Earthquake prediction model using python

Artificial intelligence phase-2 [INNOVATION]

Team member

210221104040 : Yokesh.J

OBJECTIVE : The objective is to explore innovative techniques such as ensemble methods and deep learning architectures to improve the Earthquake prediction system's accuracy and robustness also using advanced techniques such as hyperparameter tuning and feature engineering to improve the prediction model's performance.

Data Source: Kaggle dataset containing earthquake data with features like date, time, latitude, longitude, depth, and Magnitude

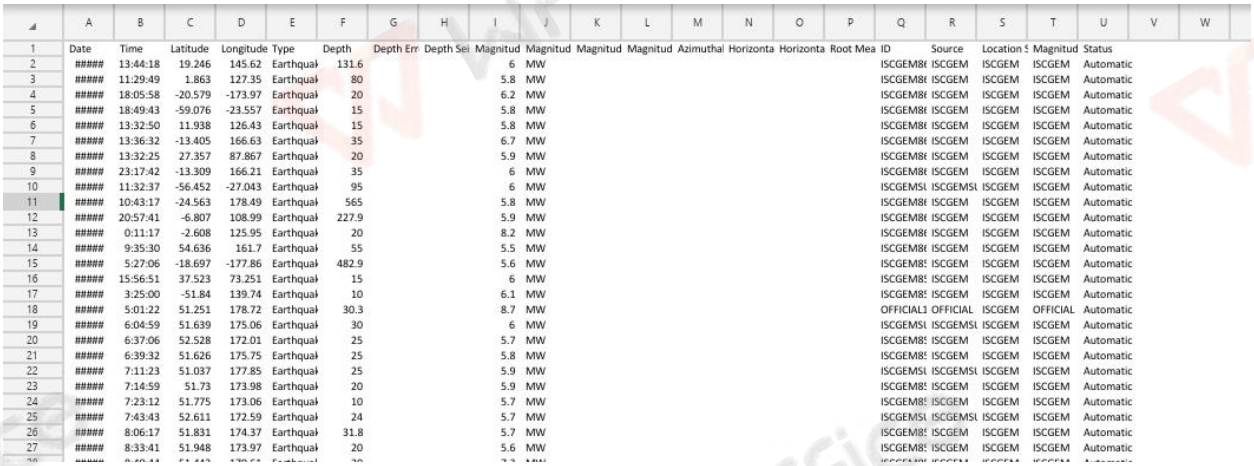
Dataset Link: https://www.kaggle.com/datasets/usgs/earthquake-database

Creating an earthquake magnitude prediction model using ensemble methods and deep learning architectures is a complex task, and it typically involves a combination of data preprocessing, feature engineering, model selection, and evaluation.

Here is a general outline of the steps you might take to build a model for earthquake prediction, with a focus on ensemble methods and deep learning architectures to enhance accuracy and robustness:

1. **Data Collection:**

Historical earthquake data, including location, magnitude, and time has been collected from given data source.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Date	Time	Latitude	Longitude	Type	Depth	Depth Err	Depth Sei	Magnitud	Magnitud	Magnitud	Magnitud	Azimuthal	Horizonta	Horizonta	Root Mea	ID	Source	Location S	Magnitud	Status		
2	#####	13:44:18	19.246	145.62	Earthquak	131.6			6	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
3	#####	11:29:49	1.863	127.35	Earthquak	80			5.8	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
4	#####	18:05:58	-20.579	-173.97	Earthquak	20			6.2	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
5	#####	18:49:43	-59.076	-23.557	Earthquak	15			5.8	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
6	#####	13:32:50	11.938	126.43	Earthquak	15			5.8	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
7	#####	13:36:32	-13.405	166.63	Earthquak	35			6.7	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
8	#####	13:32:25	27.357	87.867	Earthquak	20			5.9	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
9	#####	23:17:42	-13.309	166.21	Earthquak	35			6	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
10	#####	11:32:37	-56.452	-27.043	Earthquak	95			6	MW							ISCGEMSL	ISCGEMSL	ISCGEM	ISCGEM	Automatic		
11	#####	10:43:17	-24.563	178.49	Earthquak	565			5.8	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
12	#####	20:57:41	-6.807	108.99	Earthquak	227.9			5.9	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
13	#####	0:11:17	-2.608	125.95	Earthquak	20			8.2	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
14	#####	9:35:30	54.636	161.7	Earthquak	55			5.5	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
15	#####	5:27:06	-18.697	-177.86	Earthquak	482.9			5.6	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
16	#####	15:56:51	37.523	73.251	Earthquak	15			6	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
17	#####	3:25:00	-51.84	139.74	Earthquak	10			6.1	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
18	#####	5:01:22	51.251	178.72	Earthquak	30.3			8.7	MW							OFFICIAL(	OFFICIAL	ISCGEM	OFFICIAL	Automatic		
19	#####	6:04:59	51.639	175.06	Earthquak	30			6	MW							ISCGEMSL	ISCGEMSL	ISCGEM	ISCGEM	Automatic		
20	#####	6:37:06	52.528	172.01	Earthquak	25			5.7	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
21	#####	6:39:32	51.626	175.75	Earthquak	25			5.8	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
22	#####	7:11:23	51.037	177.85	Earthquak	25			5.9	MW							ISCGEMSL	ISCGEMSL	ISCGEM	ISCGEM	Automatic		
23	#####	7:14:59	51.73	173.98	Earthquak	20			5.9	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
24	#####	7:23:12	51.775	173.06	Earthquak	10			5.7	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
25	#####	7:43:43	52.611	172.59	Earthquak	24			5.7	MW							ISCGEMSL	ISCGEMSL	ISCGEM	ISCGEM	Automatic		
26	#####	8:06:17	51.831	174.37	Earthquak	31.8			5.7	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		
27	#####	8:33:41	51.948	173.97	Earthquak	20			5.6	MW							ISCGEM8(	ISCGEM	ISCGEM	ISCGEM	Automatic		

2. **Data Preprocessing:**

Clean and preprocess the data to remove noise and inconsistencies.

