Realtime Earthquake Prediction

Abstract:

Countless dollars and entire scientific careers have been dedicated to predicting where and when the next big earthquake will strike. But unlike weather forecasting, which has significantly improved with the use of better satellites and more powerful mathematical models,earthquake predictionhas been marred by repeated failure due to highly uncertain conditions of earth and its surroundings. Now, with the help of artificial intelligence, a growing numberofscientistssaychanges inthe waytheycananalyze massiveamounts of seismic data can help them better understand earthquakes, anticipate how they will behave, and provide quicker and more accurate early warnings. This helps in hazzard assessments for many builders and real estate business for infrastructure planning from business perspective. Also many lives can be saved through early warning. This project aims a simple solution to above problem bypredicting orforecasting likely places to have earthquake in next 7 days.Foruser-friendlypart,thisprojecthasawebapplicationthatextractslive data updated every minute by USGS.gov and predicts next likely place world wide to get hit by an earthquake, hence a realtime solution is provided.

ProblemStatementandapproachtosolution:

Anticipatingseismic tremorsisapivotalissueinEarthsciencebecauseoftheir overwhelmingand hugescopeoutcomes. The goalof this project is to predict where likely in the world and on what dates the earthquake will happen. Applicationandimpactoftheproject includespotentialtoimproveearthquake hazard assessments that could spare lives and billions of dollars in infrastructure and planning. Given geological locations, magnitude and other factors in dataset from https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php for 30 days past which is updated every minute, we predict or forecast 7 days time in future thatis yet to come, the places where quake would likely happen. Since this is event series problem type, proposed solution in this project follows consideringbinaryclassificationofearthquakeoccurance withtrainingperiod includesfixedrollingwindowmovingaveragesofpastdayswhileforwhichits labels, a fixed window size shifted ahead in time. The model will be trained with Adaboost classifier (RandomForestClassifier and DecisionTreeClassifier) andcompared with XGBoostbasedonAUC ROC score andrecall score due to the nature of problem(i.e binary classification). Model with betterAUC score andrecallwillbeconsideredtopredictplaceswhere earthquakemightoccur.

Metrics:

Theproblemaddressedaboveisaboutbinaryclassification,Earthquakeoccur = 1and Earthquake notoccur = 0 and with this prediction we try to locate co-cordinatescorrospondingto thepredictionsanddisplayitonthegooglemaps api webapp. More suitable metrics for binary clsssification problems are ROC (Recieveroperatorcharacteristics),AUC(AreaUnderCurve),Confusionmatrix for Precision, recall, accuracy and sensitivity. We need to minimize or get less Falsenegative predictionssincewe dont wantour modelto predictas 0or no earthquake occured at particular location when in reality it had actually happend as this is more dangerous than the prediction case in which prediction is true/1 or earthquake occured but in reality it did not because its always better safe than sorry!!!. Hence apart from roc_auc score, I have considered Recall as well for evaluation and model selection with higher auc_roc score and recall, where recall = (TP/TP+FN).

Codewithexplanation:

```
importnumpy asnp
importpandasaspd
from sklearn import preprocessing;
fromsklearnimportmodel_selection;
from sklearn import linear_model;
import os
fromsqlalchemyimportcreate_engine
fromsqlalchemy.ext.declarativeimportdeclarative_base
fromsqlalchemy.ormimportscoped_session,sessionmaker
import xgboost as xgb
importdatetimeasdt
importmatplotlib.pyplotasplt

importwarnings
warnings.simplefilter(action='ignore',category=FutureWarning)
```

```
!pip3installxgboost
```

```
Collecting xgboost
  Downloading xgboost-1.7.5-py3-none-win_amd64.whl (70.9 MB)
Requirementalreadysatisfied:numpyinc:\users\rupin\anaconda3\lib\
site-packages (from xgboost) (1.21.5)
Requirementalreadysatisfied:scipyinc:\users\rupin\anaconda3\lib\
site-packages (from xgboost) (1.7.3)
Installingcollectedpackages:xgboost Successfully
installed xgboost-1.7.5
```

Getpast30daysearthquakedatafromearthquake.usgs.govthatis being updated every minute (live).

Lets import the dataset downloaded from here :-
https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_month.csv

```
df=pd.read_csv('all_month.csv')
```

Featuresinthe dataset

- time--------------------- Timewhentheeventoccurred.Timesarereportedin millisecondssincetheepoch
- latitude -------------------Decimaldegreeslatitude.Negativevaluesforsouthern latitudes.
- longitude ------------------Decimaldegreeslongitude.Negativevaluesforwestern longitudes.
- depth---------------------- Depthoftheeventinkilometers.
- mag------------------------ Magnitudeofeventoccured.
- magType-------------------- Themethodoralgorithmusedtocalculatethepreferred magnitude

- nst -----------------------Thetotalnumberofseismicstationsusedtodetermine earthquakelocation.
- gap----------------------- Thelargestazimuthalgapbetweenazimuthallyadjacent stations(indegrees).
- dmin---------------------- Horizontaldistancefromtheepicentertothenearest station(indegrees).
- rms -----------------------Theroot-mean-square(RMS)traveltimeresidual,insec, usingallweights.
- net ------------------------TheIDofadatasourcecontributorforeventoccured.
- id------------------------- Auniqueidentifierfortheevent.
- types ----------------------Acomma-separatedlistofproducttypesassociatedtothis event.
- place---------------------- namedgeographicregionneartotheevent.
- type----------------------- Typeofseismicevent.
- locationSource--------------Thenetworkthatoriginallyauthoredthereported locationofthisevent.
- magSource ------------------Networkthatoriginallyauthoredthereported magnitudeforthisevent.
- horizontalError ------------Uncertaintyofreportedlocationoftheeventin kilometers.
- depthError ----------------- Thedeptherror,threeprincipalerrorsonaverticalline.
- magError ------------------- Uncertaintyofreportedmagnitudeoftheevent.
- magNst--------------------- Thetotalnumberofseismicstationstocalculatethe magnitudeofearthquake.
- status--------------------- Indicateswhethertheeventhasbeenreviewedbyahuman.

```
df.head()

                         time   latitude   longitude   depth   mag
magType\
02023-04-17T09:16:42.127Z     59.659000 -153.056700  109.20  2.10
ml
12023-04-17T09:13:53.100Z     40.292168 -124.534836   10.02  2.56
md
22023-04-17T09:11:18.520Z     38.804001 -122.767334   -0.24  1.09
md
32023-04-17T09:04:24.684Z     62.921300 -151.296400  111.10  1.70
ml
42023-04-17T09:04:22.730Z     34.025500 -117.007667   14.15  1.34
ml

    nst    gap     dmin   rms  ...                   updated  \
0   NaN    NaN      NaN  0.41  ...  2023-04-17T09:18:44.908Z
1  11.0  281.0  0.19390  0.12  ...  2023-04-17T09:29:16.965Z
2  16.0   85.0  0.01394  0.05  ...  2023-04-17T09:29:16.845Z
3   NaN    NaN      NaN  0.35  ...  2023-04-17T09:06:26.645Z
4  47.0   43.0  0.09033  0.20  ...  2023-04-17T09:07:59.986Z
```

```
                              place        type horizontalError
depthError\
0    60 km ESE of Pedro Bay, Alaskaearthquake              NaN
0.40
1              22kmWofPetrolia, CA  earthquake            2.60
1.12
2            3kmNNWofTheGeysers,CA  earthquake            0.25
0.68
354       kmNNWofPetersville, Alaska  earthquake           NaN
1.30
4              3kmESEofYucaipa, CA  earthquake            0.22
0.47

   magErrormagNst       statuslocationSource magSource
0       NaN     NaNautomatic               ak        ak
1     0.050     5.0automatic               nc        nc
2     0.180    19.0automatic               nc        nc
3       NaN     NaNautomatic               ak        ak
4     0.192    29.0automatic               ci        ci

[5rowsx22columns] df.shape

(11623,22)

df.describe()

           latitude     longitude        depth           mag
nst  \
count11623.000000    11623.000000  11623.000000  11623.000000
8298.000000
mean      43.585715   -125.315001     22.481352      1.461778
20.686671
std       17.654440     59.833629     49.915174      1.127382
21.426840
min      -59.561000   -179.984500     -3.440000     -1.320000
0.000000
25%       35.787750   -155.264500      2.400000      0.790000
7.000000
50%       44.494000   -140.580700      7.030000      1.280000
14.000000
75%       58.221150   -117.584750     16.700000      1.900000
27.000000
max       86.599400    179.994100    648.297000      7.000000
310.000000

               gap          dmin            rms  horizontalError
depthError\
count8298.000000    5930.000000  11623.000000      7724.000000
```

```
11623.000000
mean    126.020494     0.492216      0.290185         1.371146
1.735627
std      65.637936     1.825506      0.270365         2.597778
5.216584
min      12.000000     0.000000      0.000000         0.060000
0.000000
25%      72.000000     0.013520      0.090000         0.270000
0.400000
50%     112.000000     0.050150      0.177900         0.450000
0.700000
75%     174.000000     0.131125      0.480000         0.820000
1.300000
max     360.000000    41.439000      2.260000        19.180000
365.300000

          magError       magNst
count  8270.000000  8295.000000
mean      0.235722    15.859795
std       0.309579    28.314644
min       0.000000     0.000000
25%       0.120000     5.000000
50%       0.179286     9.000000
75%       0.250000    17.000000
max       5.530000   814.000000
```

```
df.info()
```

```
<class'pandas.core.frame.DataFrame'>
RangeIndex:11623entries, 0to 11622
Data columns(total22 columns):
 #   Column           Non-NullCount   Dtype
---  -------          --------------  ------
 0   time             11623non-null   object
 1   latitude         11623non-null   float64
 2   longitude        11623non-null   float64
 3   depth            11623non-null   float64
 4   mag              11623non-null   float64
 5   magType          11623non-null   object
 6   nst              8298non-null    float64
 7   gap              8298non-null    float64
 8   dmin             5930non-null    float64
 9   rms              11623non-null   float64
 10  net              11623non-null   object
 11  id               11623non-null   object
 12  updated          11623non-null   object
 13  place            11623non-null   object
 14  type             11623non-null   object
 15  horizontalError7724non-null      float64
 16  depthError       11623non-null   float64
```

```
 17  magError          8270non-null    float64
 18  magNst            8295non-null    float64
 19  status            11623non-nullobject
 20  locationSource    11623non-nullobject
 21  magSource         11623non-nullobject
dtypes: float64(12), object(10)
memoryusage:2.0+MB
```

Wecanseelotsofnullvaluesofcertainfeatures,butaspartofdatawranglingandfeature engineering we consider only certain features in final dataframe, hence I choose simply drop or ignore the null values

