EARTHQUAKE PREDICTION MODEL USING PYTHON

TEAM MEMBER

961321104010 – S.HARRY ANTONY STEPHEN

PHASE-4: Visualizing the data on a world map Splitting it into training and testing sets

Model Development Part 2

1.EVALUATION

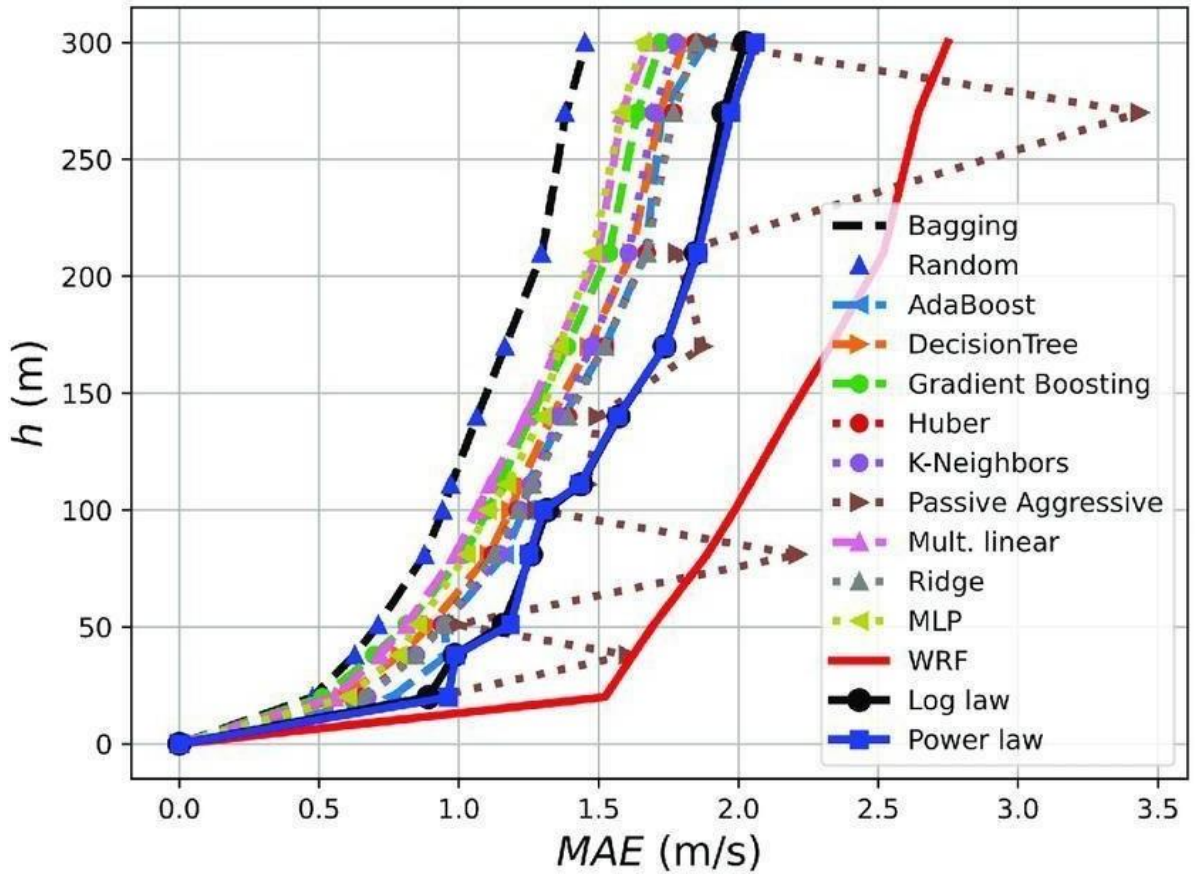
Assess the model's performance on the validation set. Common evaluation metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

Model evaluation is the process that uses some metrics which help us to analyze the performance of the model. As we all know that model development is a multi-step process and a check should be kept on how well the model generalizes future predictions. Therefore evaluating a model plays a vital role so that we can judge the performance of our model. The evaluation also helps to analyze a model's key weaknesses. There are many metrics like Accuracy, Precision, Recall, F1 score, Area under Curve, Confusion Matrix, and Mean Square Error. Cross Validation is one technique that is followed during the training phase and it is a model evaluation technique as well.

