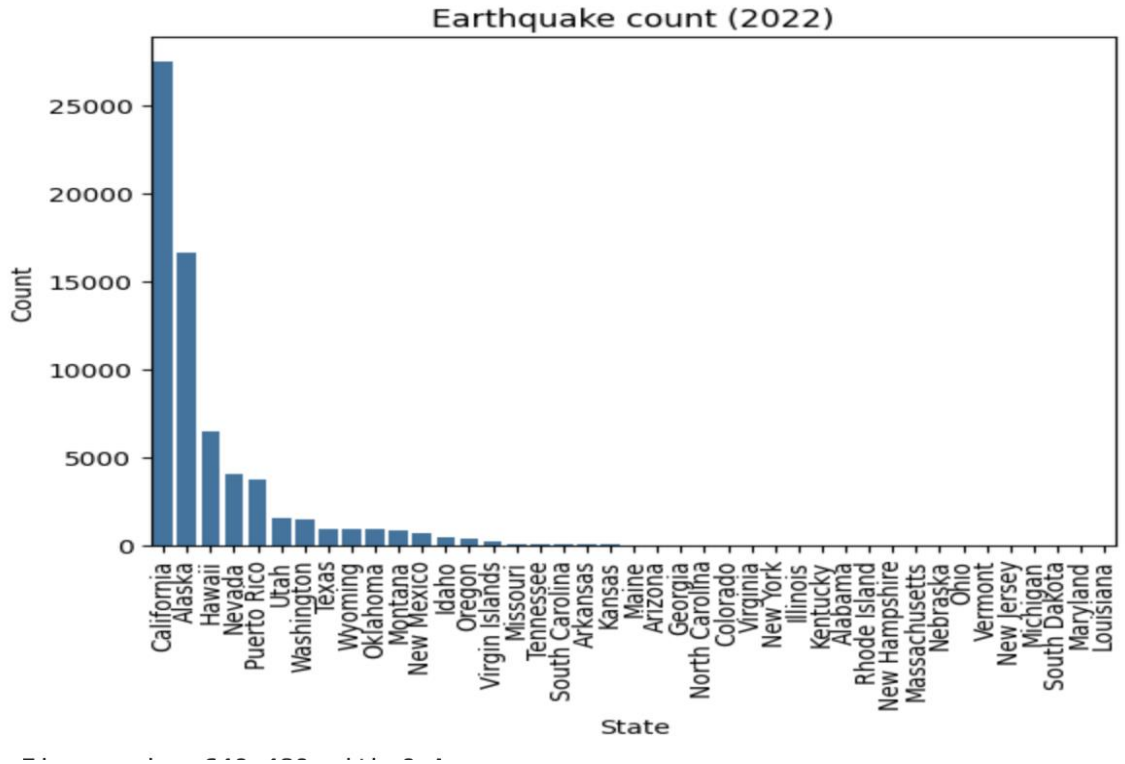
State vs count



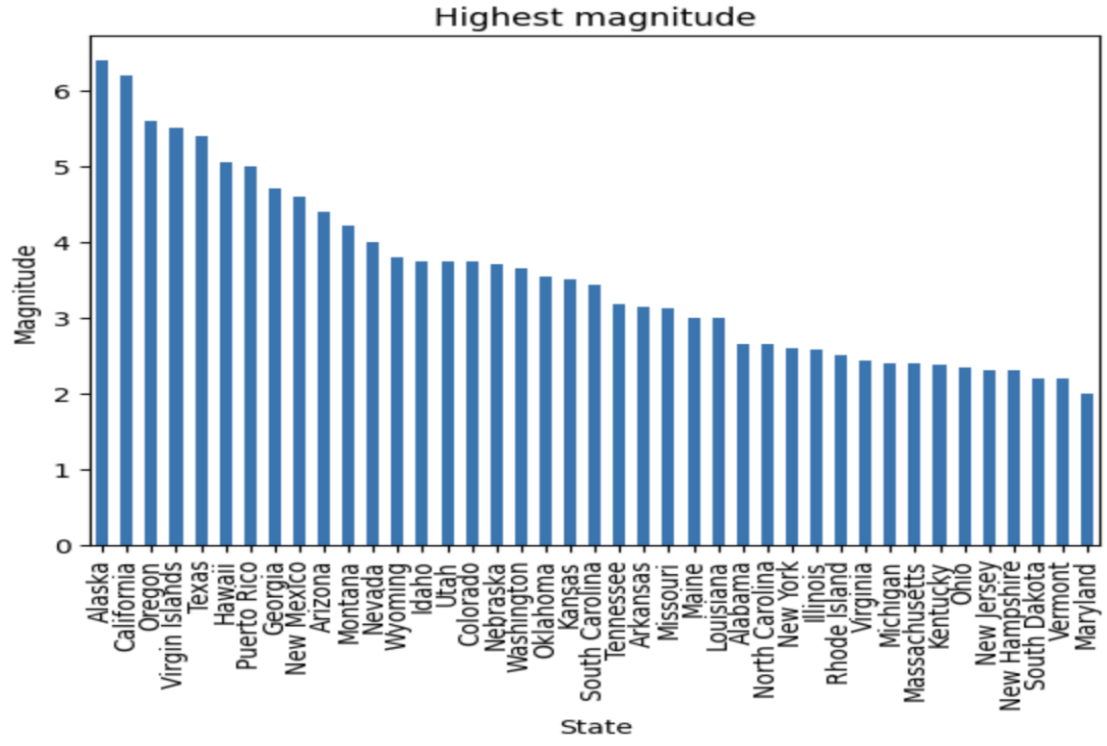
States count on a log scale



