EarthQuake Analysis

Introduction

Creating a big data analytics pipeline, using big data technologies like PySpark, MLlib, Power BI and MongoDB.

Problem Statement

Create a model to predict the possibility of earth quakes based on sampledata from 1965-2016.

Working with earthquake data that will be transformed into summary tables.Then use these tables to train predictive models and predict future earthquakes. Write the final datasets to Mongodb.After that analyze the data by building reports and dashboards in Power BI Desktop

Dataset Details

Csv files.

Dabase.csv : This file will bethe source file containing earth quake details

Steps Performed

1.Load the dataset containing earthquake details

2.Drop the columns which are not required

3.Created a year column from date
4.Build the quakes frequency dataframe using the year field and counts for each year
5.Check the schema and convert some fields from string to numeric
6.Create avg magnitude and max magnitude fields and add to df_quake_freq
7.Write the dataframes to mongodb

Details

Dataset containing earth quake details are uploaded into Filestore of databricks
/dbfs/FileStore/ProjectPyspark
1.Load the dataset containing earthquake details

```
✓ 05:20 AM (17s)                                   2

#Load the dataset
df_load= (spark.read
          .format("csv")
          .option("header","true")
          .load("dbfs:/FileStore/ProjectPyspark/database.csv")
          )

#Preview of load
df_load.count()

(3) Spark Jobs
df_load:  pyspark.sql.dataframe.DataFrame = [Date: string, Time: string ... 19 more fields]
23412
```

Just now (2s) 3 Python

```
df_load.display()
```

(1) Spark Jobs

Table

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error
1	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	null
2	01/04/1965	11:29:49	1.863	127.352	Earthquake	80	null
3	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20	null
4	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15	null
5	01/09/1965	13:32:50	11.938	126.427	Earthquake	15	null
6	01/10/1965	13:36:32	-13.405	166.629	Earthquake	35	null

Sample data:

```
Date='01/02/1965', Time='13:44:18', Latitude='19.246', Longitude='145.616',
Type='Earthquake', Depth='131.6', Depth Error=None, Depth Seismic Stations=None,
Magnitude='6', Magnitude Type='MW', Magnitude Error=None, Magnitude Seismic
Stations=None, Azimuthal Gap=None, Horizontal Distance=None, Horizontal
Error=None, Root Mean Square=None, ID='ISCGEM860706', Source='ISCGEM', Location
Source='ISCGEM', Magnitude Source='ISCGEM', Status='Automatic'
```

2.Drop the columns which are not required

05:21 AM (1s) 4 Python

```
# Drop fields we don't need from df_load
lst_dropped_columns = ['Depth Error', 'Time', 'Depth Seismic Stations','Magnitude Error','Magnitude Seismic Stations','Azimuthal Gap',
'Horizontal Distance','Horizontal Error','Root Mean Square','Source','Location Source','Magnitude Source','Status']

df_loaddb=df_load.drop(*lst_dropped_columns)
# Preview df_load
df_loaddb.show(5)
```

(1) Spark Jobs

df_loaddb: pyspark.sql.dataframe.DataFrame = [Date: string, Latitude: string ... 6 more fields]

```
+----------+--------+---------+----------+-----+---------+--------------+------------+
|      Date|Latitude|Longitude|      Type|Depth|Magnitude|Magnitude Type|          ID|
+----------+--------+---------+----------+-----+---------+--------------+------------+
|01/02/1965|  19.246|  145.616|Earthquake|131.6|        6|            MW|ISCGEM860706|
|01/04/1965|   1.863|  127.352|Earthquake|   80|      5.8|            MW|ISCGEM860737|
|01/05/1965| -20.579| -173.972|Earthquake|   20|      6.2|            MW|ISCGEM860762|
|01/08/1965| -59.076|  -23.557|Earthquake|   15|      5.8|            MW|ISCGEM860856|
|01/09/1965|  11.938|  126.427|Earthquake|   15|      5.8|            MW|ISCGEM860890|
+----------+--------+---------+----------+-----+---------+--------------+------------+
only showing top 5 rows
```

3.Create a year column from date

05:21 AM (1s) 5

```
#Create a year column and add to dataframe
df_loaddb = df_loaddb.withColumn('Year',year(to_timestamp('Date','dd/MM/yyyy')))
df_loaddb.show(5)
```