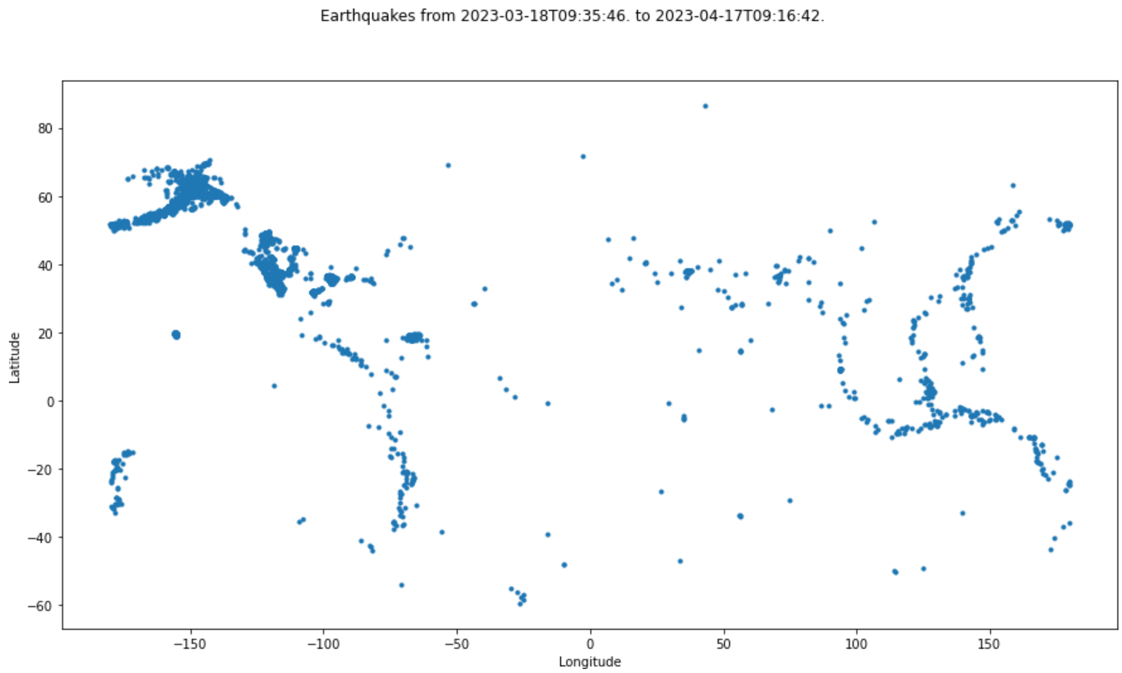
```
df.isnull().sum()
```

```
time                  0
latitude              0
longitude             0
depth                 0
mag                   0
magType               0
nst                3325
gap                3325
dmin               5693
rms                   0
net                   0
id                    0
updated               0
place                 0
type                  0
horizontalError    3899
depthError            0
magError           3353
magNst             3328
status                0
locationSource        0
magSource             0
dtype: int64
```

Visualizelatitudeandlongitudefeaturefrom'df'dataframetoseewherethepointsfall from the feature set

```
rounding_factor=10
fig,ax = plt.subplots(figsize=(15,8))

#latitude andlongitudeofearthquake siteoftop10500 samples.
plt.plot(np.round(df['longitude'].head(10500),rounding_factor),
         np.round(df['latitude'].head(10500),rounding_factor),
         linestyle='none', marker='.')

plt.suptitle('Earthquakes from '+ str(np.min(df['time']))[:20]+ 'to '+
str(np.max(df['time']))[:20])
```

```
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



Cleanupthedatabyfirstextractingonlydatefrom'time'columnby considering string[: 10]. hence we can get desired date

```
df=df.sort_values('time',ascending=True)

#Dateextraction
df['date']=df['time'].str[0:10]
df.head()
```

```
                          time   latitude   longitude  depth   mag
magType  \
11622  2023-03-18T09:35:46.450Z  19.245500 -155.126333  38.66  1.74
md
11621  2023-03-18T09:42:20.490Z  34.257667 -117.050333   3.10  1.20
ml
11620  2023-03-18T09:47:10.030Z  51.870167 -177.987833   6.08  1.17
ml
11619  2023-03-18T10:03:24.386Z  64.479800 -149.373600  15.30  1.30
ml
11618  2023-03-18T10:05:20.416Z  -7.071200  155.601000  10.00  4.70
mb

        nst    gap    dmin   rms  ...
place  \
11622  37.0  200.0     NaN  0.13  ...         24 km Sof  Fern Forest,
```

```
Hawaii
1162156.0      54.0  0.1074  0.16  ...         8kmNE of Running
Springs,CA
11620   4.0  141.0     NaN  0.12  ...               93 kmW ofAdak,
Alaska
11619   NaN    NaN     NaN  1.07...              16 km SWof Nenana,
Alaska
11618  50.0   70.0  4.4590  0.66...84kmS ofPanguna,PapuaNew
Guinea

             typehorizontalErrordepthErrormagErrormagNst
status  \
11622  earthquake           0.83      0.480  0.168011    17.0
reviewed
11621earthquake             0.14      0.510  0.118000    28.0
reviewed
11620earthquake             0.49      1.060  0.125191     4.0
reviewed
11619earthquake              NaN      0.200       NaN     NaN
reviewed
11618earthquake             8.84      1.883  0.081000    53.0
reviewed

       locationSource  magSource        date
11622              hv         hv  2023-03-18
11621              ci         ci  2023-03-18
11620              av         av  2023-03-18
11619              ak         ak  2023-03-18
11618              us         us  2023-03-18

[5rows x23 columns]
```

Datacleaningforseperating'place'column.henceonlyconsidercitybyseperatingstring by ', '

```
#only keepthecolumnsneeded
df=df[['date','latitude','longitude','depth','mag','place']]
#df['date']=df['time'].str.split(',',expand=True)
newdf=df['place'].str.split(',',expand=True)

newdf.head()
```

```
                                0                 1     2
11622        24km Sof Fern Forest            Hawaii  None
11621       8kmNEofRunningSprings                CA  None
11620                 93km WofAdak           Alaska  None
11619            16 km SWofNenana            Alaska  None
11618            84 km SofPanguna    PapuaNewGuinea  None
```

```
df['place']=newdf[1]
df=df[['date','latitude','longitude','depth','mag','place']]
```

```
df.head()
```

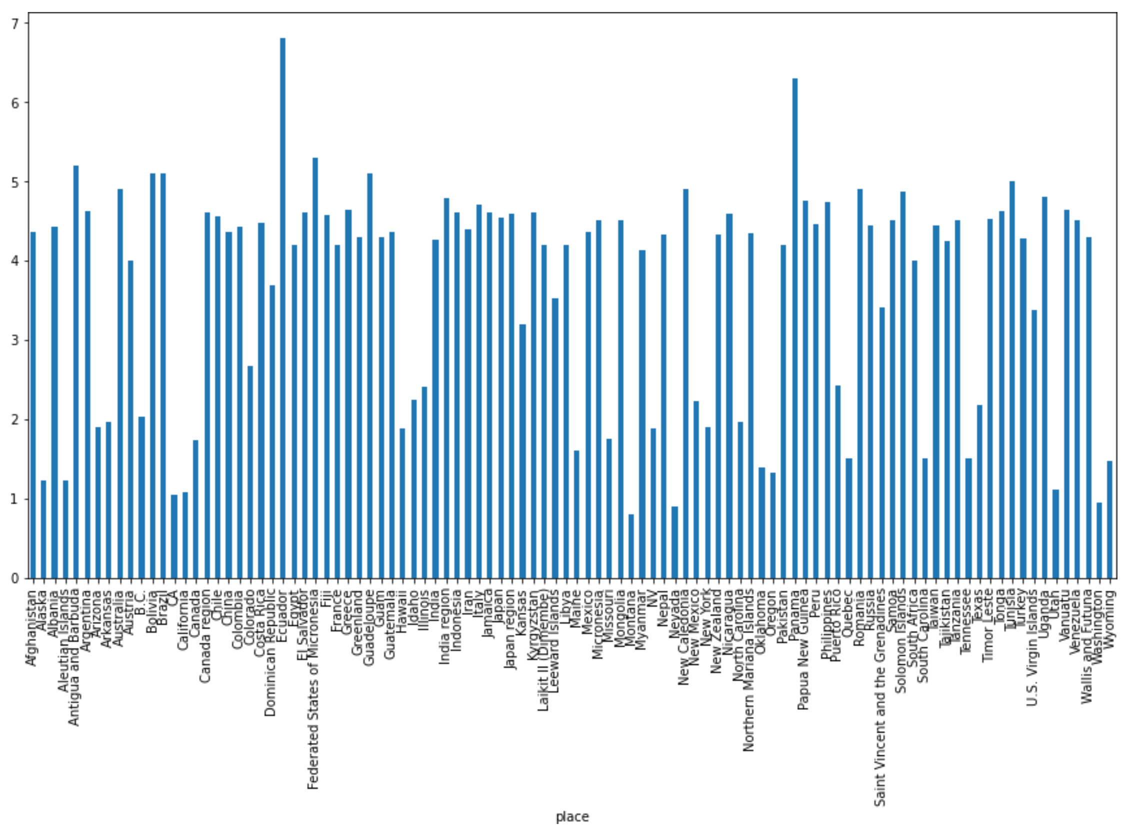
```
                date    latitude    longitude  depth    mag
place
11622   2023-03-18  19.245500 -155.126333  38.66   1.74
Hawaii
11621   2023-03-18  34.257667 -117.050333   3.10   1.20
CA
11620   2023-03-18  51.870167 -177.987833   6.08   1.17
Alaska
11619   2023-03-18  64.479800 -149.373600  15.30   1.30
Alaska
11618   2023-03-18  -7.071200  155.601000  10.00   4.70  Papua New
Guinea
```

```
print('totallocations:',len(set(df['place'])))
```

```
total locations: 101
```

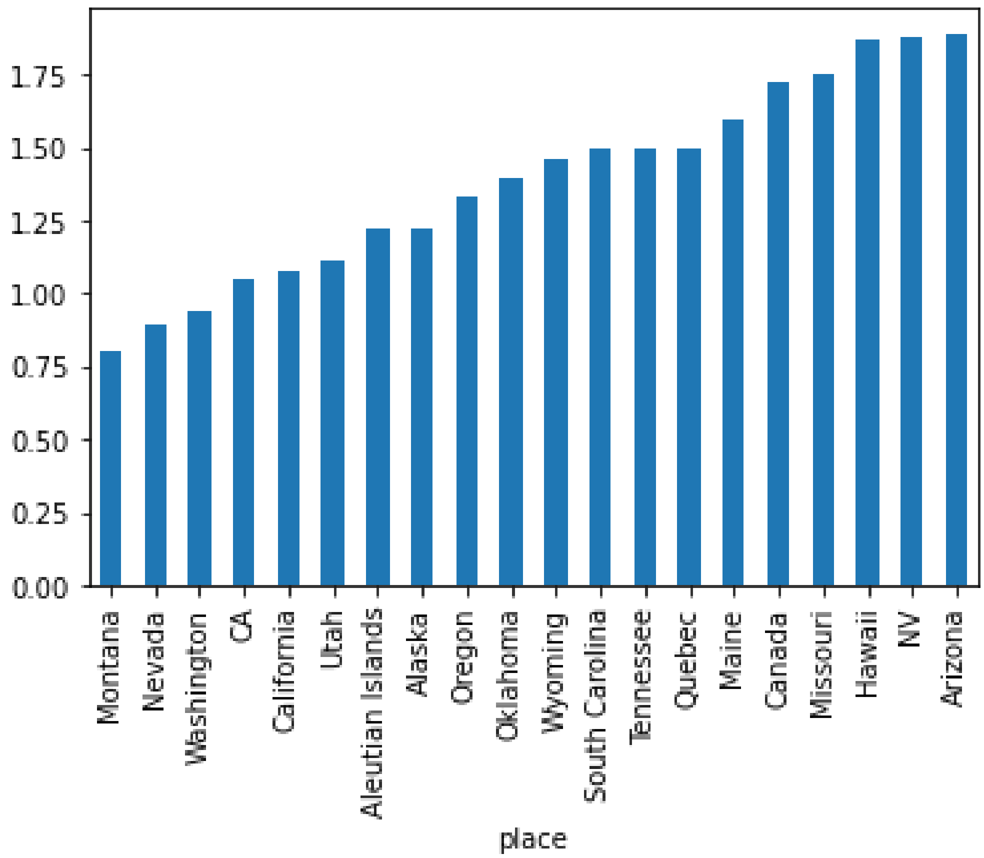
Barplotofmeanmagnitudevsplace,aswecanseefromthegraph,onlyfewcountriesare consideredasepicenterordangeroussincetheyhavemagnitudemorethan2.8(Ihave considered here)

