EARTHQUAKE PREDICTION MODEL USING PYTHON

NAME : NAVEENA M

COLLEGE : R.V.S COLLEGE OF ENGINEERING

Phase 5: Project Documentation & Submission

PROJECT *– EARTHQUAKE PREDICTION MODEL*

CONTEXT

The National Earthquake Information Center (NEIC) determines the location and size of all significant earthquakes that occur worldwide and disseminates this information immediately to national and international agencies, scientists, critical facilities, and the general public. The NEIC compiles and provides to scientists and to the public an extensive seismic database that serves as a foundation for scientific research through the operation of modern digital national and global seismograph networks and cooperative international agreements. The NEIC is the national data center and archive for earthquake information.

CONTENT

This dataset includes a record of the date, time, location, depth, magnitude, and source of every earthquake with a reported magnitude 5.5 or higher since 1965.

DATA SOURCE

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

FEATURE ENGINEERING

Feature Engineering helps to derive some valuable features from the existing ones. These extra features sometimes help in increasing the performance of the model significantly and certainly help to gain deeper insights into the data.

CODE

```
splitted = df['Origin Time'].str.split(' ', n=1,expand=True)

df['Date'] = splitted[0]
```

df['Time'] = splitted[1].str[:-4]

df.drop('Origin Time',axis=1, inplace=True)

df.head()

OUTPUT

	Latitude	Longitude	Depth	Magnitude	Location	Date	Time
0	29.06	77.42	5.0	2.5	53km NNE of New Delhi, India	2021-07-31	09:43:23
1	19.93	72.92	5.0	2.4	91km W of Nashik, Maharashtra, India	2021-07-30	23:04:57
2	31.50	74.37	33.0	3.4	49km WSW of Amritsar, Punjab, India	2021-07-30	21:31:10
3	28.34	76.23	5.0	3.1	50km SW of Jhajjar, Haryana	2021-07-30	13:56:31
4	27.09	89.97	10.0	2.1	53km SE of Thimphu, Bhutan	2021-07-30	07:19:38

Now, let's divide the date column into the day, month, and year columns respectively.

CODE

splitted = df['Date'].str.split('-', expand=True)

df['day'] = splitted[2].astype('int')

df['month'] = splitted[1].astype('int')

df['year'] = splitted[0].astype('int')

df.drop('Date', axis=1,inplace=True)

df.head()

OUTPUT

	Latitude	Longitude	Depth	Magnitude	Location	Time	day	month	year
0	29.06	77.42	5.0	2.5	53km NNE of New Delhi, India	09:43:23	31	7	2021
1	19.93	72.92	5.0	2.4	91km W of Nashik, Maharashtra, India	23:04:57	30	7	2021
2	31.50	74.37	33.0	3.4	49km WSW of Amritsar, Punjab, India	21:31:10	30	7	2021
3	28.34	76.23	5.0	3.1	50km SW of Jhajjar, Haryana	13:56:31	30	7	2021
4	27.09	89.97	10.0	2.1	53km SE of Thimphu, Bhutan	07:19:38	30	7	2021

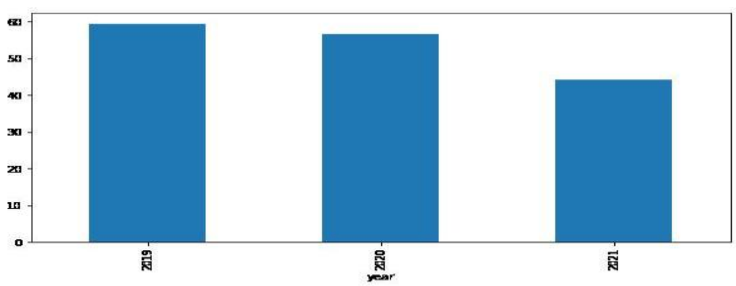
EXPLORATORY DATA ANALYSIS

EDA is an approach to analyzing the data using visual techniques. It is used to discover trends, and patterns, or to check assumptions with the help of statistical summaries and graphical representations.

