EARTHQUAKE PREDICTION MODEL USING PYTHON

922321106003-ABITHA.J

Team Leader:

PRIYA GUPTA.K (922321106301)

Team members:

ABITHA.J (922321106003)

NITHYA.A (922321106020)

SHRISHTITHA.S (922321106034)

SUBHIKSHA.V (922321106038)

SWETHA.A.B (922321106040)

PHASE 5: PROJECT DOCUMENTATION & SUBMISSION

TOPIC: EARTHQUAKE PREDICTION USING PYTHON

OBJECTIVE:

This introduction will guide you through the initial steps of the process. In the project of earthquake prediction using python the documentation of Problem statement, Design Thinking Process, Phases of Development, Dataset Used, Data Preprocessing steps, Feature extraction techniques, Machine learning algorithm, Model training, Evaluation Matrices, Innovative techniques or approaches used during the development. In the Submission part the Compiling process of Data Preprocessing, Model Training, Evaluation Steps, The README file of how to run the code and explanation and The Dataset source with the brief explanation.

PROBLEM STATEMENT:

An earthquake prediction must specify the expected magnitude range, the geographical

Area within which it will occur, and the time interval within which it will happen with sufficient precision

So, that the ultimate success or failure of the prediction can be readily judged.

- Objective:

  Develop a predictive model for earthquake magnitudes.

- Data Source:

  Utilize a kaggle dataset containing earthquake-related information, including date, time, latitude, longitude, depth, and magnitude.
- Tasks:
  - Explore and understand the dataset's structure and contents.
  - Visualize the data on a world map for a global overview.
  - Split the dataset into training and testing subsets for model validation.
  - Develop a neural network model for earthquake magnitude prediction.
  - Train the model on the training data and evaluate its performance on the test data.

DESIGN THINKING PROCESS:

DATA SOURCE SELECTION:

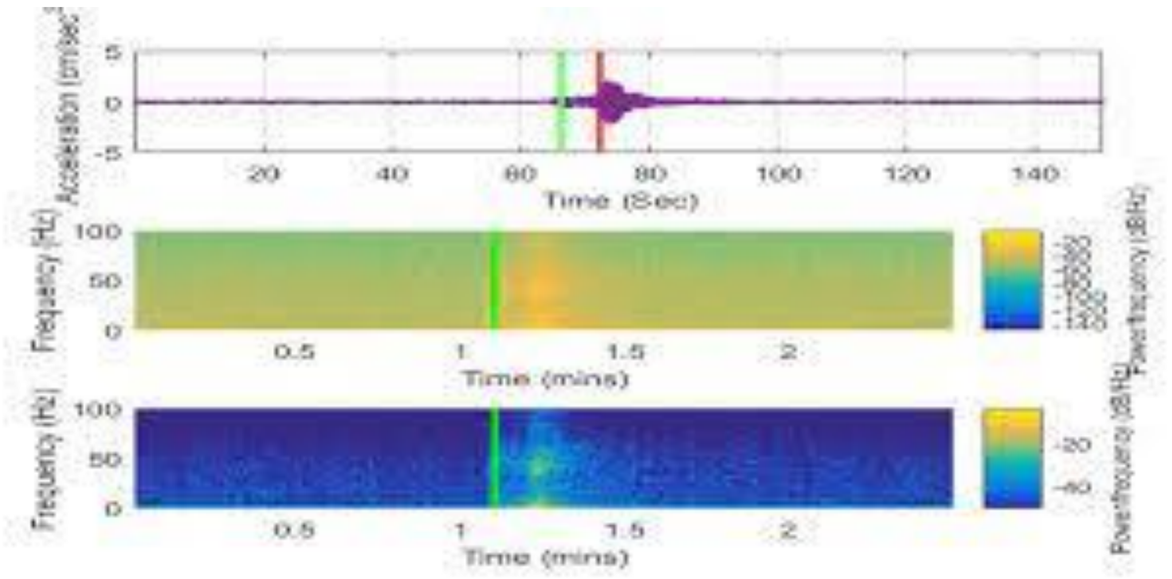
We will begin by selecting a suitable Kaggle dataset that contains earthquake data with the necessary features, including date, time, latitude, longitude, depth, and magnitude. The dataset should be comprehensive and up to date for accurate predictions.

FEATURE EXPLORATION:

Upon acquiring the dataset, we will thoroughly analyse and explore its features. This exploration will include understanding the distribution, correlations, and characteristics of key features. This step is crucial for feature engineering and model development.

VISUALIZATION:

To gain a global overview of earthquake frequency and distribution, we will create a world map visualization using latitude and longitude data. This visualization will provide valuable insights into the geographic patterns of earthquakes and can aid in feature selection.

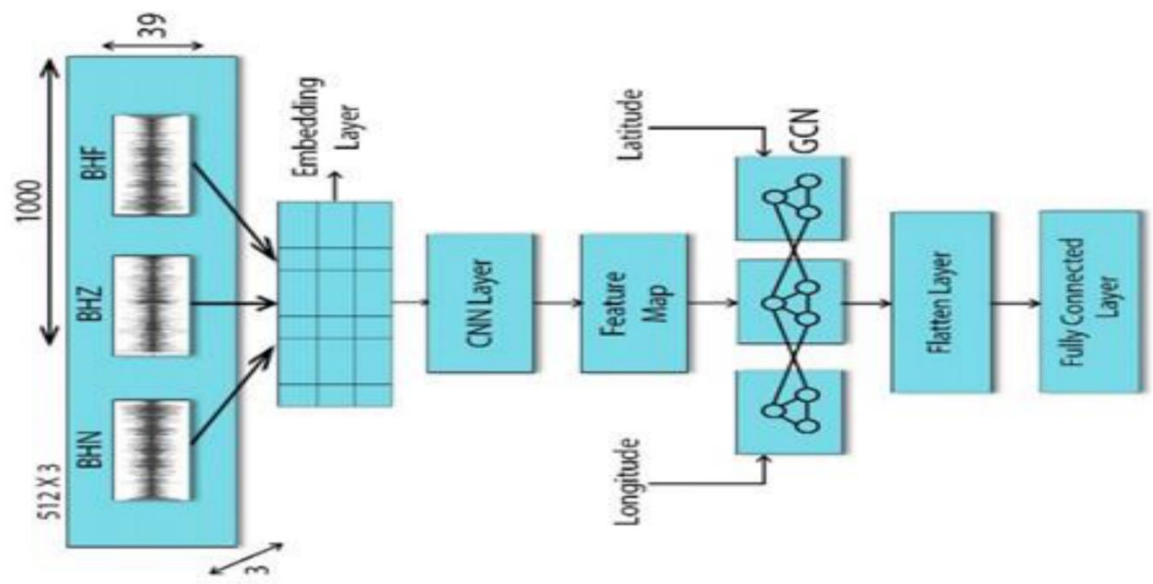


DATA SPLITTING:

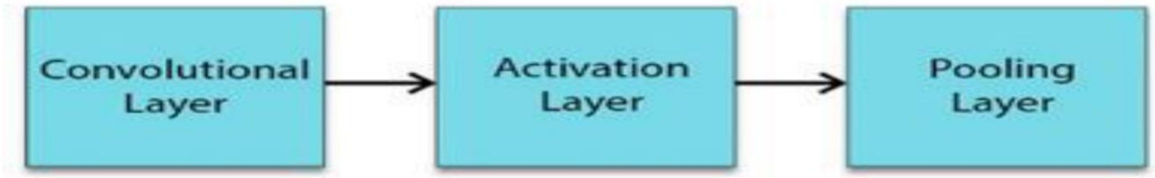
- To ensure the model's reliability, we will split the dataset into two subsets:
- A training set and a test set. Typically, an 80/20 or 70/30 split will be considered, with the larger portion allocated to training. This will allow us to train the model on one subset and evaluate its performance on an independent subset.

