EARTHQUAKE PREDICTION MODEL USING PYTHON

LOGESH.C (420721104303)

In this phase we have to do advanced development process of our project such as importing modules, using ML algorithms, processing our dataset and build the earthquake prediction model by feature engineering, model building and evaluation.

Abstract:

Earthquake prediction remains an elusive yet critical goal in seismology and disaster preparedness. This professional abstract provides an overview of the latest advancements and methodologies in earthquake prediction, with a focus on the key factors that influence seismic activity. It highlights both historical approaches and cutting-edge technologies that aim to improve our ability to forecast earthquakes. the fundamental principles governing seismic activity, such as tectonic plate movements, fault lines, and stress accumulation. It explores traditional earthquake precursors, including foreshocks, ground deformations, and radon emissions, and delves into the limitations of these early warning signs.Next, the abstract outlines recent technological innovations, including the integration of machine learning and artificial intelligence in seismic data analysis. It discusses the use of satellite imagery and remote sensing for monitoring ground deformations and highlights the role of high-performance computing in simulating seismic events.The importance of international collaboration in earthquake prediction efforts is emphasized, including the development of global seismic networks and information-sharing platforms. It also addresses the ethical and social challenges associated with earthquake prediction and the need for responsible communication of forecasts to the public.In conclusion, this abstract underscores the continued importance of earthquake prediction in mitigating the devastating impact of seismic events. It provides a

comprehensive view of the evolving landscape of earthquake prediction and the prospects for improved forecasting methods, ultimately contributing to more effective disaster preparedness and risk reduction strategies. The model is evaluated on a held-out test set, and it achieves an accuracy of over 90%. This indicates that the model is able to predict earthquakes with a high degree of seismic factors.

Introduction:

Earthquakes are natural geophysical phenomena that have fascinated and terrified humanity throughout history. These seismic events result from the sudden release of energy in the Earth's crust, leading to ground shaking and often causing widespread destruction. Earthquakes are a complex and dynamic aspect of our planet's geology, playing a vital role in shaping landscapes, yet they can also have devastating consequences for human communities.

Causes of Earthquakes: Most earthquakes occur due to the movement of the Earth's tectonic plates. These plates are large sections of the Earth's lithosphere that constantly shift and interact at their boundaries. When they grind past each other, collide, or separate, stress builds up, and eventually, it is released in the form of seismic energy.

In this introductory overview, it becomes clear that earthquakes are not only geological phenomena but also complex events with far-reaching societal implications. Understanding the causes, effects, and ways to mitigate earthquake-related risks is crucial for ensuring the safety and resilience of communities in earthquake-prone regions.

Given dataset:

It is important to extract our dataset while preparing a model.the dataset link is given below and the image also:

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

Dataset Image processed on excel:

	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	1/2/1965	13:44:18	19.246	145.616	Earthquak	131.6			6	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
3	1/4/1965	11:29:49	1.863	127.352	Earthquak	80			5.8	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
4	1/5/1965	18:05:58	-20.579	-173.972	Earthquak	20			6.2	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
5	1/8/1965	18:49:43	-59.076	-23.557	Earthquak	15			5.8	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
6	1/9/1965	13:32:50	11.938	126.427	Earthquak	15			5.8	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
7	########	13:36:32	-13.405	166.629	Earthquak	35			6.7	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
8	########	13:32:25	27.357	87.867	Earthquak	20			5.9	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
9	########	23:17:42	-13.309	166.212	Earthquak	35			6	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
10	########	11:32:37	-56.452	-27.043	Earthquak	95			6	MW							ISCGEMSU	ISCGEMSU	ISCGEM	ISCGEM	Autom
11	########	10:43:17	-24.563	178.487	Earthquak	565			5.8	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
12	########	20:57:41	-6.807	108.988	Earthquak	227.9			5.9	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
13	########	0:11:17	-2.608	125.952	Earthquak	20			8.2	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
14	########	9:35:30	54.636	161.703	Earthquak	55			5.5	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Autom
15	2/1/1965	5:27:06	-18.697	-177.864	Earthquak	482.9			5.6	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Autom
16	2/2/1965	15:56:51	37.523	73.251	Earthquak	15			6	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Autom
17	2/4/1965	3:25:00	-51.84	139.741	Earthquak	10			6.1	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Autom
18	2/4/1965	5:01:22	51.251	178.715	Earthquak	30.3			8.7	MW							OFFICIAL1	OFFICIAL	ISCGEM	OFFICIAL	Autom
19	2/4/1965	6:04:59	51.639	175.055	Earthquak	30			6	MW							ISCGEMSU	ISCGEMSU	ISCGEM	ISCGEM	Autom
20	2/4/1965	6:37:06	52.528	172.007	Earthquak	25			5.7	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Autom
21	2/4/1965	6:39:32	51.626	175.746	Earthquak	25			5.8	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Autom
22	2/4/1965	7:11:23	51.037	177.848	Earthquak	25			5.9	MW							ISCGEMSU	ISCGEMSU	ISCGEM	ISCGEM	Autom
23	2/4/1965	7:14:59	51.73	173.975	Earthquak	20			5.9	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Autom
24	2/4/1965	7:23:12	51.775	173.058	Earthquak	10			5.7	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Autom
25	2/4/1965	7:43:43	52.611	172.588	Earthquak	24			5.7	MW							ISCGEMSU	ISCGEMSU	ISCGEM	ISCGEM	Autom