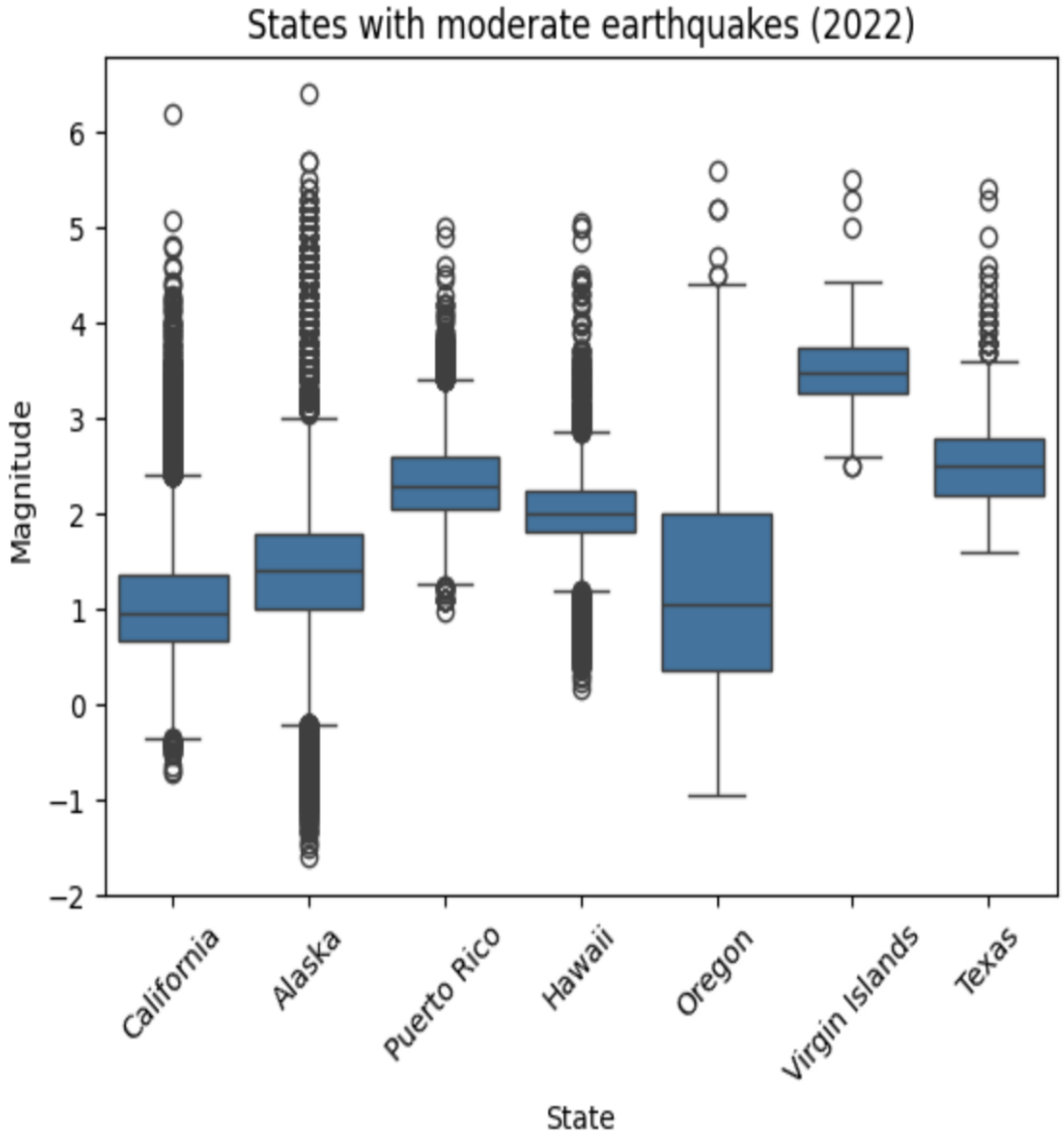
Earthquake count in year 2022



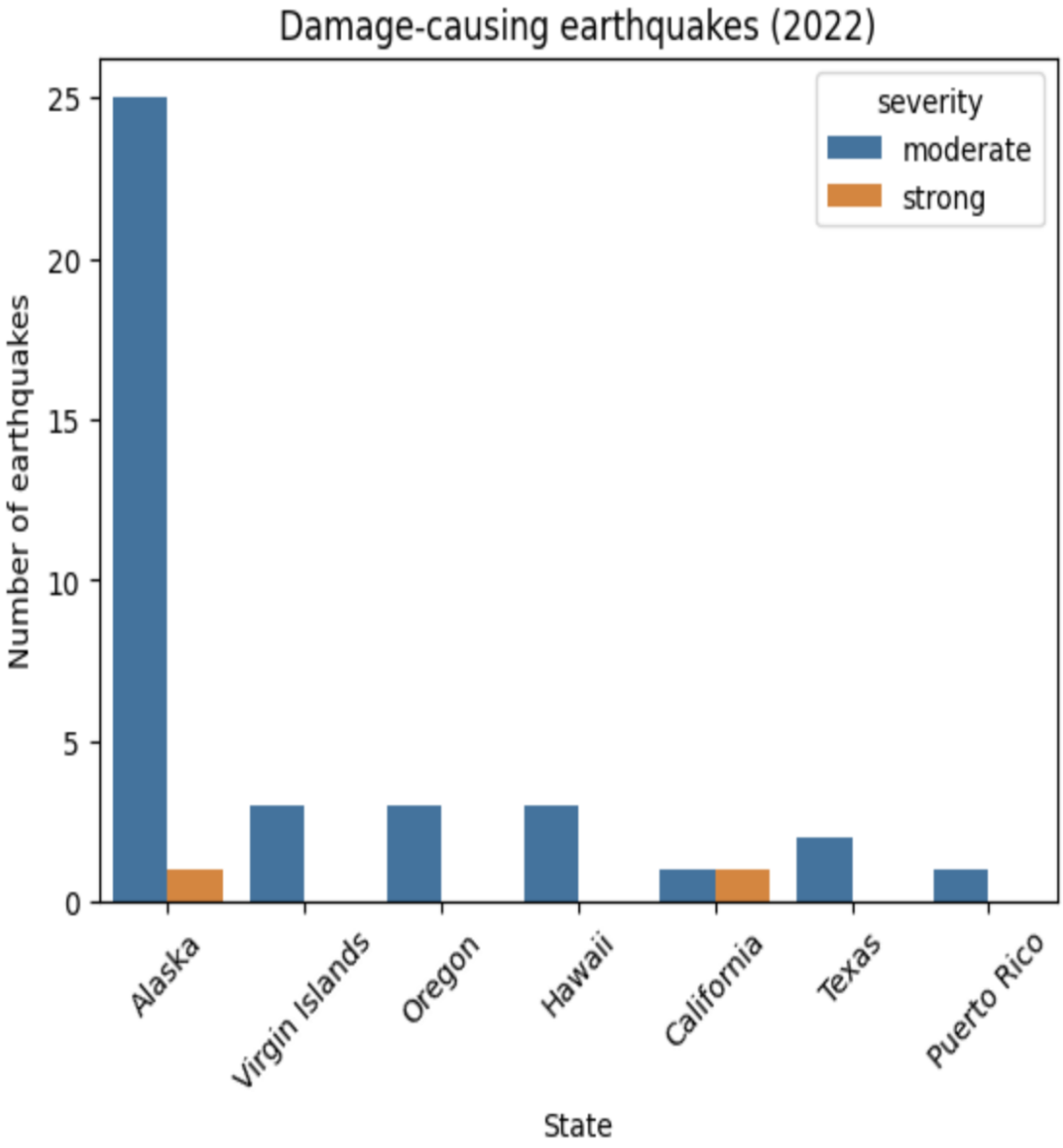
States with highest magnitude of earthquakes



