

NAME: VIGNESH K

REGISTER NUMBER: 61772231T310

EARTHQUAKE PREDICTION MODEL-PHASE- 2

MACHINE LEARNING:

Machine learning is a field of Computer Science. It evolved from study of pattern recognition as well as computational learning theory in Artificial intelligence. It is field of predictive modeling with minimizing error of a model and trying to make the most precise predictions. Machine learning uses algorithms from different fields like statistics and use them to make predictions. The algorithm used here is Regression algorithm. Regression is a topic in the field of statistics. These regression algorithms are applied on numerical variables. So it is both a statistical as well as a machine learning algorithm.

ALGORITHMS:

LINEAR REGRESSION:

Linear regression is a statistical regression method which is used for prediction analysis. It is used for solving the regression problem in machine learning.

It shows a relationship between the independent variable (x-axis) and dependent variable (y-axis). Also, it has two types of regression.

- Simple linear
- Multiple linear

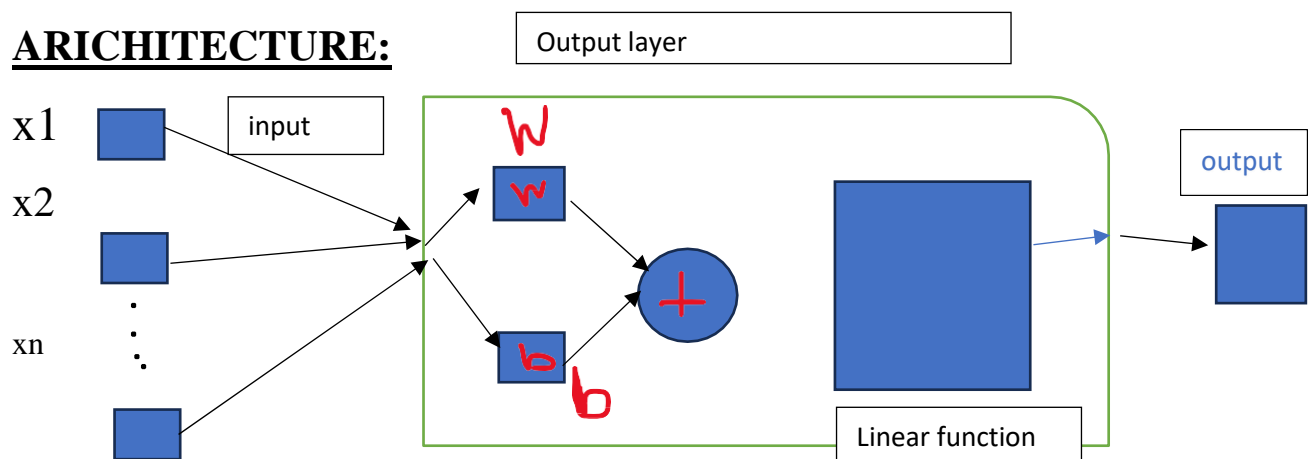
Once the model has been fit to the data, we can use it to predict the magnitude of a new earthquake given its latitude, longitude, depth, and the number of seismic stations that recorded it. This can be useful for earthquake monitoring and early warning systems, as well as for understanding the underlying causes of earthquakes and improving our ability to predict them.

EQUATION:

$$y = B_0 + B_1 * x$$

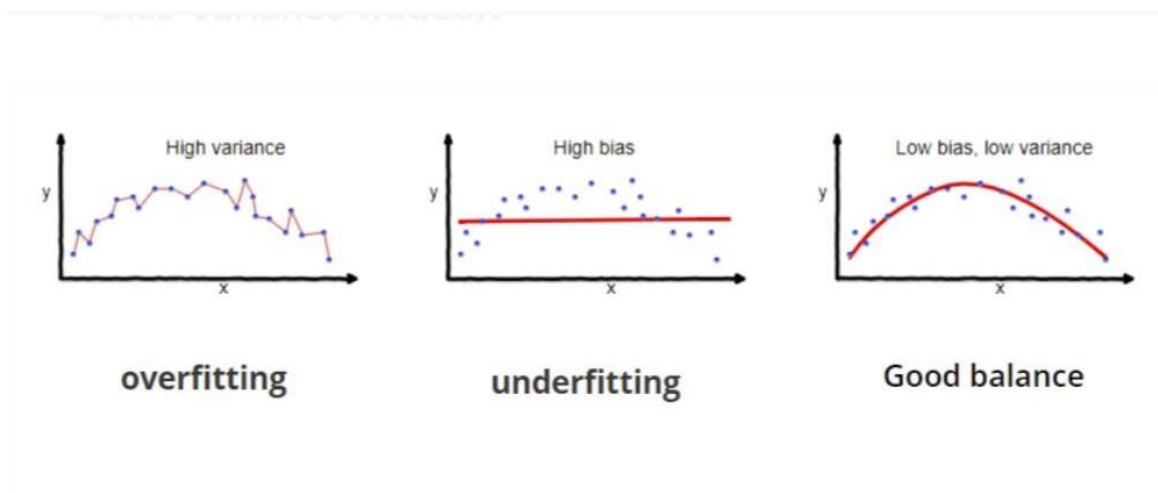
Where: B_0 = coefficient, B_1 = coefficient, x = variable, y = variable

ARCHITECTURE:



RIDGE REGRESSION:

This technique is also used when the data suffers from multicollinearity. So the leastsquares estimates are unbiased but the variance is high. Their variances are large. Due to this the observed value deviates far from the true value. Ridge regression reduces the standard errors and for this it adds a degree of bias to regression estimate. As we saw in a linear equation the prediction errors have two components. One is biased and other is variance. The prediction error can occur due to these both components. The multicollinearity problem is solved by using a shrinkage parameter λ (lambda).



EQUATION:

$$\hat{\beta}^{\text{ridge}} = (X'X + \lambda I_p)^{-1} X'Y$$

Where $X = n$ by p matrix, $Y = \text{centered } n\text{-vector}$

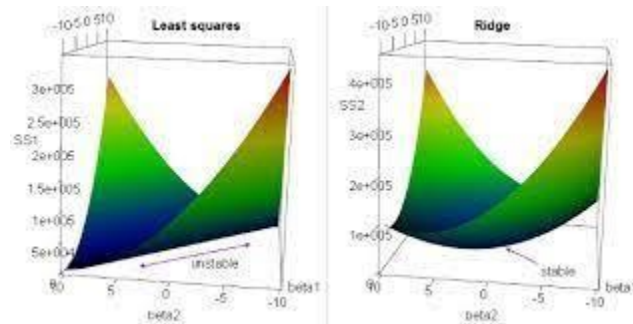
LASSO REGRESSION

The lasso regression is very similar to ridge regression. Lasso regression will penalize the absolute size of all the coefficients. This regression is used to increase the accuracy of the linear regression models. Lasso regression reduces the variability as well. Below is the equation for lasso regression.

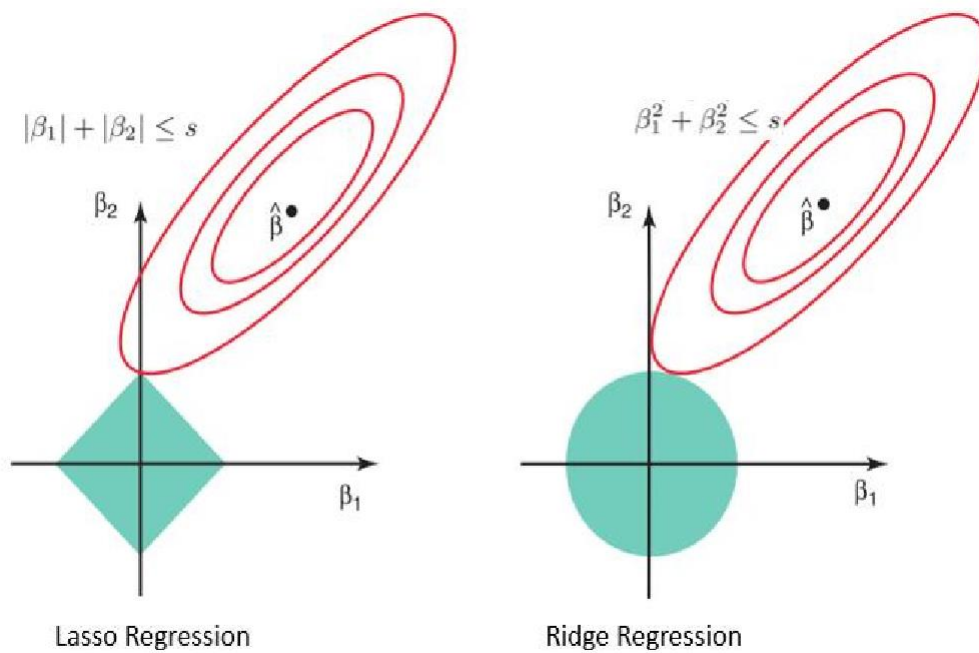
EQUATION:

$$\hat{\beta}(\text{lasso}) = \min (\beta, \beta_0) \left(\frac{1}{N} \right) (\text{Deviance} (\beta, \beta_0) + \lambda \sum |B_j|)$$

Where $N = \text{number of observations}$, $\Lambda = \text{nonnegative regularization parameter corresponding to value of Lambda}$, Parameters = β_0 and β are scalar and p -vector



ARCHITECTURE FOR LASSO AND RIDGE REGRESSION:



These are the algorithm and architecture are used via the machine learning to detect the earthquake prediction model.

IMPLEMENTATION OF DATASET CODE:

STEP 1: Importing the necessary python libraries

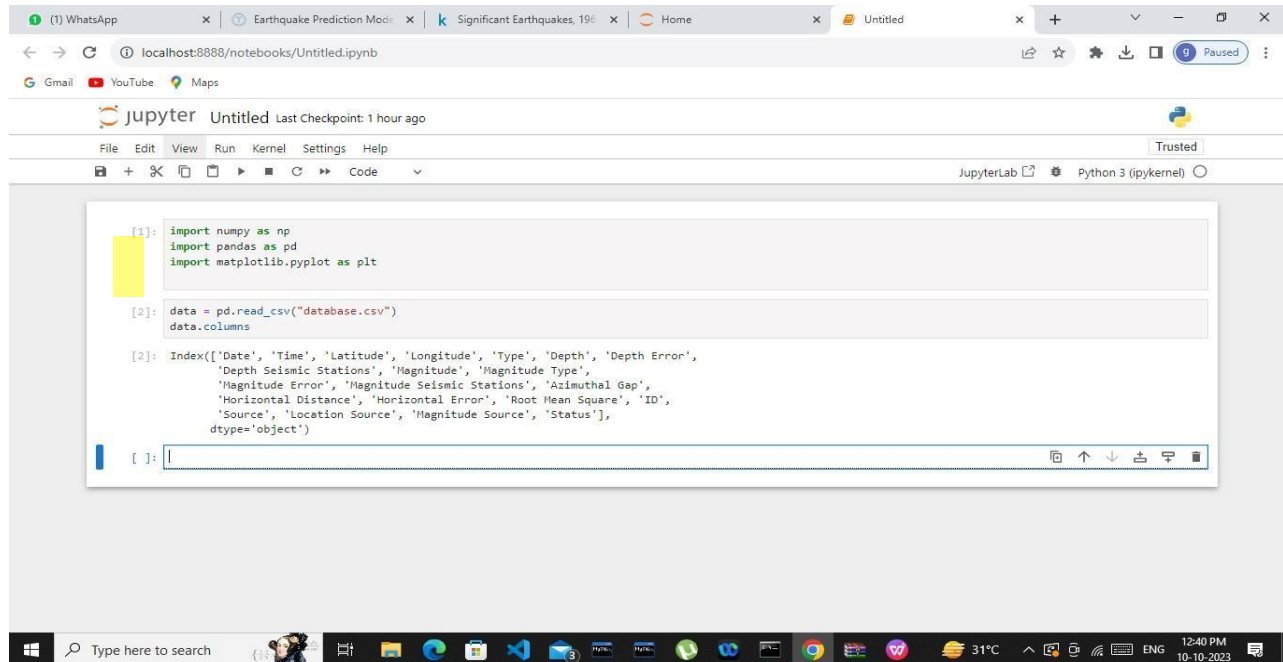
```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[2]: data = pd.read_csv("database.csv")
data.columns

[2]: Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
'Deep 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')
```

STEP 2: Importing the dataset. Download and read the dataset.

Dataset is imported from the Kaggle.com.



