Earthquake Prediction Model using Python

AI_Phase4

Development Part-2

TABLE OF CONTENT:

INTRODUCTION:

Visualization techniques play a crucial role in earthquake prediction models as they help in analyse and understanding the patterns and trends in seismic data. Python provides numerous libraries and tools.The visualization of earthquake data can include different types of plots, maps, and graphs. These visual representations assist in identifying spatial and temporal patterns, detecting correlations between variables, and exploring the seismic activity in specific regions. By visualizing earthquake data, helping to develop more accurate prediction models.Python offers various powerful libraries for earthquake data visualization, such as Matplotlib, Seaborn, and Plotly. Matplotlib provides extensive plotting capabilities, allowing users to create line plots, scatter plots, histograms, and more. In earthquake prediction models, visualizations can show the distribution of earthquake occurrences on a map.

Data splitting is a critical step in developing an earthquake prediction model to evaluate its performance accurately. The process involves dividing the dataset into distinct subsets for training, validation, and testing.In

earthquake prediction, the dataset typically consists of seismic attributes such as magnitude, depth, location coordinates, time, and additional information pertinent to earthquakes. To create an effective prediction model, it is necessary to split this dataset into three parts they are Training Set, Validation Set, Testing Set.Python provides various libraries and functions to split the earthquake dataset, such as scikit-learn's train_test_split() or K-fold cross-validation techniques. These tools enable to divide the data into appropriate proportions while ensuring the distribution of seismic attributes remains representative across the subsets.By splitting the dataset correctly, we can train, validate, and test their earthquake prediction models effectively, ensuring reliable and trustworthy results.

Visualization:

Here, all the earthquakes from the database in visualized on to the world map which shows clear representation of the locations where frequency of the earthquake will be more.

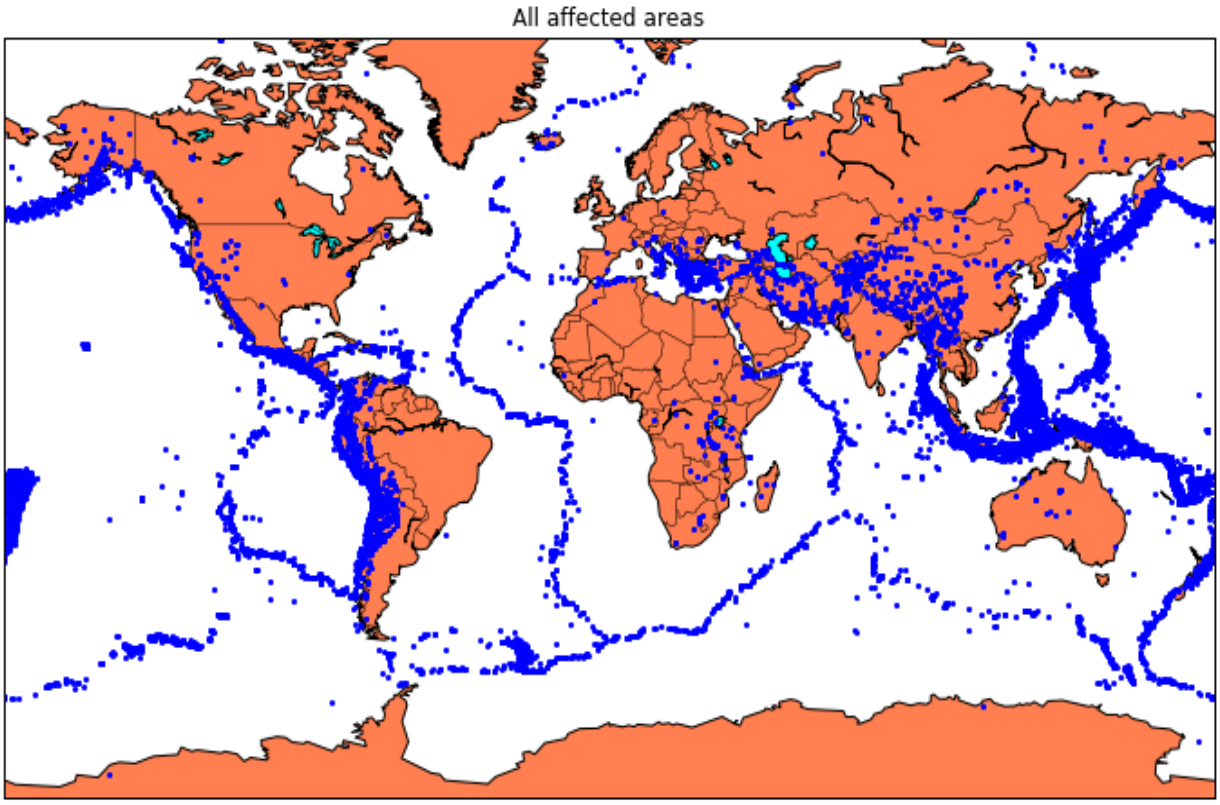
Input:

```
from mpl_toolkits.basemap import Basemap

m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat
=80, llcrnrlon=-180,urcrnrlon=180,lat_ts=20,resolutio
n='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)
```

Input:

```
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 Data:

Firstly, split the data into Xs and ys which are input to the model and output of the model respectively. Here, inputs are TImestamp, Latitude and Longitude and outputs are Magnitude and Depth. Split the Xs and ys into train and test with validation. Training dataset contains 80% and Test dataset contains 20%.

Input:

```
X = final_data[['Timestamp', 'Latitude', 'Longi
tude']]
y = final_data[['Magnitude', 'Depth']]
```

```
from sklearn.cross_validation import trai
n_test_split

X_train, X_test, y_train, y_test = train_
test_split(X, y, test_size=0.2, random_st
ate=42)
print(X_train.shape, X_test.shape, y_trai
n.shape, X_test.shape)
```

Here, we used the Random Forest Regressor model to predict the outputs, we see the strange prediction from this with score above 80% which can be assumed to be best fit but not due to its predicted values.

Input:

```
from sklearn.ensemble import RandomForestRegres
sor

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]])
```

Input:

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

Output:

```
0.8614799631765803
```

Input:

```
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 ]])
```

Input:

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

Output:

```
0.8749008584467053
```

Conclusion:

visualizing earthquake data using Python can significantly enhance our understanding and analysis of earthquake patterns. By using various visualization techniques such as scatter plots, heatmaps, and interactive maps, we can identify trends, correlations, and potential earthquake-prone areas. These visualizations help in uncovering meaningful insights and patterns that may not be apparent in raw data, assisting in decision-making, risk assessment, and early warning systems.The conclusion for data splitting is that it is essential for assessing model performance and avoiding overfitting. By using appropriate ratios for splitting the data and ensuring randomness, we can build robust earthquake prediction models that can accurately forecast seismic activity.