Earthquake Prediction Model using Python

Phase-1 Documentation Submission

Presented By: Team Member
Balaji E

LIST OF CONTENTS

INTRODUCTION:

- This project presents the development of an earthquake prediction model using Python.
- The primary objective is to leverage machine learning techniques to predict earthquake magnitudes based on a dataset obtained from Kaggle.
- A neural network model is designed and trained using TensorFlow or Keras to make predictions on earthquake magnitudes.

DATA DESCRIPTION:

- The data is provided by the United States Geological Survey (USGS), which monitors and reports on earthquake activity worldwide.

- The dataset contains information about earthquake events worldwide, including details such as location, time, magnitude, depth, and more. It is a comprehensive collection of earthquake-related data.

Dataset link:
https://www.kaggle.com/datasets/usgs/earthquake-database

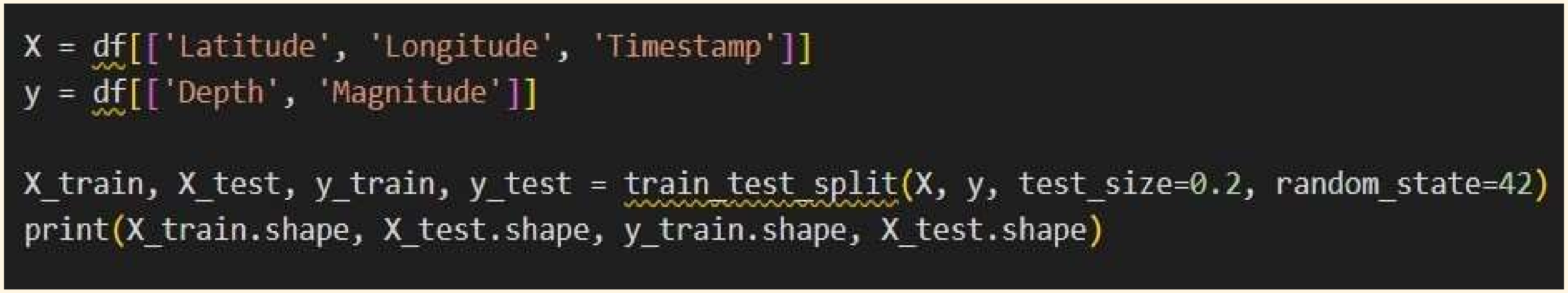
DATA PREPROCESSING:

- Data preprocessing is a crucial step in working with the USGS Earthquake Database or any dataset. It involves cleaning and preparing the data for analysis and model development.
- **Handling Missing Values**: Check for missing values in each column of the dataset. Missing value in Location, Magnitude handle by removing rows with missing values, imputing missing values with the mean or median.
- **Data Type Conversion**: Ensure that columns are of the correct data types. For example, dates should be in datetime format, and numeric columns should be represented as floats or integers.

MODEL BUILDING:

- **Train-Test Split**:

  Split the data into training and testing sets. Split ratio is 80% for training and 20% for testing.

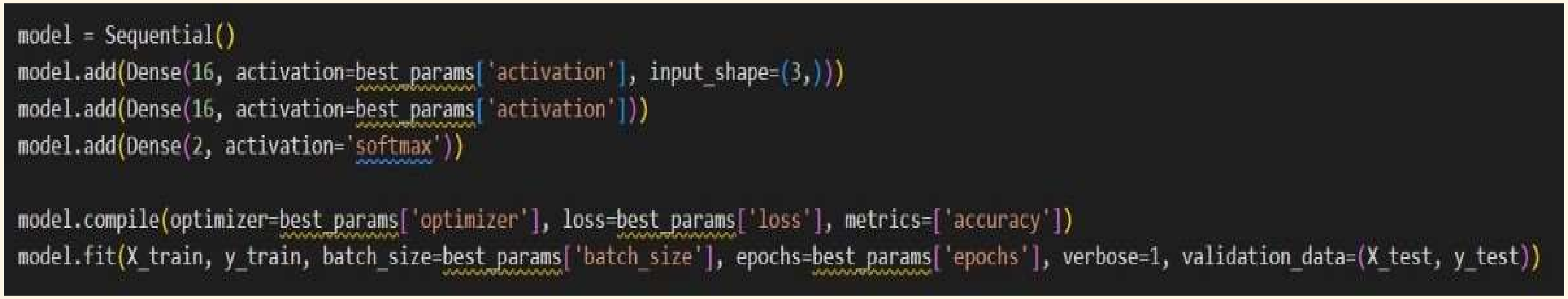


```
X = df[['Latitude', 'Longitude', 'Timestamp']]
y = df[['Depth', 'Magnitude']]

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.shape, X_test.shape)
```

- **Create a Model:**

  Create a Sequential model .Add input and hidden layers. The number of neurons in each layer is 16 and the activation functions are hyperparameters. Compile the neural network model by specifying the loss function, optimizer, and evaluation metric. An optimizer like Adam , Optimizer are used.