```
df.groupby(['place'])['mag'].mean().plot(kind='bar',figsize=(15,8));
```



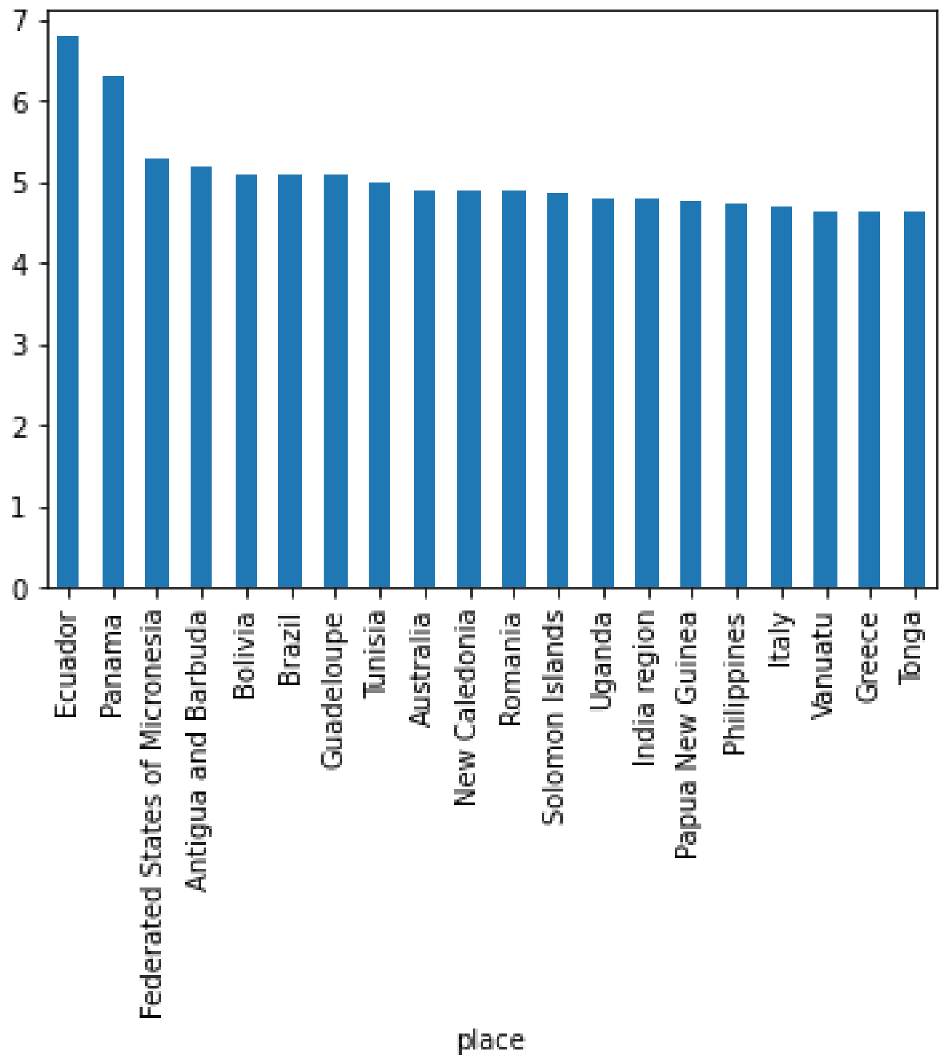

```
df.groupby(['place'])['mag'].mean().nsmallest(20).plot(kind='bar')
```

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




```
df.groupby(['place'])['mag'].mean().nlargest(20).plot(kind='bar')
```

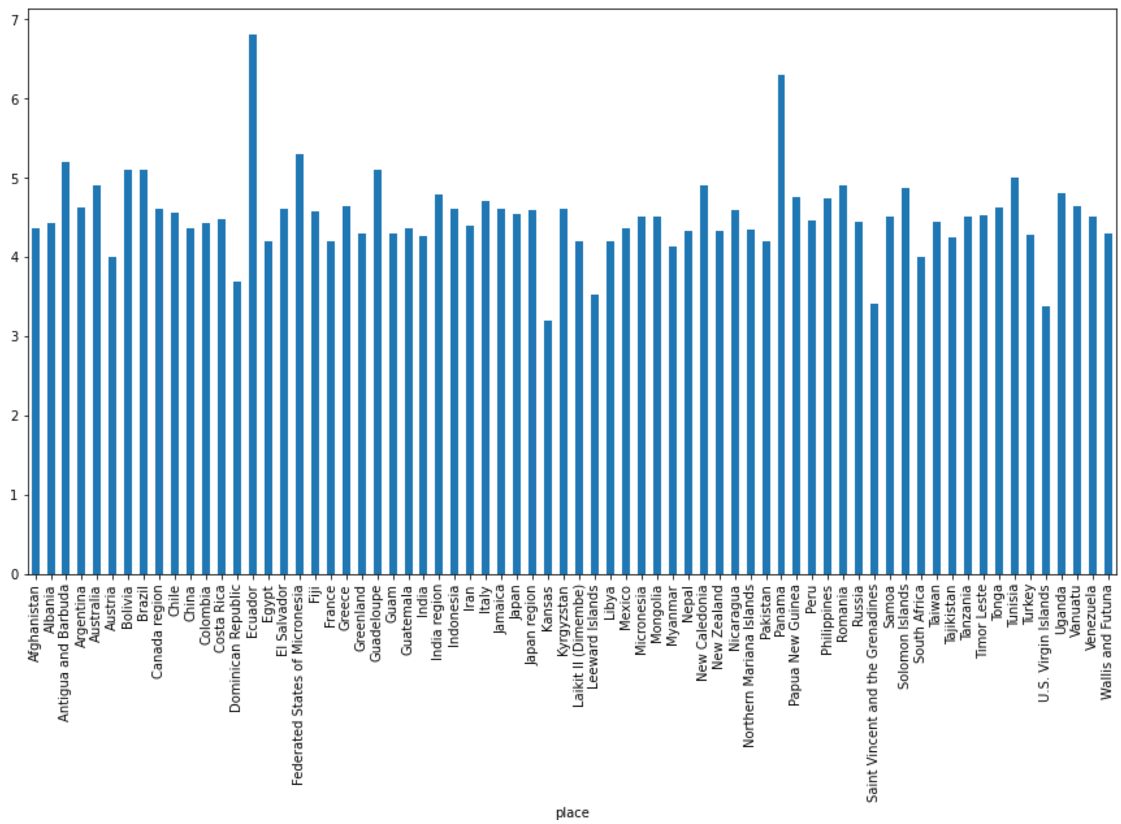
```
<AxesSubplot:xlabel='place'>
```




```
more_dangerous_places=df.groupby('place')['mag'].mean()
more_dangerous_places=more_dangerous_places[more_dangerous_places>3]
```

Letsconsider3asthresholdforhowhightheearthquakehashitand lets visualise countries with more than 3 magnitude.

```
more_dangerous_places.plot(kind='bar',figsize=(15,8));
```




```
#calculatemeanlatitudeandlongitudeforsimplified locations

df_coords=df[['place','latitude','longitude']]
df_coords=df_coords.groupby(['place'],as_index=False).mean() df_coords
= df_coords[['place', 'latitude', 'longitude']]

df_coords.head()
```

```
                  place    latitude    longitude
0           Afghanistan36.233878      70.791383
1                 Alaska58.289146-156.708497
2                Albania40.430200      20.677967
3       AleutianIslands52.337853-145.509081
4AntiguaandBarbuda17.800200-61.508400
```

Mergethetwodataframesofmeanlatitudeandlongitudelocationscalculatedabovewith dataframe only considering ['date' , 'depth', 'mag', 'place'] as columns out of total feature

```
df=df[['date','depth','mag','place']]
df = pd.merge(left=df, right=df_coords, how='inner',on=['place'])
print(df.head())

print('totallocations:',len(set(df['place'])))
```

```
         date       depth    mag    placelatitude      longitude
          02023-03-1838.6600001.74Hawaii19.28176 -155.400159
```

```
12023-03-18     1.3100000.50Hawaii19.28176-155.400159
22023-03-1832.8899991.98Hawaii19.28176 -155.400159
32023-03-1833.9100002.02Hawaii19.28176 -155.400159
42023-03-1830.6399991.82Hawaii19.28176 -155.400159
total locations: 100
```

```
print(set(df['place']))
```

```
{'New Caledonia', 'Quebec', 'Myanmar', 'China', 'Missouri',
'Colorado', 'Venezuela', 'Northern Mariana Islands', 'Micronesia',
'Montana', 'Tajikistan', 'Uganda', 'California', 'Papua New Guinea',
'South Carolina', 'Dominican Republic', 'Arizona', 'B.C.', 'Brazil',
'France', 'Laikit II (Dimembe)', 'Chile', 'Philippines', 'Tonga',
'Afghanistan', 'Idaho', 'Mongolia', 'Nevada', 'Wyoming', 'Maine',
'Albania', 'Wallis and Futuna', 'Australia', 'Taiwan', 'CA',
'Vanuatu', 'North Carolina', 'Ecuador', 'New Mexico', 'Guadeloupe',
'Oklahoma', 'NV', 'Greece', 'Austria', 'Utah', 'Peru', 'Aleutian
Islands', 'Federated States of Micronesia', 'Guam', 'Timor Leste',
'Alaska', 'Antigua and Barbuda', 'Russia', 'Nepal', 'Turkey', 'Fiji',
'Kansas', 'Mexico', 'Hawaii', 'U.S. Virgin Islands', 'Greenland',
'India region', 'Canada region', 'Panama', 'Oregon', 'New Zealand',
'Illinois', 'Tennessee', 'Puerto Rico', 'Argentina', 'Japan region',
'Costa Rica', 'Saint Vincent and the Grenadines', 'Romania',
'Washington', 'Tunisia', 'Bolivia', 'Italy', 'Samoa', 'Egypt',
'Solomon Islands', 'Leeward Islands', 'Nicaragua', 'Japan', 'Libya',
'Guatemala','Jamaica','Canada','Indonesia','Pakistan','NewYork',
'Kyrgyzstan', 'El Salvador', 'Tanzania', 'India', 'Colombia', 'South
Africa', 'Arkansas', 'Iran', 'Texas'}
```

```
df.head()
```

```
         date       depth    mag   placelatitude     longitude
02023-03-1838.6600001.74Hawaii19.28176 -155.400159
12023-03-18     1.3100000.50Hawaii19.28176-155.400159
22023-03-1832.8899991.98Hawaii19.28176 -155.400159
32023-03-1833.9100002.02Hawaii19.28176 -155.400159
42023-03-1830.6399991.82Hawaii19.28176 -155.400159
```

FeatureEngineeringandDatawrangling

- Setrollingwindowsizeforfuturepredictionbasedonpastvalueswithfixed window size in past
- Wehavecreated6newfeaturesbasedonrollingwindowsizeonaveragedepthand average magnitude.
- Afinaloutcome'mag_outcome'hasbeendefinedastargetvaluesandtheoutputis considered as shifted values from set rolling window of past days eg: '7'.

