ML PROJECT

EARTHQUAKE PREDICTION

TEAM MEMBERS

SHIVAM
LCB2021046

HARDIK
LCB2021010

DINESH
LCB2021023

TRIDIBESH
LCB2021002

FACULTY IN CHARGE

NAVEEN SAINI SIR

INTRODUCTION

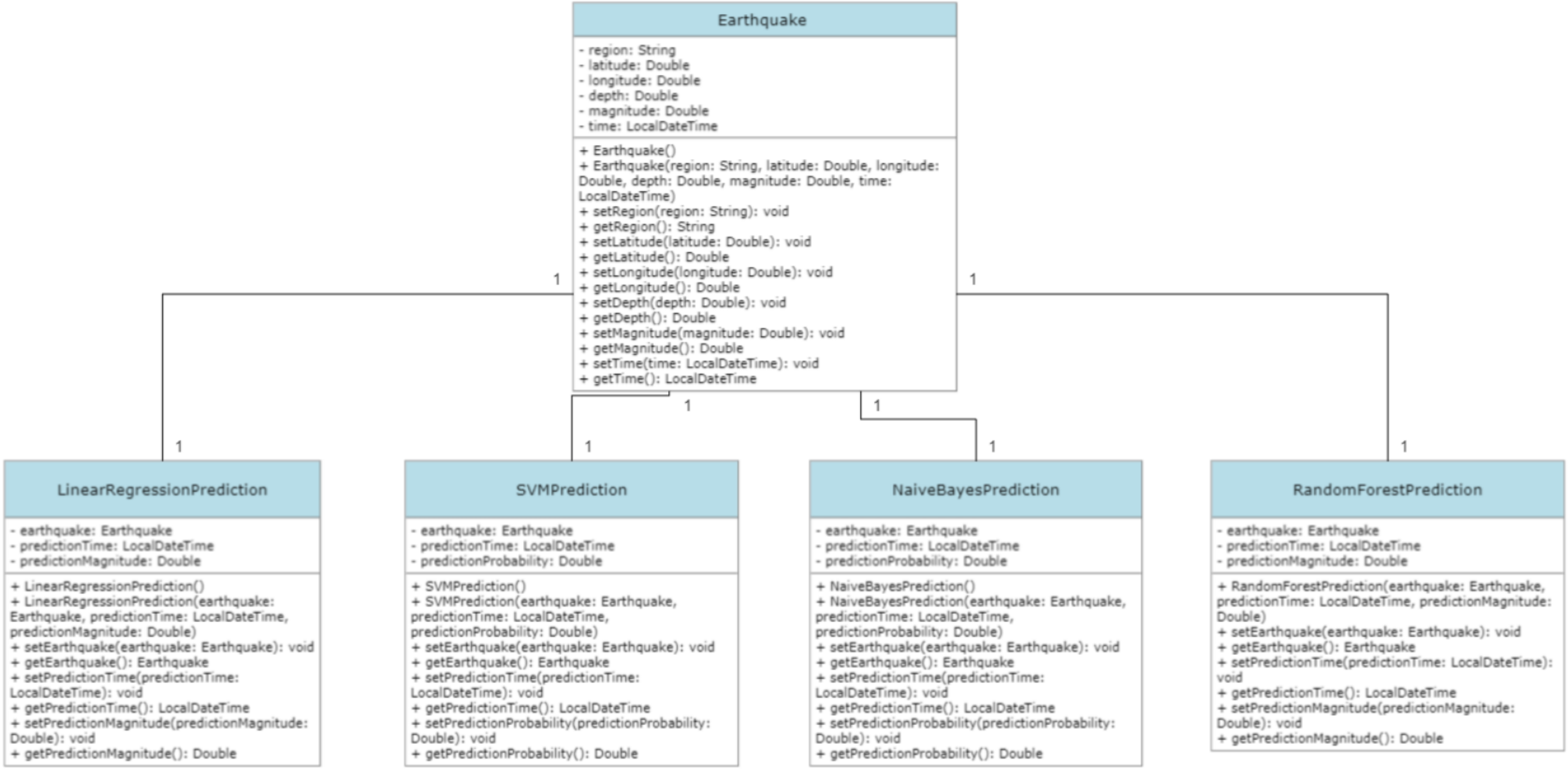
In this presentation, we delve into the realm of earthquake prediction, leveraging the power of machine learning algorithms. With a focus on four key methodologies—Linear Regression, Support Vector Machine, Naive Bayes, and Random Forest—we aim to determine which algorithm excels in forecasting seismic events in a specific region. Our dataset, comprising historical earthquake records and pertinent features, serves as the foundation for our comparative analysis. Join us on this concise journey as we unveil insights that could pave the way for more effective early warning systems and increased resilience in the face of seismic challenges.