(1) Spark Jobs

df_loaddb: pyspark.sql.dataframe.DataFrame = [Date: string, Latitude: string ... 7 more fields]

```
+----------+--------+---------+----------+-----+---------+--------------+------------+----+
|      Date|Latitude|Longitude|      Type|Depth|Magnitude|Magnitude Type|          ID|Year|
+----------+--------+---------+----------+-----+---------+--------------+------------+----+
|01/02/1965|  19.246|  145.616|Earthquake|131.6|        6|            MW|ISCGEM860706|1965|
|01/04/1965|   1.863|  127.352|Earthquake|   80|      5.8|            MW|ISCGEM860737|1965|
|01/05/1965| -20.579| -173.972|Earthquake|   20|      6.2|            MW|ISCGEM860762|1965|
|01/08/1965| -59.076|  -23.557|Earthquake|   15|      5.8|            MW|ISCGEM860856|1965|
|01/09/1965|  11.938|  126.427|Earthquake|   15|      5.8|            MW|ISCGEM860890|1965|
+----------+--------+---------+----------+-----+---------+--------------+------------+----+
only showing top 5 rows
```

4.Build the quakes frequency data frame using the year field and counts for each year

05:21 AM (4s) 6

```
# Build the quakes frequency dataframe using the year field and counts for each year
df_quake_freq = df_loaddb.groupBy("Year").count().withColumnRenamed("count","Counts")
# Preview df_quake_freq
df_quake_freq.show(5)
```

(2) Spark Jobs

df_quake_freq: pyspark.sql.dataframe.DataFrame = [Year: integer, Counts: long]

```
+----+------+
|Year|Counts|
+----+------+
|1990|   196|
|1975|   150|
|1977|   148|
|2003|   187|
|2007|   211|
+----+------+
only showing top 5 rows
```

5.Check the schema and convert some fields from string to numeric

05:21 AM (<1s)

```
# Preview df_loaddb schema
df_loaddb.printSchema()
```

```
root
 |-- Date: string (nullable = true)
 |-- Latitude: string (nullable = true)
 |-- Longitude: string (nullable = true)
 |-- Type: string (nullable = true)
 |-- Depth: string (nullable = true)
 |-- Magnitude: string (nullable = true)
 |-- Magnitude Type: string (nullable = true)
 |-- ID: string (nullable = true)
 |-- Year: integer (nullable = true)
```

05:21 AM (2s)

```
# Cast some fields from string into numeric types
df_loaddb = df_loaddb.withColumn('Latitude', df_loaddb['Latitude'].cast(DoubleType()))\
    .withColumn('Longitude', df_loaddb['Longitude'].cast(DoubleType()))\
    .withColumn('Depth', df_loaddb['Depth'].cast(DoubleType()))\
    .withColumn('Magnitude', df_loaddb['Magnitude'].cast(DoubleType()))

# Preview df_load
df_loaddb.show(5)
```

(1) Spark Jobs

df_loaddb: pyspark.sql.dataframe.DataFrame = [Date: string, Latitude: double ... 7 more fields]

```
+----------+--------+---------+----------+-----+---------+--------------+------------+----+
|      Date|Latitude|Longitude|      Type|Depth|Magnitude|Magnitude Type|          ID|Year|
+----------+--------+---------+----------+-----+---------+--------------+------------+----+
|01/02/1965|  19.246|  145.616|Earthquake|131.6|      6.0|            MW|ISCGEM860706|1965|
|01/04/1965|   1.863|  127.352|Earthquake| 80.0|      5.8|            MW|ISCGEM860737|1965|
|01/05/1965| -20.579| -173.972|Earthquake| 20.0|      6.2|            MW|ISCGEM860762|1965|
|01/08/1965| -59.076|  -23.557|Earthquake| 15.0|      5.8|            MW|ISCGEM860856|1965|
|01/09/1965|  11.938|  126.427|Earthquake| 15.0|      5.8|            MW|ISCGEM860890|1965|
+----------+--------+---------+----------+-----+---------+--------------+------------+----+
only showing top 5 rows
```

05:21 AM (<1s)

```
# Preview df_loaddb schema
df_loaddb.printSchema()
```

```
root
 |-- Date: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Type: string (nullable = true)
 |-- Depth: double (nullable = true)
 |-- Magnitude: double (nullable = true)
 |-- Magnitude Type: string (nullable = true)
 |-- ID: string (nullable = true)
 |-- Year: integer (nullable = true)
```