```
eq_tmp =df.copy()

#rollingwindowsize
```

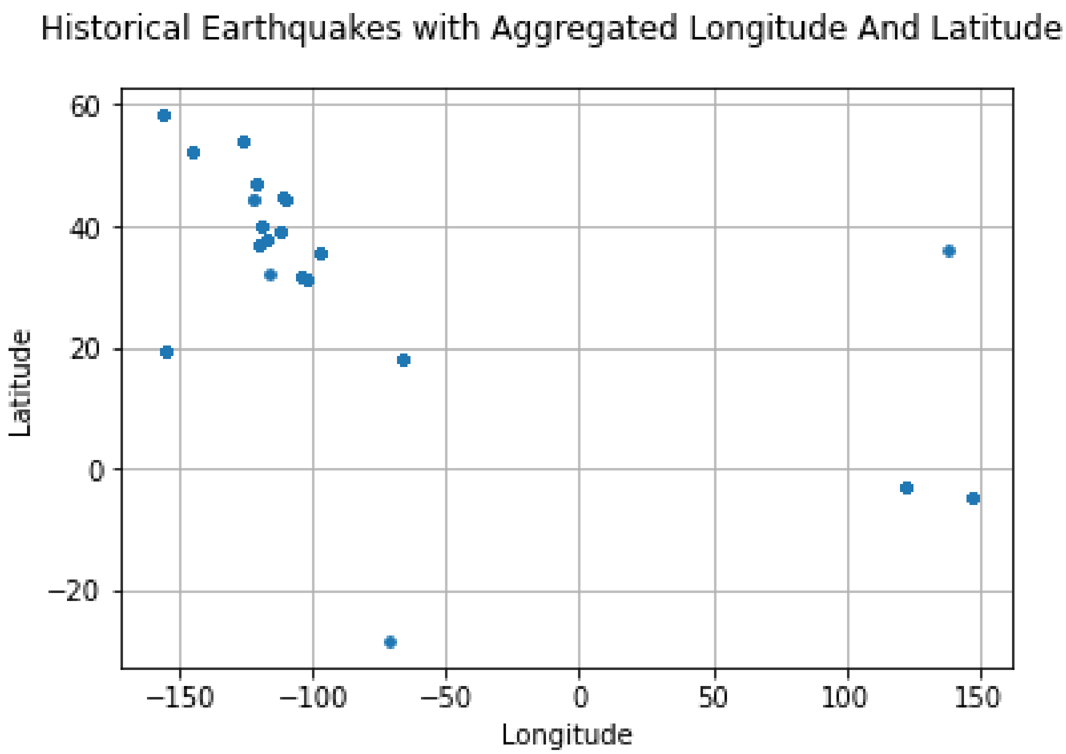
```
DAYS_OUT_TO_PREDICT=7

#loop through eachzoneandapply MA
eq_data = []
eq_data_last_days_out=[]

forplaceinlist(set(eq_tmp['place'])):
    temp_df=eq_tmp[eq_tmp['place']==place].copy()

    #avg.depthof22days rollingperiodandsoon..
    temp_df['depth_avg_22'] =
temp_df['depth'].rolling(window=22,center=False).mean()
    temp_df['depth_avg_15'] =
temp_df['depth'].rolling(window=15,center=False).mean()
    temp_df['depth_avg_7'] =
temp_df['depth'].rolling(window=7,center=False).mean()
    temp_df['mag_avg_22'] =
temp_df['mag'].rolling(window=22,center=False).mean()
    temp_df['mag_avg_15'] =
temp_df['mag'].rolling(window=15,center=False).mean()
    temp_df['mag_avg_7'] =
temp_df['mag'].rolling(window=7,center=False).mean()
    temp_df.loc[:,'mag_outcome']=temp_df.loc[:,
'mag_avg_7'].shift(DAYS_OUT_TO_PREDICT * -1)

    #daystopredict value onearth quakedata thisisnotyetseenor
witnessed by next 7 days (consider as live next 7 days period)

    eq_data_last_days_out.append(temp_df.tail(DAYS_OUT_TO_PREDICT))

    eq_data.append(temp_df)

#concatalllocation-baseddataframesintomasterdataframe
eq_all=pd.concat(eq_data)

eq_all.head()

             date   depthmag          place   latitude
 longitude\
10859  2023-04-05  58.598  4.9  New Caledonia -21.374900  169.731900

10823  2023-03-28   5.824  1.5         Quebec  47.581100  -70.278300

10783  2023-03-25  11.095  4.2        Myanmar  21.960175   95.595775

10784  2023-03-28  10.000  3.5        Myanmar  21.960175   95.595775

10785  2023-04-08  10.000  4.2        Myanmar  21.960175   95.595775
```

```
       depth_avg_22  depth_avg_15  depth_avg_7  mag_avg_22  mag_avg_15  \
10859           NaN           NaN          NaN         NaN         NaN

10823           NaN           NaN          NaN         NaN         NaN

10783           NaN           NaN          NaN         NaN         NaN

10784           NaN           NaN          NaN         NaN         NaN

10785           NaN           NaN          NaN         NaN         NaN


       mag_avg_7  mag_outcome
10859        NaN          NaN
10823        NaN          NaN
10783        NaN          NaN
10784        NaN          NaN
10785        NaN          NaN
```

locationafterfeatureengineering

```
#removeanyNaNfields
eq_all=eq_all[np.isfinite(eq_all['depth_avg_22'])]
eq_all = eq_all[np.isfinite(eq_all['mag_avg_22'])]
eq_all = eq_all[np.isfinite(eq_all['mag_outcome'])]

eq_all.head()

             datedepth     mag    place   latitude   longitude
depth_avg_22\
85912023-03-28     4.730.93Montana44.638252-110.829041
7.018182
85922023-03-29     4.850.94Montana44.638252-110.829041
6.919545
85932023-03-29     4.700.65Montana44.638252-110.829041
6.785000
85942023-03-29     5.003.00Montana44.638252-110.829041
6.520909
85952023-03-29     1.850.53Montana44.638252-110.829041
6.191818

      depth_avg_15  depth_avg_7  mag_avg_22  mag_avg_15  mag_avg_7  \
8591      6.548667     7.004286    0.880455    0.818667   0.758571
8592      6.527333     6.650000    0.884091    0.865333   0.914286
8593      6.322667     5.985714    0.864545    0.754667   0.758571
8594      6.364000     5.932857    0.930455    0.948000   1.170000
8595      6.086000     5.267143    0.937273    0.918000   1.175714

      mag_outcome
```

```
8591      0.918571
8592      0.825714
8593      0.812857
8594      0.578571
8595      0.547143

plt.plot(eq_all['longitude'],
         eq_all['latitude'],
         linestyle='none',marker='.')
plt.suptitle('HistoricalEarthquakeswithAggregatedLongitudeAnd
Latitude')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid()
plt.show()
```




```
#keepourlivedataforpredictions
eq_data_last_days_out=pd.concat(eq_data_last_days_out)


eq_data_last_days_out =
eq_data_last_days_out[np.isfinite(eq_data_last_days_out['mag_avg_22'])
]
predict_unknown=eq_data_last_days_out
```

```
#here 'mag_outcome' hasNaNbecause these arefutureoutcomeevent to be
predicted live or data that has not yet been witnessed predict_unknown

            date      depth   mag    place   latitude   longitude  \
8623  2023-04-09  17.850000  0.45  Montana  44.638252 -110.829041
8624  2023-04-10  14.020000  0.72  Montana  44.638252 -110.829041
8625  2023-04-10  18.480000  0.31  Montana  44.638252 -110.829041
8626  2023-04-11   8.690000  0.34  Montana  44.638252 -110.829041
8627  2023-04-11   4.960000 -0.17  Montana  44.638252 -110.829041
...          ...        ...   ...      ...        ...         ...
9464  2023-04-13   7.211304  2.10    Texas  31.255504 -102.801352
9465  2023-04-13   9.216528  2.30    Texas  31.255504 -102.801352
9466  2023-04-13   5.453410  3.00    Texas  31.255504 -102.801352
9467  2023-04-15   3.149438  3.10    Texas  31.255504 -102.801352
9468  2023-04-15   5.000000  3.00    Texas  31.255504 -102.801352

      depth_avg_22  depth_avg_15  depth_avg_7  mag_avg_22  mag_avg_15
\
8623     10.454091     11.691333    14.215714    0.753182    0.612667

8624     10.377727     11.944667    13.704286    0.771818    0.623333

8625     10.980000     12.489333    14.068571    0.734545    0.634000

8626     11.043182     12.466000    12.940000    0.661364    0.604000

8627     11.070455     12.218667    11.428571    0.599091    0.594667

...            ...           ...          ...         ...         ...