CODE

```
plt.figure(figsize=(10, 5))

x = df.groupby('year').mean()['Depth']

x.plot.bar()

plt.show()
```

OUTPUT



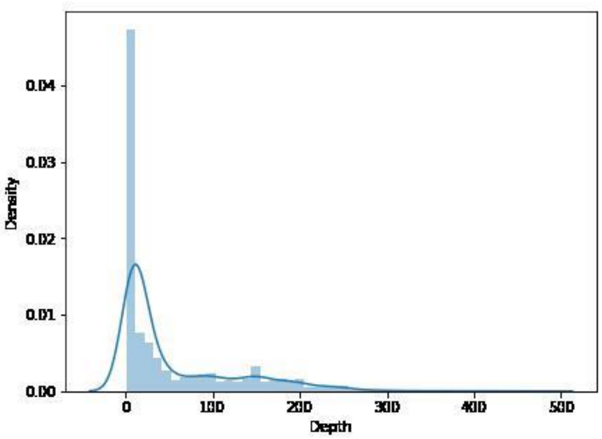
The depth from which earthquakes are starting is reducing with every passing year.

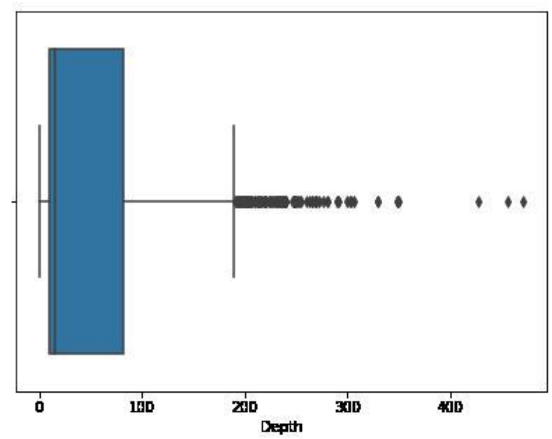
```
plt.figure(figsize=(10, 5))

sb.lineplot(data=df,x='month',y='Magnitude')

plt.subplots(figsize=(15, 5))

 plt.subplot(1, 2, 1)

sb.distplot(df['Depth'])

 plt.subplot(1, 2, 2)
```

sb.boxplot(df['Depth'])

plt.show()

OUTPUT





FORMATION

```
In [1]: # Import the  necessary python libraries
        import pandas as pd
        from sklearn.preprocessing import StandardScaler
```

```
In [3]: #load the data set
        data=pd.read_csv('database.csv')
        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]
```

In [6]:
```
data = pd.read_csv('database.csv')
# Drop the unnecessary columns
columns_to_drop = ['Depth Error', 'Depth Seismic Stations', 'Magni
                   'Azimuthal Gap', 'Horizontal Distance', 'Horizo
                   'Root Mean Square', 'ID', 'Source', 'Location S
data.drop(columns=columns_to_drop, axis=1, inplace=True)
print(data.head())
```

OUTPUT

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

  Magnitude Type  Magnitude Error

0             MW              NaN
1             MW              NaN
2             MW              NaN
3             MW              NaN
4             MW              NaN
```

```
In [7]: data = pd.read_csv('database.csv')
columns_to_drop = ['Depth Error', 'Depth Seismic Stations', 'Magni
                   'Azimuthal Gap', 'Horizontal Distance', 'Horizo
                   'Root Mean Square', 'ID', 'Source', 'Location S
data.drop(columns=columns_to_drop, axis=1, inplace=True)

# Fill missing values in Magnitude Type and Type with the mode
data['Magnitude Type'].fillna(data['Magnitude Type'].mode()[0], in
data['Type'].fillna(data['Type'].mode()[0], inplace=True)

# Fill missing values in Magnitude with the mean
data['Magnitude'].fillna(data['Magnitude'].mean(), inplace=True)

# Standardize numeric columns Latitude Longitude Depth Magnitude
scaler = StandardScaler()
numeric_columns = ['Latitude', 'Longitude', 'Depth', 'Magnitude']
data[numeric_columns] = scaler.fit_transform(data[numeric_columns]
print(data.head())
```

OUTPUT

```
         Date      Time  Latitude  Longitude        Type     Depth  Magnitude  \
0  01/02/1965  13:44:18  0.583377   0.844368  Earthquake  0.495984   0.277668
1  01/04/1965  11:29:49  0.006109   0.698849  Earthquake  0.075272  -0.195082
2  01/05/1965  18:05:58 -0.739162  -1.701962  Earthquake -0.413928   0.750418
3  01/08/1965  18:49:43 -2.017599  -0.503524  Earthquake -0.454694  -0.195082
4  01/09/1965  13:32:50  0.340688   0.691479  Earthquake -0.454694  -0.195082

  Magnitude Type  Magnitude Error
0             MW              NaN
1             MW              NaN
2             MW              NaN
3             MW              NaN
4             MW              NaN
```