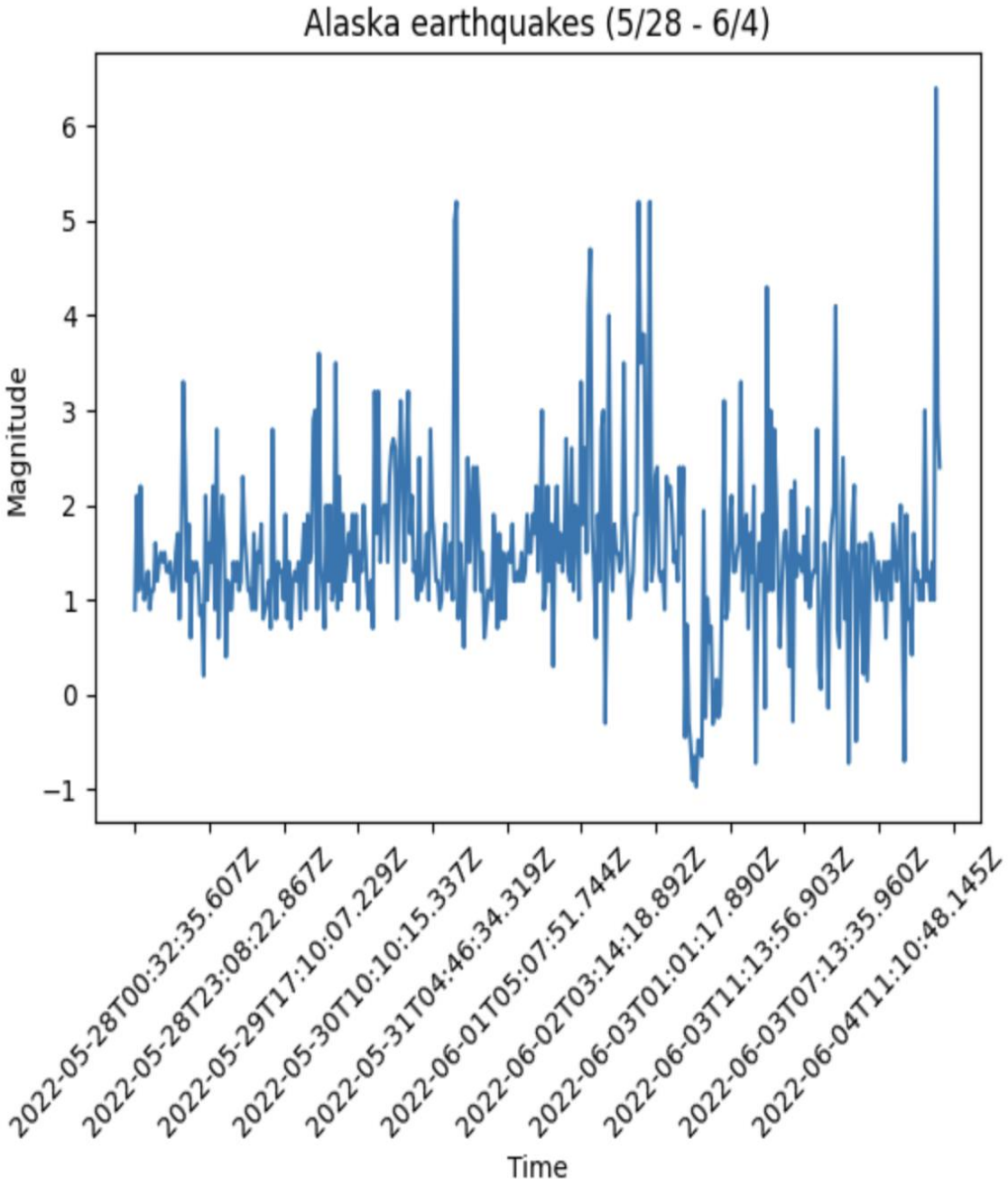
States with less number of earthquakes



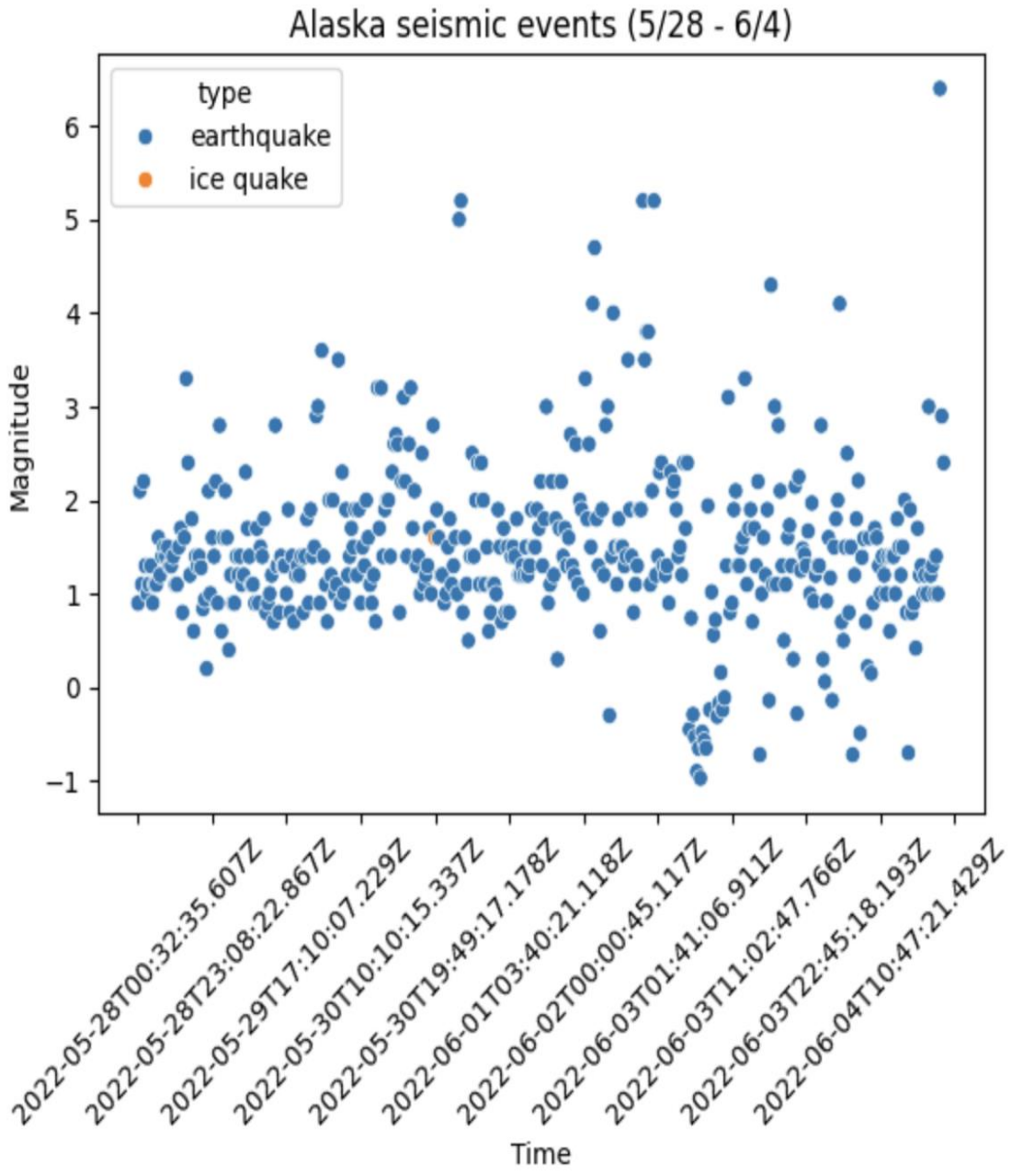
Severity of earthquakes compared

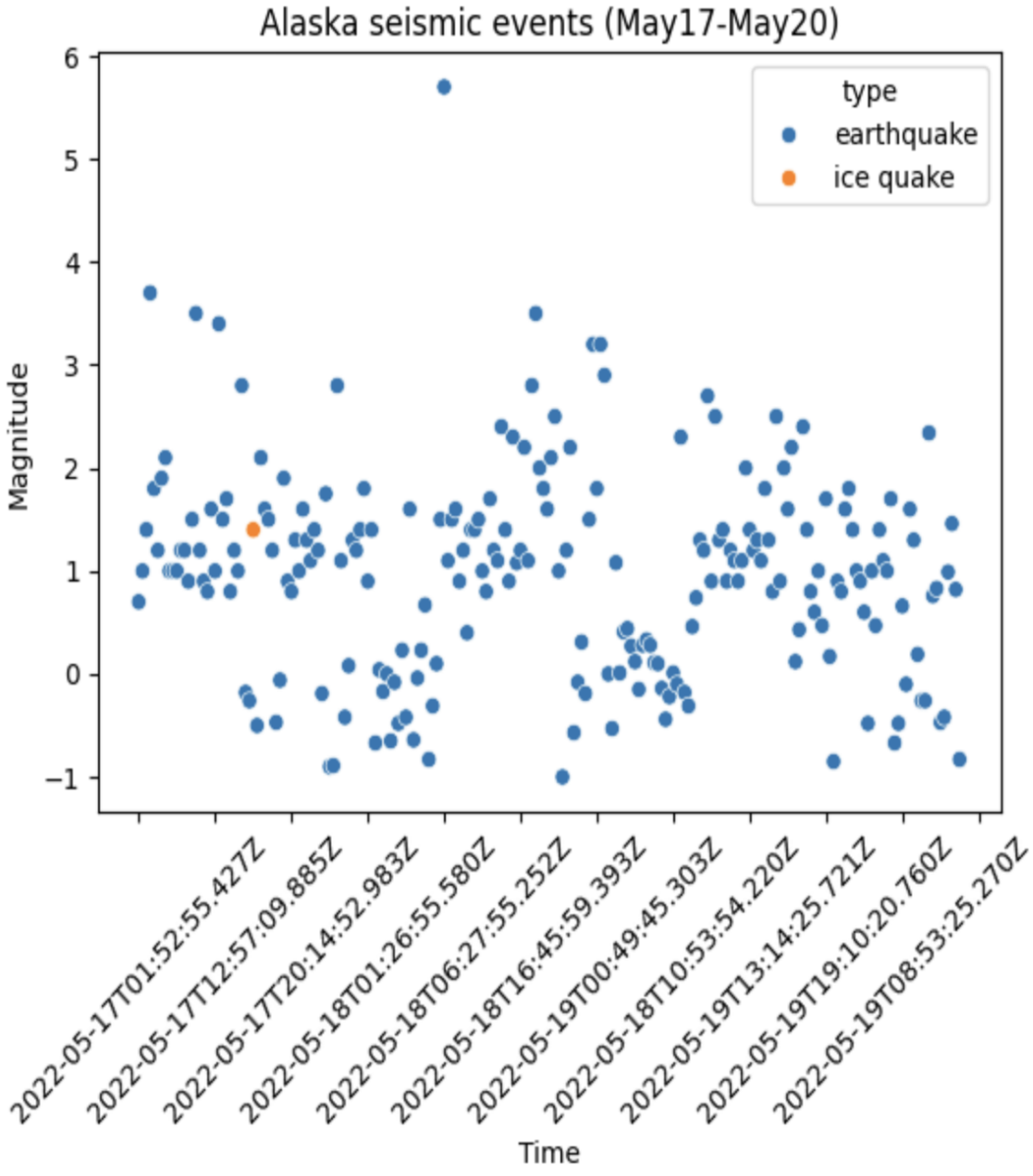


