EARTHQUAKE PREDICTION MODEL USING PYTHON

Phase 4 Submission Document:

Project Title: Earthquake Prediction

Phase 4: Development Part 2

INTRODUCTION:

- ✓ Earthquake Prediction is a way of predicting the magnitude of an earthquake based on parameters such as longitude, latitude, depth, and duration magnitude, country, and depth using machine learning to give warnings of potentially damaging earthquakes early enough to allow appropriate response to the disaster, enabling people to minimize loss of life and property.

1.Feature Engineering:

1. Extracting meaningful features from the data, such as temporal trends, spatial relationships, and interactions between different parameters.
2. Incorporating domain-specific knowledge and external factors that may influence seismic activities.

2.Machine Learning Models:

1. Developing predictive models, often based on machine learning algorithms, to analyze and learn patterns from historical earthquake data.
2. Common models include neural networks, support vector machines, and decision trees.

3.Hyperparameter Tuning:

1. Fine-tuning model parameters to optimize predictive performance through techniques like grid search or Bayesian optimization.

4.Validation and Evaluation:

1. Splitting the dataset into training and testing sets to validate the model's performance.
2. Evaluating the model's accuracy, precision, recall, and other relevant metrics.

5.Challenges in Earthquake Prediction:

1. Earthquakes are inherently unpredictable due to the dynamic and complex nature of tectonic processes.
2. Limited historical data for rare, large-magnitude earthquakes makes it challenging to train accurate models.

Development Part 2:

- Visualizing the data on a world map
- Splitting it into training and testing sets

Visualizing The Data:

Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

Data Visualization in Earthquake Prediction

Visualizing data in earthquake prediction involves conveying complex information in a clear and insightful way. Here are some data visualization techniques specifically tailored for earthquake prediction:

1.Seismic Heatmaps:

1. Use a heatmap to represent the intensity of seismic activity in a geographic area.
2. Color gradients can indicate the frequency or magnitude of earthquakes.

2.Time Series with Aftershocks:

1. Create a time series plot that shows earthquake occurrences over time.
2. Highlight significant events, such as major earthquakes and aftershocks, on the timeline.

3.Magnitude vs. Depth Scatter Plots:

1. Plot earthquake magnitude against depth to identify patterns and correlations.
2. Differentiate between shallow and deep earthquakes to understand their characteristics.

4.Animated Earthquake Maps:

1. Develop animated maps that show the evolution of seismic activity over time.
2. Animations can reveal patterns and changes in earthquake distribution.

5.Fault Line Network Graphs:

1. Use network graphs to visualize relationships between fault lines, tectonic plates, and seismic events.
2. Nodes represent regions, and edges indicate connections or correlations.

6.3D Earthquake Visualization:

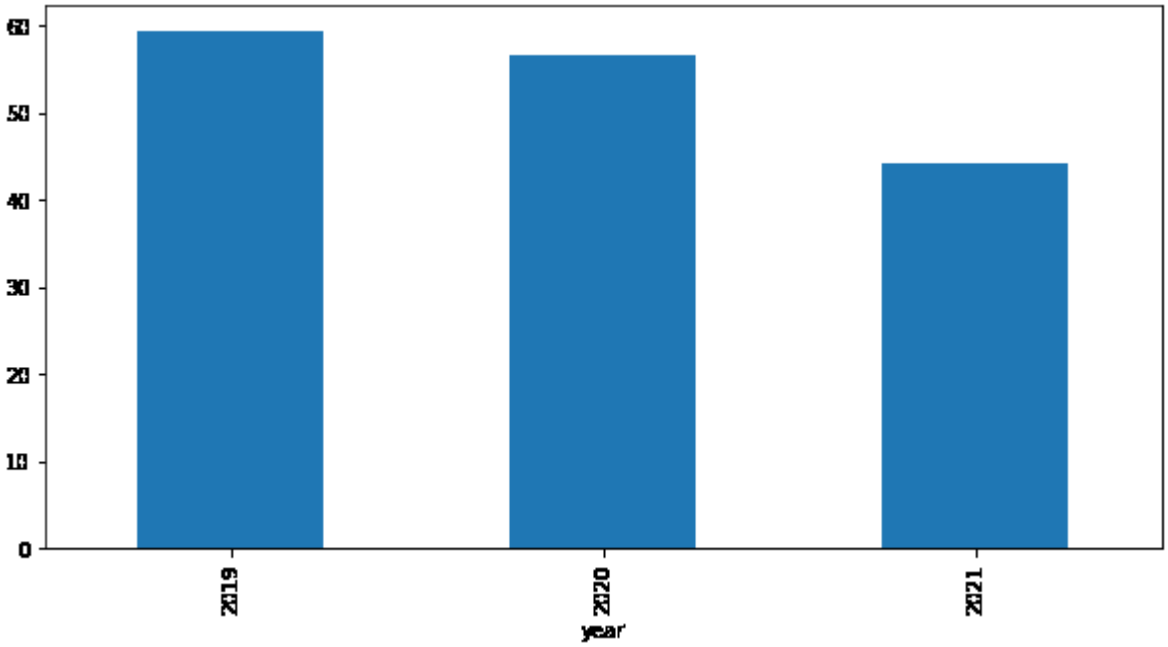
1. Represent seismic activity in a three-dimensional space, considering location, depth, and magnitude.
2. Use 3D bar charts or scatter plots for a comprehensive view.
3. visualizations to highlight relationships.

1.Real-time Earthquake Dashboard:

1. If possible, create a real-time dashboard that updates with the latest seismic data.
2. This can be valuable for monitoring ongoing seismic events.

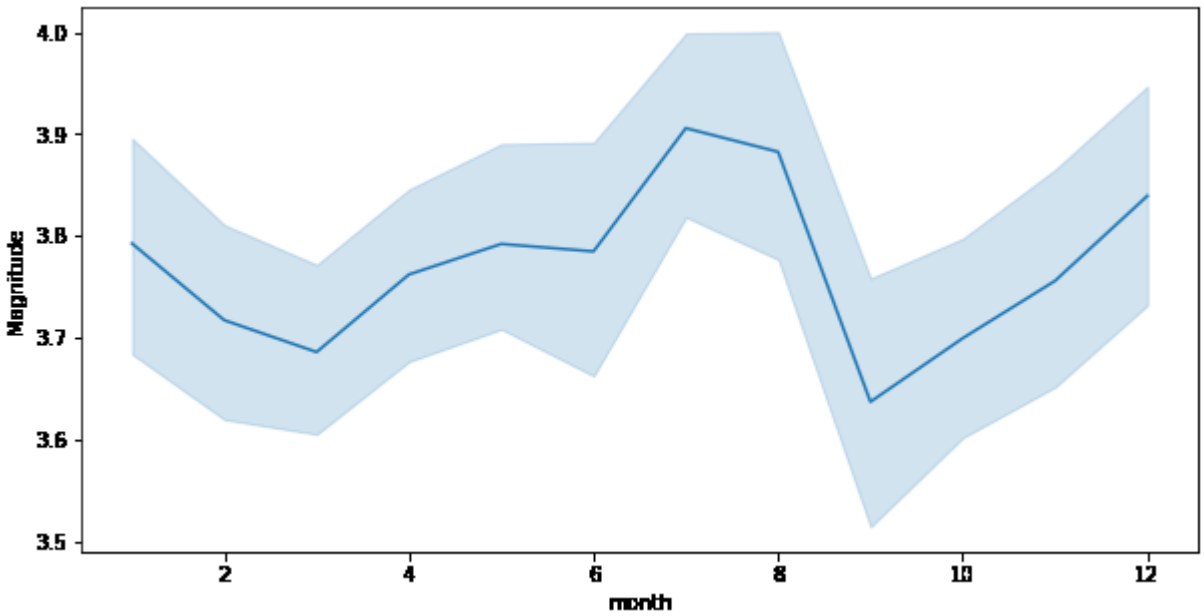
Visualizating Data:

```
plt.figure(figsize=(10, 5))
x = df.groupby('year').mean()['Depth']
x.plot.bar()
plt.show()
```