6.Create avg magnitude and max magnitude fields and join to df_quake_freq

05:21 AM (4s) 10

```
# Create avg magnitude and max magnitude fields and add to df_quake_freq
df_max_avg = df_loaddb.groupBy('Year').agg(max('Magnitude').alias('Max_Magnitude'),avg('Magnitude').alias('Avg_Magnitude'))
df_max_avg.show(5)
```

(2) Spark Jobs

df_max_avg: pyspark.sql.dataframe.DataFrame = [Year: integer, Max_Magnitude: double ... 1 more field]

```
+----+-------------+-----------------+
|Year|Max_Magnitude|    Avg_Magnitude|
+----+-------------+-----------------+
|1990|          7.6|5.858163265306125|
|1975|          7.8| 5.84866666666667|
|1977|          7.6|5.757432432432437|
|2003|          7.6|5.850802139037435|
|2007|          8.4| 5.89099526066351|
+----+-------------+-----------------+
only showing top 5 rows
```

```
05:21 AM (6s)                                  11

# Join df_max_avg to df_quake_freq
df_quake_freq=df_quake_freq.join(df_max_avg,'Year')
# Preview df_quake_freq
df_quake_freq.show(5)
```

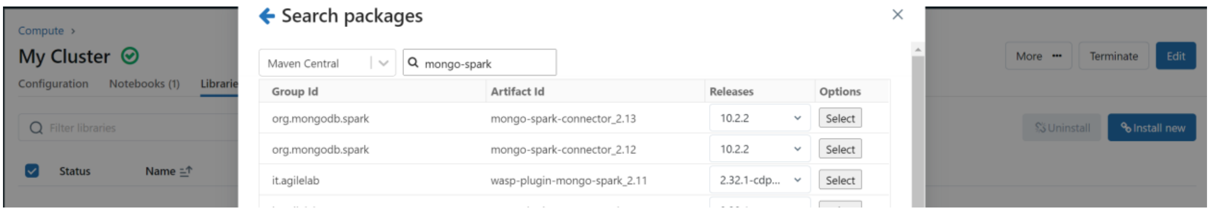
```
(4) Spark Jobs
df_quake_freq: pyspark.sql.dataframe.DataFrame = [Year: integer, Counts: long ... 2 more fields]
+----+------+-------------+-----------------+
|Year|Counts|Max_Magnitude|    Avg_Magnitude|
+----+------+-------------+-----------------+
|1990|   196|          7.6|5.858163265306125|
|1975|   150|          7.8| 5.84866666666667|
|1977|   148|          7.6|5.757432432432437|
|2003|   187|          7.6|5.850802139037435|
|2007|   211|          8.4| 5.89099526066351|
+----+------+-------------+-----------------+
only showing top 5 rows
```

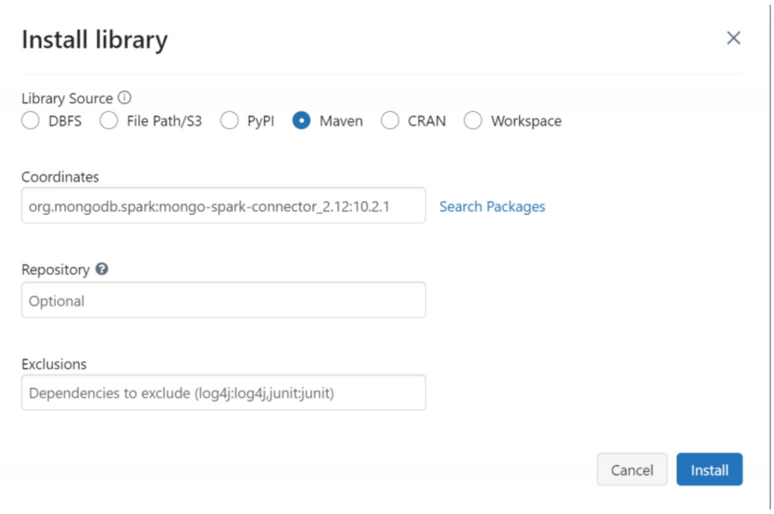
```
05:22 AM (3s)                                  14

df_quake_freq.show(5)
```

```
(4) Spark Jobs
+----+------+-------------+-----------------+
|Year|Counts|Max_Magnitude|    Avg_Magnitude|
+----+------+-------------+-----------------+
|1990|   196|          7.6|5.858163265306125|
|1975|   150|          7.8| 5.84866666666667|
|1977|   148|          7.6|5.757432432432437|
|2003|   187|          7.6|5.850802139037435|
|2007|   211|          8.4| 5.89099526066351|
+----+------+-------------+-----------------+
only showing top 5 rows
```

7.Write the data frames to mongo Db

To establish a connection from databricks to mongodb we need to install mongo-spark connector library.