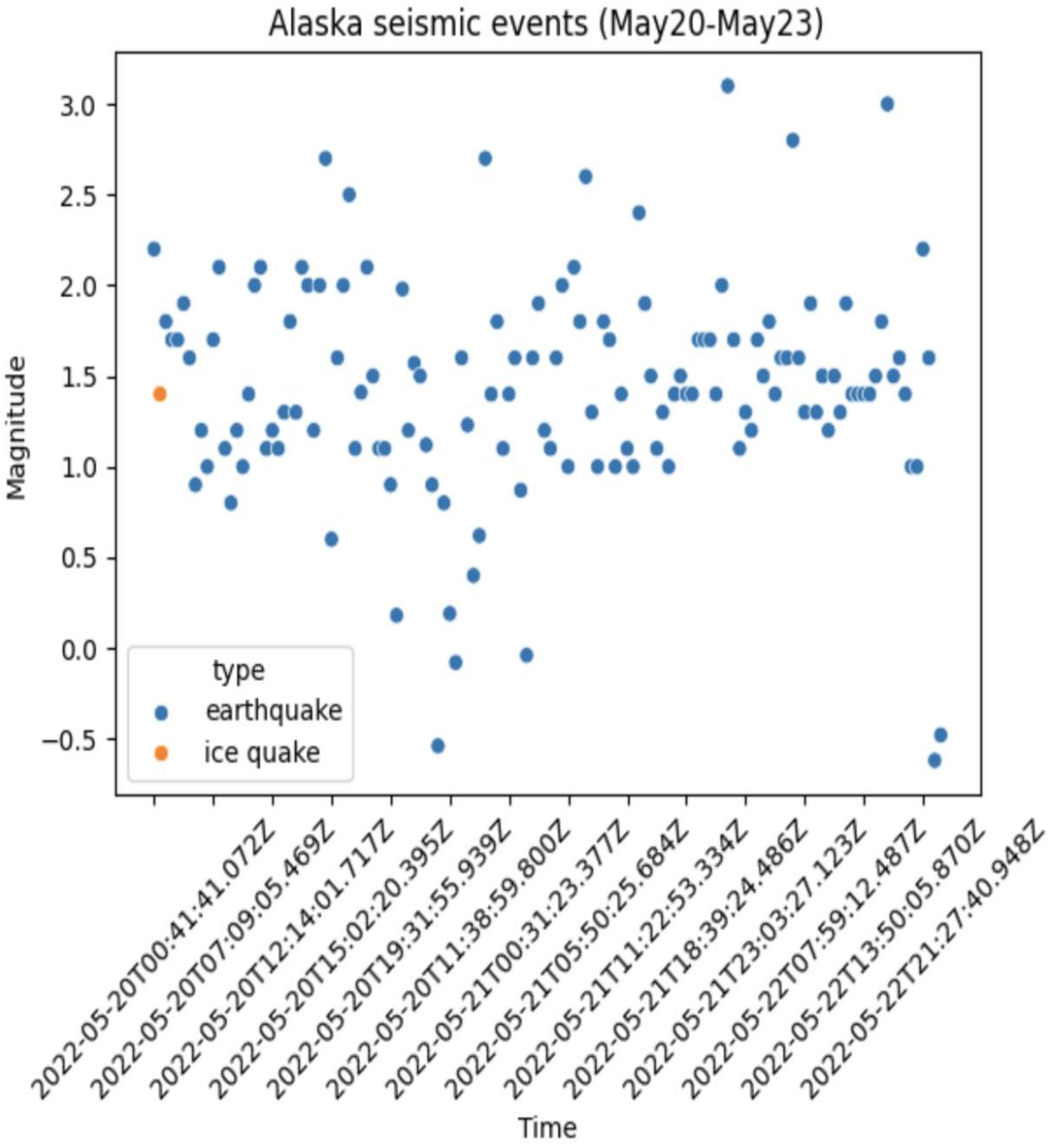
Earthquakes in Alaska earthquakes

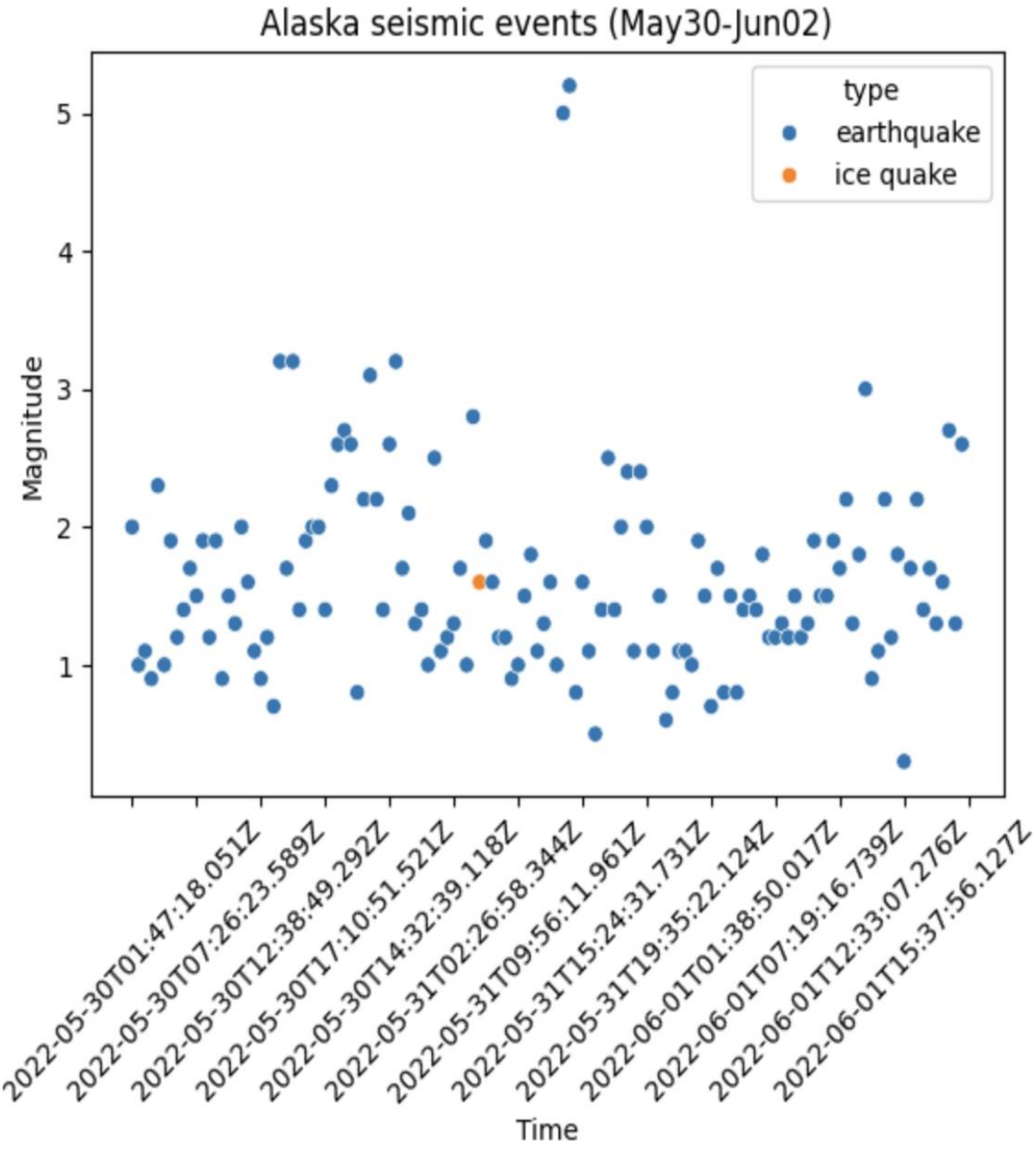


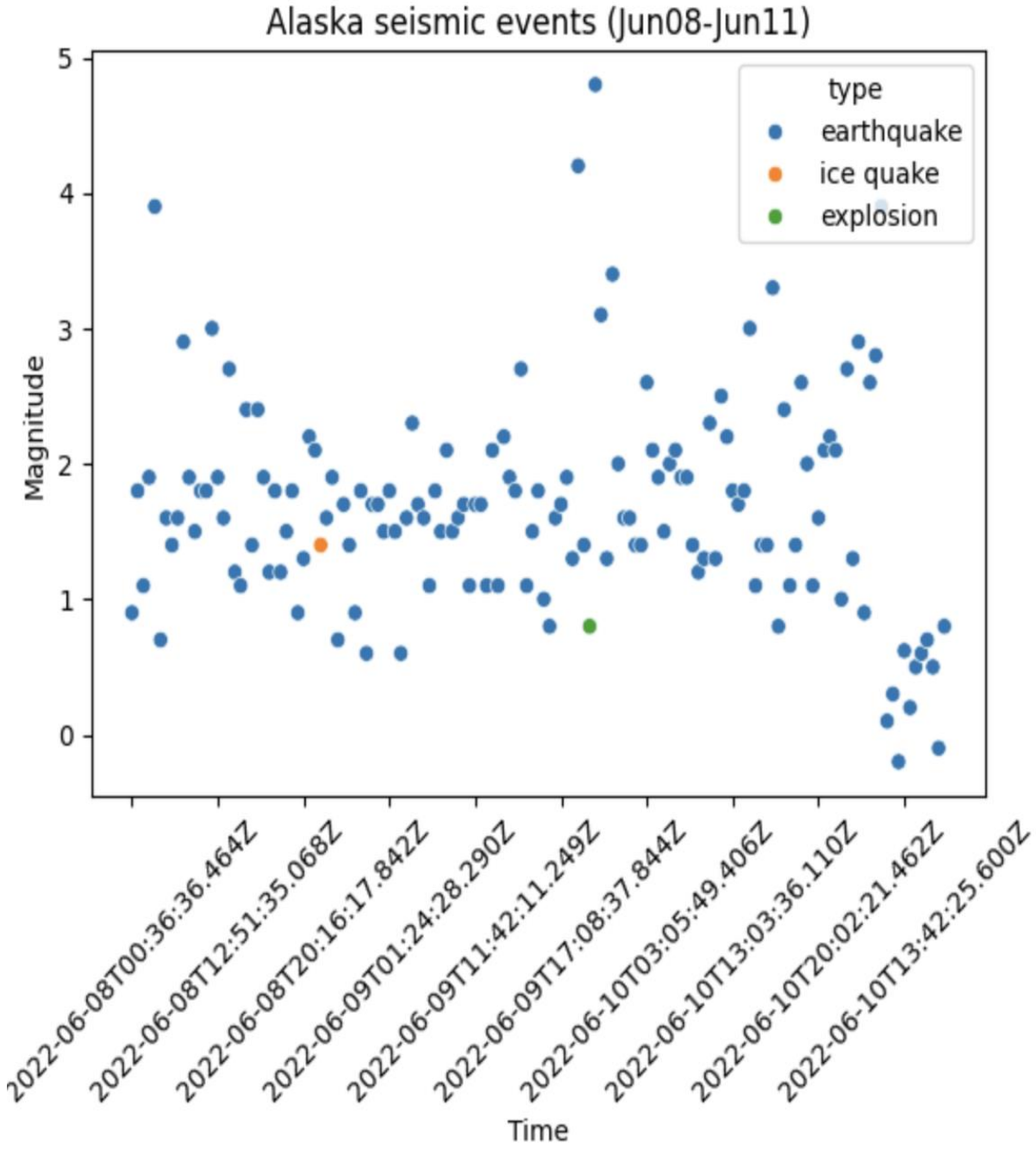
Average and other metrics of California earthquake Alaska seismic events