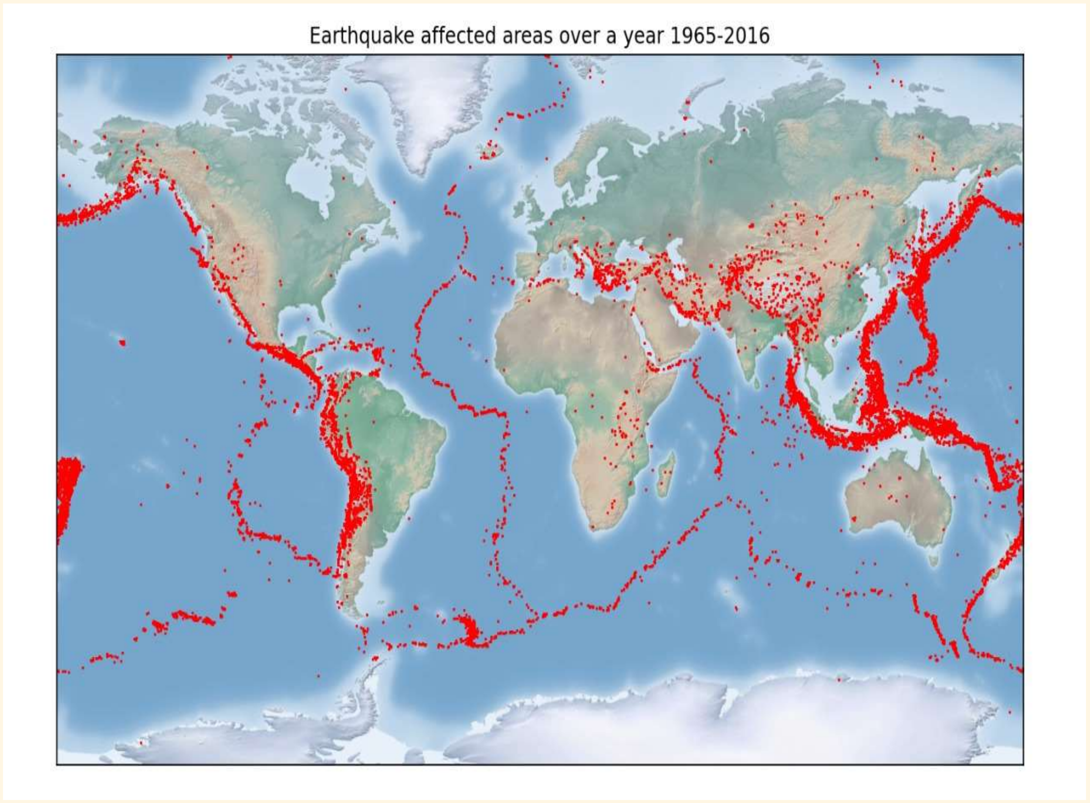


```
model = Sequential()
model.add(Dense(16, activation=best_params['activation'], input_shape=(3,)))
model.add(Dense(16, activation=best_params['activation']))
model.add(Dense(2, activation='softmax'))

model.compile(optimizer=best_params['optimizer'], loss=best_params['loss'], metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=best_params['batch_size'], epochs=best_params['epochs'], verbose=1, validation_data=(X_test, y_test))
```