Compute ›

My Cluster

Configuration | Notebooks (0) | **Libraries** | Event log | Spark UI | Driver logs | Metrics | Apps | Spark compute UI - Master

Filter libraries

Status	Name	Type	Source
✓	org.mongodb.spark:mongo-spark-connector_2.12:10.2.1	Maven	-

Write from databricks to mongodb

3 minutes ago (2m) 15

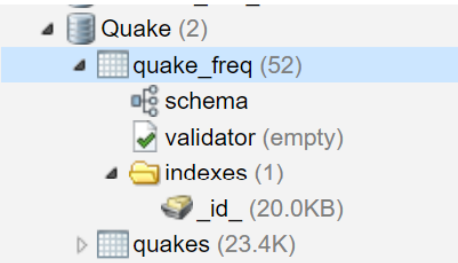
```
# Build the tables/collections in mongodb
# Write df_loaddb to mongodb
df_loaddb.write.format('mongodb')\
    .mode('overwrite')\
    .option("spark.mongodb.connection.uri", "mongodb+srv://[redacted]clusteredu.vqzezro.mongodb.net/")\
    .option("database","Quake")\
    .option("collection","quakes")\
    .save()
```

(1) Spark Jobs
Job 32 View (Stages: 1/1)
Stage 48 1/1 succeeded View

Just now (5s) 16 Python

```
# Write df_quake_freq to mongodb
df_quake_freq.write.format('mongodb')\
    .mode('overwrite')\
    .option('spark.mongodb.connection.uri', "mongodb+srv://[redacted]clusteredu.vqzezro.mongodb.net/")\
    .option("database","Quake")\
    .option("collection","quake_freq")\
    .save()
```

Mongodb:



Machine Learning

8.Load the Query.csv file which was stored in the HDFS which contains the earthquake test data

Sample

```
time,latitude,longitude,depth,mag,magType,nst,gap,dmin,rms,net,id,updated,place,type,horizontalError,depthError,magError,magNst,status,locationSource,magSource
2017-01-02T00:13:06.300Z,-36.0365,51.9288,10,5.7,mwb,,26,14.685,1.37,us,us10007p5d,2017-03-27T23:53:17.040Z,"Southwest Indian Ridge",earthquake,10.3,1.7,0.068,21,reviewed,us,us
```

Just now (2s) 18 Python

```
# Load the test data file into a dataframe
df_test = (spark.read
        .format("csv")
        .option("header","true")
        .load("dbfs:/FileStore/ProjectPyspark/query.csv")
        )
# Preview df_test
df_test.take(1)
```

(2) Spark Jobs

df_test: pyspark.sql.dataframe.DataFrame = [time: string, latitude: string ... 20 more fields]

```
[Row(time='2017-01-02T00:13:06.300Z', latitude='-36.0365', longitude='51.9288', depth='10', mag='5.7', magType='mwb', nst=None, gap='26', dm
in='14.685', rms='1.37', net='us', id='us10007p5d', updated='2017-03-27T23:53:17.040Z', place='Southwest Indian Ridge', type='earthquake', h
orizontalError='10.3', depthError='1.7', magError='0.068', magNst='21', status='reviewed', locationSource='us', magSource='us')]
```

9.Load the training data from mongodb and Perform the data cleansing activity such as selecting only required fields,Renaming fields,casting specific fields to the desired datatype

Just now (4s) 19 Python

```
# Load the training data from mongo into a dataframe
df_train = spark.read.format('mongodb')\
    .option('spark.mongodb.connection.uri', "mongodb+srv:/[redacted]clusteredu.vqzezro.mongodb.net/")\
    .option("database","Quake")\
    .option("collection","quakes")\
    .load()

# Preview df_train
df_train.show(5)
```