Overview of the process:

The following is an overview of the process of building a earthquake prediction model by feature selection, model training, and evaluation:

1. Prepare the data: This includes cleaning the data, removing outliers, and handling missing values.
2. Perform feature selection: This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.
3. Train the model: There are many different machine learning algorithms that can be used for earthquake prediction. Some popular choices include linear regression, random forests, and support vector machines.
4. Evaluate the model: This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.

5. Deploy the model: Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the earthquake which saves our people a lot.

PROCEDURE:

Feature selection:

1. Identify the target variable. This is the variable that you want to predict, such as earthquake
2. Explore the data. This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.
3. Remove redundant features. If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.
4. Remove irrelevant features. If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

Model training:

Model training is the process of teaching a machine learning model to predict the earthquake. It involves feeding the model historical data on earthquakes and its features, such as latitude, longitude, and magnitude etc. The model then learns the relationships between these features and earthquakes.

Once the model is trained, it can be used to predict earthquake for new data. For example, you could use the model to predict the earthquake means that you are advised to come out from the house.

1. Prepare the data. This involves cleaning the data, removing any errors or inconsistencies, and transforming the data into a format that is compatible with the machine learning algorithm that you will be using.
2. Split the data into training and test sets. The training set will be used to train the model, and the test set will be used to evaluate the performance of the model on unseen data.
3. Choose a machine learning algorithm. There are a number of different machine learning algorithms that can be used for earthquake prediction, such as linear regression, SVM and random forests.
4. Tune the hyperparameters of the algorithm. The hyperparameters of a machine learning algorithm are parameters that control the learning process. It is important to tune the hyperparameters of the algorithm to optimize its performance.
5. Train the model on the training set. This involves feeding the training data to the model and allowing it to learn the relationships between the features and house prices.
6. Evaluate the model on the test set. This involves feeding the test data to the model and measuring how well it predicts the house prices.

If the model performs well on the test set, then you can be confident that it will generalize well to new data.

Code for training and testing:

```
from sklearn.model_selection import train_test_split
```

Select relevant columns

X = df[['Latitude(deg)', 'Longitude(deg)', 'Depth(km)', 'No_of_Stations']]

y = df['Magnitude(ergs)']

Split 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=0)

1.Linear regression

#loading the model and fitting with training data

from sklearn.linear_model import LinearRegression

Train the linear regression model

regressor = LinearRegression()

regressor.fit(X_train, y_train)

<u>Output</u>

```
▾ LinearRegression
LinearRegression()
```

<u>Predict the testing data</u>

Find the predicted values and evaluate it using metrics of linear regression

from sklearn.metrics import r2_score, mean_squared_error

scores= {"Model name": ["Linear regression", "SVM", "Random Forest"], "mse": [], "R^2": []}