Feature engineering: Extract relevant features like seismic activity patterns, fault lines, geological data, and more.

3. Ensemble Methods:

Implement ensemble methods like Random Forest or Gradient Boosting to combine multiple models for better accuracy.

Here's a high-level Python program using scikit-learn for ensemble methods and Keras for a simple deep learning architecture (a recurrent neural network) to get you started:

Program :

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import randomforestclassifier

from sklearn.metrics import accuracy_score
```

```
from keras.models import sequential
from keras.layers import lstm, dense

# load earthquake data (you'll need a relevant dataset)
data = pd.read_csv("earthquake_data.csv")

# preprocess the data, extract features, and target labels

# split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)

# ensemble method: random forest classifier
rf_model = randomforestclassifier(n_estimators=100, random_state=42)
rf_model.fit(x_train, y_train)
rf_predictions = rf_model.predict(x_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)

# deep learning architecture: lstm
model = sequential()
model.add(lstm(100, input_shape=(x_train.shape[1], x_train.shape[2]))
model.add(dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam')
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_test))

# evaluate the lstm model
lstm_predictions = model.predict(x_test)
lstm_accuracy = accuracy_score(y_test, (lstm_predictions > 0.5).astype(int))

print(f"random forest classifier accuracy: {rf_accuracy}")
```

print(f”lstm accuracy: {lstm_accuracy}")

4. **Deep Learning Architectures:**

Develop deep learning models using neural networks, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), to capture complex patterns in the data.

Design the architecture with careful consideration of the data and the problem.

5. Hyperparameter Tuning:

Utilize techniques like grid search or random search to find optimal hyperparameters for your models.

Consider techniques like Bayesian optimization for more efficient hyperparameter tuning.

Improving the performance of your earthquake prediction model through hyperparameter tuning and feature engineering is an iterative process. Here’s a Python program using scikit-learn and scikit-learn’s GridSearchCV for hyperparameter tuning, as well as some basic feature engineering techniques:

Program :

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split, gridsearchcv

from sklearn.ensemble import randomforestclassifier

from sklearn.metrics import accuracy_score

from sklearn.preprocessing import standardscaler

from sklearn.feature_selection import selectkbest, f_classif

from keras.models import sequential

from keras.layers import lstm, dense

from keras.wrappers.scikit_learn import kerasclassifier


# load earthquake data (you’ll need a relevant dataset)

data = pd.read_csv(“earthquake_data.csv”)
```

```
# preprocess the data, extract features, and target labels

# feature engineering: standardize features
scaler = standardscaler()
x_scaled = scaler.fit_transform(features)

# feature engineering: select k best features
selector = selectkbest(score_func=f_classif, k=10)
x_selected = selector.fit_transform(x_scaled, labels)

# split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x_selected, labels, test_size=0.2, random_state=42)

# hyperparameter tuning for random forest classifier
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30]
}
rf_model = randomforestclassifier(random_state=42)
grid_search = gridsearchcv(estimator=rf_model, param_grid=param_grid, cv=5)
grid_search.fit(x_train, y_train)
best_rf_model = grid_search.best_estimator_
rf_predictions = best_rf_model.predict(x_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)

# hyperparameter tuning for lstm (kerasclassifier)
def create_lstm_model(optimizer='adam'):
    model = sequential()
    model.add(lstm(100, input_shape=(x_train.shape[1],)))
```

```
    model.add(dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer=optimizer)
    return model

lstm_model = kerasclassifier(build_fn=create_lstm_model, verbose=0)
param_grid = {
    'optimizer': ['adam', 'rmsprop'],
    'epochs': [5, 10, 20],
    'batch_size': [16, 32, 64]
}
grid_search = gridsearchcv(estimator=lstm_model, param_grid=param_grid, cv=5)
grid_search.fit(x_train, y_train)
best_lstm_model = grid_search.best_estimator_
best_lstm_model.fit(x_train, y_train)
lstm_predictions = best_lstm_model.predict(x_test)
lstm_accuracy = accuracy_score(y_test, (lstm_predictions > 0.5).astype(int))

print(f"best random forest classifier accuracy: {rf_accuracy}")
print(f"best lstm accuracy: {lstm_accuracy}")
```

Model Training:

Split the data into training, validation, and testing sets.

Train your models on the training data while monitoring performance on the validation set.

7. **Evaluation Metrics:**

- Choose appropriate evaluation metrics, like Mean Absolute Error (MAE) or Mean Squared Error (MSE), depending on your specific problem.

8. **Feature Engineering:**

- Continuously refine and improve feature engineering to capture more relevant information.

- Explore advanced techniques such as Principal Component Analysis (PCA) or t-SNE for dimensionality reduction.

9. **Regularization:**

Apply techniques like dropout and L2 regularization to prevent overfitting, especially in deep learning models.

10. **Ensemble of Ensemble:**

Consider building an ensemble of your ensemble models to improve prediction performance further.

11. **Deployment and Monitoring:**

Deploy your model in an environment suitable for earthquake monitoring.

Continuously monitor and update the model as new data becomes available.

Conclusion :

Earthquake prediction is extremely complex, and even with advanced machine learning techniques, precise short-term earthquake prediction remains a significant challenge. This process focuses more on earthquake activity analysis and anomaly detection, which can provide valuable insights for early warning systems and disaster preparedness.