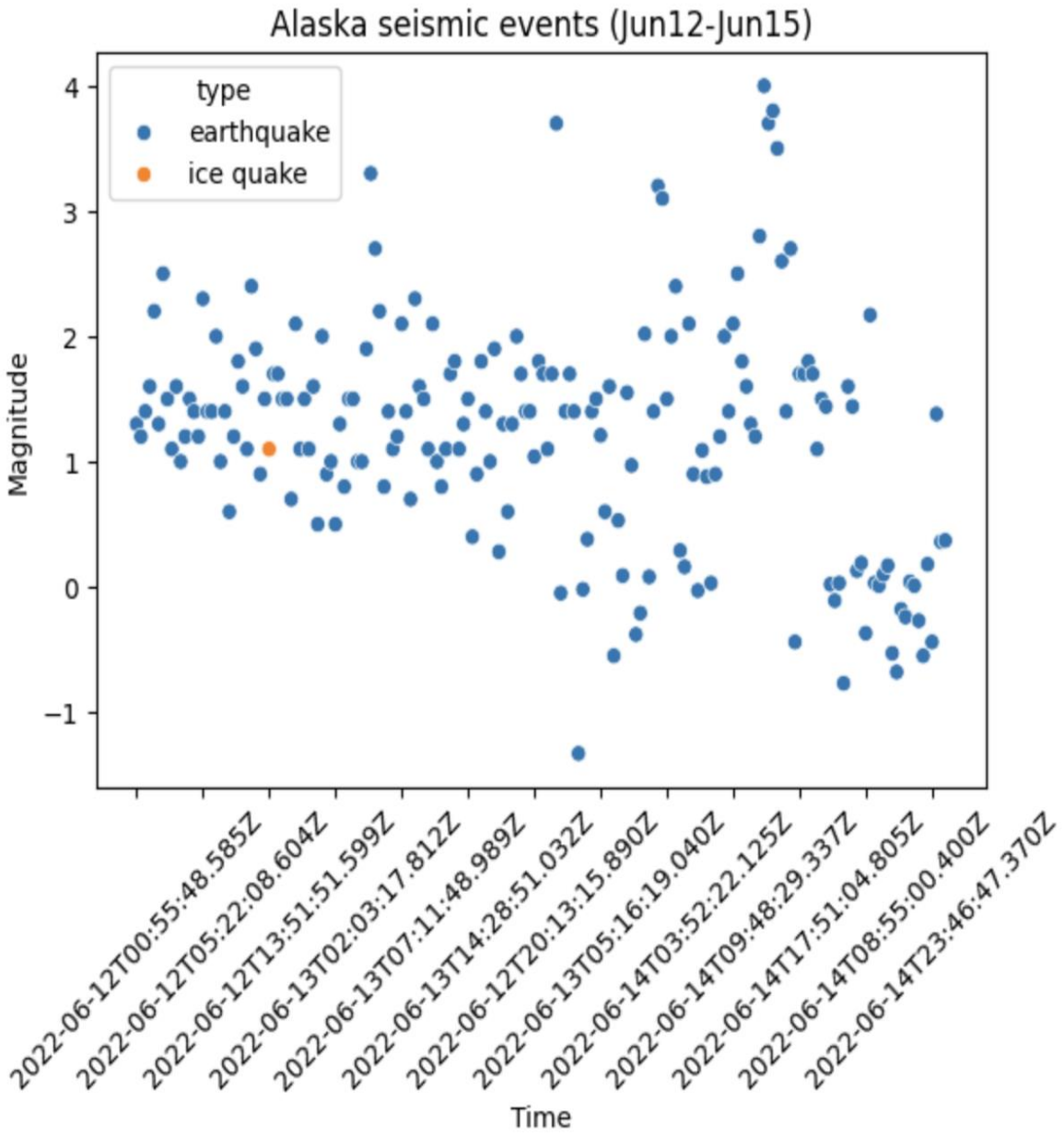


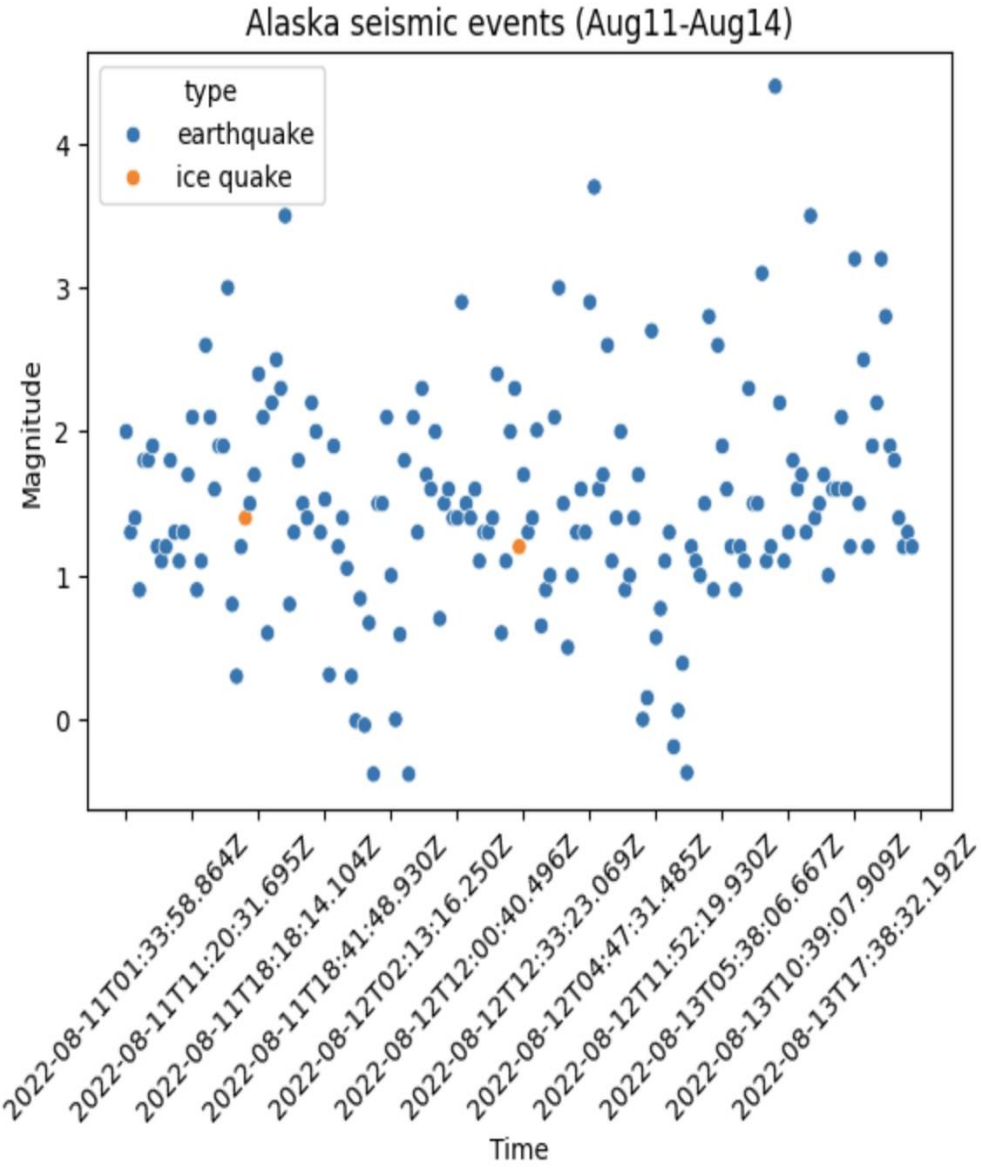

Alaska seismic events (May17-May20)
type
earthquake
ice quake
Magnitude
6
5
4
3
2
1
0
−1
2022-05-17T01:52:55.427Z
2022-05-17T12:57:09.885Z
2022-05-17T20:14:52.983Z
2022-05-18T01:26:55.580Z
2022-05-18T06:27:55.252Z
2022-05-18T16:45:59.393Z
2022-05-19T00:49:45.303Z
2022-05-18T10:53:54.220Z
2022-05-19T13:14:25.721Z
2022-05-19T19:10:20.760Z
2022-05-19T08:53:25.270Z
Time

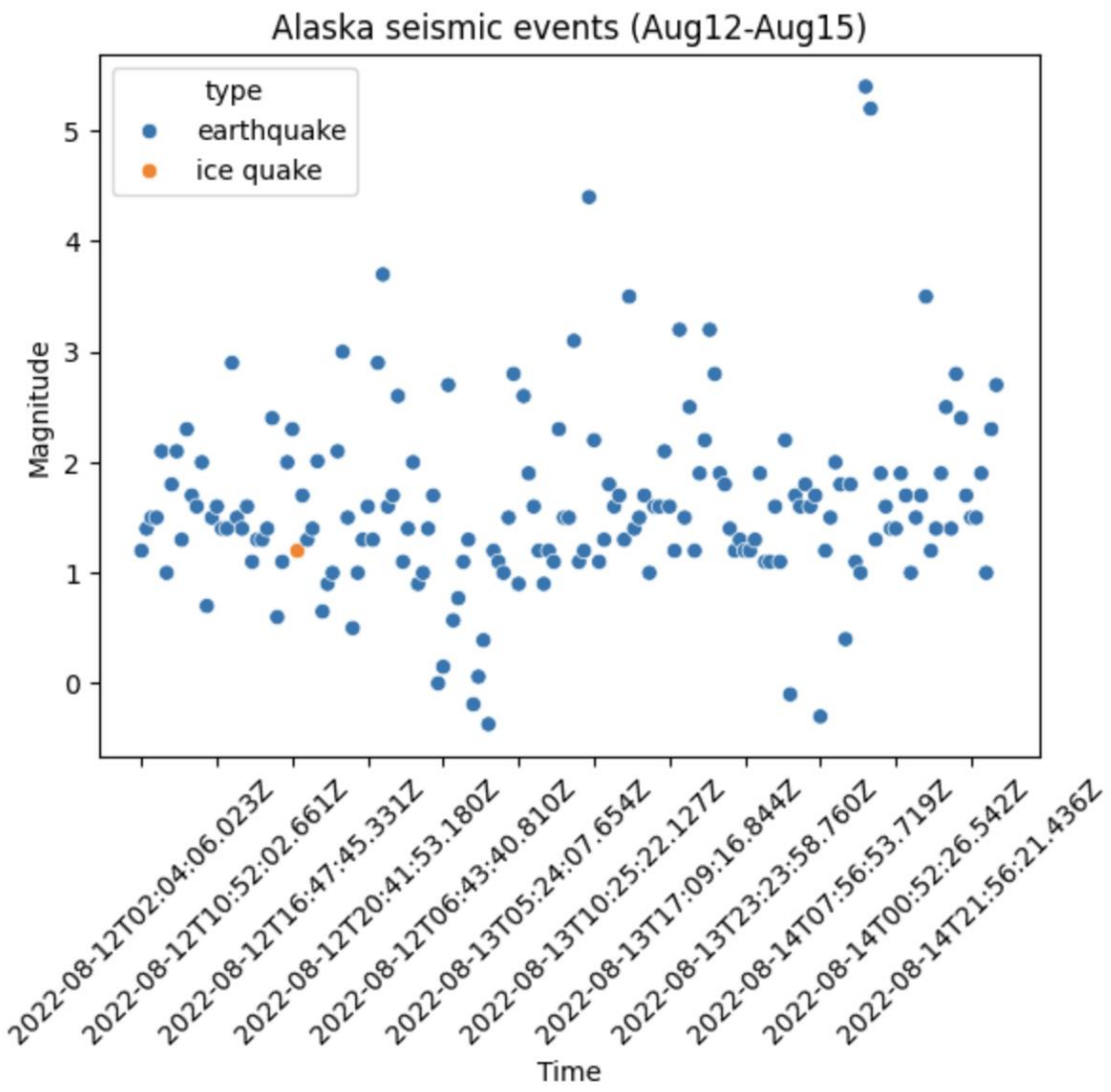

Alaska seismic events (May20-May23)
Magnitude
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
type
earthquake
ice quake
2022-05-20T00:41:41.072Z
2022-05-20T07:09:05.469Z
2022-05-20T12:14:01.717Z
2022-05-20T15:02:20.395Z
2022-05-20T19:31:55.939Z
2022-05-20T11:38:59.800Z
2022-05-21T00:31:23.377Z
2022-05-21T05:50:25.684Z
2022-05-21T11:22:53.334Z
2022-05-21T18:39:24.486Z
2022-05-21T23:03:27.123Z
2022-05-22T07:59:12.487Z
2022-05-22T13:50:05.870Z
2022-05-22T21:27:40.948Z
Time