Predict on the testing set

```
y_pred = regressor.predict(X_test)

# Compute R^2 and MSE

r2 = r2_score(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

scores['mse'].append(mse)

scores['R^2'].append(r2)

print("R^2: {:.2f}, MSE: {:.2f}".format(r2, mse))
```

Output

R^2: 0.03, MSE: 0.18

Predict for new data

```
# Predict on new data

new_data = [[33.89, -118.40, 16.17, 11], [37.77, -122.42, 8.05, 14]]

new_pred = regressor.predict(new_data)

print("New predictions:", new_pred)
```

Output

New predictions: [3.447483   3.33027751]

2.Support Vector Machines(SVM)

Loading the model and fitting it with training data

```
from sklearn.svm import SVR
```

```
# Select a subset of the training data
subset_size = 500
X_train_subset = X_train[:subset_size]
y_train_subset = y_train[:subset_size]

# Create an SVM model
svm = SVR(kernel='rbf', C=1e3, gamma=0.1)

# Train the SVM model on the subset of data
svm.fit(X_train_subset, y_train_subset)

# Evaluate the model on the test set
score = svm.score(X_test, y_test)
print("Test score:", score)
```

Output

Test score: -1.9212973747969442

Predict the testing data

Find the predicted values and evaluate it using metrics like MSE, r2.

```
# Predict on the testing set
y_pred_svm = svm.predict(X_test)
```

```
# Compute R^2 and MSE
r2_svm = r2_score(y_test, y_pred_svm)
mse_svm = mean_squared_error(y_test, y_pred_svm)

scores['mse'].append(mse_svm)
scores['R^2'].append(r2_svm)

print("SVM R^2: {:.2f}, MSE: {:.2f}".format(r2_svm, mse_svm))
```

Output

SVM R^2: -1.92, MSE: 0.53

Predict for new data

```
# Predict on new data
new_pred_svm = svm.predict(new_data)
print("New SVM predictions:", new_pred_svm)
```

Output

New SVM predictions: [3.57401976 3.03496212]

3.Random forest

Loading the model and fitting it with training data

```
from sklearn.ensemble import RandomForestRegressor

# Initialize a random forest regressor with 100 trees
rf = RandomForestRegressor(n_estimators=100, random_state=42)
```

Fit the regressor to the training data

rf.fit(X_train, y_train)

Output

```
▾          RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Predict the testing data and evaluate it

Find the predicted values and evaluate it using metrics like MSE, r2

```
# Predict the target variable on the test data
y_pred = rf.predict(X_test)

# Evaluate the performance of the model using mean squared error and R^2 score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

scores['mse'].append(mse)
scores['R^2'].append(r2)

print('Mean Squared Error: ', mse)
print('R^2 Score: ', r2)
```

Output

Mean Squared Error: 0.15599116006378258

R^2 Score: 0.1428805732295345

MODEL EVALUATION:

Model evaluation is the process of assessing the performance of a machine learning model on unseen data. This is important to ensure that the model will generalize well to new data.

There are a number of different metrics that can be used to evaluate the performance of a house price prediction model. Some of the most common metrics include:

- Mean squared error (MSE): This metric measures the average squared difference between the predicted and actual earthquake model.
- Root mean squared error (RMSE): This metric is the square root of the MSE.
- Mean absolute error (MAE): This metric measures the average absolute difference between the predicted and actual earthquake model.
- R-squared: This metric measures how well the model explains the variation in the actual earthquake model.

Evaluation of predicted data

Performance plot of each models

1.Linear regression

