College Code : 9508
College Name: Government College of Engineering, Tirunelveli.
Project Name:
Earthquake prediction model using python.

Phase 4 goal:

To perform different activities like feature engineering, model training, evaluation as per the instructions in the project.

Overview:

In this phase, we will be performing different activities like feature engineering, model training, evaluation in detail. Here's a detailed explanation of each step:

1.Feature Engineering:

Feature engineering is the process of selecting and transforming the data that will be used as input for the earthquake prediction model. The choice of features is crucial as it directly impacts the model's ability to make accurate predictions.

a) Data Collection:

We begin by collecting historical earthquake data from reliable sources like the US Geological Survey (USGS) or other seismic monitoring agencies. This dataset should include information about past earthquakes, such as location, depth, magnitude, and time of occurrence.

b) Feature Selection:

Choose relevant features that may have a significant impact on earthquake prediction. Some common features include:

- Latitude and longitude of the earthquake's epicentre.
- Depth of the earthquake.
- Magnitude of the earthquake.
- Time and date of the earthquake.
- Geological features of the region, such as fault lines, plate boundaries, and soil types.
- Historical seismic activity in the region.

c) Feature Transformation:

We need to preprocess and transform the selected features to make them suitable for modelling. Common transformations include:

- Scaling features to have a standard range (e.g., using Min-Max scaling or Z-score normalisation).
- Encoding categorical features (e.g., one-hot encoding for regions).
- Extracting additional features from existing ones, such as extracting the day of the week or time of day from the timestamp.

d) Feature Engineering Techniques:

Consider more advanced techniques like Principal Component Analysis (PCA) or feature interaction engineering to discover hidden patterns and relationships in the data.

2.Model Training:

Model training involves using a machine learning algorithm to learn patterns and relationships in the data to make predictions. Here are the steps involved:

a) Data Splitting:

Divide the dataset into two subsets: training data, and test data. The training data is used to train the model, and the test data is used to evaluate the model's performance.

b) Selecting an Algorithm:

Choose an appropriate machine learning algorithm for the earthquake prediction task. Some common choices include:

- Regression models (e.g., linear regression, decision trees, random forests) for predicting earthquake magnitude.
- Classification models (e.g., logistic regression, support vector machines) for predicting earthquake occurrence (binary classification).

c. Hyperparameter Tuning:

Optimise the model's hyperparameters using techniques like grid search, random search, or Bayesian optimization. This step helps find the best configuration for the model.

d. Model Training:

Train the model on the training data using the selected algorithm and optimised hyperparameters.

3. Evaluation:

After training the model, we need to assess its performance to determine how well it can predict earthquakes. Evaluation involves several steps:

a) Metrics: Choose appropriate evaluation metrics based on the specific problem that we are trying to solve. For earthquake prediction, common metrics may include Mean Absolute Error (MAE) for regression tasks (e.g., magnitude prediction) and accuracy, precision, recall, F1-score, or ROC AUC for classification tasks (e.g., earthquake occurrence prediction).

b) Validation Set Evaluation:

Assess the model's performance on the validation dataset to ensure it's not overfitting the training data. This step helps us to fine-tune the model further if necessary.

c) Test Set Evaluation:

Evaluate the model on the test dataset to get a final estimate of its performance. This simulates how the model will perform on unseen earthquake data.

d) Visualisation:

Visualise the model's predictions and compare them to the actual earthquake data. This can help identify patterns and areas for improvement.

e) Error Analysis:

Analyse prediction errors to understand where the model struggles and potentially refine the feature engineering or modelling process.