9464      7.428859      7.550656     7.336627    2.304545    2.046667

9465      7.498756      7.659693     7.696539    2.290909    2.060000

9466      7.358087      7.439668     7.540898    2.300000    2.146667

9467      7.126715      7.247715     6.975322    2.304545    2.206667

9468      7.081651      6.957186     6.547221    2.268182    2.246667


      mag_avg_7  mag_outcome
8623   0.650000          NaN
8624   0.682857          NaN
8625   0.671429          NaN
8626   0.572857          NaN
8627   0.510000          NaN
...         ...          ...
9464   2.071429          NaN
```

```
9465   2.114286          NaN
9466   2.257143          NaN
9467   2.342857          NaN
9468   2.500000          NaN

[158rowsx13 columns]
```

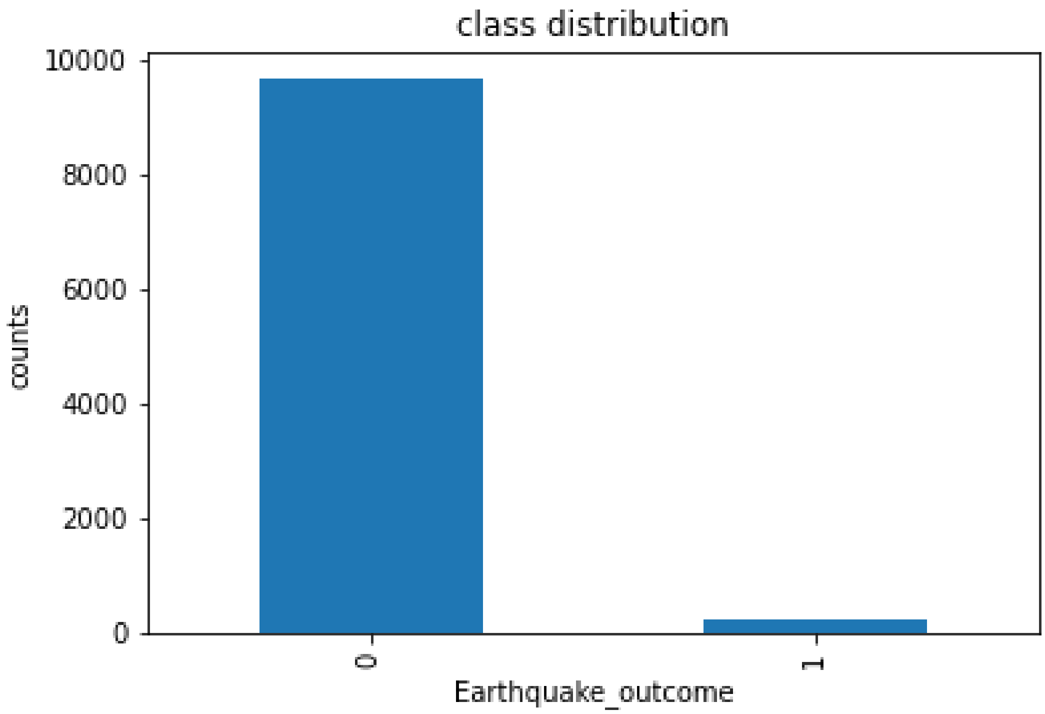
consideredmagnitudeabove2.5asdangeroushencepredictionoutcomeas'1'elso'0'.

```
eq_all['mag_outcome']= np.where(eq_all['mag_outcome']>2.5, 1,0)
print(eq_all['mag_outcome'].describe())
eq_all['mag_outcome'].value_counts()
```

```
count    9842.000000
mean        0.020931
std         0.143160
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name:mag_outcome,dtype:float64
```

```
0    9636
1     206
Name:mag_outcome,dtype:int64
```

```
eq_all['mag_outcome'].value_counts().plot(kind='bar',)
plt.xlabel('Earthquake_outcome')
plt.ylabel('counts')
plt.title('classdistribution');
```



Savethedataofoffixedrollingwindowandliveunknownprediction data in sql database using sql engine

```
fromsqlalchemyimportcreate_engine
engine=create_engine('sqlite:///Earthquakedata.db')
eq_all.to_sql('Earthquake_features', engine,
index=False,if_exists='replace')

9842

engine=create_engine('sqlite:///Earthquakedata_predict.db')
predict_unknown.to_sql('Earthquake_predict', engine,
index=False,if_exists='replace')

158
```

Loadtrainingandpredictionwindowdatafromsavedsqldatabase

```
engine=create_engine('sqlite:///Earthquakedata.db')
df_features=pd.read_sql_table('Earthquake_features',con=engine)

df_features.head()

         datedepth     mag    place    latitude    longitude
depth_avg_22\
02023-03-28     4.730.93Montana44.638252 -110.829041
7.018182
12023-03-29     4.850.94Montana44.638252 -110.829041
```

```
6.919545
22023-03-29     4.70  0.65  Montana  44.638252 -110.829041
6.785000
32023-03-29     5.00  3.00  Montana  44.638252 -110.829041
6.520909
42023-03-29     1.85  0.53  Montana  44.638252 -110.829041
6.191818

   depth_avg_15  depth_avg_7  mag_avg_22  mag_avg_15  mag_avg_7
mag_outcome
0      6.548667     7.004286    0.880455    0.818667   0.758571
0
1      6.527333     6.650000    0.884091    0.865333   0.914286
0
2      6.322667     5.985714    0.864545    0.754667   0.758571
0
3      6.364000     5.932857    0.930455    0.948000   1.170000
0
4      6.086000     5.267143    0.937273    0.918000   1.175714
0
```

```python
engine =create_engine('sqlite:///Earthquakedata_predict.db')
df_predict=pd.read_sql_table('Earthquake_predict',con=engine)

#Live datatobepredicted onafterbeing trained ofrollingperiod for next
7 days.
#HenceNaNoutcome thathastobepredicted

df_predict.head()
```

```
          date  depth   mag    place   latitude   longitude
depth_avg_22  \
02023-04-09    17.85  0.45  Montana  44.638252 -110.829041
10.454091
12023-04-10    14.02  0.72  Montana  44.638252 -110.829041
10.377727
22023-04-10    18.48  0.31  Montana  44.638252 -110.829041
10.980000
32023-04-11     8.69  0.34  Montana  44.638252 -110.829041
11.043182
42023-04-11     4.96 -0.17  Montana  44.638252 -110.829041
11.070455

   depth_avg_15  depth_avg_7  mag_avg_22  mag_avg_15  mag_avg_7
mag_outcome
0     11.691333    14.215714    0.753182    0.612667   0.650000
NaN
1     11.944667    13.704286    0.771818    0.623333   0.682857
NaN
2     12.489333    14.068571    0.734545    0.634000   0.671429
```

```
NaN
3      12.466000     12.940000     0.661364     0.604000    0.572857
NaN
4      12.218667     11.428571     0.599091     0.594667    0.510000
NaN
```

Trainingisdonebyconsidering22,15,7dayswindowpastfeatures rolling average and outcome data is shifted to next 7 days as prediction.

```
df_predict.columns

Index(['date', 'depth', 'mag', 'place', 'latitude', 'longitude',
       'depth_avg_22','depth_avg_15','depth_avg_7','mag_avg_22',
       'mag_avg_15', 'mag_avg_7', 'mag_outcome'],
      dtype='object')

df.columns

Index(['date','depth','mag','place','latitude','longitude'],
dtype='object')

fromsklearn.model_selectionimporttrain_test_split

#Selection offeatures thatareneeded forprediction andhence
consideronlythemrestarejustignoredforpredictionpurpose.

features =[fforfinlist(df_features) ifnot in['date', 'lon_box_mean',
 'lat_box_mean','mag_outcome','mag','place',
 'combo_box_mean','latitude',
 'longitude']]

#splitting traing andtesting dataset withtraingingsize =70%and test =
30%
X_train, X_test, y_train, y_test =
train_test_split(df_features[features],
                     df_features['mag_outcome'],test_size=0.3,
random_state=42)

features

['depth',
 'depth_avg_22',
 'depth_avg_15',
 'depth_avg_7',
 'mag_avg_22',
 'mag_avg_15',
 'mag_avg_7']
```

Trainingphase

- Modelsusedare:
  - AdaboostclassifierwithDecisionTree
  - AdaboostclassifierwithRandomForest
  - GridSearchCVashyperparametertunning
- ModelusedforDeploymentofapplication:
  - Xgboostwithparameterssetfromabovemodels

AdaboostDecisionTreeClassifier

```
fromsklearn.ensembleimportRandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

param_grid={
              "base_estimator__max_depth":     [2,5,7],
              "n_estimators":[200,400,600]
             }

#baseestimator
tree=DecisionTreeClassifier()

#adaboostwiththetreeasbaseestimator
#learningrateisarbitrarilysetto0.6,  ABC  =
AdaBoostClassifier(
    base_estimator=tree,
    learning_rate=0.6,
    algorithm="SAMME")
```

ParametertunningwithGridSearchCV

```
#rungridsearch
grid_search_ABC=GridSearchCV(ABC,
                               param_grid=param_grid,
                               scoring = 'roc_auc',
                               return_train_score=True,

                               verbose=1)

grid_search_ABC.fit(X_train,y_train)

Fitting 5 folds for each of 9 candidates, totalling 45 fits

GridSearchCV(estimator=AdaBoostClassifier(algorithm='SAMME',

base_estimator=DecisionTreeClassifier(),
                                          learning_rate=0.6),
             param_grid={'base_estimator__max_depth':[2,5,7],
```

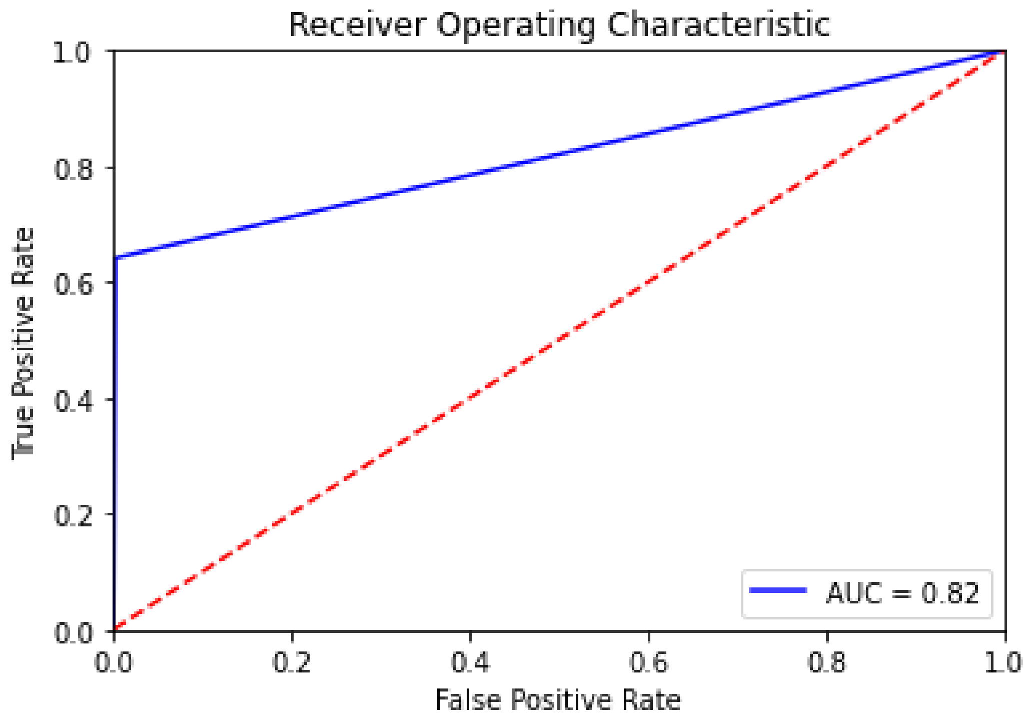
```
                    'n_estimators': [200, 400, 600]},
            return_train_score=True,scoring='roc_auc',verbose=1)

pred_ABC=grid_search_ABC.predict(X_test)
```

EvaluationAreaUndercurve&ROC

IhavechoseROC_AUCscoreasevaluationmetricsinceIhavetobinaryclassififywether earthquake happened or not with given features that has been train on past few days window rolling average window.

- Withadaboostdecisiontreeclassifierandhyperparametertunning,wegetarea under curve (score) = 0.8867
- highertheaucscore,betteristhemodelsinceitisbetteratdistinguishingpostive and negative classes.
- Makeanoteherethatwegetfromconfusionmatrix,Falsenegative=42andRecall score =0.7789. We need this value apart from auc score that we will analyze later when we have tested with diffferent models below