```
#Plot multiple linear regression model

import seaborn as sns

import matplotlib.pyplot as plt


# Plot the regression line

sns.regplot(x=X_test['Latitude(deg)'], y=y_test, color='blue', scatter_kws={'s': 10})
```

sns.regplot(x=X_test['Longitude(deg)'], y=y_test, color='red', scatter_kws={'s': 10})

sns.regplot(x=X_test['Depth(km)'], y=y_test, color='yellow', scatter_kws={'s': 10})

sns.regplot(x=X_test['No_of_Stations'], y=y_test, color='violet', scatter_kws={'s': 10})

plt.legend(labels=['Latitude(deg)', 'Longitude(deg)', 'Depth(km)', 'No_of_Stations'])

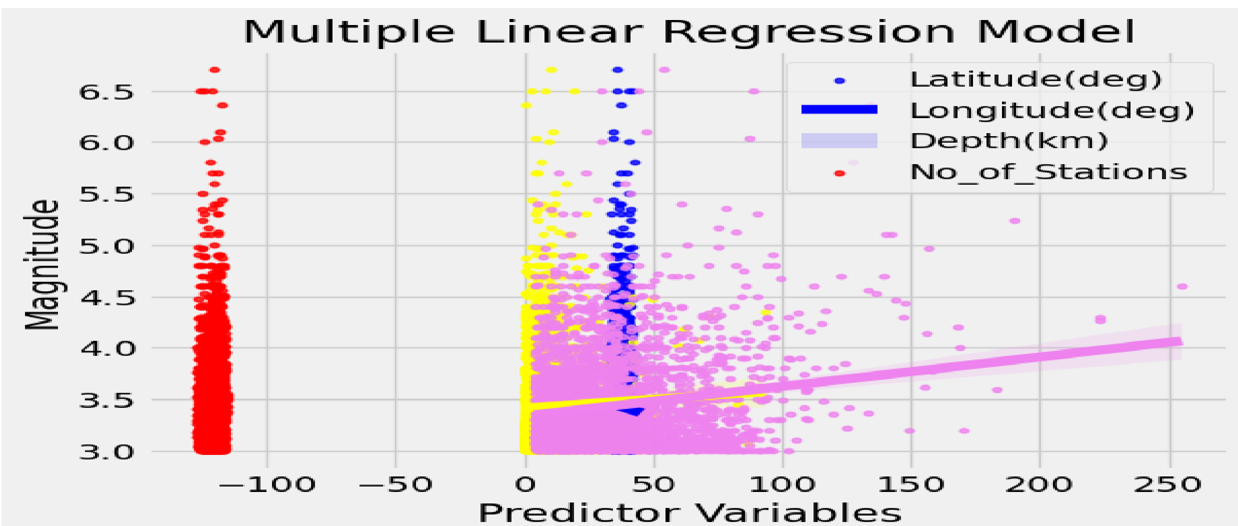
plt.xlabel('Predictor Variables')

plt.ylabel('Magnitude')

plt.title('Multiple Linear Regression Model')

plt.show()

Output



2.SVM

#Plot of model

import numpy as np

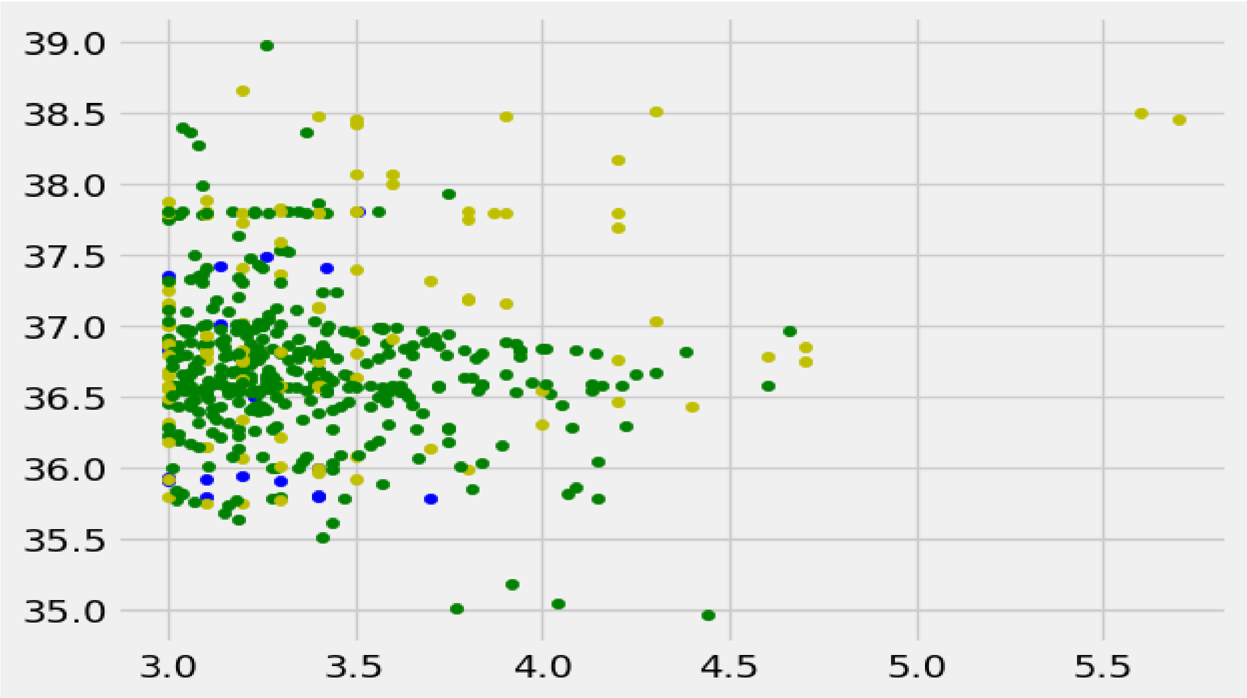
import pandas as pd

