Project 7: Earthquake Prediction Model using Python

Problem Definition:

The problem is to develop an earthquake prediction model using a dataset. The goal of earthquake prediction is to give warning of potentially damaging earthquakes early enough to allow appropriate response to the disaster, enabling people to minimize loss of life and property.

Project Objectives:

Weather forecasting, which has significantly improved with the use of better satellites and more powerful mathematical models, earthquake prediction has been marred by repeated failure due to highly uncertain conditions of earth and its surroundings. Now, with the help of artificial intelligence, a growing number of scientists say changes in the way they can analyze massive amounts of seismic data can help them better understand earthquakes, anticipate how they will behave, and provide quicker and more accurate early warnings. This helps in hazzard assessments for many builders and real estate business for infrastructure planning from business perspective. Also many lives can be saved through early warning. This project aims a simple solution to above problem by predicting or forecasting place world wide to get hit by an earthquake, hence a realtime solution is provided.

Evaluating earthquake predictions :

Machine learning has the ability to advance our knowledge of earthquakes and enable more accurate forecasting and catastrophe response.

Earthquake prediction using machine learning is an active area of research that involves analyzing seismic data to predict the likelihood of an earthquake in a specific region. Here's an overview of the process

1. **Data Collection**: Gather relevant data such as seismic wave patterns, geological information, historical earthquake occurrences, etc. This data can be sourced from geological surveys and seismic monitoring stations.
2. **Preprocessing**: Preprocess the data to remove noise and transform it into a usable format. This may include normalization, feature extraction, and filling or removing missing values.
3. **Feature Engineering**: Identify significant features or variables that are likely to have an impact on earthquake prediction. This may include the velocity of seismic waves, the depth of seismic activity, and the location of faults.
4. **Model Selection**: Select appropriate machine learning models for prediction. Common algorithms used in earthquake prediction include Support Vector Machines (SVM), Neural Networks, Random Forests, and Gradient Boosting.

5. **Training**: Split the data into training and testing sets, then train the selected model on the training data. Hyperparameter tuning and cross-validation can help in selecting the best model.
6. **Prediction**: Use the trained model to predict the likelihood of an earthquake in the given area. The output can be a binary classification (earthquake/no earthquake) or a continuous value representing the probability.
7. **Evaluation**: Evaluate the model using metrics like accuracy, precision, recall, and F1 score to assess its effectiveness in predicting earthquakes.
8. **Deployment**: If the model proves effective, it can be deployed in real-time systems to provide warnings and help in disaster preparedness.
9. **Continuous Monitoring and Updating**: Continuously monitor and update the model with new data to ensure its effectiveness in predicting earthquakes.

MODELS USED IN EARTHQUAKE PREDICTION:

- Linear Regression
- Decision Tree
- K-Nearest Neighbors

STEPS TAKEN IN EARTHQUAKE PREDICTION MODEL:

1. Preprocessing of Dataset
2. Splitting the Datasets
3. Building ML models such as Linear Regression, Decision Tree and KNN
4. Visualization with Matplotlib and Seaborn
5. Prediction

METHODOLOGY :

1.Importing Libraries

The Python code to import libraries. We have used four libraries
Numpy is imported as np.
Pandas is imported as pd.
Matplotlib is imported as plt.
Seaborn is imported as sns.

2.Importing data:

The Python code for importing data from the appropriate directory or file and allocating it to a DataFrame. It imports the data that is kept in CSV format.

3.Checking for NaN:

Checking for NaN is a critical step in the pre-processing of data. We were only able to identify a few NaNs in this test.

4.Manipulating NaN values :

It is essential to remove the NaN values.