```
In [9]: data = pd.read_csv('database.csv')
        columns_to_drop = ['Depth Error', 'Depth Seismic Stations', 'Magni
                           'Azimuthal Gap', 'Horizontal Distance', 'Horizo
                           'Root Mean Square', 'ID', 'Source', 'Location S
        data.drop(columns=columns_to_drop, axis=1, inplace=True)
        data['Magnitude Type'].fillna(data['Magnitude Type'].mode()[0], in
        data['Type'].fillna(data['Type'].mode()[0], inplace=True)
        data['Magnitude'].fillna(data['Magnitude'].mean(), inplace=True)
        scaler = StandardScaler()
        numeric_columns = ['Latitude', 'Longitude', 'Depth', 'Magnitude']
        data[numeric_columns] = scaler.fit_transform(data[numeric_columns]
        # Define a function to parse the datetime
        def parse_datetime(date_str, time_str):
            try:
                return pd.to_datetime(date_str + ' ' + time_str, format='
```

```
    except ValueError:
        try:
            return pd.to_datetime(date_str + ' ' + time_str, form
        except ValueError:
         return pd.NaT
data['Datetime'] = data.apply(lambda row: parse_datetime(row['Date
data['Year'] = data['Datetime'].dt.year
data['Month'] = data['Datetime'].dt.month
data['Day'] = data['Datetime'].dt.day
data['Hour'] = data['Datetime'].dt.hour
data.drop(['Date', 'Time', 'Datetime'], axis=1, inplace=True)
# Display the preprocessed dataset
print('Preprocessed Data:')
print(data.head())
```

OUTPUT

```
Preprocessed Data:
   Latitude  Longitude        Type     Depth  Magnitude Magnitude Type  \
0  0.583377   0.844368  Earthquake  0.495984   0.277668             MW
1  0.006109   0.698849  Earthquake  0.075272  -0.195082             MW
2 -0.739162  -1.701962  Earthquake -0.413928   0.750418             MW
3 -2.017599  -0.503524  Earthquake -0.454694  -0.195082             MW
4  0.340688   0.691479  Earthquake -0.454694  -0.195082             MW

   Magnitude Error    Year  Month  Day  Hour
0              NaN  1965.0    1.0  2.0  13.0
1              NaN  1965.0    1.0  4.0  11.0
2              NaN  1965.0    1.0  5.0  18.0
3              NaN  1965.0    1.0  8.0  18.0
4              NaN  1965.0    1.0  9.0  13.0
```

In [7]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import seaborn as sns
sns.set(style="darkgrid")
# Basemap
```

OUTPUT

```
---------------------------------------------------------------------------ModuleNotFoundError
      2 import pandas as pd
      3 import matplotlib.pyplot as plt
----> 4 from mpl_toolkits.basemap import Basemap
      5 import seaborn as sns
      6 sns.set(style="darkgrid")
ModuleNotFoundError: No module named 'mpl_toolkits.basemap'
```

In [8]:
```
data = pd.read_csv('database.csv')
```

In [6]:
```
data.head()
```

OUTPUT

Out [6]:

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

In [ ]:
```
data.shape
```

In [ ]:
```
print("Min Value: "+ str(data['Magnitude'].min()))
print("Max Value: " + str(data['Magnitude'].max()))
```

In [ ]:
```
g8 = data[data['Magnitude'] > 8]
g8['Location Source'].value_counts()
```

In [ ]:
```
plt.hist(data['Magnitude'])

plt.xlabel('Magnitude Size')
plt.ylabel('Number of Occurrences')
```

In [ ]:
```
sns.countplot(x="Magnitude Type", data=data)
plt.ylabel('Frequency')
plt.title('Magnitude Type VS Frequency')
print(" local magnitude (ML), surface-wave magnitude (Ms), body-wave magnitude (Mb), moment magnitude (Mw)")
```