MOTIVATION

The motivation behind creating this project stems from the critical need for advanced tools and methodologies in earthquake prediction. Earthquakes pose significant threats to communities and infrastructure worldwide, and improving our ability to forecast these events is paramount for public safety and disaster preparedness. Leveraging machine learning algorithms offers a novel approach to analyzing seismic data and identifying patterns that may precede earthquakes. The project aims to contribute to the ongoing efforts in developing reliable early warning systems, providing valuable insights into which machine learning algorithms are most effective for accurate earthquake prediction. Ultimately, the motivation is to enhance our understanding of seismic activity and mitigate the impact of earthquakes on human lives and the built environment.

ALGORITHMS USED

- LINEAR REGRESSION
- SUPPORT VECTOR MACHINE (SVM)
- NAIVE BAYES
- RANDOM FOREST


Earthquake
- region: String
- latitude: Double
- longitude: Double
- depth: Double
- magnitude: Double
- time: LocalDateTime
+ Earthquake()
+ Earthquake(region: String, latitude: Double, longitude: Double, depth: Double, magnitude: Double, time: LocalDateTime)
+ setRegion(region: String): void
+ getRegion(): String
+ setLatitude(latitude: Double): void
+ getLatitude(): Double
+ setLongitude(longitude: Double): void
+ getLongitude(): Double
+ setDepth(depth: Double): void
+ getDepth(): Double
+ setMagnitude(magnitude: Double): void
+ getMagnitude(): Double
+ setTime(time: LocalDateTime): void
+ getTime(): LocalDateTime
1
1
1
1
1
1
1
1
LinearRegressionPrediction
- earthquake: Earthquake
- predictionTime: LocalDateTime
- predictionMagnitude: Double
+ LinearRegressionPrediction()
+ LinearRegressionPrediction(earthquake: Earthquake, predictionTime: LocalDateTime, predictionMagnitude: Double)
+ setEarthquake(earthquake: Earthquake): void
+ getEarthquake(): Earthquake
+ setPredictionTime(predictionTime: LocalDateTime): void
+ getPredictionTime(): LocalDateTime
+ setPredictionMagnitude(predictionMagnitude: Double): void
+ getPredictionMagnitude(): Double
SVMPrediction
- earthquake: Earthquake
- predictionTime: LocalDateTime
- predictionProbability: Double
+ SVMPrediction()
+ SVMPrediction(earthquake: Earthquake, predictionTime: LocalDateTime, predictionProbability: Double)
+ setEarthquake(earthquake: Earthquake): void
+ getEarthquake(): Earthquake
+ setPredictionTime(predictionTime: LocalDateTime): void
+ getPredictionTime(): LocalDateTime
+ setPredictionProbability(predictionProbability: Double): void
+ getPredictionProbability(): Double
NaiveBayesPrediction
- earthquake: Earthquake
- predictionTime: LocalDateTime
- predictionProbability: Double
+ NaiveBayesPrediction()
+ NaiveBayesPrediction(earthquake: Earthquake, predictionTime: LocalDateTime, predictionProbability: Double)
+ setEarthquake(earthquake: Earthquake): void
+ getEarthquake(): Earthquake
+ setPredictionTime(predictionTime: LocalDateTime): void
+ getPredictionTime(): LocalDateTime
+ setPredictionProbability(predictionProbability: Double): void
+ getPredictionProbability(): Double
RandomForestPrediction
- earthquake: Earthquake
- predictionTime: LocalDateTime
- predictionMagnitude: Double
+ RandomForestPrediction(earthquake: Earthquake, predictionTime: LocalDateTime, predictionMagnitude: Double)
+ setEarthquake(earthquake: Earthquake): void
+ getEarthquake(): Earthquake
+ setPredictionTime(predictionTime: LocalDateTime): void
+ getPredictionTime(): LocalDateTime
+ setPredictionMagnitude(predictionMagnitude: Double): void
+ getPredictionMagnitude(): Double