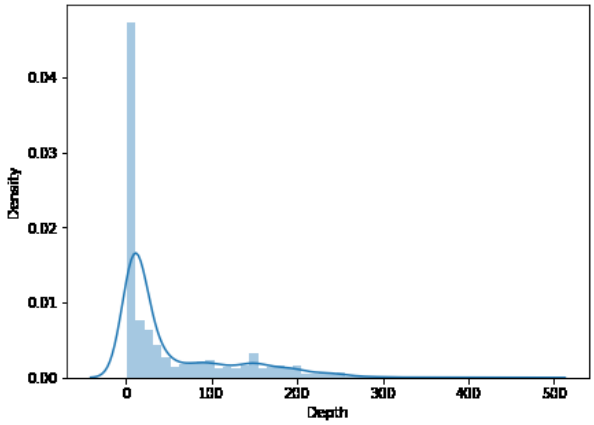
LINEPLOTS:

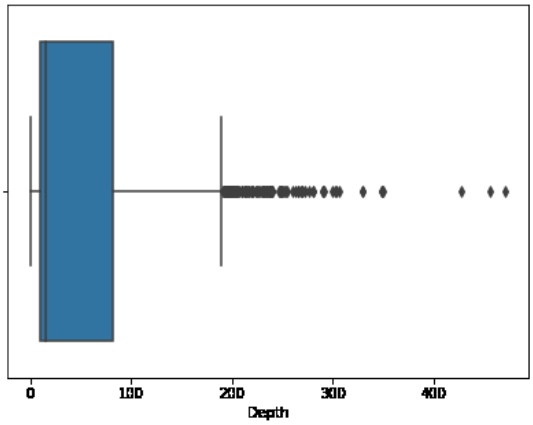
```
plt.figure(figsize=(10, 5))
sb.lineplot(data=df,
            x='month',
            y='Magnitude')
plt.show()
```



SUBPLOTS:

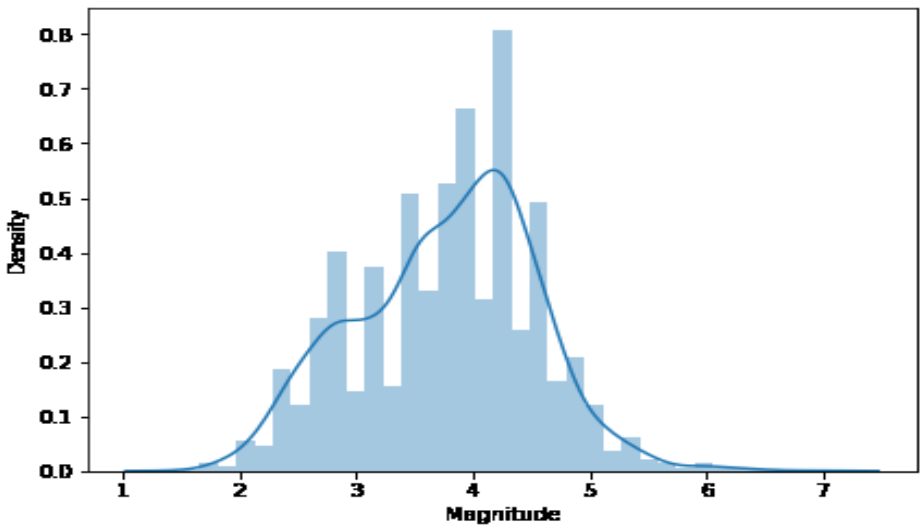
```
plt.subplots(figsize=(15, 5))

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

plt.show()
```