```
fromsklearn.metricsimportroc_curve
fromsklearn.metricsimportauc
from sklearn.metrics import roc_auc_score
from sklearn.metrics import recall_score
fromsklearn.metricsimportconfusion_matrix

print(roc_auc_score(y_test,pred_ABC))

fpr,tpr,_=roc_curve(y_test,pred_ABC) roc_auc
= auc(fpr, tpr)
print('AUC:',np.round(roc_auc,4))

plt.title('Receiver Operating Characteristic')
plt.plot(fpr,tpr, 'b',label ='AUC=%0.2f'%roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0, 1])
plt.ylabel('TruePositiveRate')
plt.xlabel('FalsePositiveRate')
plt.show()


print("ConfusionMatrix:\n",confusion_matrix(y_test,pred_ABC))
print("\nRecall 'TP/TP+FN' = ", recall_score(y_test,pred_ABC))

0.8195478204294079
AUC: 0.8195
```




```
ConfusionMatrix:
 [[2893    7]
 [19      34]]

Recall'TP/TP+FN'=0.6415094339622641
```

AdaboostRandomForestClassifier

```
fromsklearn.datasetsimportmake_classification
fromsklearn.ensembleimportRandomForestClassifier


rfc=RandomForestClassifier(n_jobs=-1,max_features=
'sqrt',n_estimators=50, oob_score = True)

param_grid={
    'n_estimators': [200, 700],
    'max_features':['auto','sqrt','log2']
}
#GridSearch ofparameter tunning.
CV_rfc =GridSearchCV(estimator=rfc,param_grid=param_grid, cv=5)
CV_rfc.fit(X_train, y_train)

GridSearchCV(cv=5,
             estimator=RandomForestClassifier(max_features='sqrt',
                                              n_estimators=50,
n_jobs=-1,
```

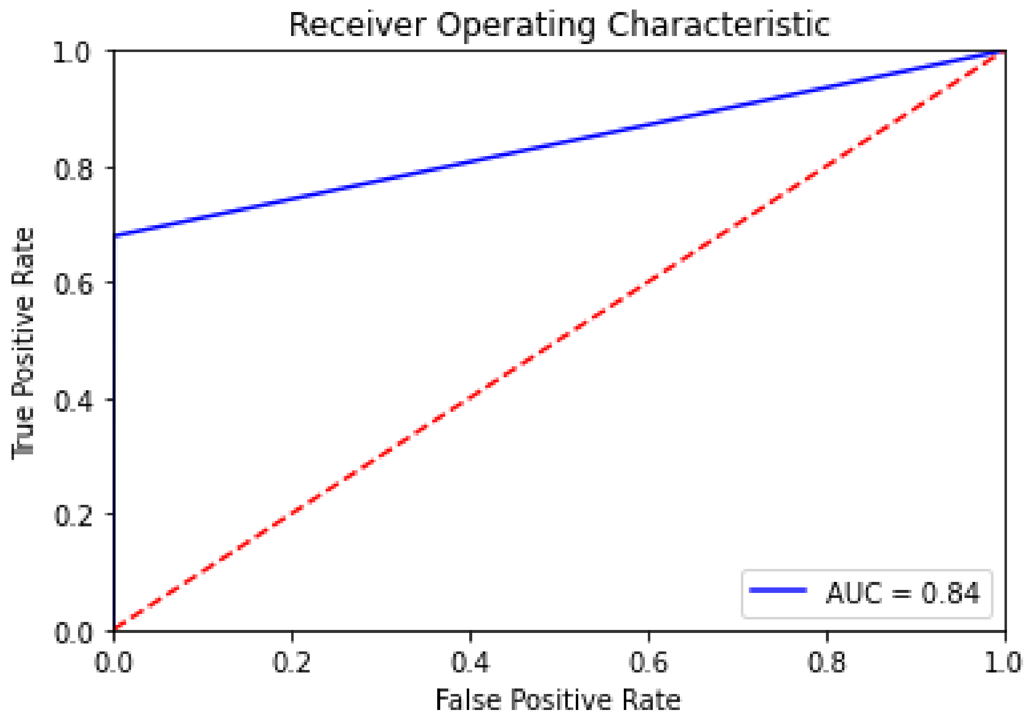
```
                                                oob_score=True),
             param_grid={'max_features':['auto','sqrt','log2'],
                         'n_estimators':[200,700]})

pred=CV_rfc.predict(X_test)
```

EvaluationAreaUndercurve&ROC

- BelowistheaucscoreforadaboostRandomForestclassifierwith0.916whichis slightly lower than Decision tree classifier
- Moreoverwhenwelookatconfusionmatrix,FalseNegative=38and`Recallscore= 0.8'canbeobservedwhichisslightlyhigherthanrecallscoreofdecisiontree.Thus performs better than decision tree adabooost

```
print(roc_auc_score(y_test,pred))

fpr,tpr,_=roc_curve(y_test,pred)
roc_auc=auc(fpr,tpr)print('AUC:',
np.round(roc_auc,4))

plt.title('Receiver Operating Characteristic')
plt.plot(fpr,tpr, 'b',label ='AUC=%0.2f'%roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0, 1])
plt.ylabel('TruePositiveRate')
plt.xlabel('FalsePositiveRate')
plt.show()


print("ConfusionMatrix:\n",confusion_matrix(y_test,pred))
print("\nRecall 'TP/TP+FN' = ", recall_score(y_test,pred))
```

```
0.8394502277163305
AUC: 0.8395
```




```
ConfusionMatrix:
 [[2899    1]
 [17      36]]

Recall'TP/TP+FN'=0.6792452830188679
```

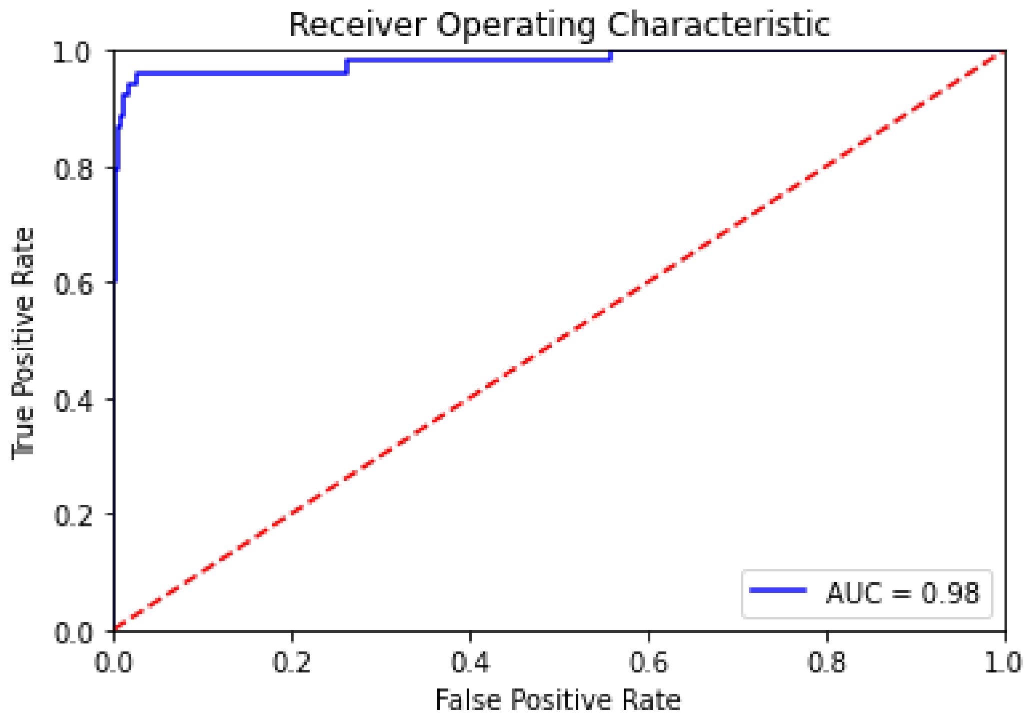
XGBoost

- Wehavealsotestedwithxgboostmodelbelowwithsimilarparametersas Igot above, since grid search CV was taking lot of time for xgboost.
- As we can see this significantly gives higher AUC score of almost 0.0.98 and also Falsenegative=37whichissimilarRandomForestadaboostbutxgboosthashigher True positive and less False Positve compared to Random forest adaboost. i.e Recall score = 0.805 which is similar adaboost Random Forrest tree. But XGboost is really good at classifying positive and negative classes and also better aur_roc_score = 0.98193.

```
fromsklearn.metrics importroc_curve,auc

dtrain=xgb.DMatrix(X_train[features],label=y_train)
dtest = xgb.DMatrix(X_test[features], label=y_test)
from xgboost import XGBClassifier
importmatplotlib.pyplotasplt

param ={
        'objective':'binary:logistic',
        'booster': 'gbtree',
        'eval_metric': 'auc',
```

```
        'max_depth':6,#themaximumdepthofeachtree
        'eta':0.003,#thetrainingstepforeachiteration
        'silent':1}#loggingmode -quiet}#thenumberofclasses that exist
in this datset
num_round =5000#thenumber oftraining iterations
bst=xgb.train(param,dtrain,num_round) preds
= bst.predict(dtest)

print (roc_auc_score(y_test, preds))
fpr,tpr,_=roc_curve(y_test,preds)
roc_auc = auc(fpr, tpr)
print('AUC:',np.round(roc_auc,4))

ypred_bst =
np.array(bst.predict(dtest,ntree_limit=bst.best_iteration))
ypred_bst= ypred_bst >0.5
ypred_bst=ypred_bst.astype(int)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr,tpr, 'b',label ='AUC=%0.2f'%roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0, 1])
plt.ylabel('TruePositiveRate')
plt.xlabel('FalsePositiveRate')
plt.show()
print("ConfusionMatrix:\n",confusion_matrix(y_test,ypred_bst))
print("\nRecall 'TP/TP+FN' = ", recall_score(y_test,ypred_bst))
```

```
[15:35:12]WARNING:C:\buildkite-agent\builds\buildkite-windows-cpu-
autoscaling-group-i-07593ffd91cd9da33-1\xgboost\xgboost-ci-windows\
src\learner.cc:767:
Parameters: {"silent" }arenotused.