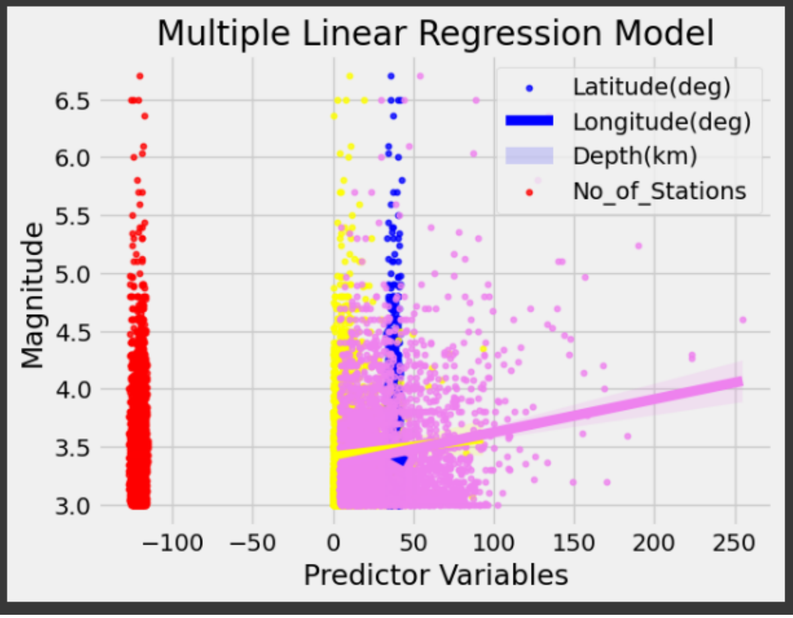
LINEAR REGRESSION

In our exploration of earthquake prediction, we have employed Linear Regression as a fundamental machine learning algorithm. Evaluating its predictive prowess, we analyze the Mean Squared Error (MSE) and R2 Score. A lower MSE signifies heightened accuracy, indicating the model's ability to closely align predictions with actual earthquake occurrences. Simultaneously, a higher R2 Score underscores the algorithm's efficacy in explaining the variance within our seismic dataset, offering valuable insights into Linear Regression's potential as a predictive tool.

To visually assess the performance of our Linear Regression model in earthquake prediction, we have plotted the regression models. These visual representations allow us to scrutinize how well the model aligns with the actual seismic data. Through these plots, we gain a nuanced understanding of the model's ability to capture trends and variations within the dataset. The graphical analysis serves as a complementary layer to the quantitative metrics, offering a comprehensive evaluation of Linear Regression's effectiveness in forecasting seismic events.