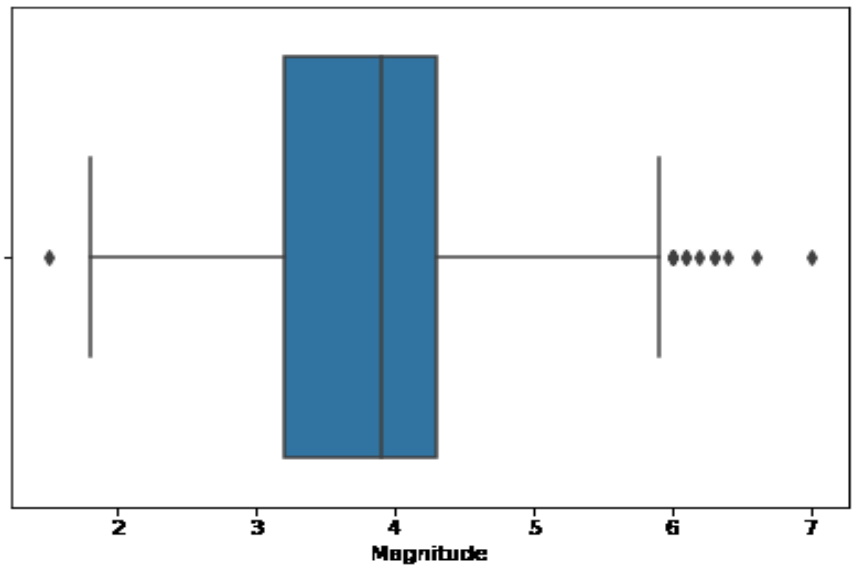




BOXPLOTS:

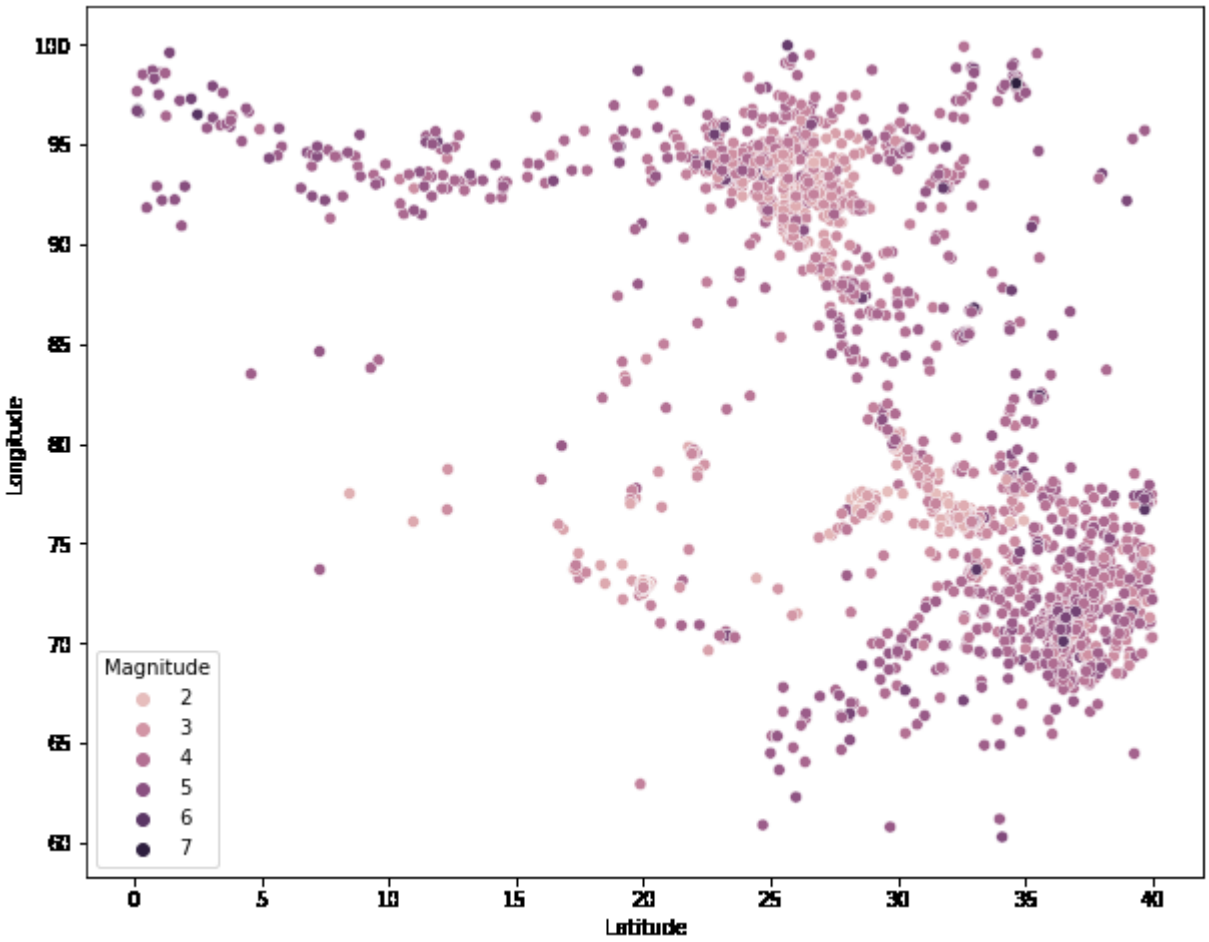
```
plt.subplots(figsize=(15, 5))

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

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

plt.show()
```





