Visualization

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

```
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='l
cc',

#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()

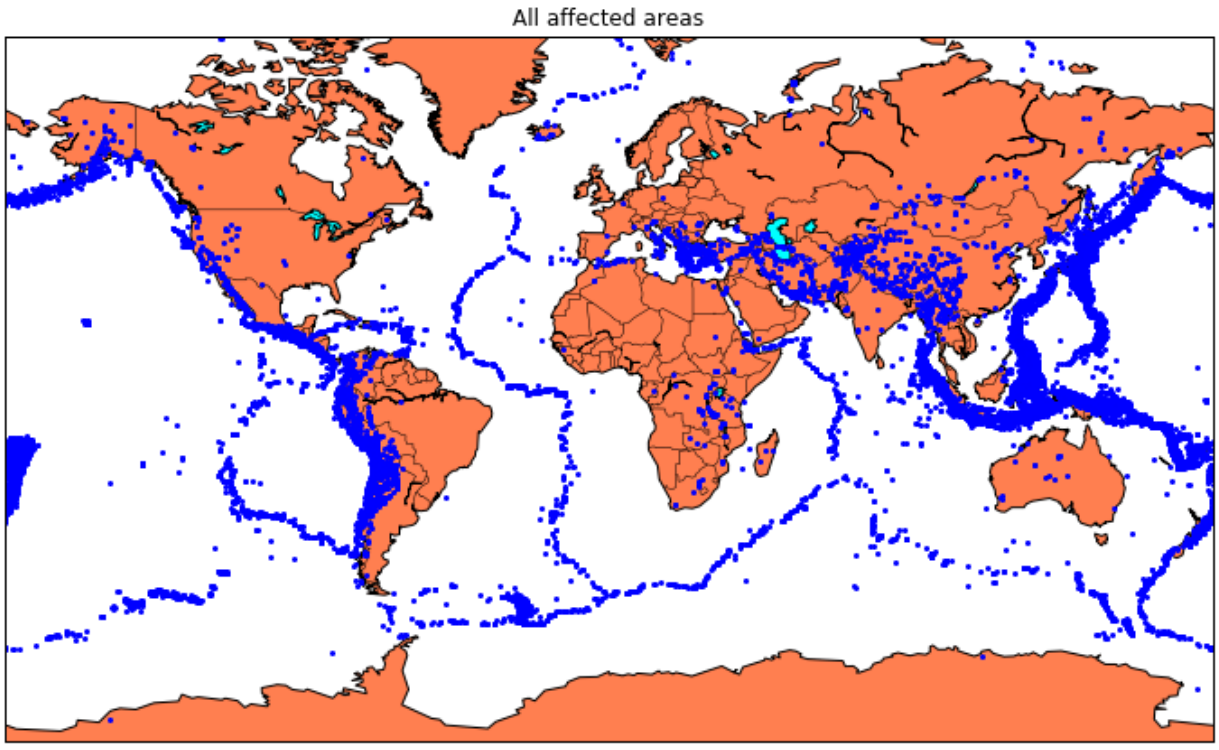
m.fillcontinents(color=’coral’,lake_color=’aqua’)

m.drawmapboundary()

m.drawcountries()

plt.show()

Output:



Splitting The Dataset

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

```
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,
random_state=42)

Print(X_train.shape, X_test.shape, y_train
```

Output:

Earthquake Prediction Using Machine Learning

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

```
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]])
```

Reg.score(X_test, y_test)

Output:

```
0.8614799631765803
```

86% of accuracy is quite high.

Now we will shift to GridSearch.

From sklearn.model_selection import GridSearchCV

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

Grid_obj = 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 ]])
```

best_fit.score(X_test, y_test)

Output:

```
0.8749008584467053
```

Considering it's a natural phenomenon, we have got a high accuracy number.