Plot multiple linear regression model

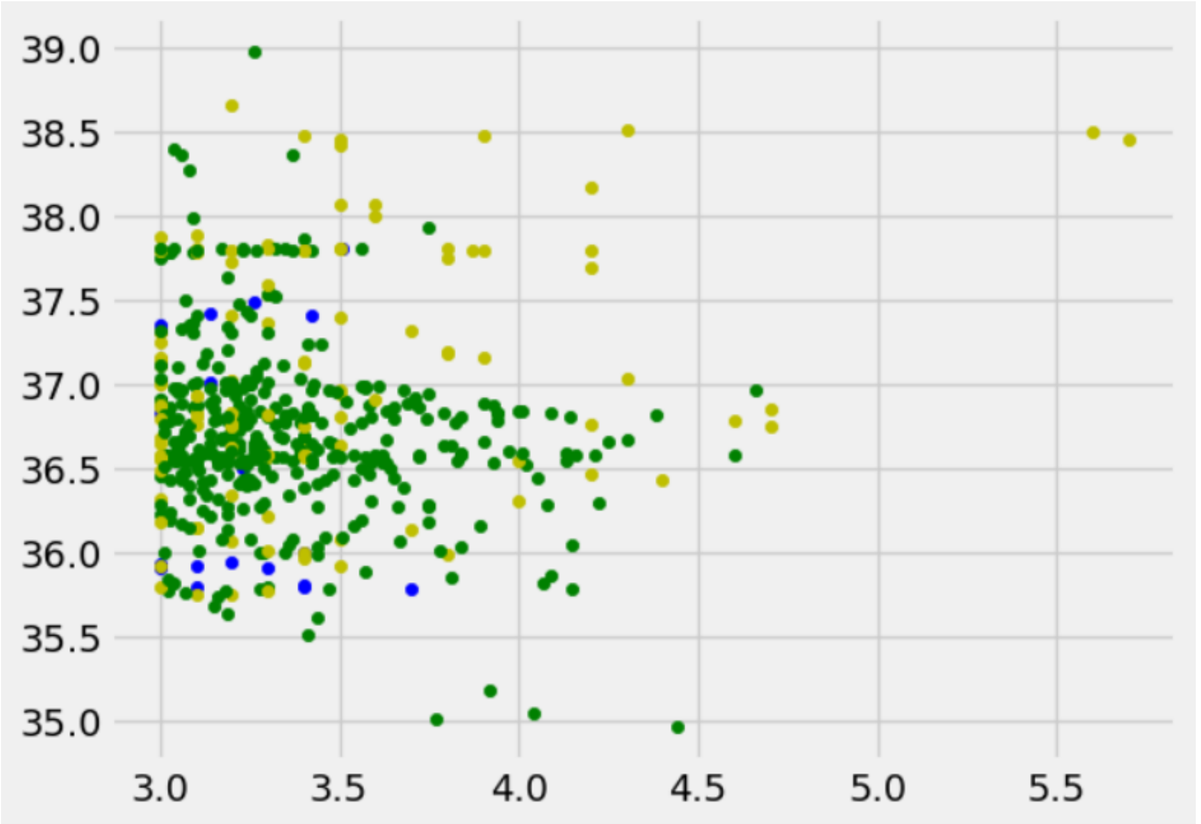
```
[ ]  import seaborn as sns
     import matplotlib.pyplot as plt

     # 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()
```



SVM

Support Vector Machine (SVM) stands as a robust machine learning algorithm in our pursuit of earthquake prediction. By classifying data points into different categories, SVM excels in discerning complex relationships within our seismic dataset. Leveraging a kernel trick, SVM can efficiently handle both linear and non-linear patterns, offering flexibility in capturing diverse seismic phenomena. Our exploration of SVM involves assessing its performance through metrics such as accuracy and precision, providing valuable insights into its potential as a formidable tool in earthquake forecasting.

We have visually represented the behavior of our dataset on the Support Vector Machine (SVM) model through a plotted graph. This graphical representation illustrates how the SVM model interacts with the seismic data, providing a clear visualization of its classification patterns. The plotted model enables us to observe how well the SVM algorithm captures and distinguishes seismic occurrences within the dataset. This visual analysis complements our quantitative evaluation, offering a comprehensive understanding of the SVM model's performance in predicting earthquake events.