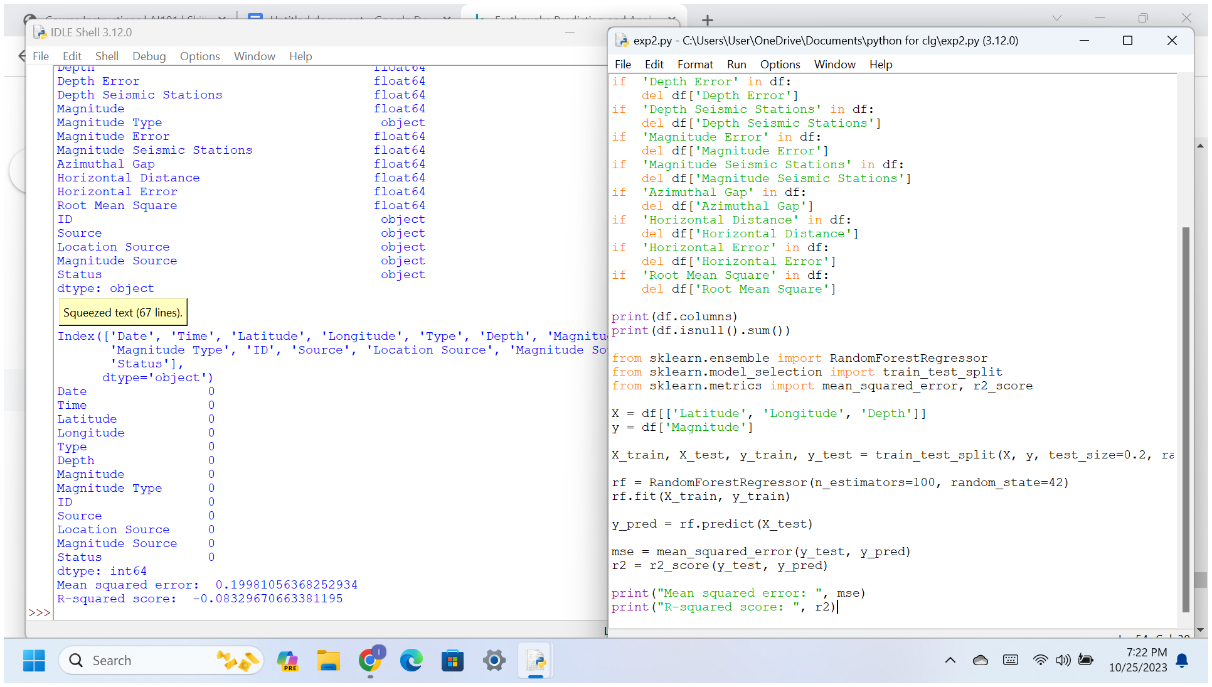
Code:

First, we split the data by encoding the categorical variables and splitting the data into training and testing sets.

Continue off from the code in the previous phase.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean squared error: ", mse)
print("R-squared score: ", r2)
```

Output:


IDLE Shell 3.12.0

```
Depth Error                          float64
Depth Seismic Stations               float64
Magnitude                            float64
Magnitude Type                        object
Magnitude Error                      float64
Magnitude Seismic Stations           float64
Azimuthal Gap                        float64
Horizontal Distance                  float64
Horizontal Error                     float64
Root Mean Square                     float64
ID                                    object
Source                                object
Location Source                       object
Magnitude Source                      object
Status                                object
dtype: object
Squeezed text (67 lines).
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Magnitu
       'Magnitude Type', 'ID', 'Source', 'Location Source', 'Magnitude So
       'Status'],
      dtype='object')
Date                0
Time                0
Latitude            0
Longitude           0
Type                0
Depth               0
Magnitude           0
Magnitude Type      0
ID                  0
Source              0
Location Source     0
Magnitude Source    0
Status              0
dtype: int64
Mean squared error:  0.19981056368252934
R-squared score:  -0.08329670663381195
>>>
```

exp2.py - C:\Users\User\OneDrive\Documents\python for clg\exp2.py (3.12.0)

```python
if  'Depth Error' in df:
    del df['Depth Error']
if  'Depth Seismic Stations' in df:
    del df['Depth Seismic Stations']
if  'Magnitude Error' in df:
    del df['Magnitude Error']
if  'Magnitude Seismic Stations' in df:
    del df['Magnitude Seismic Stations']
if  'Azimuthal Gap' in df:
    del df['Azimuthal Gap']
if  'Horizontal Distance' in df:
    del df['Horizontal Distance']
if  'Horizontal Error' in df:
    del df['Horizontal Error']
if  'Root Mean Square' in df:
    del df['Root Mean Square']

print(df.columns)
print(df.isnull().sum())