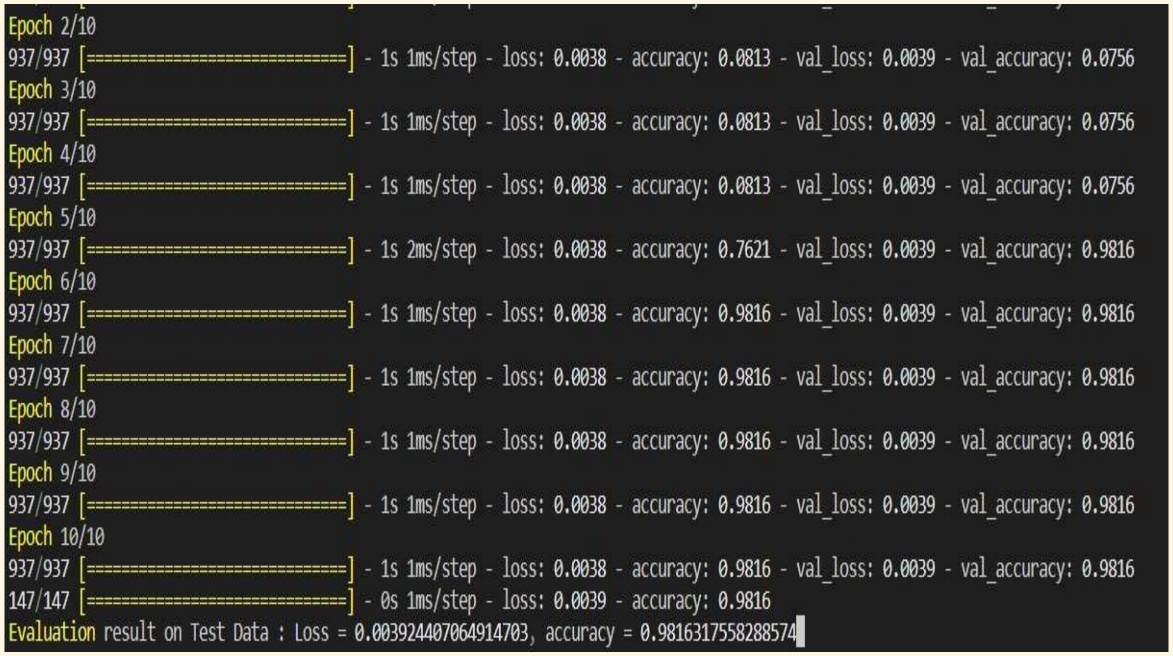
MODEL VISUALIZATION:

- Visualizing the frequency distribution of earthquakes across the world by plot place where earthquake occurs.
- Visualizations contribute to a better understanding of earthquake-prone areas and can be valuable for disaster preparedness and risk assessment.



MODEL EVALUATION:

The evaluation on the test data shows a test loss of approximately 0.0039 and a test accuracy of approximately 98.16%. This means that the model achieved a high accuracy rate on the test dataset, indicating that it performs well on unseen earthquake predicit of magnitude.



```
Epoch 2/10
937/937 [==============================] - 1s 1ms/step - loss: 0.0038 - accuracy: 0.0813 - val_loss: 0.0039 - val_accuracy: 0.0756
Epoch 3/10
937/937 [==============================] - 1s 1ms/step - loss: 0.0038 - accuracy: 0.0813 - val_loss: 0.0039 - val_accuracy: 0.0756
Epoch 4/10
937/937 [==============================] - 1s 1ms/step - loss: 0.0038 - accuracy: 0.0813 - val_loss: 0.0039 - val_accuracy: 0.0756
Epoch 5/10
937/937 [==============================] - 1s 2ms/step - loss: 0.0038 - accuracy: 0.7621 - val_loss: 0.0039 - val_accuracy: 0.9816
Epoch 6/10
937/937 [==============================] - 1s 1ms/step - loss: 0.0038 - accuracy: 0.9816 - val_loss: 0.0039 - val_accuracy: 0.9816
Epoch 7/10
937/937 [==============================] - 1s 1ms/step - loss: 0.0038 - accuracy: 0.9816 - val_loss: 0.0039 - val_accuracy: 0.9816
Epoch 8/10
937/937 [==============================] - 1s 1ms/step - loss: 0.0038 - accuracy: 0.9816 - val_loss: 0.0039 - val_accuracy: 0.9816
Epoch 9/10
937/937 [==============================] - 1s 1ms/step - loss: 0.0038 - accuracy: 0.9816 - val_loss: 0.0039 - val_accuracy: 0.9816
Epoch 10/10
937/937 [==============================] - 1s 1ms/step - loss: 0.0038 - accuracy: 0.9816 - val_loss: 0.0039 - val_accuracy: 0.9816
147/147 [==============================] - 0s 1ms/step - loss: 0.0039 - accuracy: 0.9816
Evaluation result on Test Data : Loss = 0.003924407064914703, accuracy = 0.9816317558288574
```

Loss = 0.003924407064914703

Accuracy = 0.9816317558288574