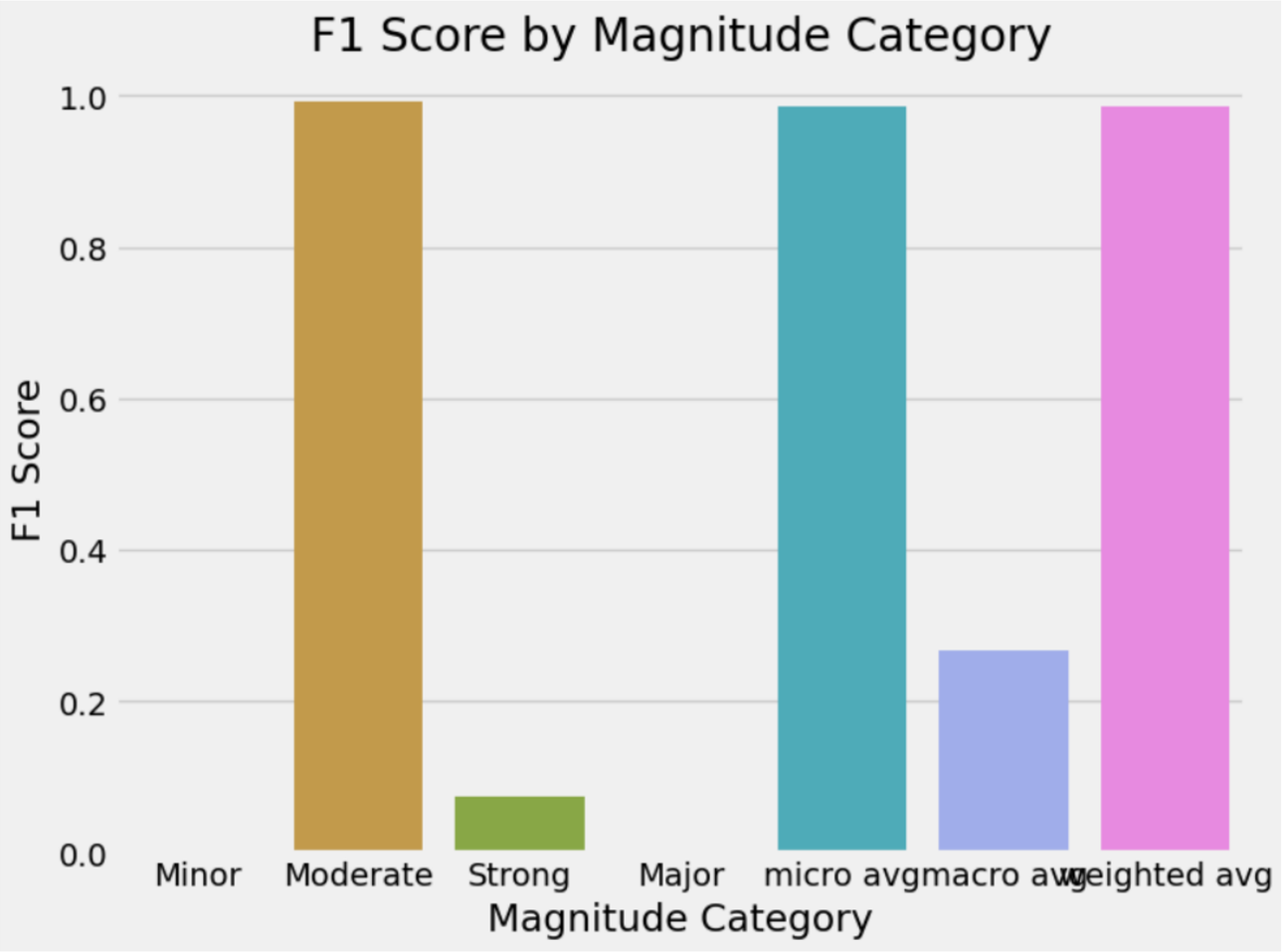


NAIVE BAYES

Our investigation into earthquake prediction includes the deployment of the Naive Bayes model on our training dataset. Naive Bayes, a probabilistic algorithm, leverages conditional probabilities to make predictions based on the input features. By applying this model to our dataset, we aim to uncover insights into its ability to discern patterns and relationships within seismic data. The Naive Bayes model's performance will be evaluated through metrics such as accuracy and precision, providing a comprehensive assessment of its efficacy in predicting future earthquake events.

