Earthquake Prediction using python

Earthquake Prediction

It is well known that if a disaster has happened in a region, it is likely to happen there again. Some regions really have frequent earthquakes, but this is just a comparative quantity compared to other regions. So, predicting the earthquake with Date and Time, Latitude and Longitude from previous data is not a trend which follows like other things, it is natural occurring

Visualization

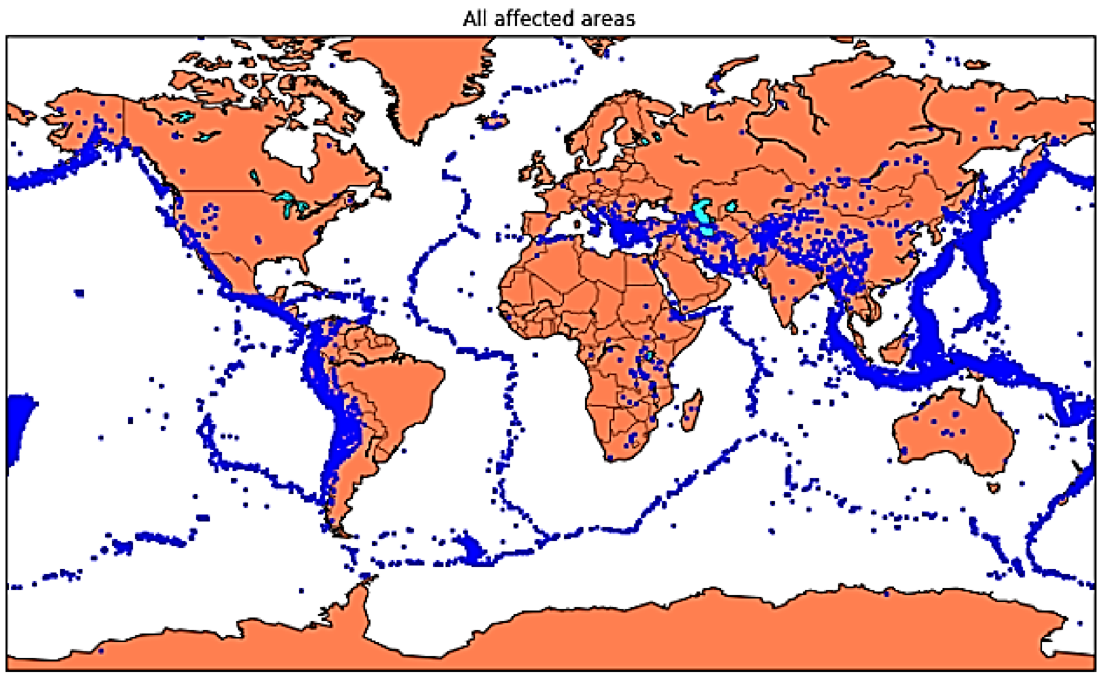
Here, all the earthquakes from the database in visualized on to the world map which shows clear representation of the locations where frequency of the earthquake will be more

Example

```
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()
```

```
/opt/conda/lib/python3.6/site-packages/mpl_toolkits/basemap
/__init__.py:1704: MatplotlibDeprecationWarning: The axesPa
tch function was deprecated in version 2.1. Use Axes.patch
instead.
  limb = ax.axesPatch
/opt/conda/lib/python3.6/site-packages/mpl_toolkits/basemap
/__init__.py:1707: MatplotlibDeprecationWarning: The axesPa
tch function was deprecated in version 2.1. Use Axes.patch
instead.
  if limb is not ax.axesPatch:
```

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%.

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

```
In [11]:
linkcode
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, t
est_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, X_test.sh
ape)
```

Neural Network model

In the above case it was more kind of linear regressor where the predicted values are not as expected. So, Now, we build the neural network to fit the data for training set. Neural Network consists of three Dense layer with each 16, 16, 2 nodes and relu, relu and softmax as activation function