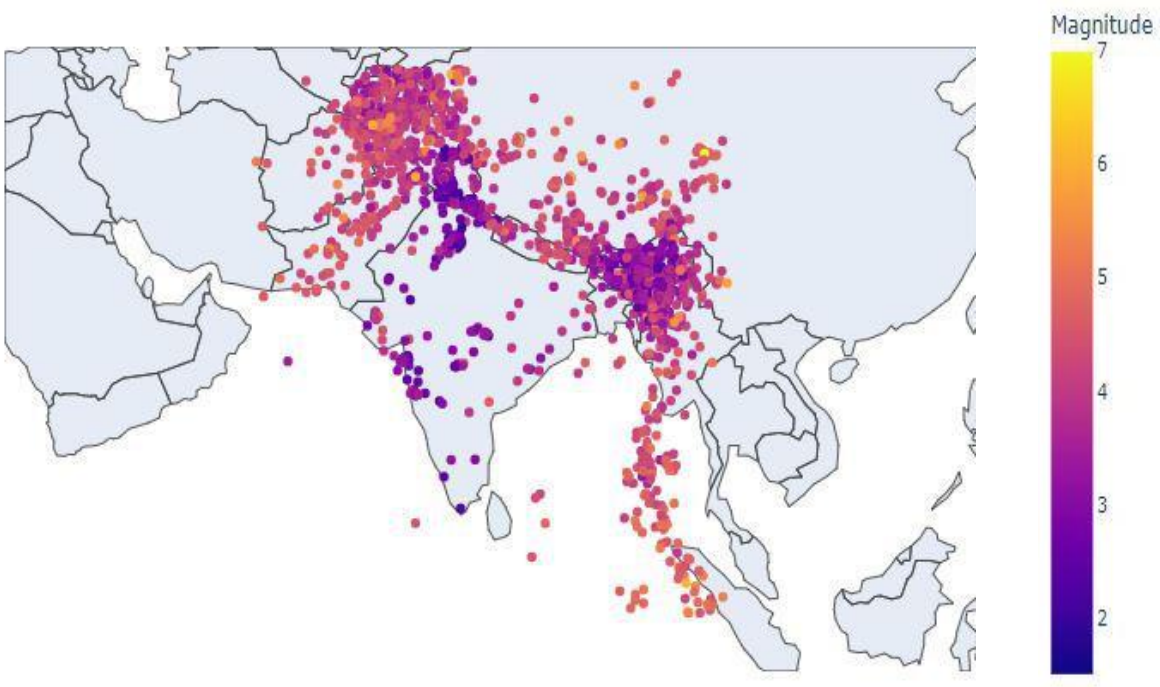
SCATTERPLOTS:

```
plt.figure(figsize=(10, 8))
sb.scatterplot(data=df,
               x='Latitude',
               y='Longitude',
               hue='Magnitude')
plt.show()
```