The screenshot shows a JupyterLab interface with a notebook titled 'Untitled'. The notebook contains three code cells. The first cell imports the necessary libraries: numpy, pandas, and matplotlib.pyplot. The second cell reads a CSV file named 'database.csv' and displays the columns. The third cell displays the index of the data, showing a list of column names and their data types.

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[2]: data = pd.read_csv("database.csv")
data.columns

[3]: Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
'Deep 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')
```

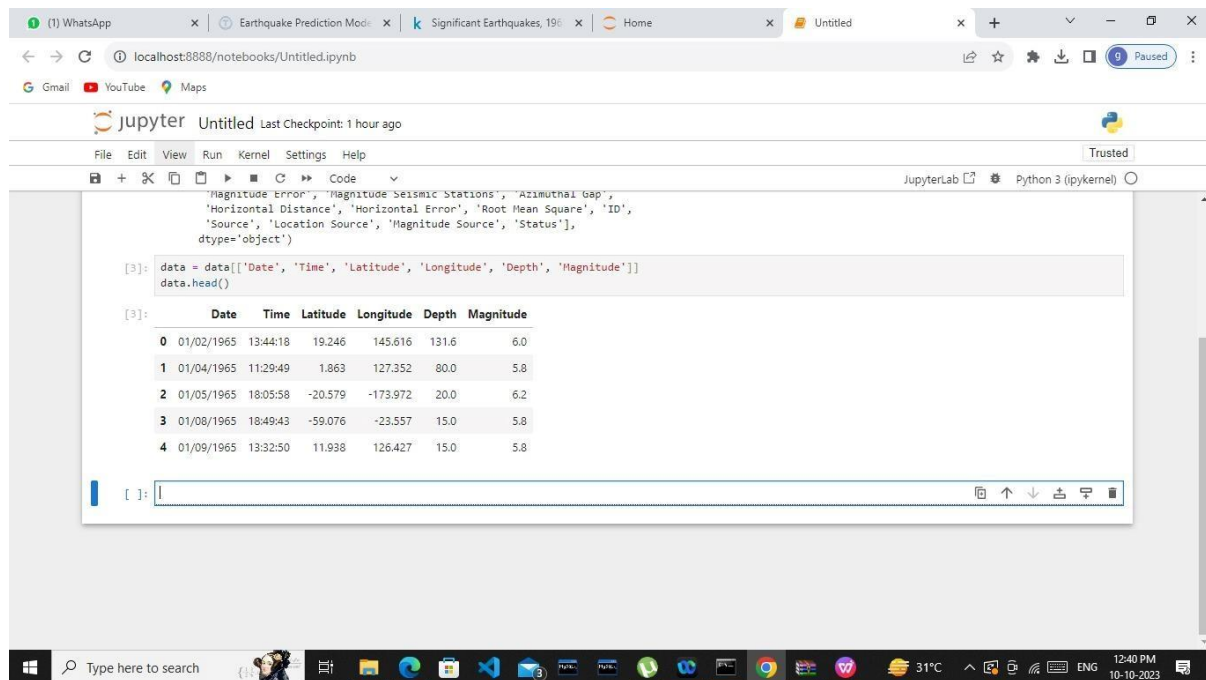
DATA CLEANING:

Raw data may contain errors, missing values, or inconsistency. Data preprocessing involves cleaning and transforming the data to make it suitable for analysis. This can include tasks like data imputation, outlier detection, and normalization.

STEP 3: According to the given dataset we are creating the object for these characteristics, namely, date, time, latitude, longitude, depth, magnitude.

By doing this process we can remove the unwanted data for the dataset according to the model we have chosen.

Panda's library which is used here helps us to examine the dataset's structure, including, columns, datatypes, and any missing values.



```
['Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',  
'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',  
'Source', 'Location Source', 'Magnitude Source', 'Status'],  
dtype='object')  
  
[3]: data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]  
data.head()  
  