(1) Spark Jobs

df_train: pyspark.sql.dataframe.DataFrame = [Date: string, Depth: double ... 8 more fields]

```
+----------+-----+------------+--------+---------+---------+--------------+----------+----+--------------------+
|      Date|Depth|          ID|Latitude|Longitude|Magnitude|Magnitude Type|      Type|Year|                 _id|
+----------+-----+------------+--------+---------+---------+--------------+----------+----+--------------------+
|01/02/1965|131.6|ISCGEM860706|  19.246|  145.616|      6.0|            MW|Earthquake|1965|65f24936bdf037582...|
|01/04/1965| 80.0|ISCGEM860737|   1.863|  127.352|      5.8|            MW|Earthquake|1965|65f24936bdf037582...|
|01/05/1965| 20.0|ISCGEM860762| -20.579| -173.972|      6.2|            MW|Earthquake|1965|65f24936bdf037582...|
|01/08/1965| 15.0|ISCGEM860856| -59.076|  -23.557|      5.8|            MW|Earthquake|1965|65f24936bdf037582...|
|01/09/1965| 15.0|ISCGEM860890|  11.938|  126.427|      5.8|            MW|Earthquake|1965|65f24936bdf037582...|
+----------+-----+------------+--------+---------+---------+--------------+----------+----+--------------------+
only showing top 5 rows
```

1 minute ago (1s) 20

```
# Select fields we will use and discard fields we don't need
df_test_clean = df_test['time', 'latitude', 'longitude', 'mag', 'depth']
# Preview df_test_clean
df_test_clean.show(5)
```

(1) Spark Jobs

df_test_clean: pyspark.sql.dataframe.DataFrame = [time: string, latitude: string ... 3 more fields]

```
+--------------------+--------+---------+---+------+
|                time|latitude|longitude|mag| depth|
+--------------------+--------+---------+---+------+
|2017-01-02T00:13:...|-36.0365|  51.9288|5.7|    10|
|2017-01-02T13:13:...|  -4.895| -76.3675|5.9|   106|
|2017-01-02T13:14:...|-23.2513| 179.2383|6.3|551.62|
|2017-01-03T09:09:...| 24.0151|  92.0177|5.7|    32|
|2017-01-03T21:19:...|-43.3527| -74.5017|5.5| 10.26|
+--------------------+--------+---------+---+------+
only showing top 5 rows
```

```
df_test_clean.show()
```

(1) Spark Jobs

```
+--------------------+--------+---------+---------+------+
|                Date|Latitude|Longitude|Magnitude| Depth|
+--------------------+--------+---------+---------+------+
|2017-01-02T00:13:...|-36.0365|  51.9288|      5.7|  10.0|
|2017-01-02T13:13:...|  -4.895| -76.3675|      5.9| 106.0|
|2017-01-02T13:14:...|-23.2513| 179.2383|      6.3|551.62|
|2017-01-03T09:09:...| 24.0151|  92.0177|      5.7|  32.0|
|2017-01-03T21:19:...|-43.3527| -74.5017|      5.5| 10.26|
```

10.Create testing and training dataframes

```
# Create training and testing dataframes
df_testing = df_test_clean['Latitude', 'Longitude', 'Magnitude', 'Depth']
df_training = df_train['Latitude', 'Longitude', 'Magnitude', 'Depth']
```

```
df_testing: pyspark.sql.dataframe.DataFrame
    Latitude: double
    Longitude: double
    Magnitude: double
    Depth: double
df_training: pyspark.sql.dataframe.DataFrame
    Latitude: double
    Longitude: double
    Magnitude: double
    Depth: double
```

11. Import the relevant modules for machine learning and create mode land evaluate it

RandomForestRegressor
VectorAssembler
RegressionEvaluator

Select features to parse into model and then create the feature vector. After that create and train the model and make the predictions.

```
from pyspark.ml import Pipeline
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import RegressionEvaluator
```

```
# Select features to parse into our model and then create the feature vector
assembler = VectorAssembler(inputCols=['Latitude', 'Longitude', 'Depth'], outputCol='features')

# Create the Model
model_reg = RandomForestRegressor(featuresCol='features', labelCol='Magnitude')

# Chain the assembler with the model in a pipeline
pipeline = Pipeline(stages=[assembler, model_reg])

# Train the Model
model = pipeline.fit(df_training)

# Make the prediction
pred_results = model.transform(df_testing)
```