MODEL DEVELOPMENT:

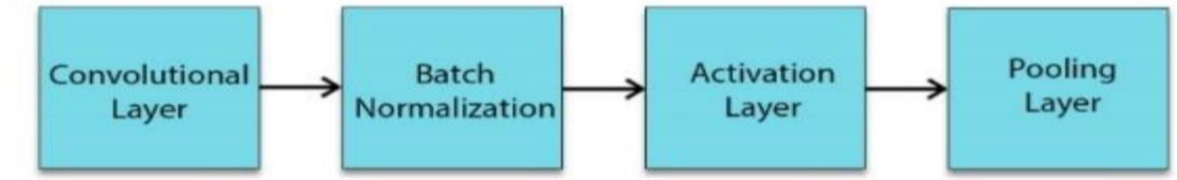
We will design and build a neural network model tailored for earthquake magnitude prediction. The architecture of the neural network, including number of layers and neurons, will be determined based on experimentation and optimization.



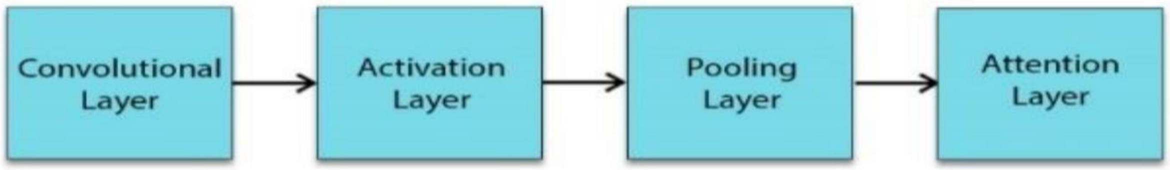
(**a**) The simple architecture of GCNN model



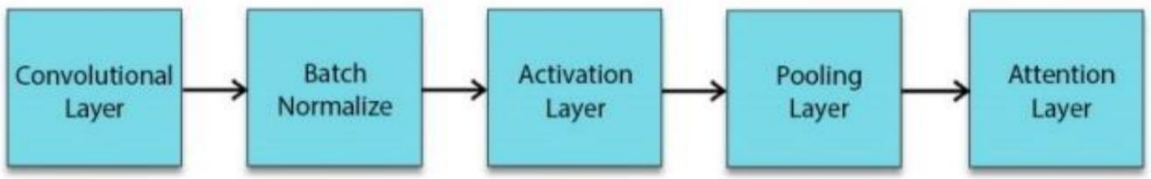
(**b**) Representation of the CNN block



(**c**) CNN block with batch normalization



(**d**) CNN block with attention layer



(**e**) CNN block with batch normalization and attention layer

SVM:

In another terms, Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data.

NN:

Ss largest seismic event in the following month based on the analysis of eight mathematically computed parameters known as seismicity indicators.

PHASE DEVELOPMENT:

During the development phase, everything that will be needed to implement the project is arranged. Potential suppliers or subcontractors are brought in, a schedule is made, materials and tools are ordered, instructions are given to the personnel and so forth. The development phase is complete when implementation is ready to start. All matters must be clear for the parties that will carry out the implementation.

SIX PHASES OF PROJECT MANAGEMENT:

- **Initiation phase**
- **Definition phase**
- **Design phase**
- **Development phase**
- **Implementation phase**
- **Follow-up phase**

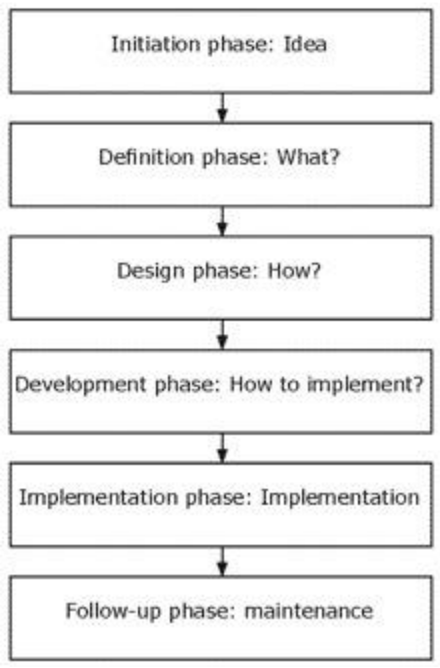

Initiation phase: Idea
Definition phase: What?
Design phase: How?
Development phase: How to implement?
Implementation phase: Implementation
Follow-up phase: maintenance

FIGURE: Project management in six phases, with the central theme of each phase.

INITIATION PHASE:

The initiation phase is the beginning of the project. In this phase, the idea for the project is explored and elaborated. The goal of this phase is to examine the feasibility of the project. In addition, decisions are made concerning who is to carry out the project, which party (or parties) will be involved and whether the project has an adequate base of support among those who are involved.

In this phase, the current or prospective project leader writes a proposal, which contains a description of the above-mentioned matters. Examples of this type of project proposal include business plans and grant applications. The prospective sponsors of the project evaluate the proposal and, upon approval, provide the necessary financing. The project officially begins at the time of approval.

DEFINITION PHASE:

After the project plan (which was developed in the initiation phase) has been approved, the project enters the second phase: the definition phase. In this phase, the requirements that are associated with a project result are specified as clearly as possible. This involves identifying the expectations that all of the involved parties have with regard to the project result. How many files are to be archived? Should the metadata conform to the Data Documentation Initiative format, or will the Dublin Core (DC) format suffice? May files be deposited in their original format, or will only those that conform to the Preferred Standards be accepted? Must the depositor of a dataset ensure that it has been processed adequately in the archive, or is this the responsibility of the archivist? Which guarantees will be made on the results of the project? The list of questions goes on and on.

'After the brain-storming session, all of the members of the "New Archive" project team were in agreement about the desired outcome'.

Figure: Expectations of a project

It is important to identify the requirements as early in the process as possible. Wijnen (2004) distinguishes several categories of project requirements that can serve as a memory aid:

- Preconditions
- Functional requirements
- Operational requirements
- Design limitations

Preconditions form the context within which the project must be conducted. Examples include legislation, working-condition regulations and approval requirements. These requirements cannot be influenced from within the project. Functional requirements are requirements that have to do with the quality of the project result (e.g. how energy-efficient must an automobile be or how many rooms must a new building have?). Operational requirements involve the use of the project result. For example, after a software project has been realised, the number of malfunctions that occur must be reduced by ninety per cent. Finally, design limitations are requirements that involve the actual realisation of the project.

DESIGN PHASE:

The list of requirements that is developed in the definition phase can be used to make design choices. In the design phase, one or more designs are developed, with which the project result can apparently be achieved. Depending on the subject of the project, the products of the design phase can include dioramas, sketches, flow charts, site trees, HTML screen designs, prototypes, photo impressions and UML schemas. The project supervisors use these designs to choose the definitive design that will be produced in the project. This is followed by the development phase. As in the definition phase, once the design has been chosen, it cannot be changed in a later stage of the project.

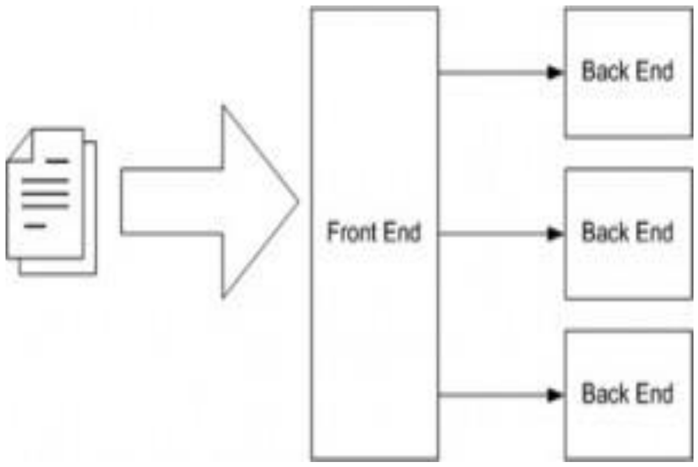


Figure: Example: Global design for the DANS Architecture Archive

In a young, very informal company, the design department was run by an artist. The term design department was not accurate in this case; it was more a group of designers who were working together. In addition, everyone was much too busy, including the head of the department.

One project involved producing a number of designs, which were quite important to the success of the project. A young designer on the project team created the designs. Although the head of the design department had ultimate responsibility for the designs, he never attended the meetings of the project team when the designs were to be discussed. The project leader always invited him, and sent him e-mails containing his young colleagues sketches, but the e-mails remained unanswered. The project leader and the young designer erroneously assumed that the department head had approved the designs. The implementation phase began. When the project was nearly finished, the result was presented to the department head, who became furious and demanded that it be completely redone. The budget, however, was almost exhausted.