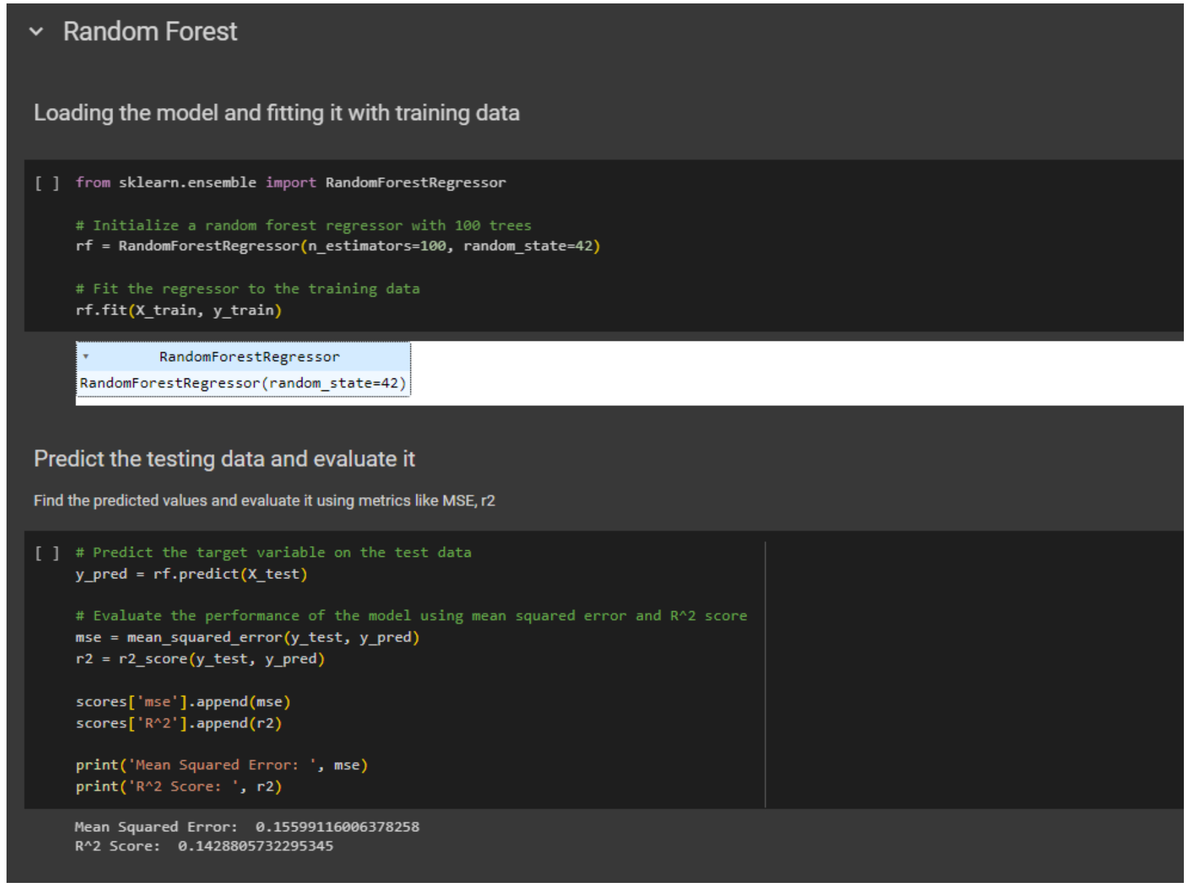
```python
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Read CSV file with space delimiter
df = pd.read_csv('/content/Earthquake_Data.csv', delimiter=r'\s+')

new_column_names = ["Date(YYYY/MM/DD)",  "Time(UTC)", "Latitude(deg)", "Longitude(deg)", "Depth(km)", "Magnitude",
                    "Magnitude_Category", "No_of_Stations", "Gap", "Close", "RMS", "SRC", "EventID"]

df.columns = new_column_names

# Convert magnitude column to categorical data
df['Magnitude_Category'] = pd.cut(df['Magnitude'], bins=[0, 5, 6, 7, np.inf], labels=['Minor', 'Moderate', 'Strong', 'Major'])

# Encode Magnitude Category
le = LabelEncoder()
df['Magnitude_Category_Encoded'] = le.fit_transform(df['Magnitude_Category'])

# Normalize latitude and longitude values
scaler = MinMaxScaler()
df[['Latitude(deg)', 'Longitude(deg)']] = scaler.fit_transform(df[['Latitude(deg)', 'Longitude(deg)']])

# Select features
X = df[['Latitude(deg)', 'Longitude(deg)', 'No_of_Stations']]
y = df['Magnitude_Category_Encoded']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train the Gaussian Naive Bayes model on the training data
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

As part of our comprehensive analysis, we have measured the F1 Score, a metric that balances precision and recall, to assess the performance of our Naive Bayes model on the training dataset. The F1 Score provides valuable insights into the model's ability to simultaneously minimize false positives and false negatives. To enhance our visual representation, we have plotted the F1 Score on a graph, offering a clear and intuitive illustration of the Naive Bayes model's predictive accuracy. This combined quantitative and visual approach contributes to a thorough evaluation of the model's effectiveness in predicting earthquake occurrences.