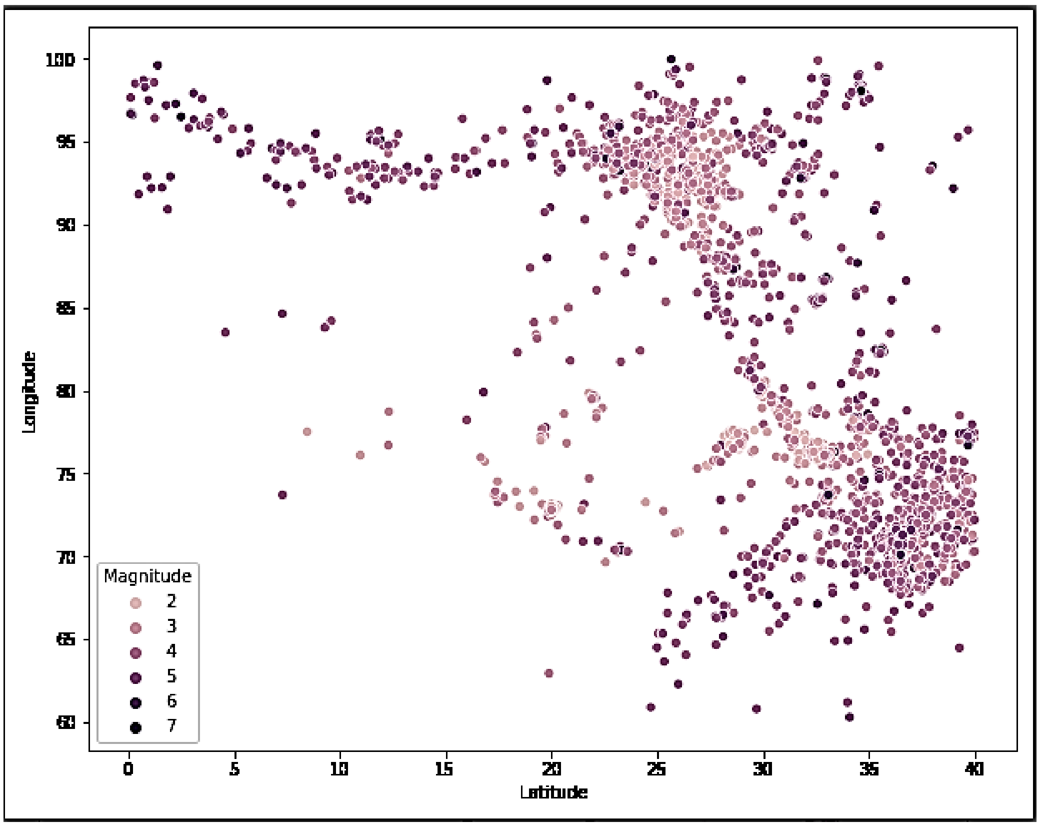
```
from keras.models import Sequential
from keras.layers import Dense

def create_model(neurons, activation, optimizer, loss):
    model = Sequential()
    model.add(Dense(neurons, activation=activation, input_shape=(3,)
))
    model.add(Dense(neurons, activation=activation))
    model.add(Dense(2, activation='softmax'))

    model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy
'])

    return model
```

As we know that many natural phenomena follow a normal distribution and here we can observe that the magnitude of the earthquake also follows a normal distribution.

Example

```
plt.figure(figsize=(10, 8))

sb.scatterplot(data=df,

               x='Latitude',

               y='Longitude',

               hue='Magnitude')

plt.show()
```

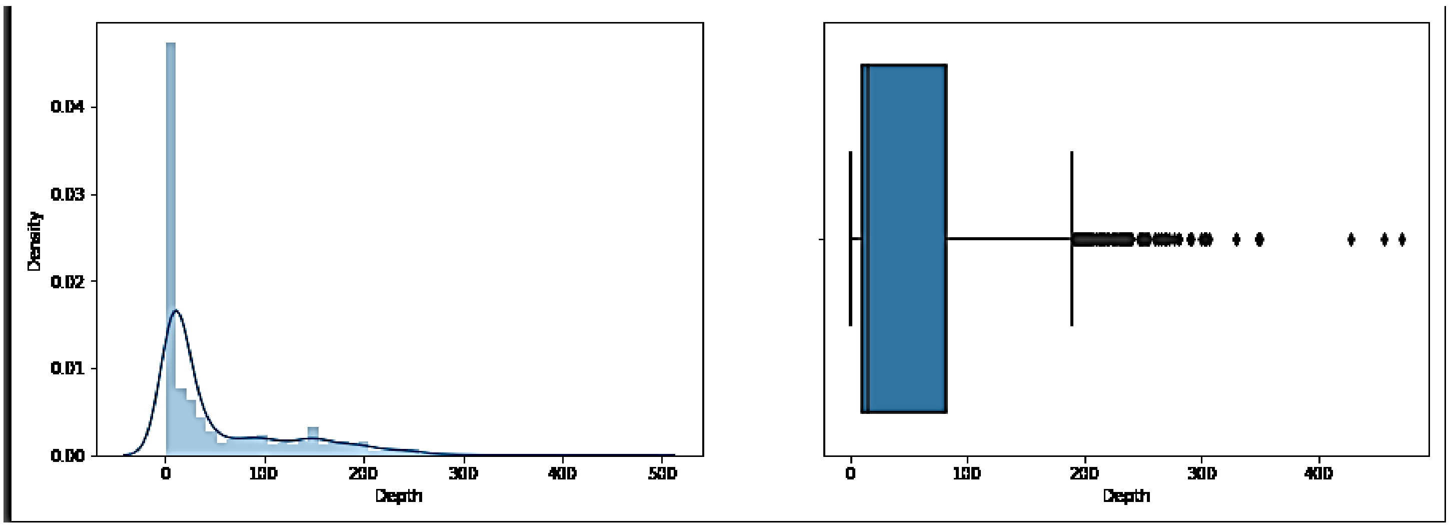
output



Example

```
plt.subplots(figsize=(15, 5))

plt.subplot(1, 2, 1)
sb.distplot(df['Depth'])

plt.subplot(1, 2, 2)
sb.boxplot(df['Depth'])

plt.show()
```

From the distribution graph, it is visible that there are some outliers that can be confirmed by using the boxplot. But the main point to observe here is that the distribution of the depth at which the earthquake rises is left-skewed.


Density
0.00
0.01
0.02
0.03
0.04
0
100
200
300
400
500
Depth
0
100
200
300
400
Depth