DEVELOPMENT PHASE:

During the development phase, everything that will be needed to implement the project is arranged. Potential suppliers or subcontractors are brought in, a schedule is made, materials and tools are ordered, instructions are given to the personnel and so forth. The development phase is complete when implementation is ready to start. All matters must be clear for the parties that will carry out the implementation.

In some projects, particularly smaller ones, a formal development phase is probably not necessary. The important point is that it must be clear what must be done in the implementation phase, by whom and when.

IMPLEMENTATION PHASE:

- The project takes shape during the implementation phase. This phase involves the construction of the actual project result. Programmers are occupied with encoding, designers are involved in developing graphic material, contractors are building, the actual reorganisation takes place. It is during this phase that the project becomes visible to outsiders, to whom it may appear that the project has just begun. The implementation phase is the doing phase, and it is important to maintain the momentum.

- In one project, it had escaped the project teams attention that one of the most important team members was expecting to become a father at any moment and would thereafter be completely unavailable for about a month. When the time came, an external specialist was brought in to take over his work, in order to keep the team from grinding to a halt. Although the team was able to proceed, the external expertise put a considerable dent in the budget.
- At the end of the implementation phase, the result is evaluated according to the list of requirements that was created in the definition phase. It is also evaluated according to the designs. For example, tests may be conducted to determine whether the web application does indeed support Explorer 5 and Firefox 1.0 and higher. It may be determined whether the trim on the building has been made according to the agreement, or whether the materials that were used were indeed those that had been specified in the definition phase. This phase is complete when all of the requirements have been met and when the result corresponds to the design.
- Those who are involved in a project should keep in mind that it is hardly ever possible to achieve a project result that precisely meets all of the requirements that were originally specified in the definition phase. Unexpected events or advancing insight sometimes require a project team to deviate from the original list of requirements or other design documents during the implementation of the project. This is a potential source of conflict, particularly if an external customer has ordered the project result. In such cases, the customer can appeal to the agreements that were made during the definition phase.

- As a rule, the requirements cannot be changed after the end of the definition phase. This also applies to designs: the design may not be changed after the design phase has been completed. Should this nonetheless be necessary (which does sometimes occur), the project leader should ensure that the changes are discussed with those involved (particularly the decision-makers or customers) as soon as possible. It is also important that the changes that have been chosen are well documented, in order to prevent later misunderstandings. More information about the documentation of the project follows later in this handbook.

FOLLOW-UP PHASE:

- Although it is extremely important, the follow-up phase is often neglected. During this phase, everything is arranged that is necessary to bring the project to a successful completion. Examples of activities in the follow-up phase include writing handbooks, providing instruction and training for users, setting up a help desk, maintaining the result, evaluating the project itself, writing the project report, holding a party to celebrate the result that has been achieved, transferring to the directors and dismantling the project team.
- The central question in the follow-up phase concerns when and where the project ends. Project leaders often joke among themselves that the first ninety per cent of a project proceeds quickly and that the final ten per cent can take years. The boundaries of the project should be considered in the beginning of a project, so that the project can be closed in the follow-up phase, once it has reached these boundaries.
- It is sometimes unclear for those concerned whether the project result is to be a prototype or a working product. This is particularly common in innovative projects in which the outcome is not certain. Customers may expect to receive a product, while the project team assumes that it is building a prototype. Such situations are particularly likely to manifest themselves in the follow-up phase. Consider the case of a software project to test a very new concept.

- There was some anxiety concerning whether any results would be produced at all. The project eventually produced good results. The team delivered a piece of software that worked well, at least within the testing context. The customer, who did not know much about IT, thought that he had received a working product. After all, it had worked on his office computer. The software did indeed work, but when it was installed on the computers of fifty employees, the prototype began to have problems, and it was sometimes instable.
- Although the programmers would have been able to repair the software, they had no time, as they were already involved in the next project. Furthermore, they had no interest in patching up something that they considered a trial piece. Several months later, when Microsoft released its Service Pack 2 for Windows, the software completely stopped functioning. The customer was angry that the product once again did not work. Because the customer was important, the project leader tried to persuade the programmers to make a few repairs. The programmers were resistant, however, as repairing the bugs would cause too much disruption in their new project. Furthermore, they perceived the software as a prototype. Making it suitable for large-scale use would require changing the entire architectural structure. They wondered if the stream of complaints from the customer would ever stop.
- The motto, Think before you act is at the heart of the six-phase model. Each phase has its own work package. Each work package has its own aspects that should be the focus of concentration. It is therefore unnecessary to continue discussing what is to be made during the implementation phase. If all has gone well, this was already determined in the definition phase and the design phase.

DATASET:

- A data set is an ordered collection of data. As we know, a collection of information obtained through observations, measurements, study, or analysis is referred to as data. It could include information such as facts, numbers, figures, names, or even basic descriptions of objects. For our study, data can be organized in the form of graphs, charts, or tables. Through data mining, data scientists assist in the analysis of gathered data.

- A dataset is a set of numbers or values that pertain to a specific topic. A dataset is, for example, each student's test scores in a certain class. Datasets can be written as a list of integers in a random order, a table, or with curly brackets around them. The data sets are normally labelled so you understand what the data represents, however, while dealing with data sets, you don't always know what the data stands for, and you don't necessarily need to realize what the data represents to accomplish the problem.

DATASET USED:

For building the model, we used one dataset that is given in below link, it includes the factors of earthquake to build the model such as longitude, latitude, date, time, depth, depth error, type, etc...,

- **Dataset link:**

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

TYPES OF DATASETS:

In Statistics, we have different types of data sets available for different types of information. They are:

- **Numerical data sets**
- **Bivariate data sets**
- **Multivariate data sets**
- **Categorical data sets**
- **Correlation data sets**

NUMERICAL DATASETS:

The numerical data set is a data set, where the data are expressed in numbers rather than natural language. The numerical data is sometimes called quantitative data. The set of all the quantitative data/numerical data is called the numerical data set. The numerical data is always in the numbers form, such that we can perform arithmetic operations on it.

- Weight and height of a person
- The count of RBC in a medical report
- Number of pages present in a book

BIVARIATE DATASETS:

A data set that has two variables is called a Bivariate data set. It deals with the relationship between the two variables. Bivariate dataset usually contains two types of related data.

MULTIVARIATE DATASETS:

A data set with multiple variables. When the dataset contains three or more than three data types (variables), then the data set is called a multivariate dataset. In other words, the multivariate dataset consists of individual measurements that are acquired as a function of three or more than three variables.

Example: If we have to measure the length, width, height, volume of a rectangular box, we have to use multiple variables to distinguish between those entities.

CATEGORICAL DATASETS:

Categorical data sets represent features or characteristics of a person or an object. The categorical dataset consists of a categorical variable also called the qualitative variable, that can take exactly two values. Hence, it is termed as a dichotomous variable. Categorical data/variables with more than two possible values are called polytomous variables. The qualitative/categorical variables are often assumed to be polytomous variable unless otherwise specified.