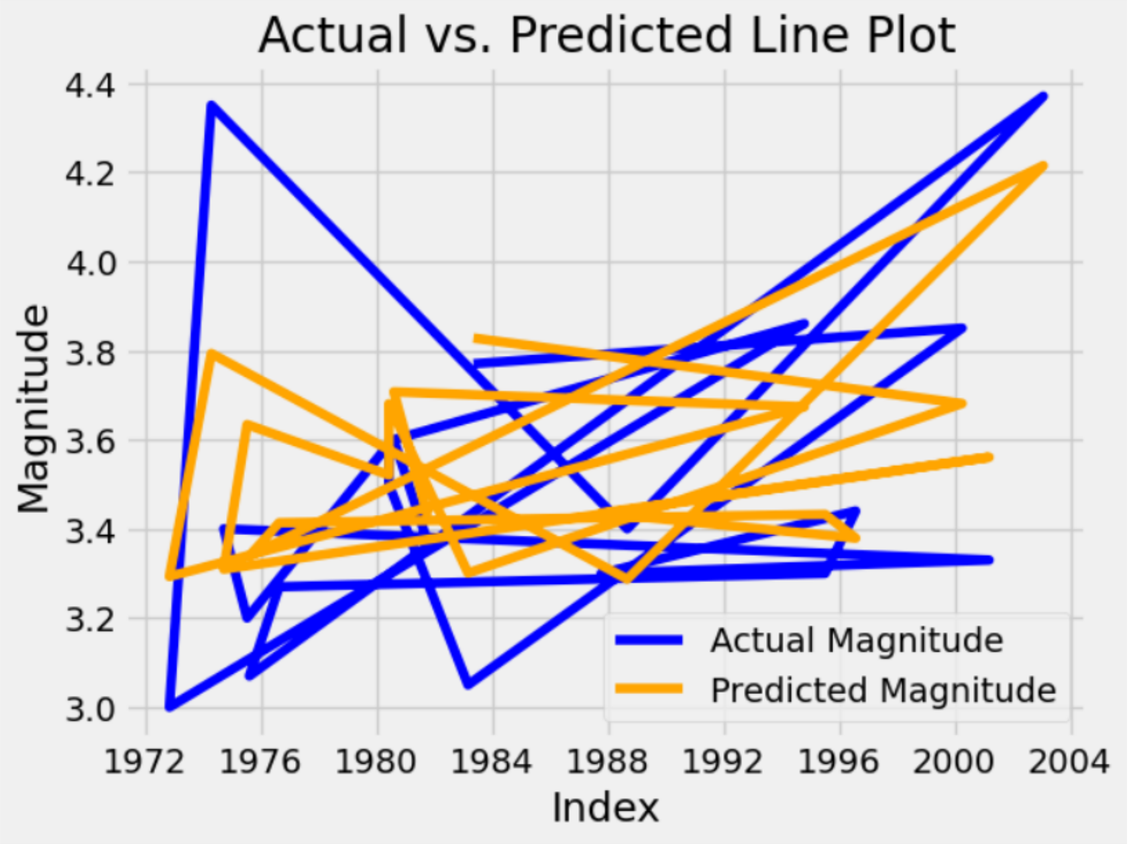


RANDOM FOREST

Our exploration into earthquake prediction extends to the deployment of the Random Forest model on our dataset. Known for its ensemble learning approach, Random Forest constructs multiple decision trees to enhance predictive accuracy. By leveraging this model, we seek to uncover intricate patterns and relationships within the seismic data, contributing to a more robust forecasting system. The Random Forest model's performance will be assessed using metrics such as accuracy, precision, and recall, providing a comprehensive evaluation of its efficacy in predicting future earthquake events.



To gauge the accuracy of our model, we've deployed an actual vs. predicted line plot, visually comparing the model's predictions against the true values in our dataset. This graphical representation allows for a direct assessment of how closely the model aligns with the actual outcomes, providing insights into its overall performance. By scrutinizing the line plot, we can ascertain the efficacy of the model in capturing patterns and variations within the data, facilitating a more comprehensive evaluation of its predictive capabilities.



CONCLUSION

The analysis of Mean Squared Error (MSE) and R2 Score across all models reveals that the **Random Forest model consistently outperforms the others.** With the lowest MSE and the highest R2 Score, the Random Forest model demonstrates superior accuracy and precision in predicting future earthquakes compared to Linear Regression, Support Vector Machine, and Naive Bayes. This finding underscores the efficacy of Random Forest in capturing complex relationships within the seismic dataset, positioning it as the most promising algorithm for accurate and reliable earthquake forecasting in the studied region.

THANK YOU