[3]:
```

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

```
[ ]:
```

Data cleaning can also be done by the data splitting.

Data Splitting:

We need to divide the data into Xs and ys which respectively will be entered into the model as inputs to receive the output from the model.

Here the inputs are Timestamp, latitude and longitude and the outputs are magnitude and depth. Now I am going to split the xs and ys into train and test with validation. The training set contains 80% and the test set contains 20%.

Output: (18727,3)(4682,3)(18727,2)(4682,3)

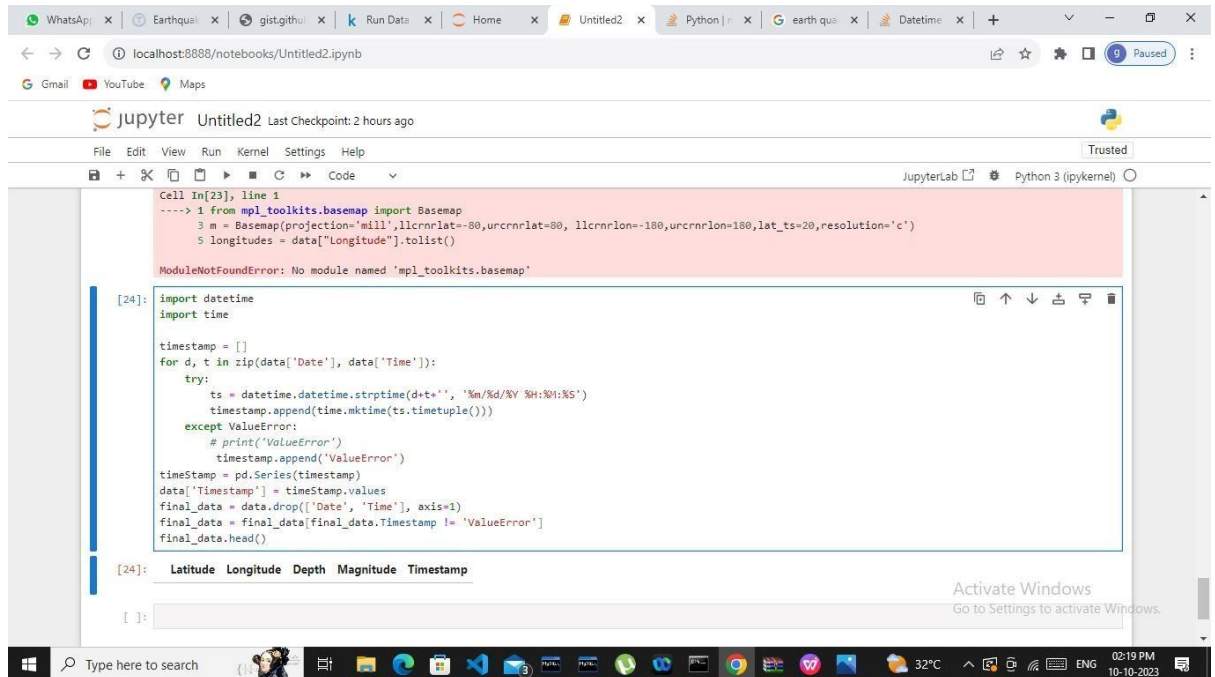
```
[49]: X = final_data[['Timestamp', 'Latitude', 'Longitude']]  
y = final_data[['Magnitude', 'Depth']]  
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  
print(X_train.shape, X_test.shape, y_train.shape, X_test.shape)
```

DATA ANALYSIS:

Predictive analysis: Once the model is trained and validated, it can be used for making predictive.

STEP 4: From this we can analysis the dataset for the particular data and time so we can get the values for the latitude, longitude, depth, magnitude, timestamp.

Data splitting also helps in the data analysis.



The screenshot displays a JupyterLab environment with a browser window at the top showing tabs for WhatsApp, Earthquake, gist.github, Run Data, Home, and Untitled2. The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar. A code cell is active, showing the following Python code:

```
Cell In[23], line 1
----> 1 from mpl_toolkits.basemap import Basemap
      2 m = Basemap(projection='mill', llcrnrlat=-80, urcrnrlat=80, llcrnrlon=-180, urcrnrlon=180, lat_ts=20, resolution='c')
      3 longitudes = data["Longitude"].tolist()

ModuleNotFoundError: No module named 'mpl_toolkits.basemap'
```

Below the error message, another code cell is shown with the following Python code:

```
[24]: 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()
```

The output of the second cell is displayed as a table with columns: Latitude, Longitude, Depth, Magnitude, and Timestamp. The output is currently empty, showing only the column headers.

At the bottom of the screen, the Windows taskbar is visible, showing the search bar, taskbar icons, and system tray with the date 10-10-2023 and time 02:19 PM.