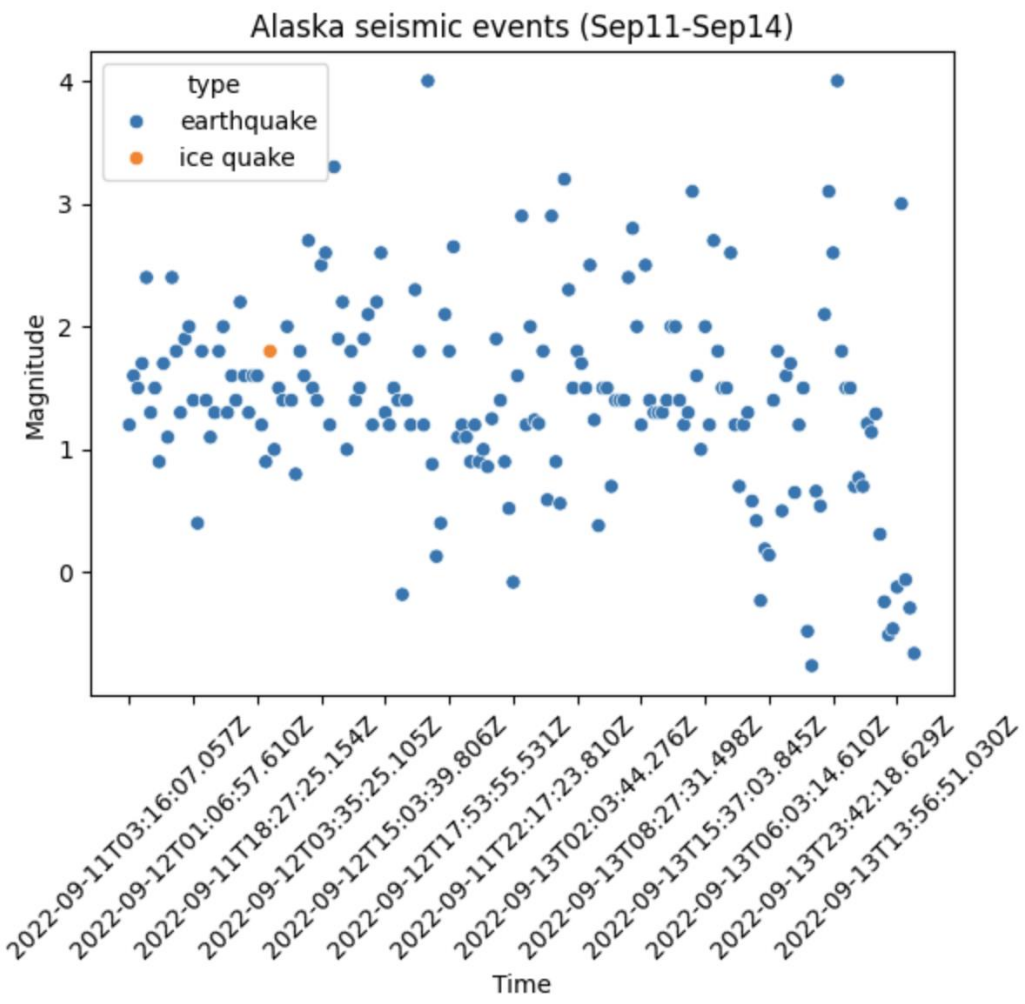

Alaska seismic events (May30-Jun02)
type
earthquake
ice quake
Magnitude
5
4
3
2
1
2022-05-30T01:47:18.051Z
2022-05-30T07:26:23.589Z
2022-05-30T12:38:49.292Z
2022-05-30T17:10:51.521Z
2022-05-30T14:32:39.118Z
2022-05-31T02:26:58.344Z
2022-05-31T09:56:11.961Z
2022-05-31T15:24:31.731Z
2022-05-31T19:35:22.124Z
2022-06-01T01:38:50.017Z
2022-06-01T07:19:16.739Z
2022-06-01T12:33:07.276Z
2022-06-01T15:37:56.127Z
Time

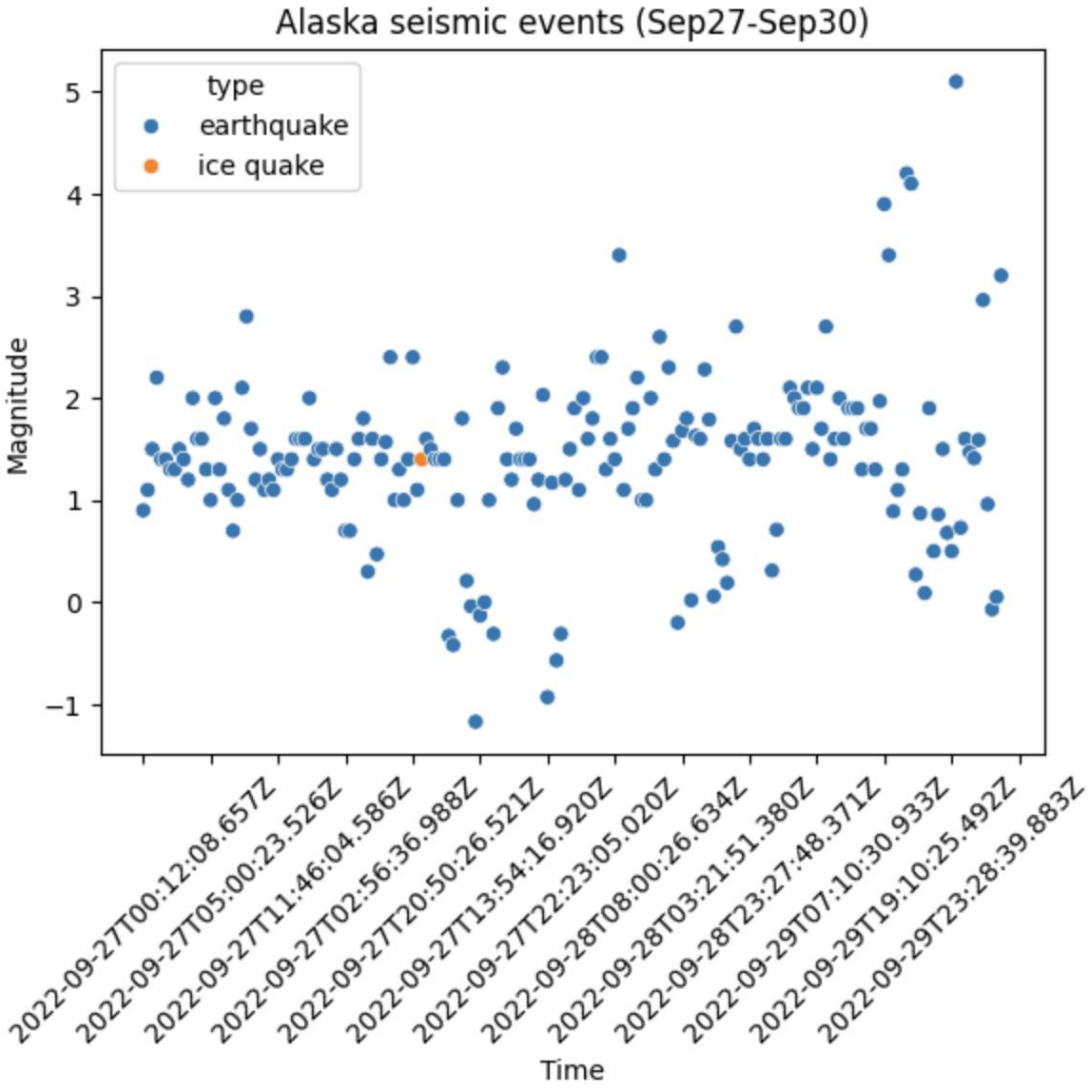

Alaska seismic events (Jun08-Jun11)
type
earthquake
ice quake
explosion
Magnitude
Time
0
1
2
3
4
5
2022-06-08T00:36:36.464Z
2022-06-08T12:51:35.068Z
2022-06-08T20:16:17.842Z
2022-06-09T01:24:28.290Z
2022-06-09T11:42:11.249Z
2022-06-09T17:08:37.844Z
2022-06-10T03:05:49.406Z
2022-06-10T13:03:36.110Z
2022-06-10T20:02:21.462Z
2022-06-10T13:42:25.600Z