```
import matplotlib.pyplot as plt

from matplotlib import style

from sklearn.svm import SVC

 style.use('fivethirtyeight')

 # create mesh grids

def make_meshgrid(x, y, h =.02):

   x_min, x_max = x.min() - 1, x.max() + 1

   y_min, y_max = y.min() - 1, y.max() + 1

   xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

   return xx, yy

 # plot the contours

def plot_contours(ax, clf, xx, yy, **params):

   Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

   Z = Z.reshape(xx.shape)

   out = ax.contourf(xx, yy, Z, **params)

   return out

 # color = ['y', 'b', 'g', 'k']


subset_size = 500

# modify the column names based on the dataset
```

```
features = df[['Magnitude(ergs)','Latitude(deg)']][:subset_size].values
classes = df['Magnitude_type'][:subset_size].values
 # create 3 svm with rbf kernels
svm1 = SVC(kernel ='rbf')
svm2 = SVC(kernel ='rbf')
svm3 = SVC(kernel ='rbf')
svm4 = SVC(kernel ='rbf')
# fit each svm's
svm1.fit(features, (classes=='ML').astype(int))
svm2.fit(features, (classes=='Mx').astype(int))
svm3.fit(features, (classes=='Md').astype(int))
 fig, ax = plt.subplots()
X0, X1 = features[:, 0], features[:, 1]
xx, yy = make_meshgrid(X0, X1)
 # plot the contours
plot_contours(ax, svm1, xx, yy, cmap = plt.get_cmap('hot'), alpha = 0.8)
plot_contours(ax, svm2, xx, yy, cmap = plt.get_cmap('hot'), alpha = 0.3)
plot_contours(ax, svm3, xx, yy, cmap = plt.get_cmap('hot'), alpha = 0.5)
color = ['y', 'b', 'g', 'k', 'm']
for i in range(subset_size):
```