Now by using Plotly let's plot the latitude and the longitude data on the map to visualize which areas are more prone to earthquakes.

```
import plotly.express as px
import pandas as pd

fig = px.scatter_geo(df, lat='Latitude',
                     lon='Longitude',
                     color="Magnitude",
                     fitbounds='locations',
                     scope='asia')
fig.show()
```

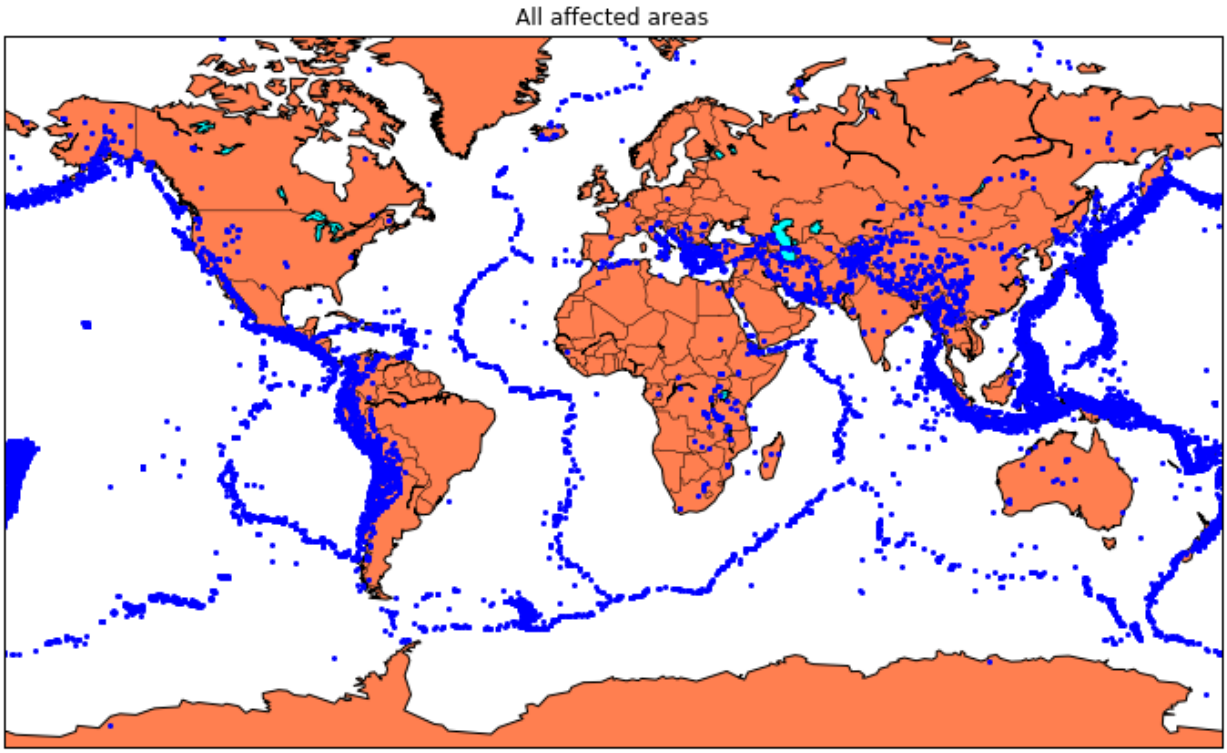


Visualization On The World Map:

```
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:



Data Spiliting:

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']]
```

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

```
(18727, 3) (4682, 3)(1237, 2) (4682,3)
```

```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
reg.predict(X_test)
```

```
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)
```

```
0.8614799631765803
```

```
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)
```

```
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)
```

```
0.8749008584467053
```

Neural Network model

```
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
```

```
Using TensorFlow backend.
```