5. Train/Test split :

Creating train and test sets from the data is our next step towards developing a Machine Learning model. The Python code to divide the data set into train and test data

Evaluating earthquake predictions :

1. **Data Collection**
2. **Preprocessing**

3. **Feature Engineering**
4. **Model Selection**
5. **Training**
6. **Prediction**
7. **Evaluation**
8. **Deployment**
9. **Monitoring and Updating**

CODING FOR PROJECT :

Dataset Link:

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

Step 1: Import Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Load the dataset
data = pd.read_csv('/content/database.csv')
# Display the first few rows of the dataset
print(data.head())
```

output

```
Date Time Latitude Longitude Type Depth Depth Error \
0 01/02/1965 13:44:18 19.246 145.616 Earthquake 131.6 NaN
1 01/04/1965 11:29:49 1.863 127.352 Earthquake 80.0 NaN
2 01/05/1965 18:05:58 -20.579 -173.972 Earthquake 20.0 NaN
3 01/08/1965 18:49:43 -59.076 -23.557 Earthquake 15.0 NaN
4 01/09/1965 13:32:50 11.938 126.427 Earthquake 15.0 NaN
Depth Seismic Stations Magnitude Magnitude Type … \
0 NaN 6.0 MW ...
1 NaN 5.8 MW ...
2 NaN 6.2 MW ...
3 NaN 5.8 MW ...
4 NaN 5.8 MW ...
Magnitude Seismic Stations Azimuthal Gap Horizontal Distance \
0 NaN NaN NaN
1 NaN NaN NaN
2 NaN NaN NaN
3 NaN NaN NaN
4 NaN NaN NaN
Horizontal Error Root Mean Square ID Source Location Source \
0 NaN NaN ISCGEM860706 ISCGEM ISCGEM
1 NaN NaN ISCGEM860737 ISCGEM ISCGEM
2 NaN NaN ISCGEM860762 ISCGEM ISCGEM
3 NaN NaN ISCGEM860856 ISCGEM ISCGEM
4 NaN NaN ISCGEM860890 ISCGEM ISCGEM
Magnitude Source Status
0 ISCGEM Automatic
1 ISCGEM Automatic
2 ISCGEM Automatic
3 ISCGEM Automatic
4 ISCGEM Automatic
[5 rows x 21 columns]
```

Preprocessing the data:

# Select features (attributes) and target variable (magnitude of earthquake)
X = data.drop('magnitude', axis=1)
y = data['magnitude']

output:

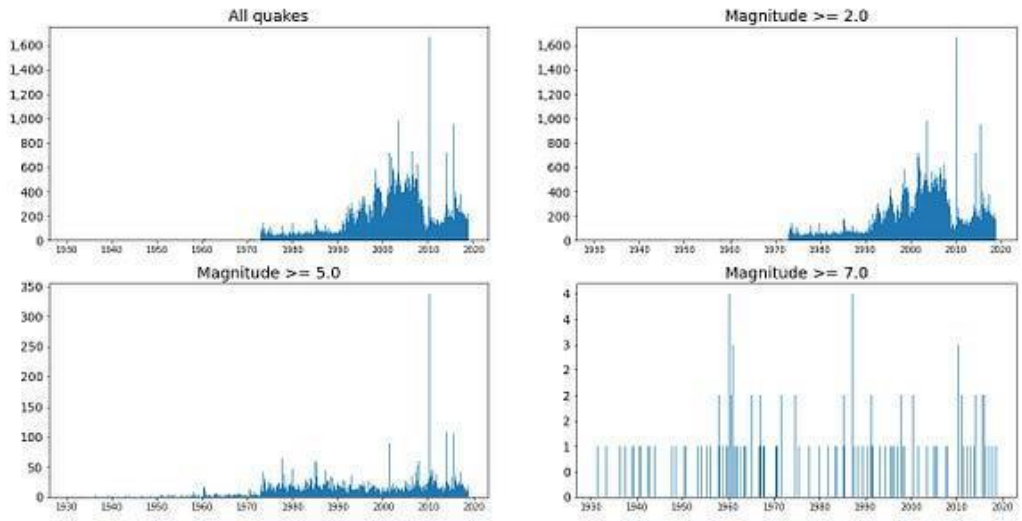
date	depth	mag	place	latitude	longitude	depth_avg_22	depth_avg_15	depth_avg_7	mag_avg_22	mag_avg_15	mag_avg_7	mag_outcome
2020-07-14	6.70	1.58	Oklahoma	36.171483	-97.718347	6.717727	6.560000	7.100000	1.352273	1.271333	1.357143	1.338571
