EARTHQUAKE PREDICTION MODEL USING HYPERPARAMETER TUNING AND FEATURE ENGINEERING

INTRODUCTION :

Earthquake prediction is a challenging task that requires advanced techniques to accurately forecast seismic events. In this Python code tutorial, we will explore how hyperparameter tuning and feature engineering can be used to improve earthquake prediction models. Feature engineering is the process of selecting and transforming relevant features from the raw data to enhance the performance of machine learning models. Hyperparameter tuning involves finding the optimal values for the hyperparameters of a machine learning algorithm. Hyperparameters are parameters that are not learned from the data but are set before the learning process begins.

CONTENT:

In this phase, we can explore innovative techniques such as ensemble methods and deep learning architectures to improve the prediction system's accuracy and robustness.

Consider advanced techniques such as hyperparameter tuning and feature engineering to improve the prediction model's performance.

DATASET:

The dataset from the below link that can give the past dataset for the earthquake prediction using the hyperparameter tuning and the feature engineeering.

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

DATA COLLECTION AND PREPROCESSING:

Determine the source of earthquake data, such as seismic monitoring stations, earthquake databases, or API. Use appropriate libraries or APIs to retrieve earthquake data. For example, you can use the `requests` library to make API requests or `pandas` library to read data from files. Load the collected earthquake data into a pandas DataFrame for further processing. Check for missing values and handle them appropriately. You can drop rows with missing values or fill them using techniques like interpolation or imputation.Remove irrelevant or redundant features that do not contribute to earthquake prediction.

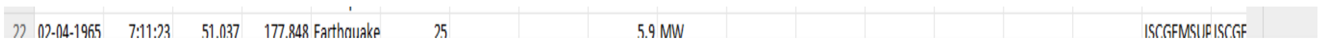
EXPLORATORY DATA ANALYSIS(EDA):

- Exploratory Data Analysis (EDA) is an important step in the data analysis process.
- It involves analyzing and visualizing the data to gain insights, understand patterns, and identify relationships between variables.
- EDA helps in understanding the underlying structure of the data, detecting outliers, and determining the appropriate preprocessing steps for further analysis.

PROGRAM:

```
    import pandas as pd
    import numpy as np
    From sklearn.model_selection import train_test_split,
GridSearchCV
    from sklearn.preprocessing import StandardScaler
    from sklearn.pipeline import Pipeline
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score
    data = pd.read_csv('/content/database.csv.zip')
```

OUTPUT:



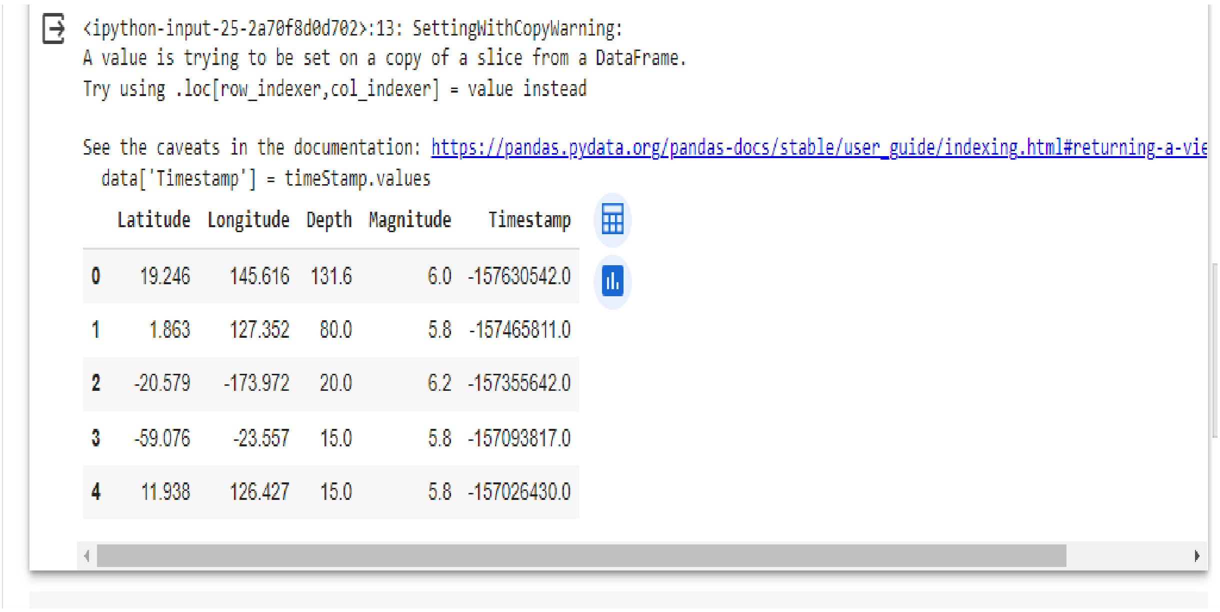
22	02-04-1965	7:11:23	51.037	177.848	Earthquake	25			5.9	MW						ISCGEMSUP ISCGE

```
    data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth',
'Magnitude']]
    data.head()
```

OUTPUT:

```
 import datetime
 import time
timestamp = []
for d, t in zip(data['Date'], data['Time']):
    try:
        ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y
%H:%M:%S')
        timestamp.append(time.mktime(ts.timetuple()))
    except ValueError:
        # print('ValueError')
        timestamp.append('ValueError')
    timeStamp = pd.Series(timestamp)
    data['Timestamp'] = timeStamp.values
    final_data = data.drop(['Date', 'Time'], axis=1)
    final_data = final_data[final_data.Timestamp != 'ValueError']
    final_data.head()
```

OUTPUT:



```
<ipython-input-25-2a70f8d0d702>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  data['Timestamp'] = timeStamp.values
```

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-157630542.0
1	1.863	127.352	80.0	5.8	-157465811.0
2	-20.579	-173.972	20.0	6.2	-157355642.0
3	-59.076	-23.557	15.0	5.8	-157093817.0
4	11.938	126.427	15.0	5.8	-157026430.0

```
from mpl_toolkits.basemap import Basemap

m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat_ts=20,resolution='c')


longitudes = data["Longitude"].tolist()

latitudes = data["Latitude"].tolist()

#m = Basemap(width=12000000,height=9000000,projection='lcc',

            #resolution=None,lat_1=80.,lat_2=55,lat_0=80,lon_0=-107.)

x,y = m(longitudes,latitudes)


m.plot(x, y, "o", markersize = 2, color = 'blue')

m.drawcoastlines()

m.fillcontinents(color='coral',lake_color='aqua')

m.drawmapboundary()

m.drawcountries()
```

OUTPUT:

CONCLUSION:

In this article we learned about the python’s matplotlib library and how to use it. We have also analyzed and visualized the earthquake dataset using the matplotlib library.