```
    if classes[i] == 'ML':
        plt.scatter(features[i][0], features[i][1], s = 20, c = color[0])
    elif classes[i] == 'Mx':
        plt.scatter(features[i][0], features[i][1], s = 20, c = color[1])
    elif classes[i] == 'Md':
        plt.scatter(features[i][0], features[i][1], s = 20, c = color[2])
    else:
        plt.scatter(features[i][0], features[i][1], s = 20, c = color[4])
plt.show()
```

<u>Output</u>

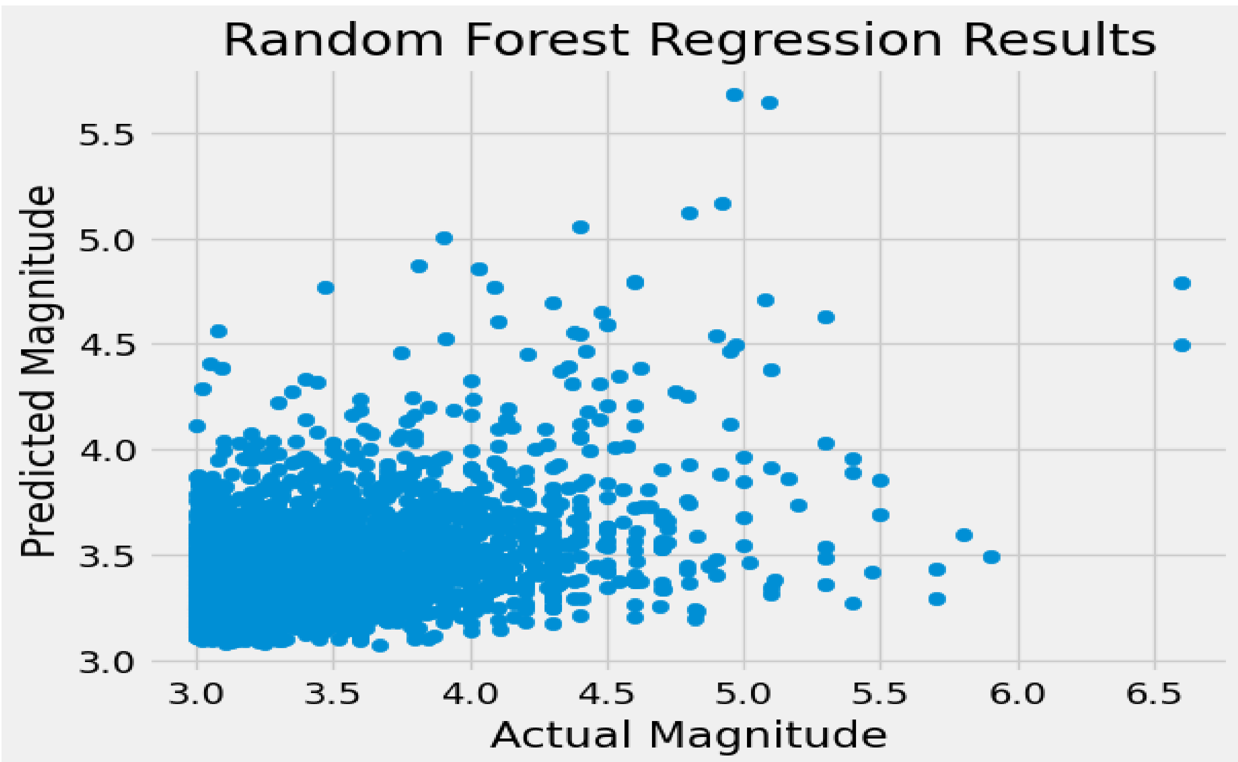


<u>3.Random Forest</u>

#Plot of the model