(8) Spark Jobs

pred_results: pyspark.sql.dataframe.DataFrame = [Latitude: double, Longitude: double ... 4 more fields]

```
Just now (1s)    32    Python
```

```python
# Preview pred_results dataframe
pred_results.show(5)
```

(1) Spark Jobs

```
+--------+---------+---------+------+--------------------+------------------+
|Latitude|Longitude|Magnitude| Depth|            features|        prediction|
+--------+---------+---------+------+--------------------+------------------+
|-36.0365|  51.9288|      5.7|  10.0|[-36.0365,51.9288...| 5.850136724800972|
|  -4.895| -76.3675|      5.9| 106.0|[-4.895,-76.3675,...| 5.869637793167875|
|-23.2513| 179.2383|      6.3|551.62|[-23.2513,179.238...| 5.896591561887881|
| 24.0151|  92.0177|      5.7|  32.0|[24.0151,92.0177,...| 5.931774818629581|
|-43.3527| -74.5017|      5.5| 10.26|[-43.3527,-74.501...|5.9204458792780486|
+--------+---------+---------+------+--------------------+------------------+
only showing top 5 rows
```

Evaluate the model.rmse should be less than 0.5 for the model to be useful

```
Just now (1s)    33
```

```python
# Evaluate the model
# rmse should be less than 0.5 for the model to be useful
evaluator = RegressionEvaluator(labelCol='Magnitude', predictionCol='prediction', metricName='rmse')
rmse = evaluator.evaluate(pred_results)
print('Root Mean Squared Error (RMSE) on test data = %g' % rmse)
```

(1) Spark Jobs

```
Root Mean Squared Error (RMSE) on test data = 0.403077
```

12.Create the prediction Dataset

```
Just now (1s)    35    Python
```

```python
# Create the prediction dataset
df_pred_results = pred_results['Latitude', 'Longitude', 'prediction']

# Rename the prediction field
df_pred_results = df_pred_results.withColumnRenamed('prediction', 'Pred_Magnitude')

# Add more columns to our prediction dataset
df_pred_results = df_pred_results.withColumn('Year', lit(2017))\
    .withColumn('RMSE', lit(rmse))

# Preview df_pred_results
df_pred_results.show(5)
```

(1) Spark Jobs

df_pred_results: pyspark.sql.dataframe.DataFrame = [Latitude: double, Longitude: double ... 3 more fields]

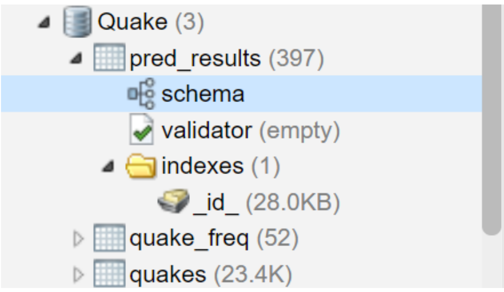
```
+--------+---------+------------------+----+-------------------+
|Latitude|Longitude|    Pred_Magnitude|Year|               RMSE|
+--------+---------+------------------+----+-------------------+
|-36.0365|  51.9288| 5.850136724800972|2017|0.40307689171713734|
|  -4.895| -76.3675| 5.869637793167875|2017|0.40307689171713734|
|-23.2513| 179.2383| 5.896591561887881|2017|0.40307689171713734|
| 24.0151|  92.0177| 5.931774818629581|2017|0.40307689171713734|
|-43.3527| -74.5017|5.9204458792780486|2017|0.40307689171713734|
+--------+---------+------------------+----+-------------------+
only showing top 5 rows
```