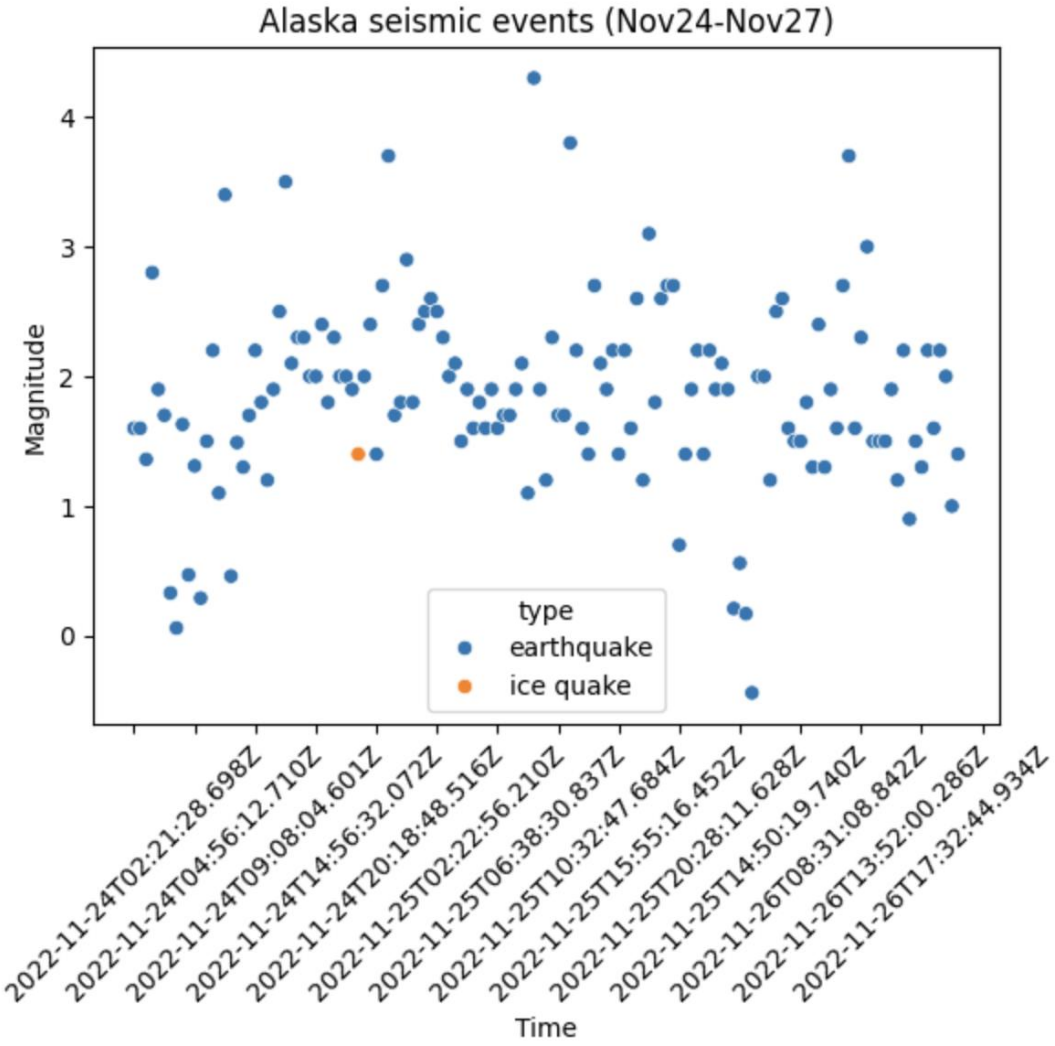

Alaska seismic events (Jun12-Jun15)
type
earthquake
ice quake
Magnitude
4
3
2
1
0
−1
2022-06-12T00:55:48.585Z
2022-06-12T05:22:08.604Z
2022-06-12T13:51:51.599Z
2022-06-13T02:03:17.812Z
2022-06-13T07:11:48.989Z
2022-06-13T14:28:51.032Z
2022-06-12T20:13:15.890Z
2022-06-13T05:16:19.040Z
2022-06-14T03:52:22.125Z
2022-06-14T09:48:29.337Z
2022-06-14T17:51:04.805Z
2022-06-14T08:55:00.400Z
2022-06-14T23:46:47.370Z
Time

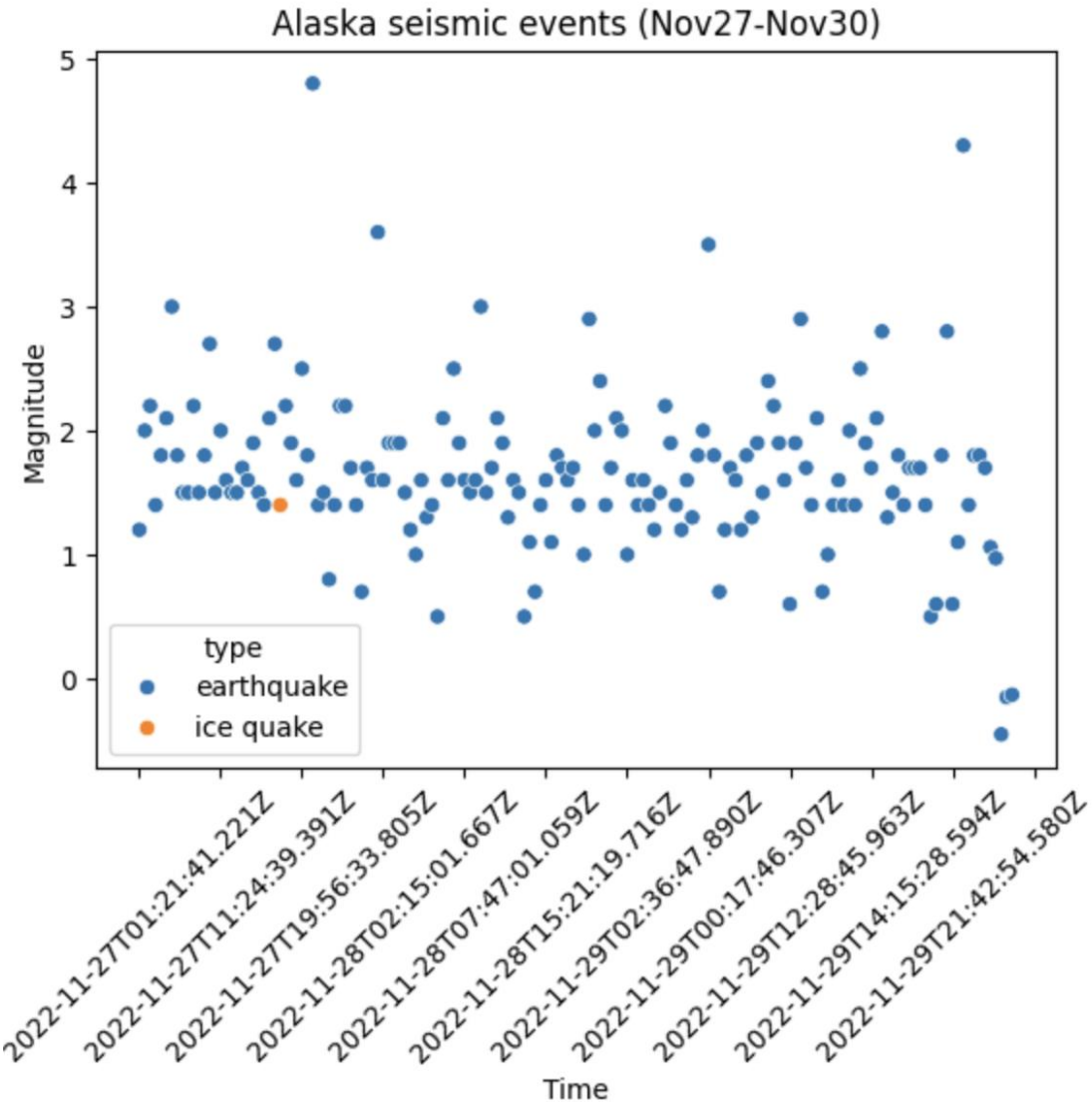

Alaska seismic events (Aug11-Aug14)
type
earthquake
ice quake
Magnitude
0
1
2
3
4
2022-08-11T01:33:58.864Z
2022-08-11T11:20:31.695Z
2022-08-11T18:18:14.104Z
2022-08-11T18:41:48.930Z
2022-08-12T02:13:16.250Z
2022-08-12T12:00:40.496Z
2022-08-12T12:33:23.069Z
2022-08-12T04:47:31.485Z
2022-08-12T11:52:19.930Z
2022-08-13T05:38:06.667Z
2022-08-13T10:39:07.909Z
2022-08-13T17:38:32.192Z
Time


Alaska seismic events (Aug12-Aug15)
type
earthquake
ice quake
Magnitude
0
1
2
3
4
5
2022-08-12T02:04:06.023Z
2022-08-12T10:52:02.661Z
2022-08-12T16:47:45.331Z
2022-08-12T20:41:53.180Z
2022-08-12T06:43:40.810Z
2022-08-13T05:24:07.654Z
2022-08-13T10:25:22.127Z
2022-08-13T17:09:16.844Z
2022-08-13T23:23:58.760Z
2022-08-14T07:56:53.719Z
2022-08-14T00:52:26.542Z
2022-08-14T21:56:21.436Z
Time