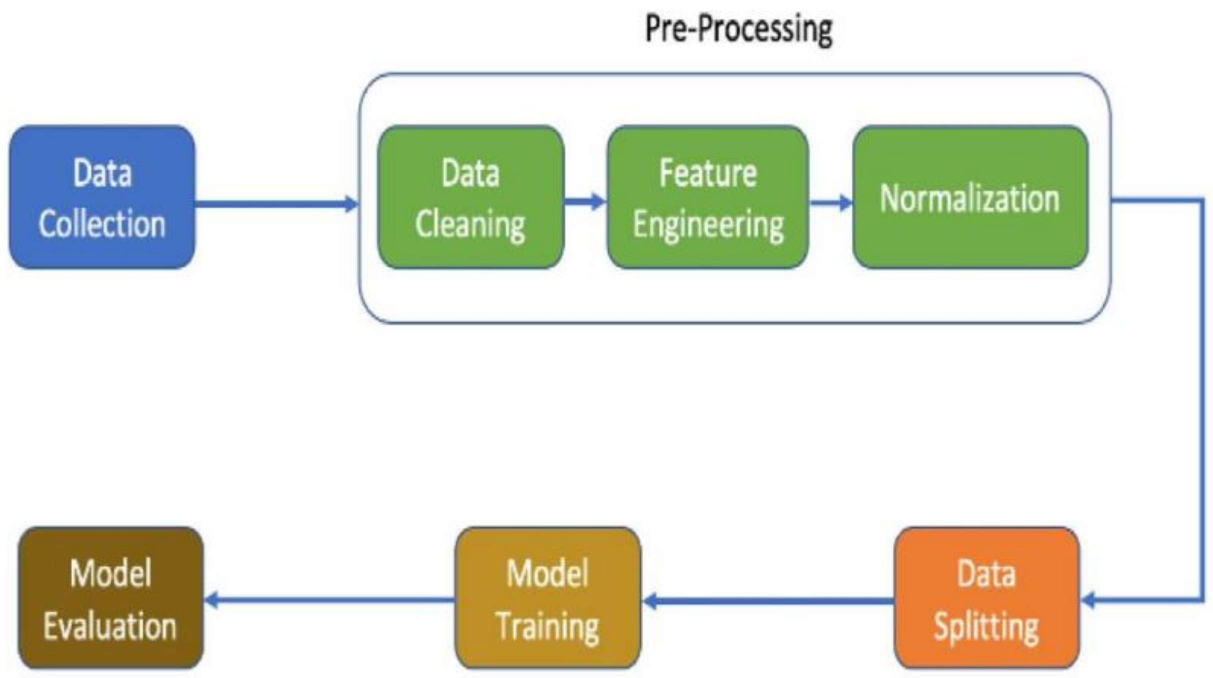
CORRELATION DATASETS:

The set of values that demonstrate some relationship with each other indicates correlation data sets. Here the values are found to be dependent on each other.

Generally, correlation is defined as a statistical relationship between two entities/variables. In some scenarios, you might have to predict the correlation between the things. It is essential to understand how correlation works.

DATA PREPROESSING:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.



GET THE DATASET:

- To create a machine learning model, the first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the dataset.
- To use the dataset in our code, we usually put it into a CSV file. However, sometimes, we may also need to use an HTML or xlsx file.

IMPORTING LIBRARIES:

In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing,
which are:

- Numpy
- pandas
- matplotlib
- seaborn

HANDLING MISSING DATA:

The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

Ways to handle missing data:

- By deleting the particular row
- By calculating the mean

SPLITTING THE DATASET INTO THE TRAINING SET AND TEST SET:

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model.

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

FEATURE SCALING:

- Feature scaling is the final step of data preprocessing in machine learning.
- It is a technique to standardize the independent variables of the dataset in a specific range.
- In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominate the other variable.

VISUALIZATION:

- Visualization or visualization (see spelling differences) is any technique for creating images, diagrams, or animations to communicate a message.
- Visualization through visual imagery has been an effective way to communicate both abstract and concrete ideas since the dawn of humanity.
- Visualization is a crucial aspect of earthquake prediction models. It helps you understand the data, discover patterns, and communicate your findings effectively. Here are some types of visualizations and their content that can be helpful in the context of earthquake prediction.

FEATURE EXTRACTION TECHNIQUES:

Feature extraction is a part of the dimensionality reduction process, in which, an initial set of the raw data is divided and reduced to more manageable groups. So when you want to process it will be easier. The most important characteristic of these large data sets is that they have a large number of variables. These variables require a lot of computing resources to process. So Feature extraction helps to get the best feature from those big data sets by selecting and combining variables into features, thus, effectively reducing the amount of data. These features are easy to process, but still able to describe the actual data set with accuracy and originality.

APPLICATION OF FEATURE EXTRACTION:

- **Bag of Words-** Bag-of-Words is the most used technique for natural language processing. In this process they extract the words or the features from a sentence, document, website, etc. and then they classify them into the frequency of use. So in this whole process feature extraction is one of the most important parts.
- **Image Processing** –Image processing is one of the best and most interesting domain. In this domain basically you will start playing with your images in order to understand them. So here we use many techniques which includes feature extraction as well and algorithms to detect features such as shaped, edges, or motion in a digital image or video to process them.
- **Auto-encoders:** The main purpose of the auto-encoders is efficient data coding which is unsupervised in nature. This process comes under unsupervised learning. So Feature extraction procedure is applicable here to identify the key features from the data to code by learning from the coding of the original data set to derive new ones.

MACHINE LEARNING ALGORITHM:

Machine learning algorithms are mathematical model mapping methods used to learn or uncover underlying patterns embedded in the data. Machine learning comprises a group of computational algorithms that can perform pattern recognition, classification, and prediction on data by learning from existing data. The most common machine learning approaches in biology are support vector machines and artificial neural networks. ANN model with deep neuron layers can be used to predict sequence specificities of DNA- and RNA-binding proteins, noncoding variants, alternative splicing, and quantitative structure–activity relationship of drugs. Deep learning models have outperformed other machine learning methods in identifying more complex features from data. To achieve complex results, deep learning techniques require a higher volume of data and computational time, compared to other machine learning algorithms. Omic data such as genome, transcriptome, epigenome, proteome, and metabolome may be integrated into a single model, which has large dimensions, and requires extensive time to build an appropriate model. Data collection can be minimized by reducing the dimension of input data, which can be done before or after data integration with principal component analysis (PCA), or after data integration with feature selection algorithms. Mode of action by network identification (MNI) combines reverse engineering network modeling with machine learning to decipher regulatory interactions. MNI uses a training set of multidimensional omic data to identify genetic components and network that correspond to a specific state. MNI, using a set of ordinary differential equations, directed graph relating the amounts of biomolecules to each other can be generated. For example, when transcriptomic data are used as training data, regulatory influences between genes can be inferred. In addition to MNI, another network-based system CellNet classifies cellular states based on the status of gene regulatory network. Both MNI and CellNet utilize machine learning integrated reverse engineering methods.

MODEL TRAINING:

Training data is the initial dataset you use to teach a machine learning application to recognize patterns or perform to your criteria, while testing or validation data is used to evaluate your model’s accuracy.

- **Relevant:** You will, of course, need data relevant to the task at hand or the problem you’re trying to solve. If your goal is to automate customer support processes, you’d use a dataset of your actual customer support data, or it would be skewed. If you’re training a model to analyze social media data, you’ll need a dataset from, Whatsapp, Twitter, Facebook, Instagram, or whichever site you’ll be analyzing.

- **Uniform:** All data should come from the same source with the same attributes.
- **Representative:** Your training data must have the same data points and factors as the data you’ll be analyzing.
- **Comprehensive:** Your training dataset must be large enough for your needs and have the proper scope and range to encompass all of the model’s desired use cases.
- **Diverse:** The dataset should reflect the training and user base, or the results will end up skewed. Ensure those tasked with training the model have no hidden biases or bring in a third party to audit the criteria.

EVALUATION MATRICES:

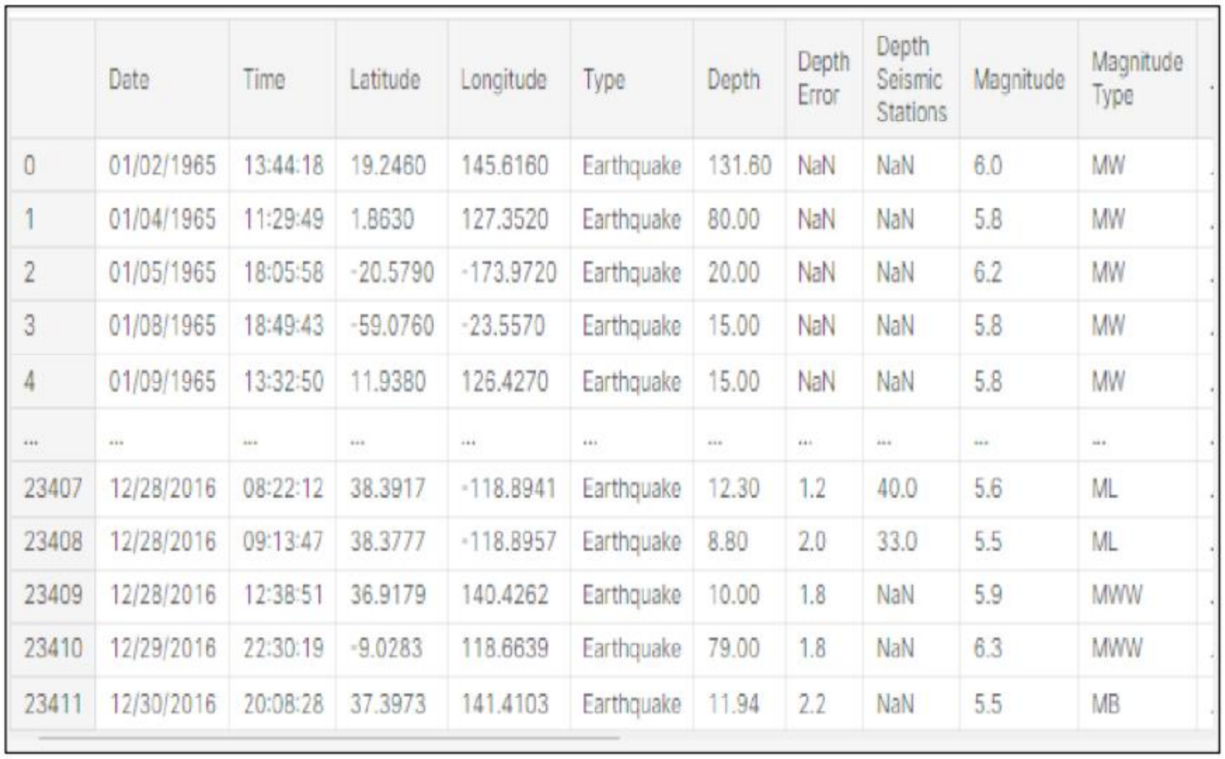
- Evaluation metrics are quantitative measures used to assess the performance and effectiveness of a statistical or machine learning model. These metrics provide insights into how well the model is performing and help in comparing different models or algorithms.
- When evaluating a machine learning model, it is crucial to assess its predictive ability, generalization capability, and overall quality. Evaluation metrics provide objective criteria

to measure these aspects. The choice of evaluation metrics depends on the specific problem domain, the type of data, and the desired outcome.

- I have seen plenty of analysts and aspiring data scientists not even bothering to check how robust their model is. Once they are finished building a model, they hurriedly map predicted values on unseen data. This is an incorrect approach. The ground truth is building a predictive model is not your motive. It’s about creating and selecting a model which gives a high accuracy score on out-of-sample data. Hence, it is crucial to check the accuracy of your model prior to computing predicted values.

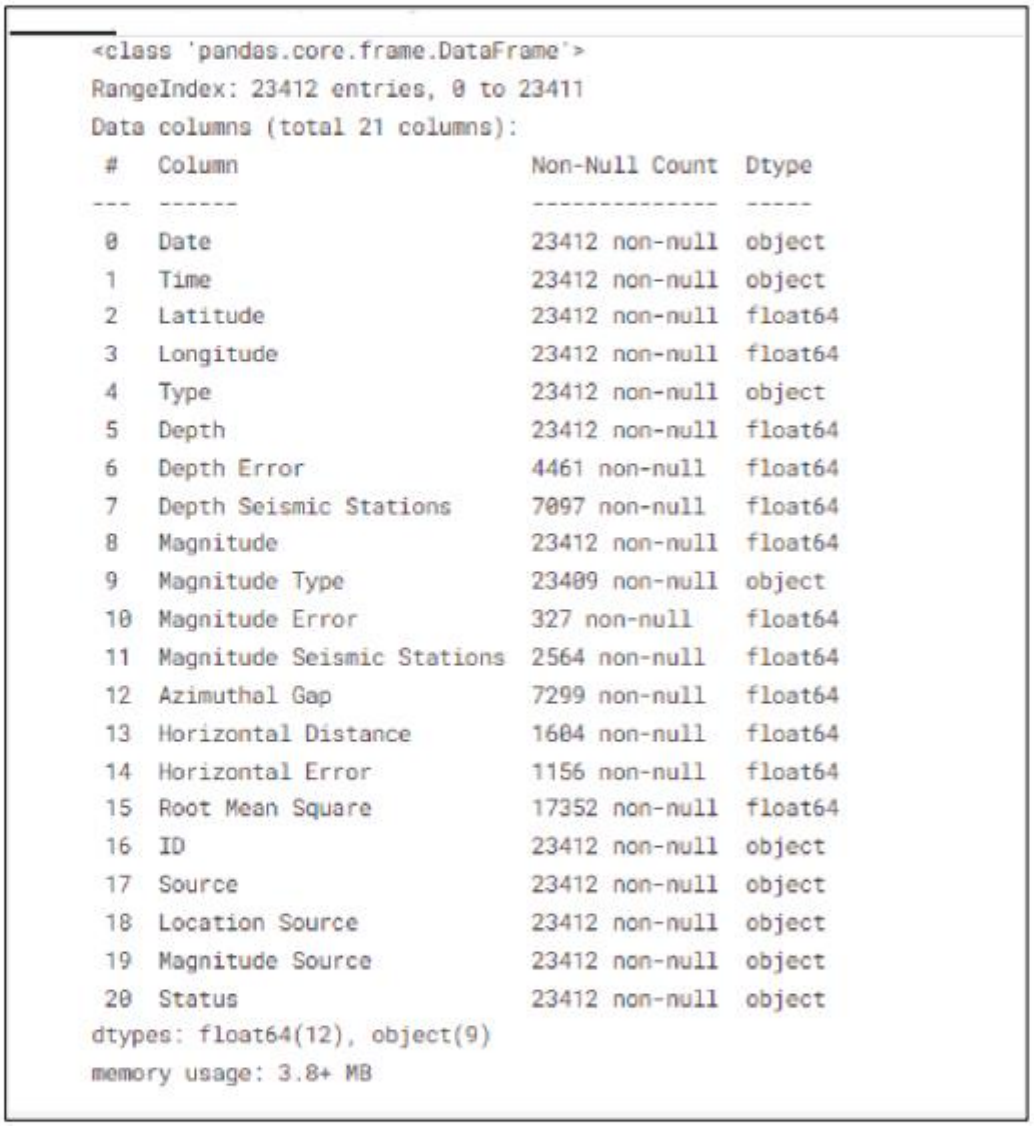
PROGRAM OF DATA PREPROCESSING:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pltimport seaborn
as sns
from sklearn.preprocessing import StandardScaler from
sklearn.model_selection import train_test_split
import tensorflow as tf
data =pd.read_csv('C:/earthquake-database/database.csv')data
```



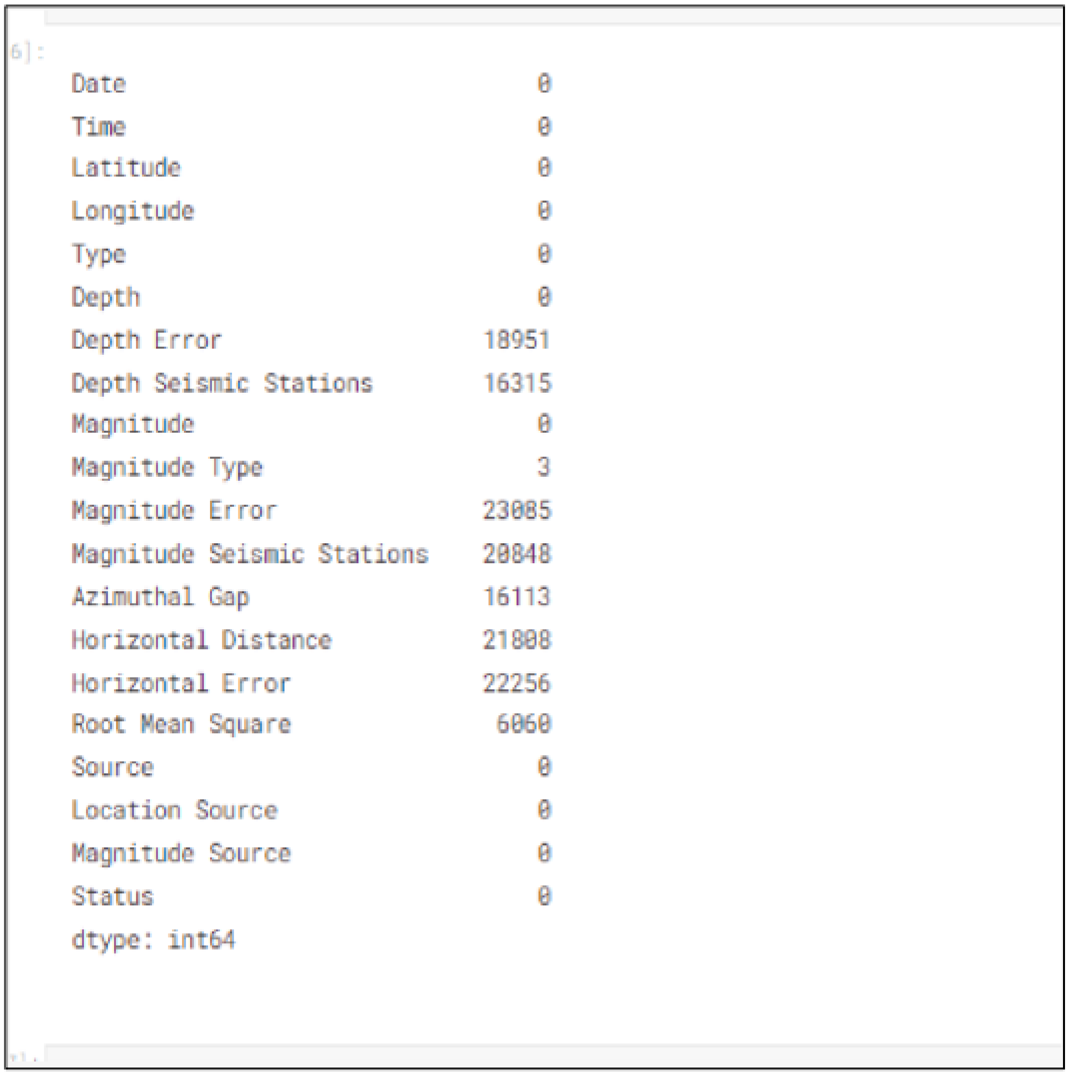
	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	.
0	01/02/1965	13:44:18	19.2460	145.6160	Earthquake	131.60	NaN	NaN	6.0	MW	.
1	01/04/1965	11:29:49	1.8630	127.3520	Earthquake	80.00	NaN	NaN	5.8	MW	.
2	01/05/1965	18:05:58	-20.5790	-173.9720	Earthquake	20.00	NaN	NaN	6.2	MW	.
3	01/08/1965	18:49:43	-59.0760	-23.5570	Earthquake	15.00	NaN	NaN	5.8	MW	.
4	01/09/1965	13:32:50	11.9380	126.4270	Earthquake	15.00	NaN	NaN	5.8	MW	.
...	...	...	...	...	...	...	...	...	...	...	.
23407	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake	12.30	1.2	40.0	5.6	ML	.
23408	12/28/2016	09:13:47	38.3777	-118.8957	Earthquake	8.80	2.0	33.0	5.5	ML	.
23409	12/28/2016	12:38:51	36.9179	140.4262	Earthquake	10.00	1.8	NaN	5.9	MWW	.
23410	12/29/2016	22:30:19	-9.0283	118.6639	Earthquake	79.00	1.8	NaN	6.3	MWW	.
23411	12/30/2016	20:08:28	37.3973	141.4103	Earthquake	11.94	2.2	NaN	5.5	MB	.