13.Load the Prediction dataset to Mongodb

```
Just now (4s)    36    Python
```

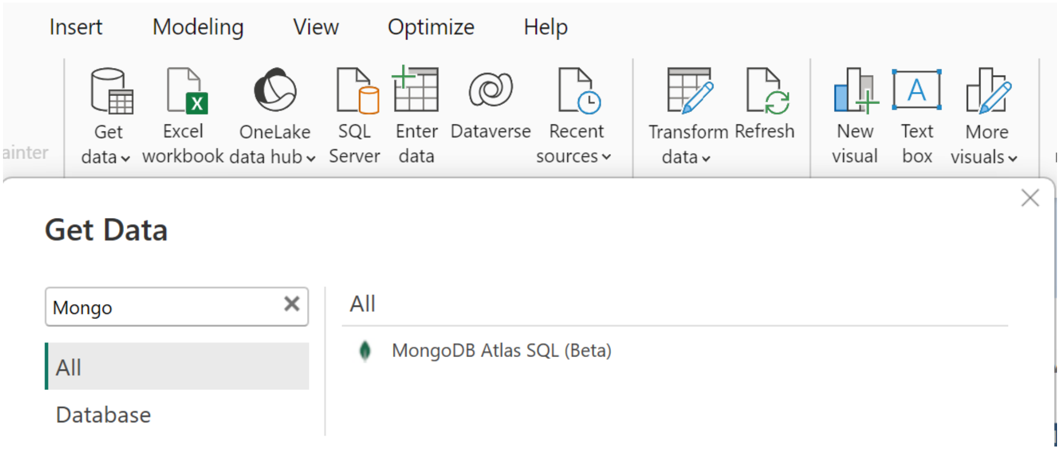
```python
# Load the prediction dataset into mongodb
# Write df_pred_results

df_pred_results.write.format('mongodb')\
    .mode('overwrite')\
    .option("spark.mongodb.connection.uri", "mongodb+srv:/[redacted]clusteredu.vqzezro.mongodb.net/")\
    .option("database","Quake")\
    .option("collection","pred_results")\
    .save()
```

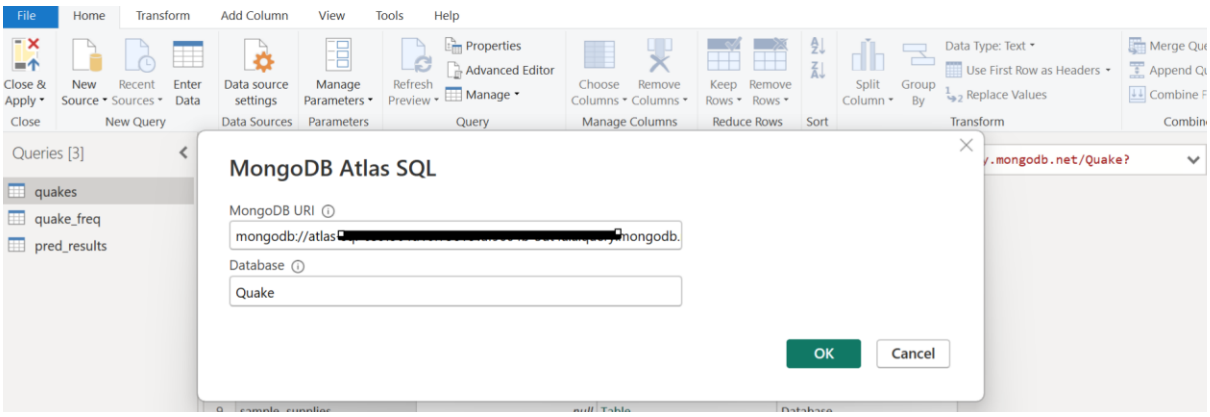
(1) Spark Jobs



14.Connect the tables from MongoDB using Power BI

https://www.mongodb.com/docs/atlas/data-federation/query/sql/powerbi/connect/#std-label-sql-connect-powerbi





15.Create a report in Power BI which will showcase the different metrics and predictions.