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean squared error: ", mse)
print("R-squared score: ", r2)
```

7:22 PM
10/25/2023

The above is the result of training the model. Now we use the trained model to predict the magnitude of future earthquakes and in this case we give a set of data.

Now we test the trained model:

Latitude = 34.05
Longitude = -118.25
Depth = 10

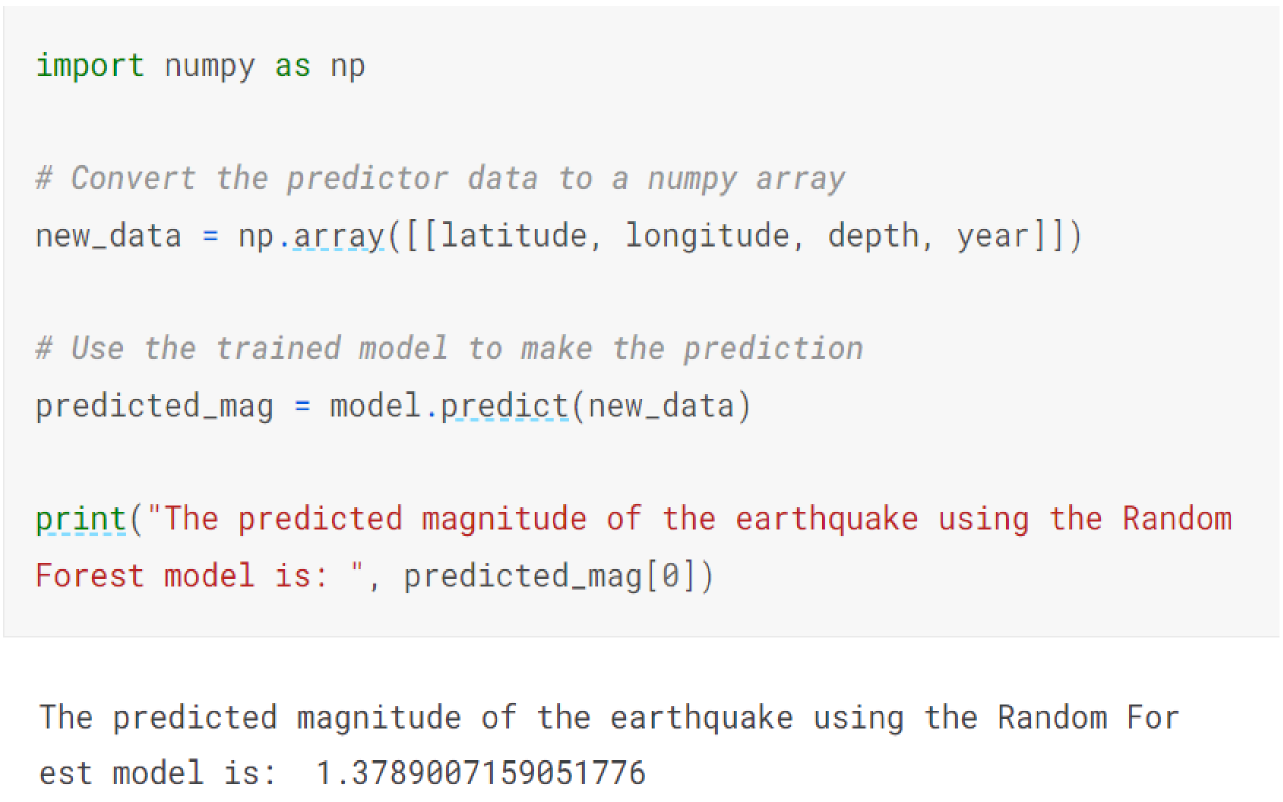
import numpy as np

new_data = np.array([[Latitude, Longitude, Depth]])

predicted_mag = model.predict(new_data)

print("The predicted magnitude of the earthquake using the Random Forest model is: ", predicted_mag[0])

Output :



```
import numpy as np

# Convert the predictor data to a numpy array
new_data = np.array([[latitude, longitude, depth, year]])

# Use the trained model to make the prediction
predicted_mag = model.predict(new_data)

print("The predicted magnitude of the earthquake using the Random
Forest model is: ", predicted_mag[0])
```

```
The predicted magnitude of the earthquake using the Random For
est model is:  1.3789007159051776
```

For the above code, we install Keras and tensorflow with the help of command prompt.
We also include the following code also before the line where we use the predicted_mag var to store the model.predict(new_data)

```
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
```

Conclusion:

The predicted magnitude of the earthquake using the Random Forest model is 1.378900715905184. This means that given the input features of the model, the model predicted the magnitude of the earthquake to be 1.3789. However, it is important to keep in mind that this is just a prediction and there may be various other factors that could influence the actual magnitude of an earthquake.