```
data.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23412 entries, 0 to 23411
Data columns (total 21 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Date                        23412 non-null  object
 1   Time                        23412 non-null  object
 2   Latitude                    23412 non-null  float64
 3   Longitude                   23412 non-null  float64
 4   Type                        23412 non-null  object
 5   Depth                       23412 non-null  float64
 6   Depth Error                 4461 non-null   float64
 7   Depth Seismic Stations      7097 non-null   float64
 8   Magnitude                   23412 non-null  float64
 9   Magnitude Type              23409 non-null  object
 10  Magnitude Error             327 non-null    float64
 11  Magnitude Seismic Stations  2564 non-null   float64
 12  Azimuthal Gap               7299 non-null   float64
 13  Horizontal Distance         1604 non-null   float64
 14  Horizontal Error            1156 non-null   float64
 15  Root Mean Square            17352 non-null  float64
 16  ID                          23412 non-null  object
 17  Source                      23412 non-null  object
 18  Location Source             23412 non-null  object
 19  Magnitude Source            23412 non-null  object
 20  Status                      23412 non-null  object
dtypes: float64(12), object(9)
memory usage: 3.8+ MB
```

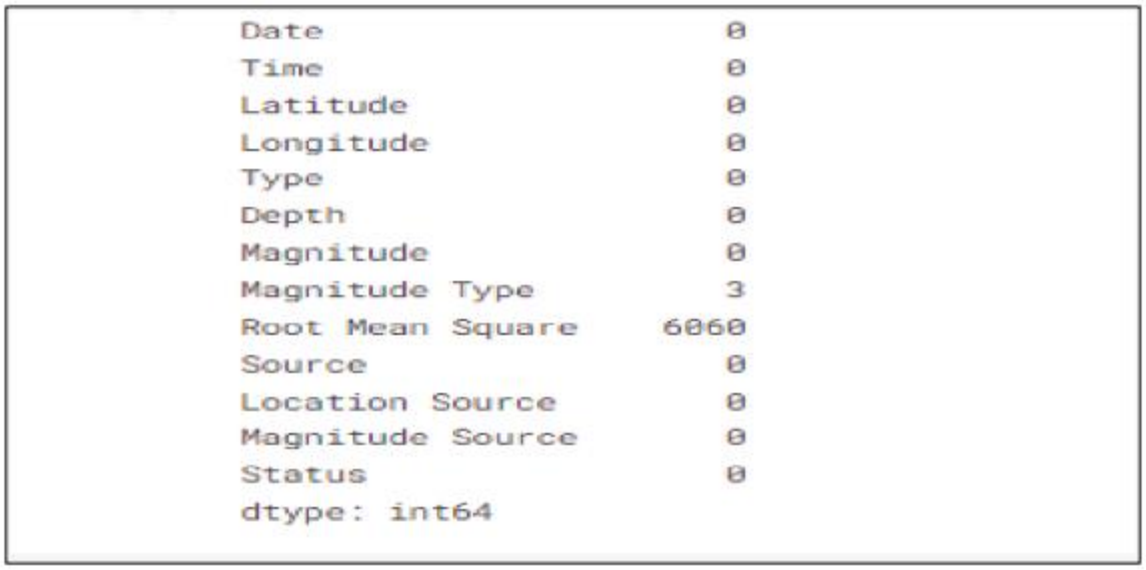
```
data = data.drop('ID', axis=1)
data.isna().sum()
```



```
6]:
    Date                            0
    Time                            0
    Latitude                        0
    Longitude                       0
    Type                            0
    Depth                           0
    Depth Error                 18951
    Depth Seismic Stations      16315
    Magnitude                       0
    Magnitude Type                  3
    Magnitude Error             23085
    Magnitude Seismic Stations  20848
    Azimuthal Gap               16113
    Horizontal Distance         21808
    Horizontal Error            22256
    Root Mean Square             6060
    Source                          0
    Location Source                 0
    Magnitude Source                0
    Status                          0
    dtype: int64
```

null_columns = data.loc[:, data.isna().sum() > 0.66 *data.shape[0]].columns

data.isna().sum()