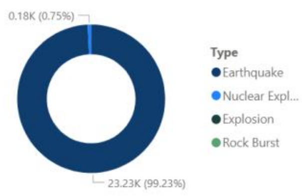
Earthquake Analysis

9201
Total Earthquakes

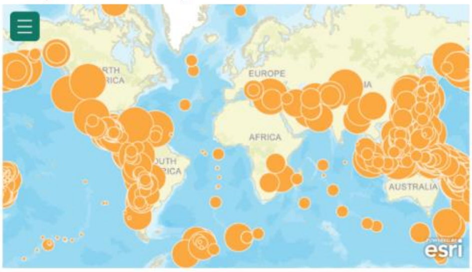
5.89
Average Magnitude

9.10
Maximum Magnitude

Types of Earthquakes

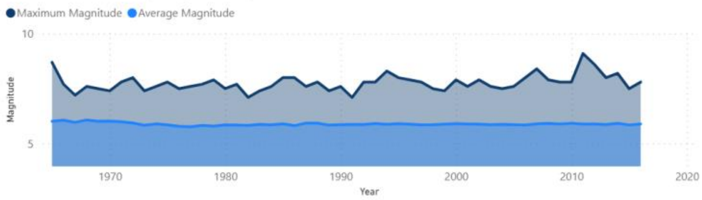


Earthquake Prediction Map

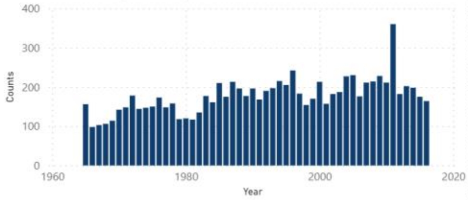


Latitude	Longitude	Predicted Magnitude	RMSE	Year
-23.73	-175.68	5.80	0.40	2017
-36.29	-98.43	5.80	0.40	2017
-36.16	-100.62	5.80	0.40	2017
86.88	54.15	5.80	0.40	2017
86.92	55.20	5.80	0.40	2017
86.90	53.21	5.80	0.40	2017
-62.74	-161.09	5.80	0.40	2017
-59.62	-151.12	5.80	0.40	2017
-56.97	-141.41	5.80	0.40	2017
-56.56	-142.17	5.80	0.40	2017

Maximum and Average Magnitude by Year



Number of Earthquakes by Year



Earthquake Analysis - Detail

Category
- [x] Danger
- [ ] Extreme Danger
- [ ] Moderate
- [ ] Normal

9201
Total Earthquakes

5.89
Average Magnitude

9.10
Maximum Magnitude

Year	Date	Category	Latitude	Longitude	Magnitude	Depth	Type	Magnitude Type	ID
2006	01/02/2006	Danger	-60.96	-21.61	7.40	13.00	Earthquake	MWC	USP000E7DD
2003	08/04/2003	Danger	-60.53	-43.41	7.60	10.00	Earthquake	MWC	USP000C41F
1970	06/11/1970	Danger	-59.23	158.90	7.30	15.00	Earthquake	MW	ISCGEM794513
1987	09/03/1987	Danger	-58.89	158.51	7.40	33.00	Earthquake	MW	USP000381S
1965	08/02/1965	Danger	-56.05	157.92	7.30	10.00	Earthquake	MW	ISCGEM854292
1971	01/03/1971	Danger	-55.92	-2.67	7.10	15.00	Earthquake	MW	ISCGEM787884
2008	04/12/2008	Danger	-55.66	158.45	7.10	16.00	Earthquake	MWC	USP000G3Q9
2015	12/04/2015	Danger	-47.62	85.09	7.10	35.00	Earthquake	MWW	US100043Z2
1979	10/12/1979	Danger	-46.68	165.71	7.40	33.00	Earthquake	MS	USP00013CK
2001	12/12/2001	Danger	-42.81	124.69	7.10	10.00	Earthquake	MWC	USP000AUFX
2011	01/02/2011	Danger	-38.36	-73.33	7.20	24.00	Earthquake	MWW	USP000HSFQ
1975	05/10/1975	Danger	-38.18	-73.23	7.70	6.00	Earthquake	MS	USP0000AZ9
1995	02/05/1995	Danger	-37.76	178.75	7.10	21.10	Earthquake	MW	USP0006SEF
1985	04/09/1985	Danger	-34.13	-71.62	7.20	37.80	Earthquake	MW	USP0002DM4
1985	03/04/1985	Danger	-33.21	-71.66	7.40	33.00	Earthquake	MW	USP0002CD3
1985	03/03/1985	Danger	-33.14	-71.87	8.00	33.00	Earthquake	MW	USP0002CCZ
1971	07/09/1971	Danger	-32.60	-71.08	7.80	60.30	Earthquake	MW	ISCGEM782010
1978	02/09/1978	Danger	-30.68	-177.36	7.20	33.00	Earthquake	MS	USP0000SWB
2001	06/03/2001	Danger	-29.67	-178.63	7.20	178.10	Earthquake	MWC	USP000AFYG
2011	07/06/2011	Danger	-29.54	-176.34	7.60	17.00	Earthquake	MWW	USP000J48H
1995	07/03/1995	Danger	-29.21	-177.59	7.20	35.30	Earthquake	MWB	USP000702Z
1974	07/02/1974	Danger	-29.08	-175.95	7.20	33.00	Earthquake	MS	USP00006ZM
1983	10/04/1983	Danger	-26.54	-70.56	7.40	14.80	Earthquake	MW	USP0001YT[illegible]