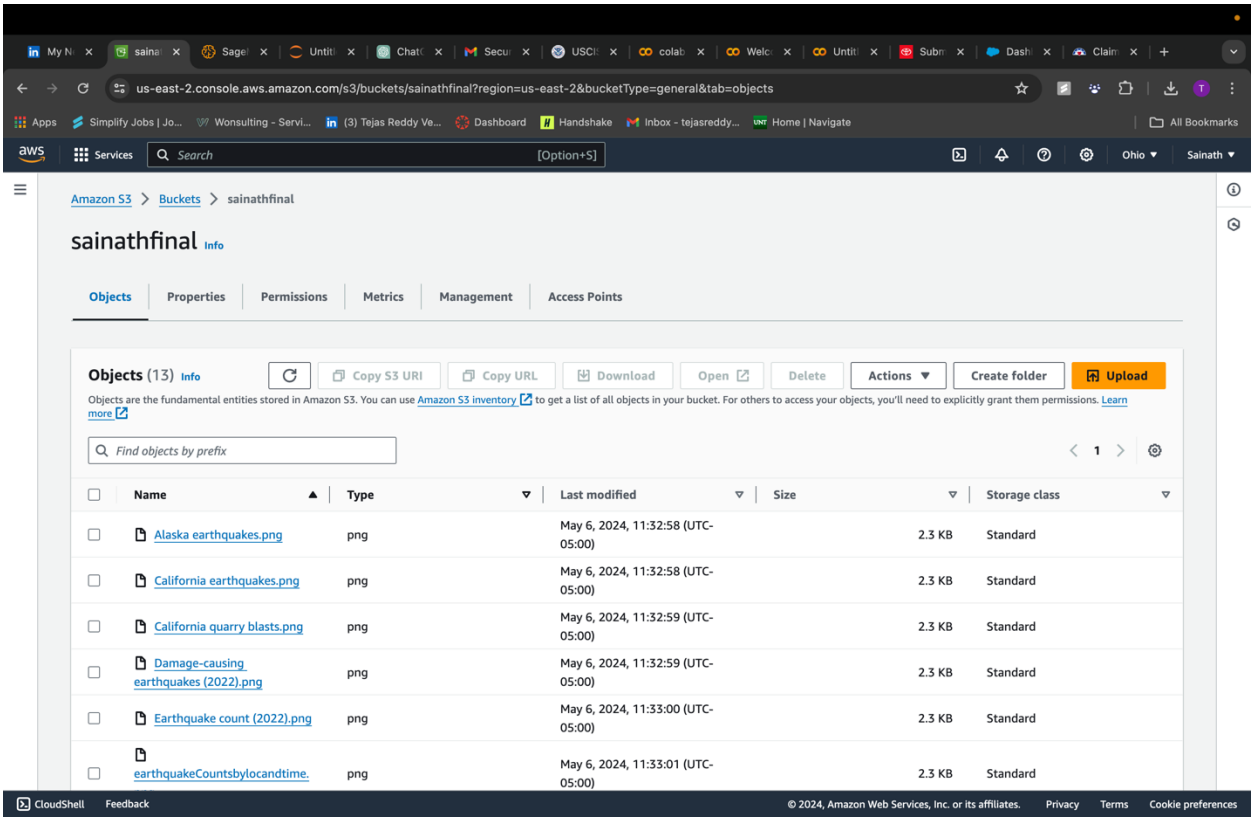

Alaska seismic events (Sep11-Sep14)
type
earthquake
ice quake
Magnitude
4
3
2
1
0
2022-09-11T03:16:07.057Z
2022-09-12T01:06:57.610Z
2022-09-11T18:27:25.154Z
2022-09-12T03:35:25.105Z
2022-09-12T15:03:39.806Z
2022-09-12T17:53:55.531Z
2022-09-11T22:17:23.810Z
2022-09-13T02:03:44.276Z
2022-09-13T08:27:31.498Z
2022-09-13T15:37:03.845Z
2022-09-13T06:03:14.610Z
2022-09-13T23:42:18.629Z
2022-09-13T13:56:51.030Z
Time

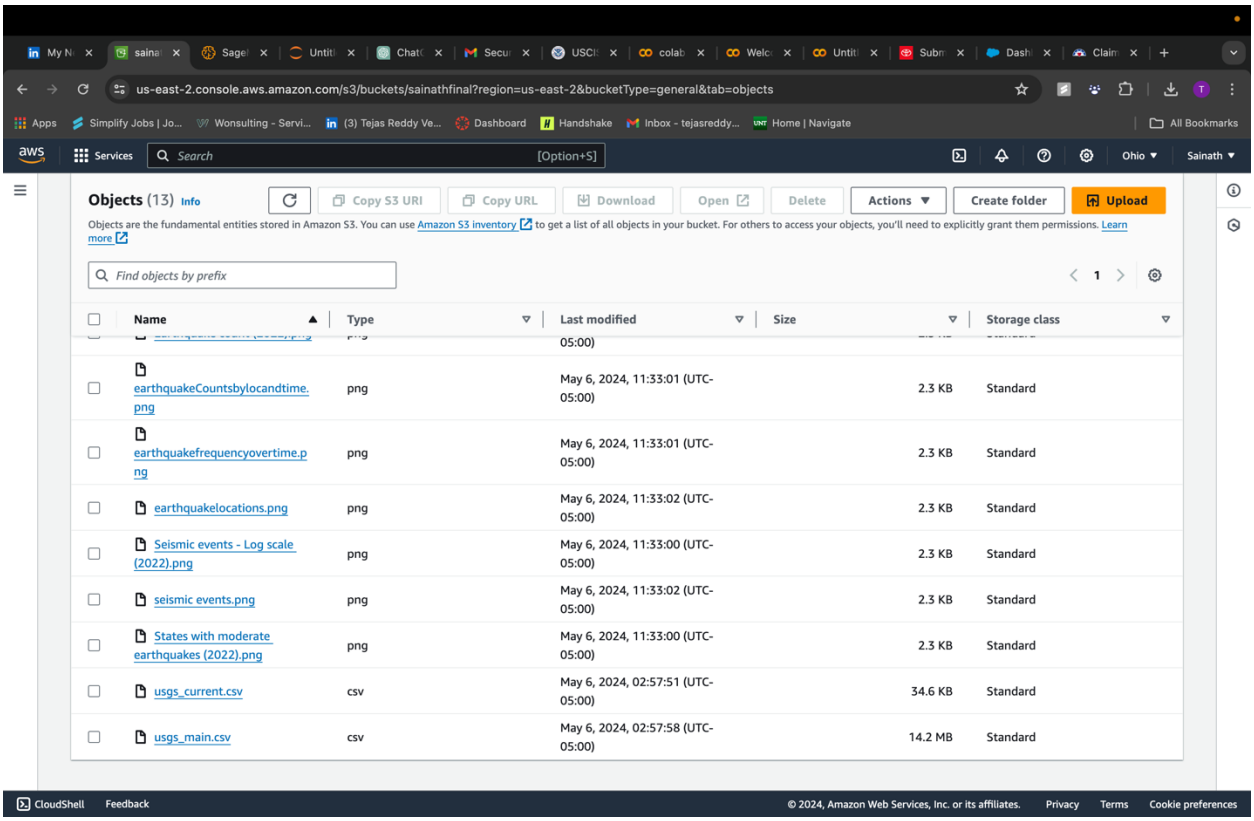

Alaska seismic events (Sep27-Sep30)
type
earthquake
ice quake
Magnitude
5
4
3
2
1
0
−1
2022-09-27T00:12:08.657Z
2022-09-27T05:00:23.526Z
2022-09-27T11:46:04.586Z
2022-09-27T02:56:36.988Z
2022-09-27T20:50:26.521Z
2022-09-27T13:54:16.920Z
2022-09-27T22:23:05.020Z
2022-09-28T08:00:26.634Z
2022-09-28T03:21:51.380Z
2022-09-28T23:27:48.371Z
2022-09-29T07:10:30.933Z
2022-09-29T19:10:25.492Z
2022-09-29T23:28:39.883Z
Time


Alaska seismic events (Nov24-Nov27)
Magnitude
Time
type
earthquake
ice quake
0
1
2
3
4
2022-11-24T02:21:28.698Z
2022-11-24T04:56:12.710Z
2022-11-24T09:08:04.601Z
2022-11-24T14:56:32.072Z
2022-11-24T20:18:48.516Z
2022-11-25T02:22:56.210Z
2022-11-25T06:38:30.837Z
2022-11-25T10:32:47.684Z
2022-11-25T15:55:16.452Z
2022-11-25T20:28:11.628Z
2022-11-25T14:50:19.740Z
2022-11-26T08:31:08.842Z
2022-11-26T13:52:00.286Z
2022-11-26T17:32:44.934Z



Saved png files into d3 from jupyter hub


us-east-2.console.aws.amazon.com/s3/buckets/sainathfinal?region=us-east-2&bucketType=general&tab=objects
Amazon S3 > Buckets > sainathfinal
sainathfinal Info
Objects | Properties | Permissions | Metrics | Management | Access Points
Objects (13) Info
Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name | Type | Last modified | Size | Storage class
Alaska earthquakes.png | png | May 6, 2024, 11:32:58 (UTC-05:00) | 2.3 KB | Standard
California earthquakes.png | png | May 6, 2024, 11:32:58 (UTC-05:00) | 2.3 KB | Standard
California quarry blasts.png | png | May 6, 2024, 11:32:59 (UTC-05:00) | 2.3 KB | Standard
Damage-causing earthquakes (2022).png | png | May 6, 2024, 11:32:59 (UTC-05:00) | 2.3 KB | Standard
Earthquake count (2022).png | png | May 6, 2024, 11:33:00 (UTC-05:00) | 2.3 KB | Standard
earthquakeCountsbylocandtime. | png | May 6, 2024, 11:33:01 (UTC-05:00) | 2.3 KB | Standard
CloudShell | Feedback | © 2024, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences


us-east-2.console.aws.amazon.com/s3/buckets/sainathfinal?region=us-east-2&bucketType=general&tab=objects
Objects (13) Info
Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name | Type | Last modified | Size | Storage class
earthquakeCountsbylocandtime.png | png | May 6, 2024, 11:33:01 (UTC-05:00) | 2.3 KB | Standard
earthquakefrequencyovertime.png | png | May 6, 2024, 11:33:01 (UTC-05:00) | 2.3 KB | Standard
earthquakelocations.png | png | May 6, 2024, 11:33:02 (UTC-05:00) | 2.3 KB | Standard
Seismic events - Log scale (2022).png | png | May 6, 2024, 11:33:00 (UTC-05:00) | 2.3 KB | Standard
seismic events.png | png | May 6, 2024, 11:33:02 (UTC-05:00) | 2.3 KB | Standard
States with moderate earthquakes (2022).png | png | May 6, 2024, 11:33:00 (UTC-05:00) | 2.3 KB | Standard
usgs_current.csv | csv | May 6, 2024, 02:57:51 (UTC-05:00) | 34.6 KB | Standard
usgs_main.csv | csv | May 6, 2024, 02:57:58 (UTC-05:00) | 14.2 MB | Standard
CloudShell | Feedback | © 2024, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences

Conclusion:

This project successfully demonstrated how to use cloud technologies to handle and analyze massive amounts of seismic data to forecast earthquake features using PySpark and AWS SageMaker. This project not only improved our capacity to anticipate natural disasters but also showed the scalability and real-time data processing advantages of cloud computing. Among the things that have been acquired are the value of properly preparing data and the advantages of incorporating real-time data analysis for quick reaction. Subsequent research efforts could investigate the integration of more diverse data sources and sophisticated machine learning models to strengthen prediction precision and broaden the project's relevance to alternative disaster situations.

References:

1. Jordan, M. I., & Mitchell, T. M. (2015). Machine Learning: Trends, Perspectives, and Prospects. Science, 349(6245), 255-260.
2. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2016). Apache Spark: A Unified Engine for Big Data Processing. Communications of the ACM, 59(11), 56-65.
3. Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. R News, 2(3), 18-22. Retrieved from https://CRAN.R-project.org/doc/Rnews/
4. Stein, S., & Wysession, M. (2003). An Introduction to Seismology, Earthquakes, and Earth Structure. Blackwell Publishing.