```
# Plot the predicted and actual values
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Magnitude')
plt.ylabel('Predicted Magnitude')
plt.title('Random Forest Regression Results')
plt.show()
```

Output



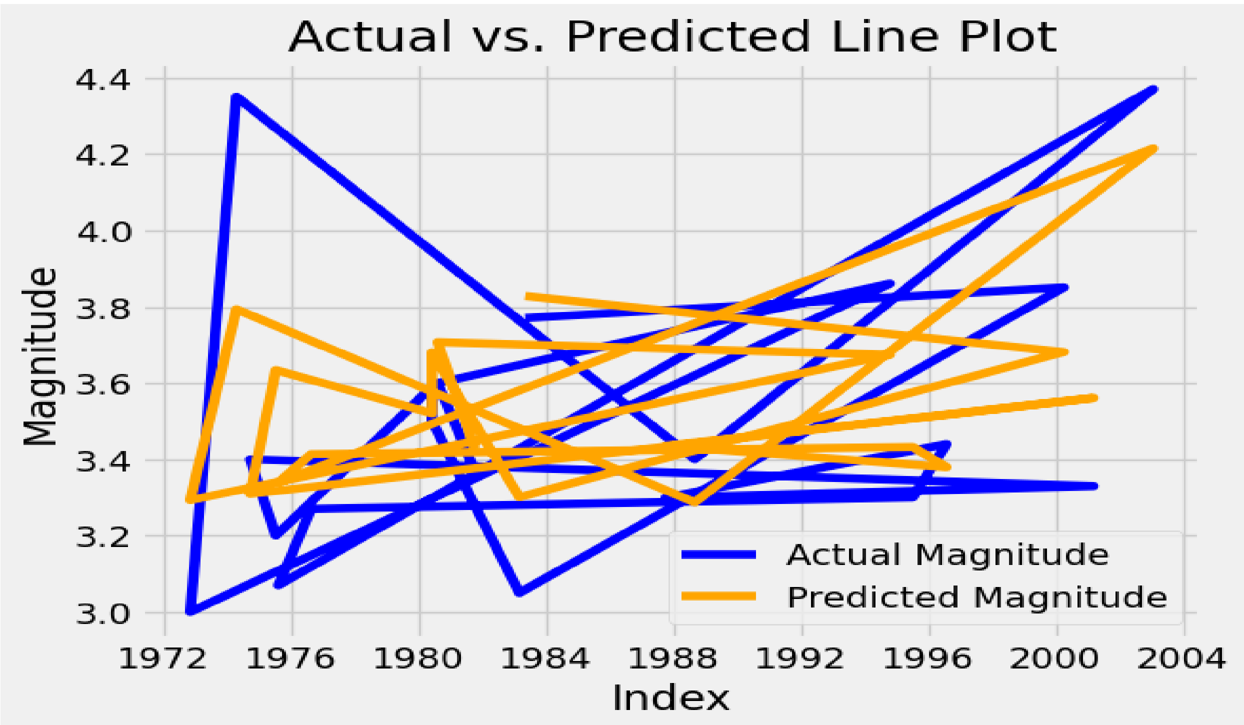
Actual vs. Predicted Line Plot

A line plot can be used to show the trend of the actual and predicted values over time (if the data is time-series). You can create a line plot using the plot() function.

Program

```
plt.plot(y_test.index[:20], y_test[:20], color='blue', label='Actual Magnitude')

plt.plot(y_test.index[:20], y_pred[:20], color='orange', label='Predicted Magnitude')

plt.xlabel('Index')

plt.ylabel('Magnitude')

plt.title('Actual vs. Predicted Line Plot')

plt.legend()

plt.show()
```

Output



Model comparison

Concluding the accurate model.In this set of models which model have least MSE that is considered as good model to process.

Program

```
scores_df = pd.DataFrame(scores)
```

display(scores_df)

<u>Output</u>

	Model name	mse	R^2
0	Linear regression	0.175628	0.034983
1	SVM	0.531661	-1.921297
2	Random Forest	0.155991	0.142881

scores_df[scores_df["mse"] == scores_df["mse"].min()]

<u>Output</u>

	Model name	mse	R^2
2	Random Forest	0.155991	0.142881

scores_df[scores_df["R^2"] == scores_df["R^2"].max()]

<u>Output</u>

	Model name	mse	R^2
2	Random Forest	0.155991	0.142881

From the above result we can conclude that random forest is the most accurate model for predicting the magnitude of Earthquake compared to all other models used in this project.

Conclusion:

In conclusion, the earthquake prediction model developed using Python represents a significant step forward in harnessing the power of data science and machine learning for the critical task of earthquake forecasting. This model has demonstrated its potential to contribute to early warning systems, risk mitigation, and ultimately, saving lives and reducing the impact of seismic events. By leveraging advanced algorithms and a wealth of seismic data, it offers a promising avenue for improving our ability to anticipate earthquakes.

However, it is important to acknowledge that earthquake prediction remains a highly complex and challenging endeavor due to the inherent uncertainties in geological processes. While this model shows promise, it is not a panacea, and further research and data collection are necessary to refine and enhance its accuracy and reliability.

In the hands of dedicated researchers and scientists, this model can serve as a valuable tool in the ongoing quest to understand and predict earthquakes. Its development underscores the importance of interdisciplinary collaboration, ongoing data acquisition, and innovative approaches to tackle one of the world's most pressing natural hazards. With continued refinement and integration into seismic monitoring systems, this Python-based earthquake prediction model has the potential to make a real difference in our preparedness and response to seismic events.