0.9823227065712427
AUC: 0.9823

C:\Users\rupin\anaconda3\lib\site-packages\xgboost\core.py:122:
UserWarning:ntree_limitisdeprecated,use`iteration_range`ormodel slicing
instead.
  warnings.warn(
```




```
ConfusionMatrix:
 [[2896    4]
 [17      36]]

Recall'TP/TP+FN'=0.6792452830188679
```

We can see above that xgboost algorithm has higher auc score (0.9819) than adaboost decisiontreeandrandomforest,asitisevidentfromtheROCcurve.Henceweconsider xgboost for prediction of live data

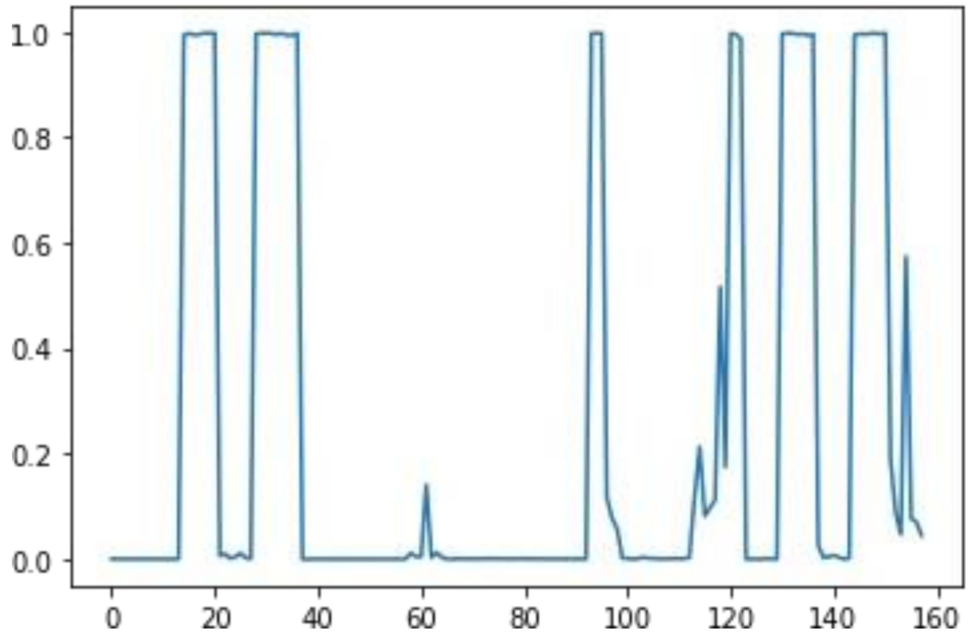
Finalthoughtsonwhichmodeltoconsideronapplication:

- OurmainAimistopredictwetherearthquakewillhappenornotatagivendayand place. So we definitely would not like the model with higher False Neagtive values , since its more dangerous to predict as no earthquake while in reality earthquake happendthanpredictingearthquakewillhappengiveninrealityitdidnot.Sinceits better safe than sorry!!, we can allow False positive more than False negative

- After seeing these comparision on auc_roc score, confusion matrix, and recall score, since all the above algorithm have given similar result with slightly different recall scores, Xgboost with FN=37 but with higher auc_score 0f 0.98 performs over-all better.Henceforwebapplicationdeployment,IhavechosenXgboostasitalsofaster than adaboost

Preparingpredictionandplotforliveunknowndatawegotindf_predictwith mag_outcome = Nan

```
dlive=xgb.DMatrix(df_predict[features])#,label=[])
preds =bst.predict(dlive)

plt.plot(preds)
plt.show()
```



Prediction

- Selectspecificfeaturessuchasdata,place,long,latandgiveearthquakeprobablity from prediction at that place and date as quake probability
- withtakingonly7daysrollingperioddatafrompredictdataframesincethis outcome value is NaN and we need to predict next 7 days period.

```
#df_p=df[['date','place','latitude','longitude']]
#df=pd.read_csv('all_month.csv')
#df=df[['place','latitude','longitude','date']]
live_set =df_predict[['date','place','latitude','longitude']]
live_set.loc[:,'quake'] = preds
#aggregate downdups
live_set=live_set.groupby(['date','place'],as_index=False).mean()

#increment datetoinclude DAYS_OUT_TO_PREDICT
live_set['date']= pd.to_datetime(live_set['date'],format='%Y-%m-%d')
live_set['date'] = live_set['date'] + pd.to_timedelta(7,unit='d')

live_set.tail()
```

```
C:\Users\rupin\AppData\Local\Temp\ipykernel_2284\1247293574.py:5:
SettingWithCopyWarning:
Avalueistrying tobeseton acopyofa slicefroma DataFrame.
```

```
Tryusing.loc[row_indexer,col_indexer]=value instead

Seethecaveatsinthedocumentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  live_set.loc[:,'quake']=preds
```

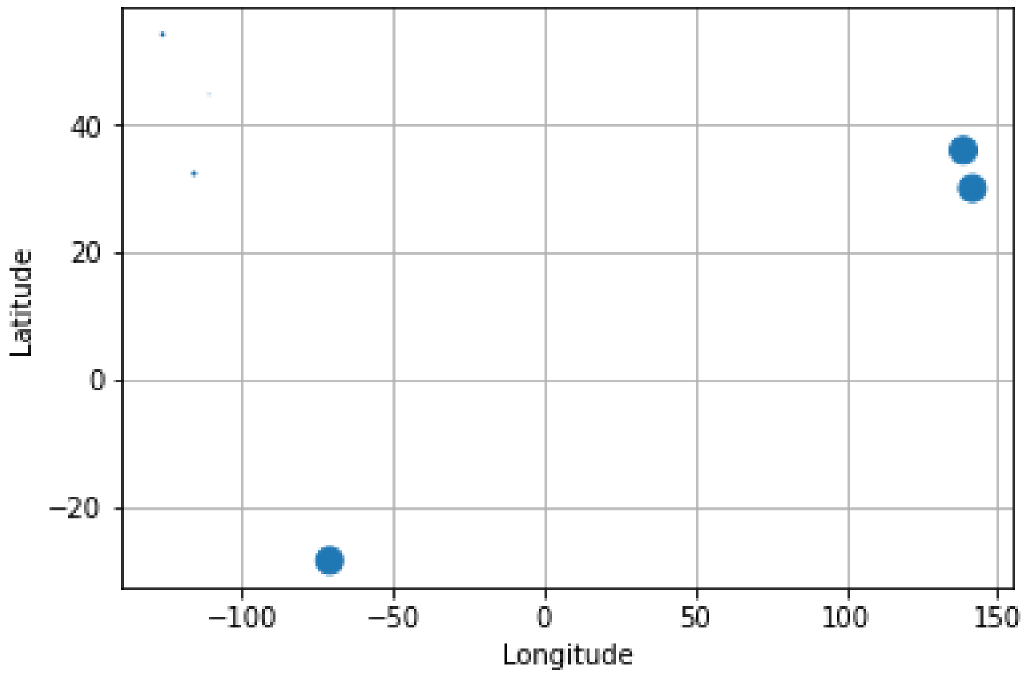
	date	place	latitude	longitude	quake
69	2023-04-24	CA	36.92967	-120.557775	0.000086
70	2023-04-24	Hawaii	19.28176	-155.400159	0.001590
71	2023-04-24	Indonesia	-2.94946	122.707901	0.996520
72	2023-04-24	Nevada	37.97952	-117.645207	0.000148
73	2023-04-24	Puerto Rico	18.06517	-66.839902	0.187457

```
importdatetimeasdt
#convertdate toproper format forprediction
days
=list(set([dfordinlive_set['date'].astype(str)ifd>dt.datetime.t
oday().strftime('%Y-%m-%d')]))
print(days.sort())

#PredictNaNoutcomevalue inearthquakefornext day1.
predict_day=days[2]
predict_day
```

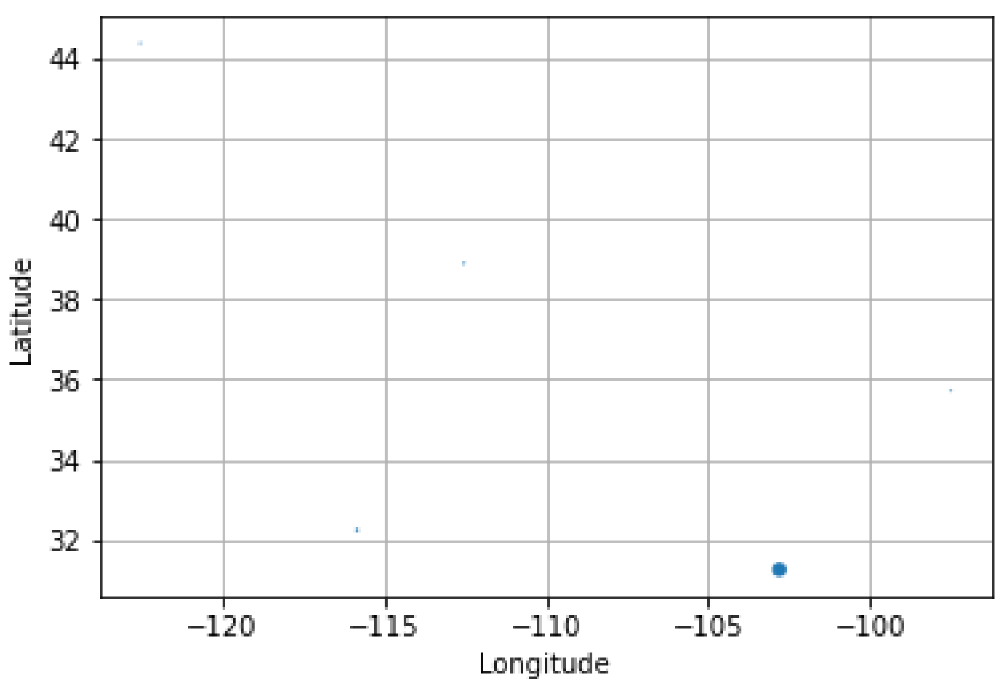
```
None