In this, we define the hyperparameters with two or more options to find the best fit.

```
from keras.wrappers.scikit_learn import KerasClassifier

model = KerasClassifier(build_fn=create_model, verbose=0)

# neurons = [16, 64, 128, 256]
neurons = [16]
# batch_size = [10, 20, 50, 100]
batch_size = [10]
epochs = [10]
# activation = ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear', 'exponential']
activation = ['sigmoid', 'relu']
# optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
optimizer = ['SGD', 'Adadelta']
loss = ['squared_hinge']

param_grid = dict(neurons=neurons, batch_size=batch_size, epochs=epochs, activation=activation, opti
mizer=optimizer, loss=loss)
```

```
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X_train, y_train)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.666684 using {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_h
inge', 'neurons': 16, 'optimizer': 'SGD'}
0.666684 (0.471398) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squa
red_hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.000000 (0.000000) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squa
red_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}
0.666684 (0.471398) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared
_hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.000000 (0.000000) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared
_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}
```

The best fit parameters are used for same model to compute the score with training data and testing data.

```
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(3,)))
model.add(Dense(16, activation='relu'))
model.add(Dense(2, activation='softmax'))

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

```
model.fit(X_train, y_train, batch_size=10, epochs=20, verbose=1, validation_data=(X_test, y_test))
```

```
Train on 18727 samples, validate on 4682 samples
Epoch 1/20
18727/18727 [==============================] - 4s 233us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 2/20
18727/18727 [==============================] - 4s 220us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 3/20
18727/18727 [==============================] - 4s 228us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 4/20
18727/18727 [==============================] - 4s 222us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 5/20
18727/18727 [==============================] - 5s 262us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 6/20
18727/18727 [==============================] - 4s 223us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 7/20
18727/18727 [==============================] - 4s 220us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 8/20
18727/18727 [==============================] - 4s 224us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 9/20
18727/18727 [==============================] - 4s 220us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 10/20
18727/18727 [==============================] - 4s 224us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 11/20
18727/18727 [==============================] - 4s 221us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
```

```
Epoch 12/20
18727/18727 [==============================] - 4s 231us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 13/20
18727/18727 [==============================] - 5s 248us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 14/20
18727/18727 [==============================] - 4s 220us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 15/20
18727/18727 [==============================] - 4s 223us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 16/20
18727/18727 [==============================] - 4s 222us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 17/20
18727/18727 [==============================] - 4s 225us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 18/20
18727/18727 [==============================] - 4s 219us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 19/20
18727/18727 [==============================] - 4s 220us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
Epoch 20/20
18727/18727 [==============================] - 5s 258us/step - loss: 0.5038 - acc: 0.9182 - val_l
oss: 0.5038 - val_acc: 0.9242
```

```
Out[20]:
<keras.callbacks.History at 0x78dfa2107ef0>
```

```
[test_loss, test_acc] = model.evaluate(X_test, y_test)
print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))
```

```
4682/4682 [==============================] - 0s 29us/step
Evaluation result on Test Data : Loss = 0.5038455790406056, accuracy = 0.9241777017858995
```

```
model.save('earthquake.h5')
```

CONCLUSION:

Continuing the development of the earthquake prediction model involves critical steps like visualizing the data on a world map and splitting it into training and testing sets. These actions not only contribute to a comprehensive understanding of the data but also lay the groundwork for a robust and accurate predictive model.

Visualizing seismic data on a world map provides a holistic view of the geographical distribution of earthquakes. This visualization can reveal patterns, hotspots, and trends that may be crucial for the model's learning process. Understanding how seismic activity varies across regions is essential for creating a model that can adapt to diverse geological conditions.

In essence, by visualizing the data on a world map, we gain a spatial perspective that informs our understanding of seismic patterns globally. Simultaneously, splitting the data into training and testing sets is a strategic move towards building a predictive model that is not only accurate but also capable of making reliable predictions on future, unseen seismic events. These steps set the stage for a comprehensive and well-informed earthquake prediction model, laying the foundation for further refinement and optimization as the development process continues.