```
Date                  0
Time                  0
Latitude              0
Longitude             0
Type                  0
Depth                 0
Magnitude             0
Magnitude Type        3
Root Mean Square   6060
Source                0
Location Source       0
Magnitude Source      0
Status                0
dtype: int64
```

data['Root Mean Square'] = data['Root Mean

Square'].fillna(data['Root Mean Square'].mean())data =

data.dropna(axis=0).reset_index(drop=True)data.isna().sum().sum()

0

y = data.loc[:, 'Status']

X = data.drop('Status', axis=1)

scaler = StandardScaler()

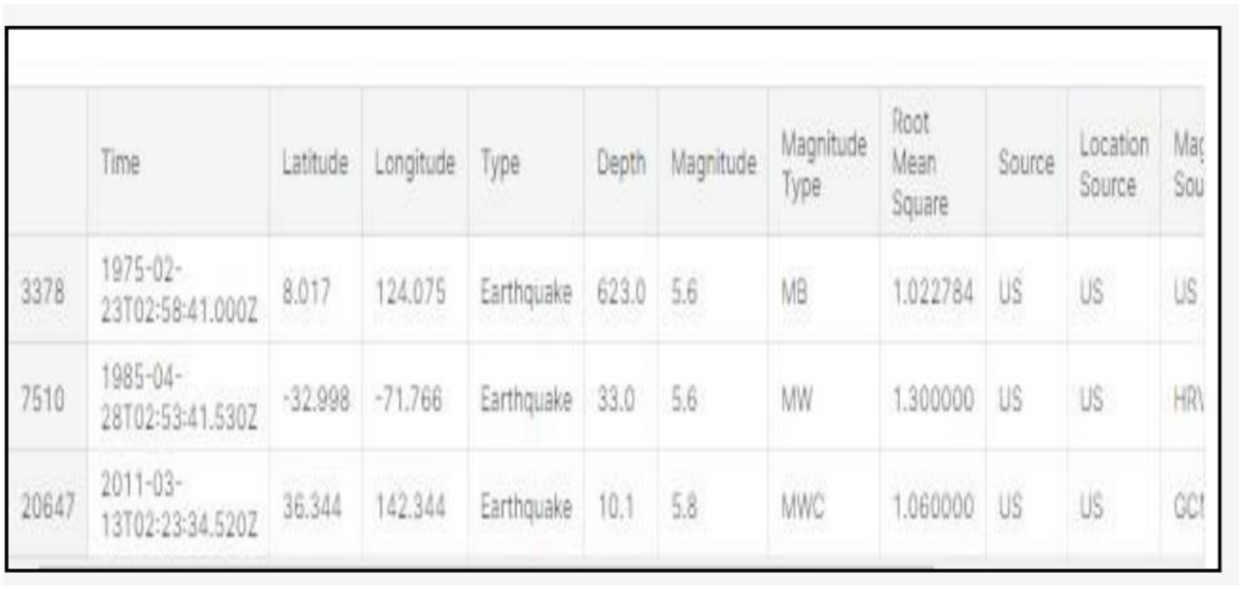
X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y,train_size=0.7, random_state=56)

Split X.shape

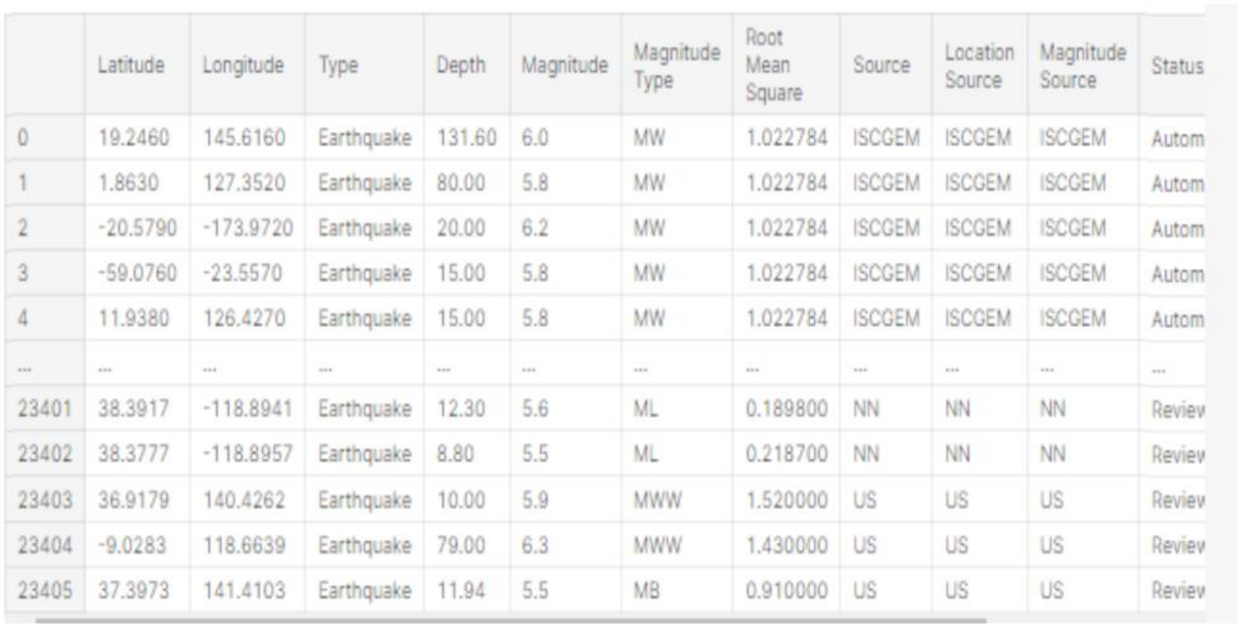
(23406, 104)

```
y.mean()
0.88737930445185
inputs = tf.keras.Input(shape=(104,))
x = tf.keras.layers.Dense(64, activation='relu')(inputs)x =
tf.keras.layers.Dense(64, activation='relu')(x)
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)model =
tf.keras.Model(inputs, outputs)
model.compile( optimizer='adam',
        loss='binary_crossentropy',
        metrics=[tf.keras.metrics.AUC(name='auc')]
)
batch_size = 32
epochs = 30
history = model.fit(
        X_train, y_train,
        validation_split=0.2, batch_size=batch_size,
        epochs=epochs,
        callbacks=[tf.keras.callbacks.ReduceLROnPlateau()],verbose=0
)
)
data['Month'] = data['Date'].apply(lambda x: x[0:2])
data['Year'] = data['Date'].apply(lambda x: x[-4:])data = data.drop('Date',
axis=1)
data['Month'] = data['Month'].astype(np.int)data[data['Year'].str.contains('Z')]
```



	Time	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type	Root Mean Square	Source	Location Source	Mag Sou
3378	1975-02-23T02:58:41.000Z	8.017	124.075	Earthquake	623.0	5.6	MB	1.022784	US	US	US
7510	1985-04-28T02:53:41.530Z	-32.998	-71.766	Earthquake	33.0	5.6	MW	1.300000	US	US	HR
20647	2011-03-13T02:23:34.520Z	36.344	142.344	Earthquake	10.1	5.8	MWC	1.060000	US	US	GC