2020-07-14	7.55	2.07	Oklahoma	36.171483	-97.718347	6.730000	6.682667	7.132857	1.372727	1.334667	1.527143	1.535714
2020-07-14	7.39	1.89	Oklahoma	36.171483	-97.718347	6.747727	6.708667	6.940000	1.396818	1.377333	1.570000	1.335714
2020-07-15	7.75	1.48	Oklahoma	36.171483	-97.718347	6.834545	6.764000	6.848571	1.383182	1.388667	1.581429	1.251429
2020-07-15	7.81	1.50	Oklahoma	36.171483	-97.718347	6.841364	6.854667	6.964286	1.404545	1.385333	1.602857	1.291429

Feature Engineering:

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))
output:

Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
Output:

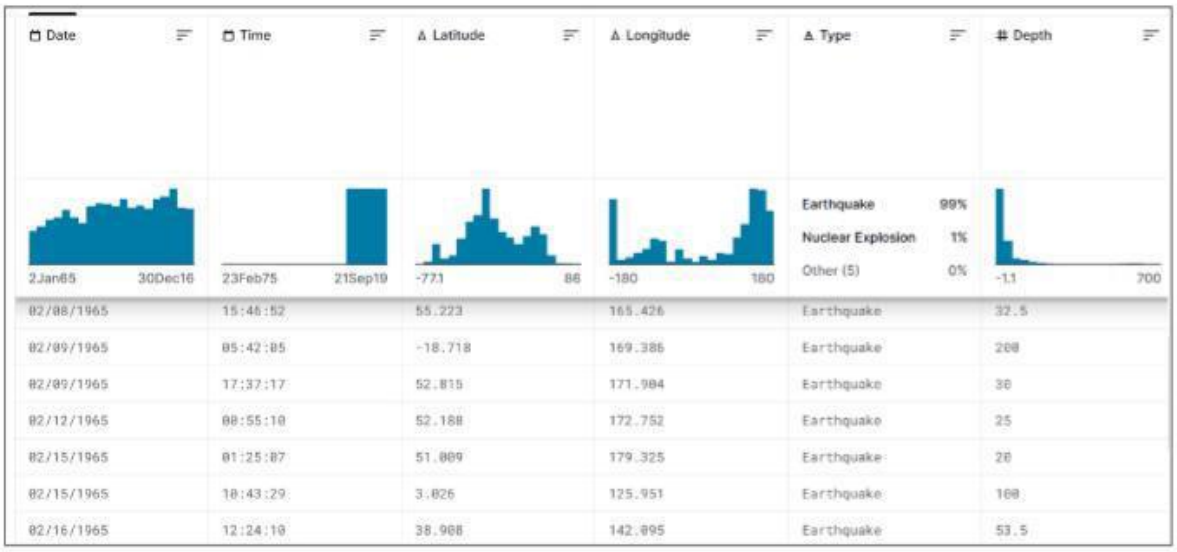


Program:

X = final_data[['Timestamp', 'Latitude', 'Longitude']]
y = final_data[['Magnitude', 'Depth']]
X_train,X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Output:

Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
'Depth Seismic Stations', 'Magnitude', 'Magnitude Type',
'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
'Source', 'Location Source', 'Magnitude Source', 'Status'],
dtype='object')



02/08/1965	15:46:52	55.223	165.426	Earthquake	32.5
02/09/1965	05:42:05	-18.718	169.386	Earthquake	200
02/09/1965	17:37:17	52.815	171.984	Earthquake	30
02/12/1965	08:55:10	52.188	172.752	Earthquake	25
02/15/1965	01:25:07	51.009	179.325	Earthquake	20
02/15/1965	10:43:29	3.026	125.951	Earthquake	100
02/16/1965	12:24:10	38.908	142.095	Earthquake	53.5

reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
reg.predict(X_test)

output:

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW

Magnitude Error	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Error	Root Mean Square	ID	Source	Location Source	Magnitu Source
NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860706	ISCGEM	ISCGEM	ISCGEM
NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860737	ISCGEM	ISCGEM	ISCGEM
NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860762	ISCGEM	ISCGEM	ISCGEM
NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860856	ISCGEM	ISCGEM	ISCGEM
NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860890	ISCGEM	ISCGEM	ISCGEM