ARTIFICIAL INTELLIGENCE -Earthquake prediction model of python

TEAM MEMBER

511821104037– N.THARUNRAJ

Phase – 4 Development Part-2

PROJECT: *Continue building the earthquake prediction model by visualizing the data on a world map and splitting it into training and testing sets.*

ARTIFICIAL INTELLIGENCE -Earthquake prediction

DEPARTMENT OF COMPUTERSCIENCE and ENGINEERING

YEAR – 3rd **SEMESTER** – 5th

Introduction:

For predicting earthquake consider 100% of data as training set. By using the machine learning library of spark we build the regression models on the training set. They also check the accuracy of the algorithm, for this purpose we take 60% of data as training set and 40% as test data.

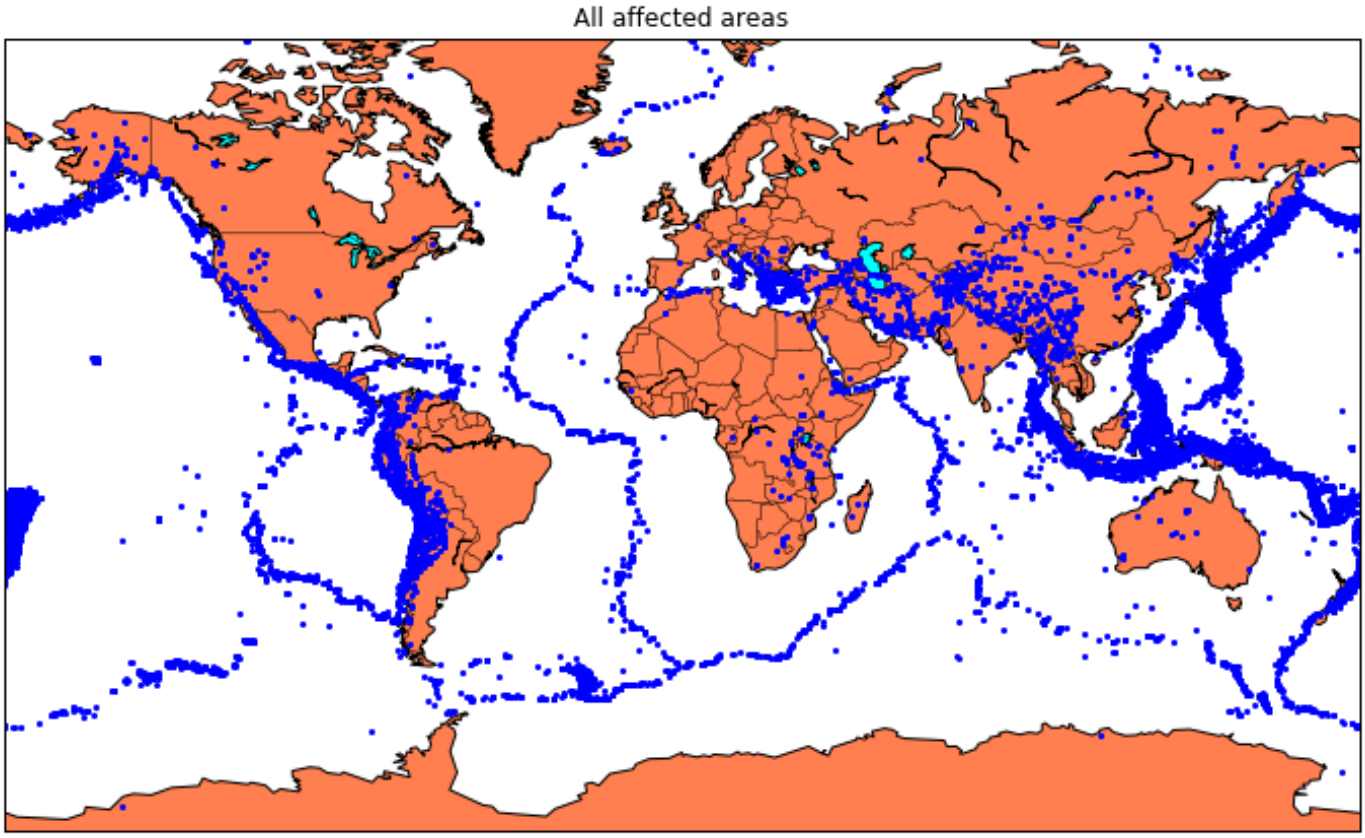
Visualization:

- **Here, we will visualize the earthquakes that have occurred all around the world.**

Program:

```
from mpl_toolkits.basemap import Basemap
m = Basemap(projection='mill',llcrnrlat=80,urcrnrlat=80, llcrnrlon=180,urcrnrlon=180,lat_ts=20,resolution='c')
longitudes = data["Longitude"].tolist()
latitudes = data["Latitude"].tolist()
#m = Basemap(width=12000000,height=9000000,projection='lcc',
#resolution=None,lat_1=80.,lat_2=55,lat_0=80,lon_0=-107.)
x,y = m(longitudes,latitudes)
fig = plt.figure(figsize=(12,10))
plt.title("All affected areas")
m.plot(x, y, "o", markersize = 2, color = 'blue')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```

Output:



- **We have located on the world map where earthquakes happened in the last few years.**

Splitting the Dataset:

- **To do this, we split our dataset into training, validation, and testing data splits.**
- **Data splitting is when data is divided into two or more subsets.**
- **Typically, with a two-part split, one part is used to evaluate or test the data and the other to train the model. Data splitting is an important aspect of data science, particularly for creating models based on data.**

Now we will split the dataset into training and testing set.

Program:

```
X = final_data[['Timestamp', 'Latitude', 'Longitude']]

y = final_data[['Magnitude', 'Depth']]

from sklearn.cross_validation import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
ndom_state=42)
print(X_train.shape, X_test.shape, y_train.shape, X_test.shape)
```

Output:

```
(18727, 3) (4682, 3) (18727, 2) (4682, 3)
```

- **We will be using the RandomForestRegressor model to predict the earthquake, here will look for its accuracy.**

Program:

```
reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
reg.predict(X_test)
```

Output:

```
array([[  5.96,  50.97],
       [  5.88,  37.8 ],
       [  5.97,  37.6 ],
       ...,
       [  6.42,  19.9 ],
       [  5.73, 591.55],
       [  5.68,  33.61]])
```

Program:

```
reg.score(X_test, y_test)

reg.score(X_test, y_test)
```

Output:

```
0.8614799631765803
```

- 86% of accuracy is quite high.
- Now we will shift to Grid Search

Program:

```
from sklearn.model_selection import GridSearchCV

parameters = {'n_estimators':[10, 20, 50, 100, 200, 500]}grid_ob

j = GridSearchCV(reg, parameters)

grid_fit = grid_obj.fit(X_train, y_train)

best_fit = grid_fit.best_estimator_

best_fit.predict(X_test)
```

Output:

```
array([[  5.8888 ,  43.532  ],
       [  5.8232 ,  31.71656],
       [  6.0034 ,  39.3312 ],
       ...,
       [  6.3066 ,  23.9292 ],
       [  5.9138 , 592.151  ],
       [  5.7866 ,  38.9384 ]])
```

Program:

```
best_fit.score(X_test, y_test)
```

Output:

```
0.8749008584467053
```

1. **Considering it's a natural phenomenon, we have got a high accuracy number.**
2. **We will employ a neural network for predicting the earthquake.**