```
invalid_year_indices =
data[data['Year'].str.contains('Z')].indexdata =
data.drop(invalid_year_indices,
axis=0).reset_index(drop=True) data['Year'] =
data['Year'].astype(np.int)
data['Hour'] = data['Time'].apply(lambda x: np.int(x[0:2]))data = data.drop('Time',
axis=1)
data
```

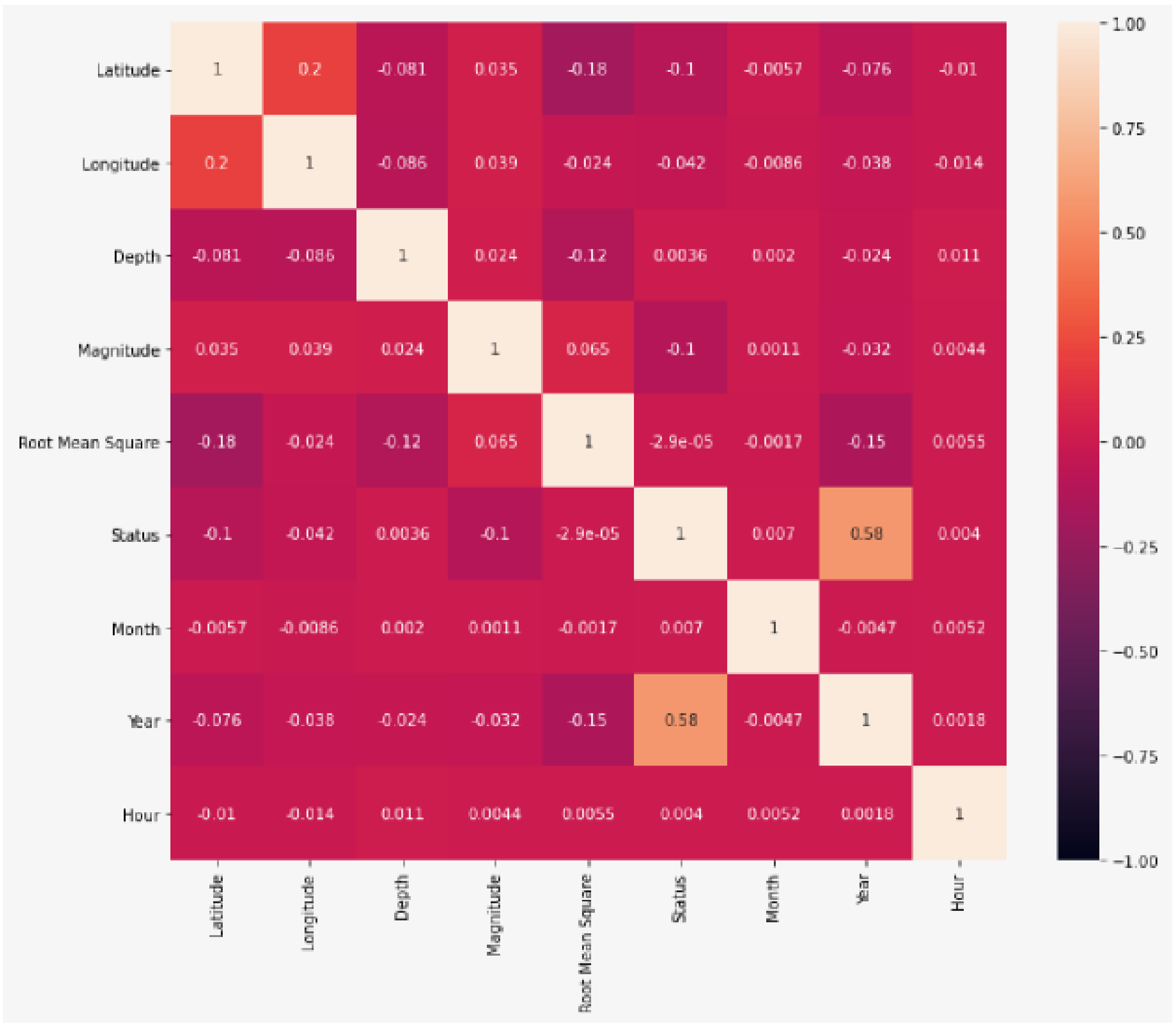


	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type	Root Mean Square	Source	Location Source	Magnitude Source	Status
0	19.2460	145.6160	Earthquake	131.60	6.0	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
1	1.8630	127.3520	Earthquake	80.00	5.8	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
2	-20.5790	-173.9720	Earthquake	20.00	6.2	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
3	-59.0760	-23.5570	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
4	11.9380	126.4270	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
...	...	...	...	...	...	...	...	...	...	...	...
23401	38.3917	-118.8941	Earthquake	12.30	5.6	ML	0.189800	NN	NN	NN	Review
23402	38.3777	-118.8957	Earthquake	8.80	5.5	ML	0.218700	NN	NN	NN	Review
23403	36.9179	140.4262	Earthquake	10.00	5.9	MWW	1.520000	US	US	US	Review
23404	-9.0283	118.6639	Earthquake	79.00	6.3	MWW	1.430000	US	US	US	Review
23405	37.3973	141.4103	Earthquake	11.94	5.5	MB	0.910000	US	US	US	Review

23406 rows × 14 columns

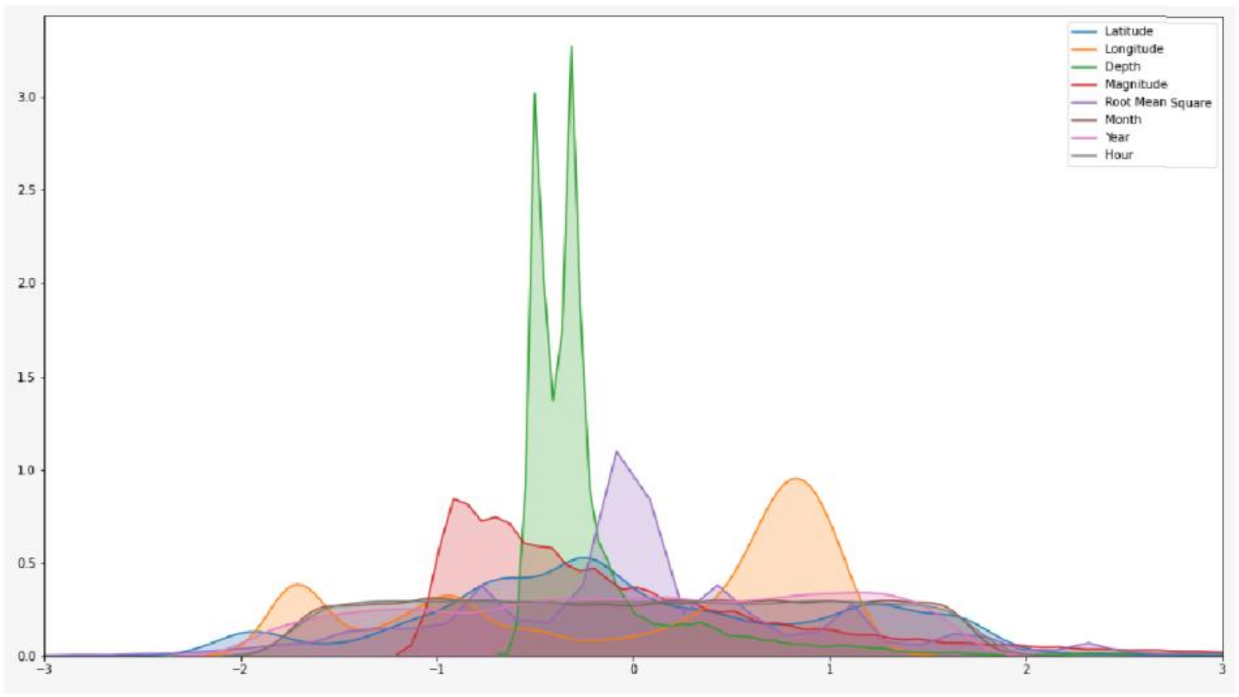
```
data['Status'].unique()
            array(['Automatic', 'Reviewed'], dtype=object) data['Status'] =
data['Status'].apply(lambda x: 1 if x =='Reviewed' else 0)
```

```
numeric_columns = [column for column in data.columns ifdata.dtypes[column]
!= 'object']
            corr = data[numeric_columns].corr()plt.figure(figsize=(12, 10))
            sns.heatmap(corr, annot=True, vmin=-1.0, vmax=1.0)
            plt.show()
```



	Latitude	Longitude	Depth	Magnitude	Root Mean Square	Status	Month	Year	Hour
Latitude	1	0.2	-0.081	0.035	-0.18	-0.1	-0.0057	-0.076	-0.01
Longitude	0.2	1	-0.086	0.039	-0.024	-0.042	-0.0086	-0.038	-0.014
Depth	-0.081	-0.086	1	0.024	-0.12	0.0036	0.002	-0.024	0.011
Magnitude	0.035	0.039	0.024	1	0.065	-0.1	0.0011	-0.032	0.0044
Root Mean Square	-0.18	-0.024	-0.12	0.065	1	-2.9e-05	-0.0017	-0.15	0.0055
Status	-0.1	-0.042	0.0036	-0.1	-2.9e-05	1	0.007	0.58	0.004
Month	-0.0057	-0.0086	0.002	0.0011	-0.0017	0.007	1	-0.0047	0.0052
Year	-0.076	-0.038	-0.024	-0.032	-0.15	0.58	-0.0047	1	0.0018
Hour	-0.01	-0.014	0.011	0.0044	0.0055	0.004	0.0052	0.0018	1

```
numeric_columns.remove('Status')scaler =
        StandardScaler()
standardized_df = pd.DataFrame(scaler.fit_transform(data[numeric_columns].copy()),
columns=numeric_columns)
plt.figure(figsize=(18, 10)) for column in
numeric_columns:
sns.kdeplot(standardized_df[column],shade=True)
```



CONCLUSION:

In this project of Earthquake Prediction using python the documentation of Problem Statement, Design Thinking Process, Phases of Development, Dataset, Data Preprocessing, Feature Extraction, Machine Learning Algorithms, Model training, Evaluation Matrices, And the python program of Data preprocessing.