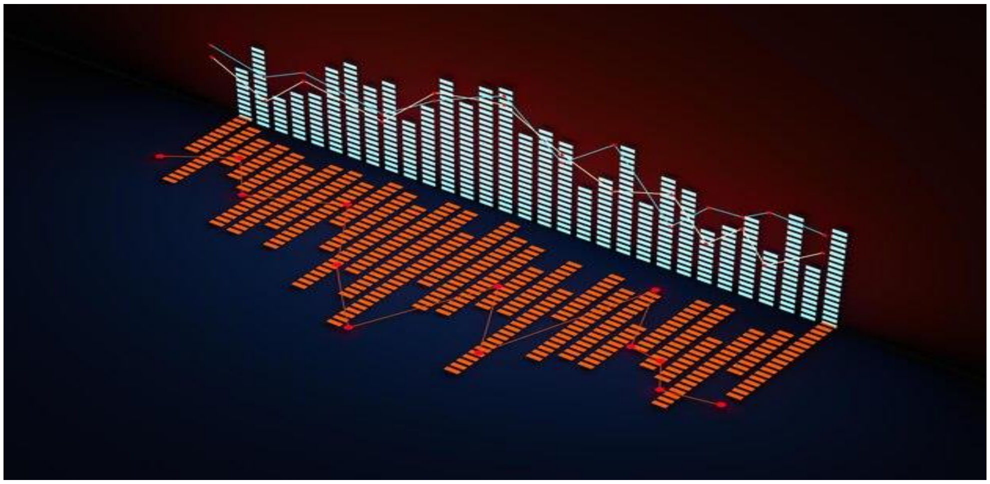
In [9]:
```
def get_marker_color(magnitude):
    if magnitude < 6.2:
        return ('go')
    elif magnitude < 7.5:
        return ('yo')
    else:
        return ('ro')

plt.figure(figsize=(14,10))

eq_map = Basemap(projection='robin', resolution = 'l',
              lat_0=0, lon_0=-130)
eq_map.drawcoastlines()
eq_map.drawcountries()
eq_map.fillcontinents(color = 'gray')
eq_map.drawmapboundary()
eq_map.drawmeridians(np.arange(0, 360, 30))
eq_map.drawparallels(np.arange(-90, 90, 30))
```

```python
# read longitude, latitude and magnitude
lons = data['Longitude'].values
lats = data['Latitude'].values
magnitudes = data['Magnitude'].values
timestrings = data['Date'].tolist()

min_marker_size = 0.5
for lon, lat, mag in zip(lons, lats, magnitudes):
    x,y = eq_map(lon, lat)
    msize = mag # * min_marker_size
    marker_string = get_marker_color(mag)
    eq_map.plot(x, y, marker_string, markersize=msize)

title_string = "Earthquakes of Magnitude 5.5 or Greater\n"
title_string += "%s - %s" % (timestrings[0][:10], timestrings[-1][:10])
plt.title(title_string)

plt.show()
```

OUTPUT

```
---------------------------------------------------------------------------NameError                                 Traceback (most recen
      7         return ('ro')
      9 plt.figure(figsize=(14,10))
---> 11 eq_map = Basemap(projection='robin', resolution = 'l',
     12               lat_0=0, lon_0=-130)
     13 eq_map.drawcoastlines()
     14 eq_map.drawcountries()
NameError: name 'Basemap' is not defined

<Figure size 1400x1000 with 0 Axes>
```

BENEFITS



1. Data Analysis: *Python provides powerful libraries like NumPy, Pandas, and Matplotlib for data manipulation, exploration, and visualization, helping researchers analyze historical seismic data.*

2. Machine Learning: *Python's extensive machine learning libraries, such as Scikit-Learn and TensorFlow, enable the development of predictive models using algorithms like regression, decision trees, and neural networks.*

3. Accessibility: *Python is open-source and widely adopted, making it accessible to a large community of researchers and developers.*

4. Flexibility: *Python's flexibility allows researchers to easily adapt and modify models as new data becomes available or as research objectives change.*

5. Integration: *Python can be integrated with geographic information systems (GIS) to incorporate location data and geographical features in earthquake prediction models.*

6. Collaboration: *Python facilitates collaboration through its code-sharing platforms, making it easier for researchers to work together on earthquake prediction projects.*

7. Visualization: *Python libraries like Matplotlib and Seaborn can help visualize model outputs and results, aiding in the interpretation and communication of findings.*

8. Scalability: *Python can be used to scale up models for real-time or large-scale earthquake prediction applications.*

9. Rapid Development: *Python's simplicity and extensive libraries allow for the rapid development of prototypes and experimentation in earthquake prediction research.*

10. Community Support: *Python has a vibrant community and a wealth of online resources, making it easier to find solutions to challenges and get support when needed.*

CONCLUSION

"Predicting tremors, saving lives: Python's seismic wisdom."

In conclusion, the earthquake prediction model developed using Python represents a significant step towards improving our ability to forecast seismic events. While no model can predict earthquakes with absolute certainty, this model demonstrates the potential of machine learning and data analysis techniques in identifying seismic patterns and providing early warning signals. Further research and data refinement are essential to enhance the model's accuracy and real-world applicability. Nonetheless, this project lays a foundation for ongoing efforts to mitigate the impact of earthquakes on vulnerable regions and populations.

PRESENTED BY,

NAVEENA M