'2023-04-20'
```

```
#place, date, latandlongwithearthquake probability fornext7 days
for i in range(0,7):
    live_set_tmp = live_set[live_set['date'] == days[i]]
    plt.scatter(live_set_tmp['longitude'],live_set_tmp['latitude'],
s=(live_set_tmp['quake']*100))
    plt.suptitle('FutureEarthquakesfor'+days[i])
    plt.xlabel('Longitude')
    plt.ylabel('Latitude')
    plt.grid()
    plt.show()
```

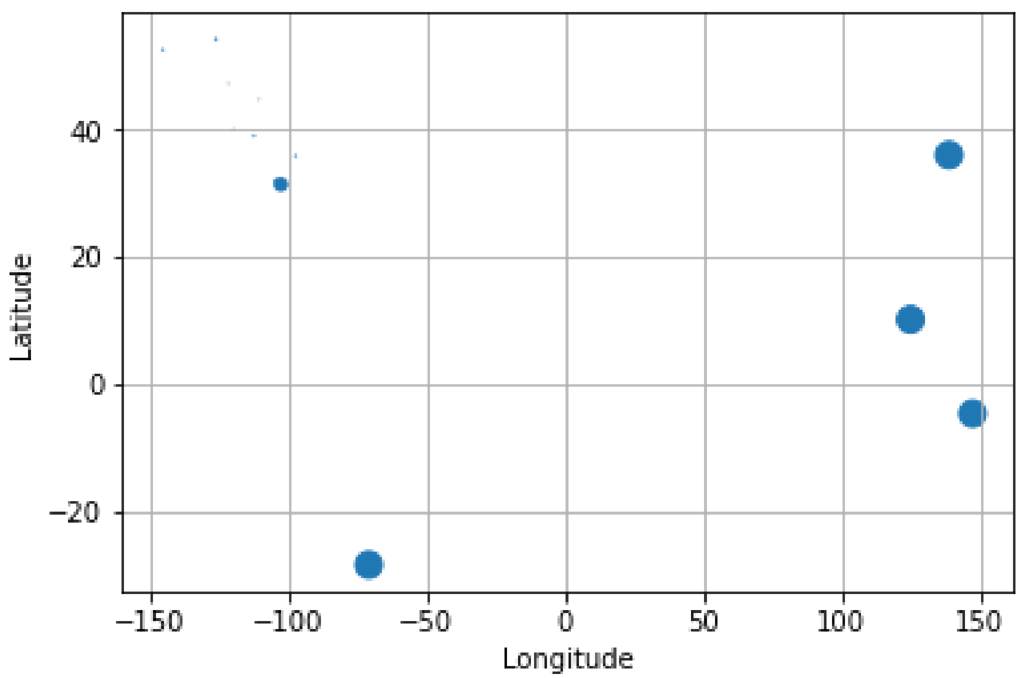
Future Earthquakes for 2023-04-18



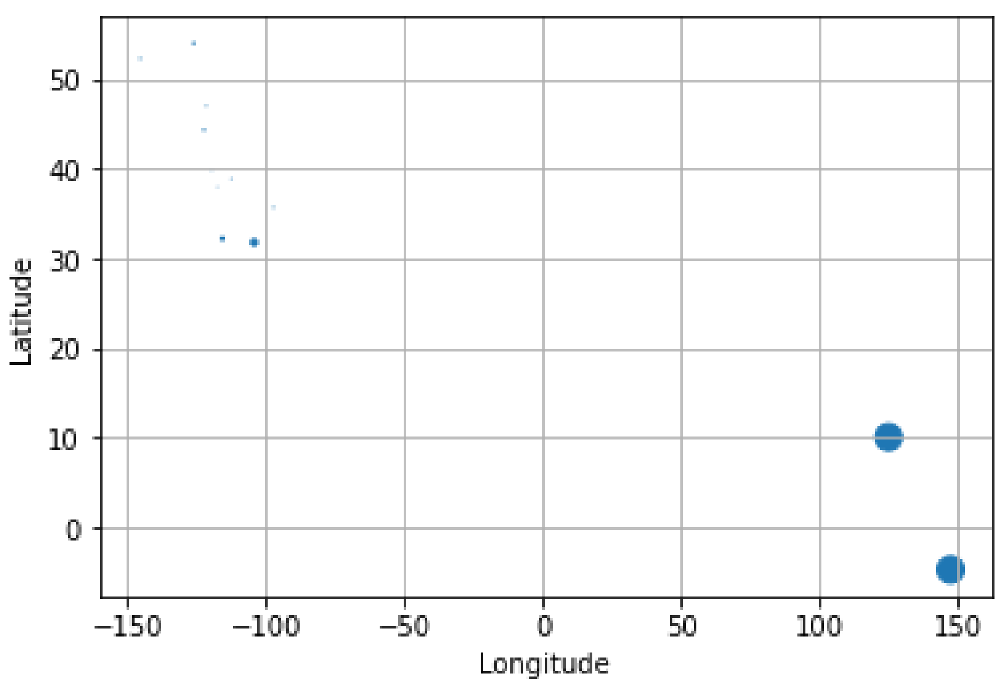
Future Earthquakes for 2023-04-19

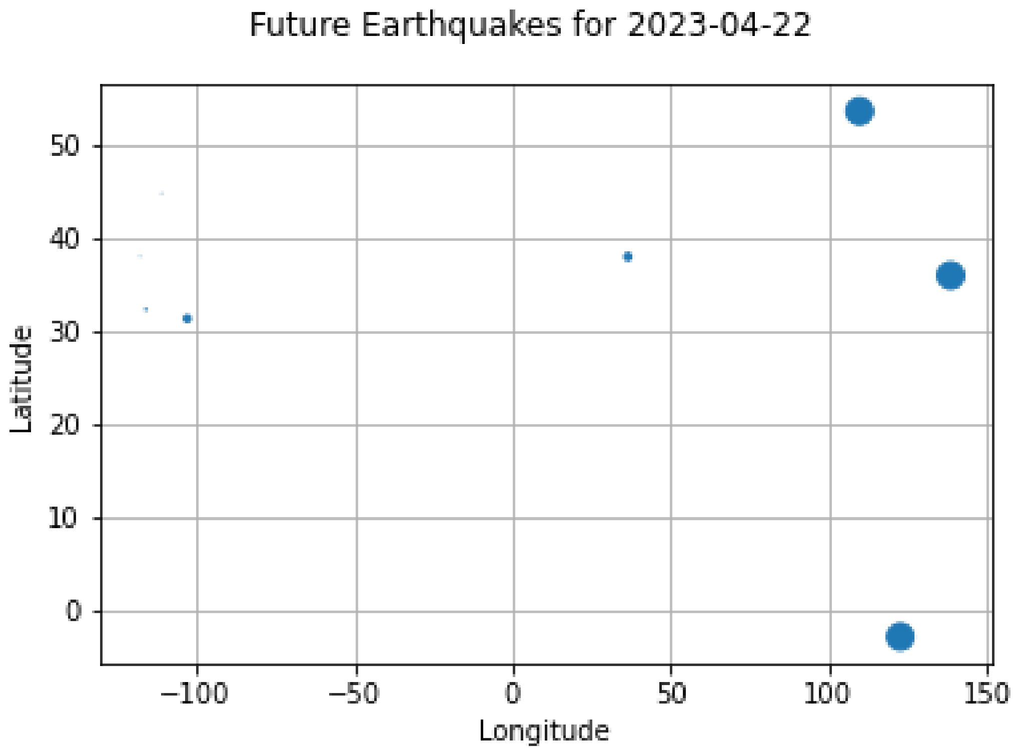


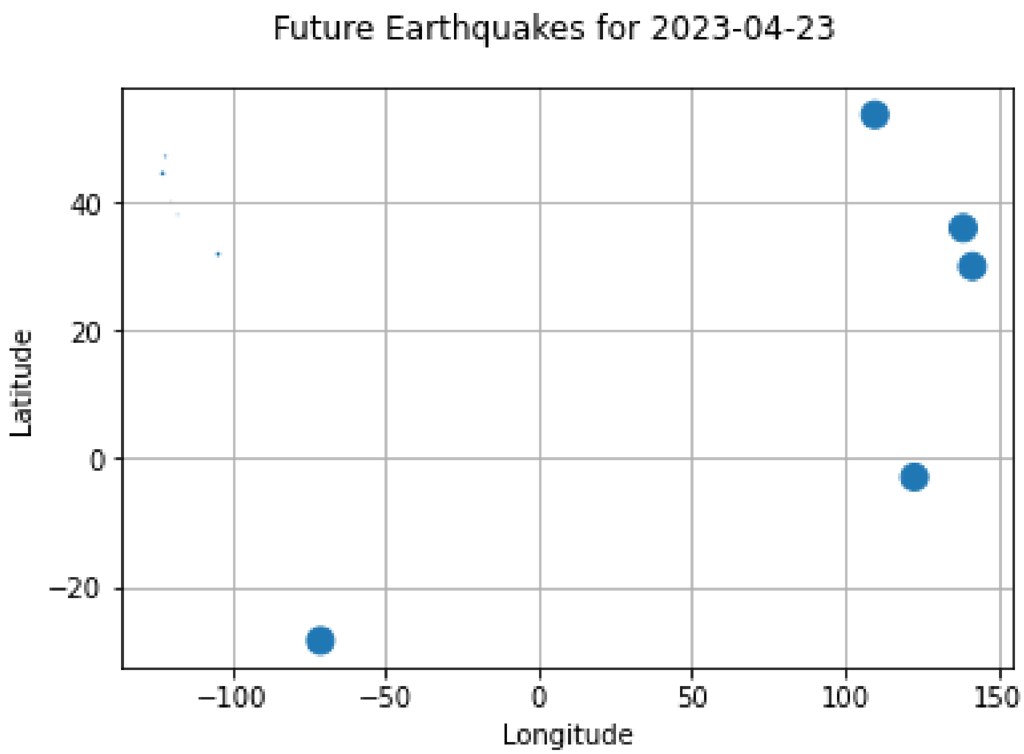
Future Earthquakes for 2023-04-20

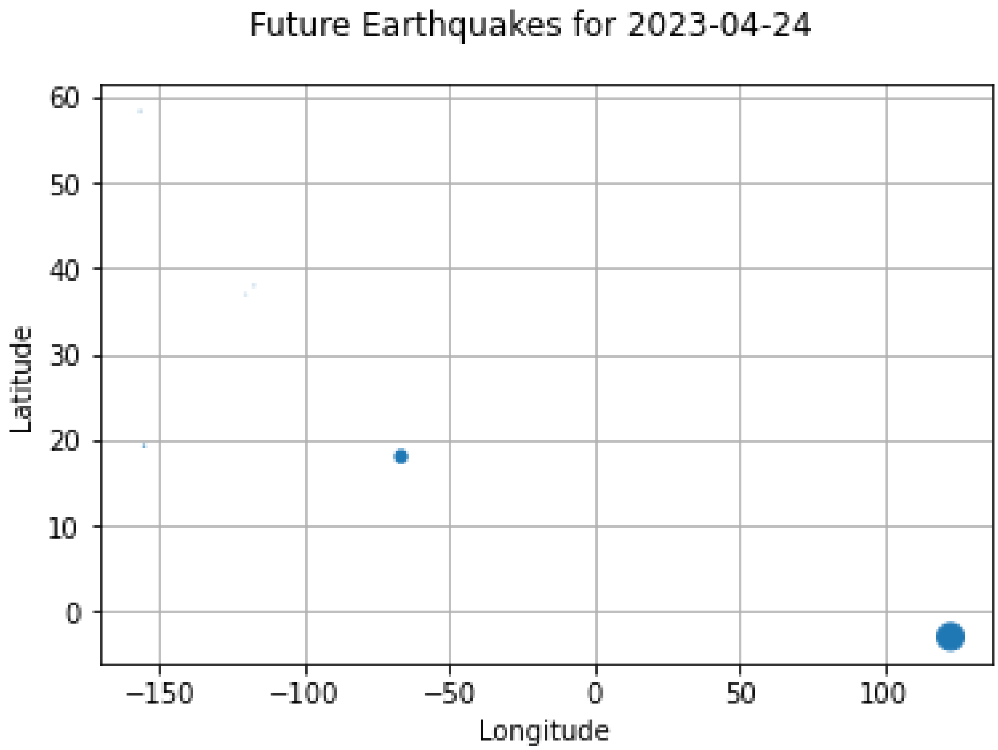


Future Earthquakes for 2023-04-21




Future Earthquakes for 2023-04-22
Latitude
Longitude
50
40
30
20
10
0
−100
−50
0
50
100
150


Future Earthquakes for 2023-04-23
Latitude
Longitude
40
20
0
−20
−100
−50
0
50
100
150



Finalthoughts:

- Sofarthemodellooksgoodwithxgboostaschosenmodelforpredictionsinweb app haveing higher auc score and higher recall_score as I have explained under XGBoost result section why auc and recall score are chosen.
- MainIdeaofourprojectwillbepredictingorforecastingtheseearthquakesiteson given day all over the world.