Mean Absolute Error(MAE)

This is the simplest metric used to analyze the loss over the whole dataset. As we all know the error is basically the difference between the predicted and actual values. Therefore MAE is defined as the average of the errors calculated. Here we calculate the modulus of the error, perform the summation and then divide the result by the number of data points. It is a positive quantity and is not concerned about the direction. The formula of MAE is given by

$$MSE = \sum (y_{pred} - y_{actual})^2 / N$$



Mean Squared Error(MSE)

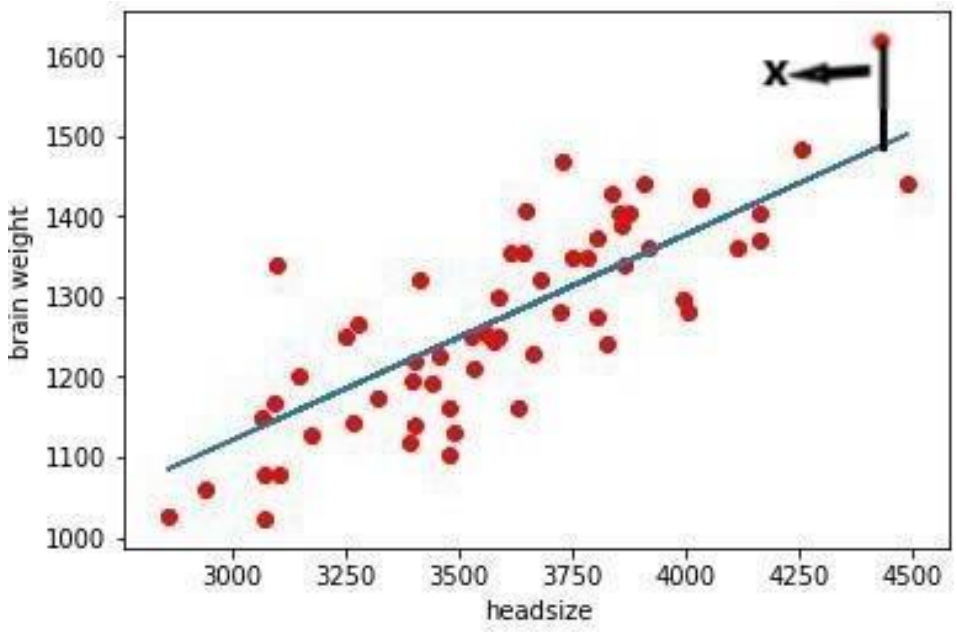
The most commonly used metric is Mean Square error or MSE. It is a function used to calculate the loss. We find the difference between the predicted values and the truth variable, square the result and then find the average over the whole dataset. MSE is always positive as we square the values. The small the MSE better is the performance of our model. The formula of MSE is given:

$$MSE = \sum (y_{pred} - y_{actual})^2 / N$$

Root Mean Squared Error(RMSE)

RMSE is a popular method and is the extended version of MSE(Mean Squared Error). This method is basically used to evaluate the performance of our model. It indicates how much the data points are spread around the best line. It is the standard deviation of the Mean squared error. A lower value means that the data point lies closer to the best fit line.

$$RMSE=\sqrt{(\sum (y_{pred} - y_{actual})^2 / N)}$$



2.HYPERPARAMETER TUNING

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters. However, there is another kind of parameter, known as ***Hyperparameters***, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

Some examples of model hyperparameters include:

1. The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization
2. The learning rate for training a neural network.
3. The C and sigma hyperparameters for support vector machines.
4. The k in k-nearest neighbors.

The aim of this article is to explore various strategies to tune hyperparameters for Machine learning models.

Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem.

GridSearchCV

In GridSearchCV approach, the machine learning model is evaluated for a range of hyperparameter values. This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.

For example, if we want to set two hyperparameters C and Alpha of the Logistic Regression Classifier model, with different sets of values. The grid search technique will construct many versions of the model with all possible combinations of hyperparameters and will return the best one.

As in the image, for C = [0.1, 0.2, 0.3, 0.4, 0.5] and Alpha = [0.1, 0.2, 0.3, 0.4]. For a combination of ***C=0.3 and Alpha=0.2***, the performance score comes out to be **0.726(Highest)**, therefore it is selected.

C	Alpha = 0.1	0.2	0.3	0.4
0.5	0.701	0.703	0.697	0.696
0.4	0.699	0.702	0.698	0.702
0.3	0.721	0.726	0.713	0.703
0.2	0.706	0.705	0.704	0.701
0.1	0.698	0.692	0.688	0.675

Program:

```
From sklearn.linear_model import LogisticRegression

From sklearn.model_selection import GridSearchCV

C_space = np.logspace(-5, 8, 15)

Param_grid = {‘C’: c_space}

Logreg = LogisticRegression()

Logreg_cv = GridSearchCV(logreg, param_grid, cv = 5)
Logreg_cv.fit(X, y)
```

Print(“Tuned Logistic Regression Parameters: {}”.format(logreg_cv.best_params_))

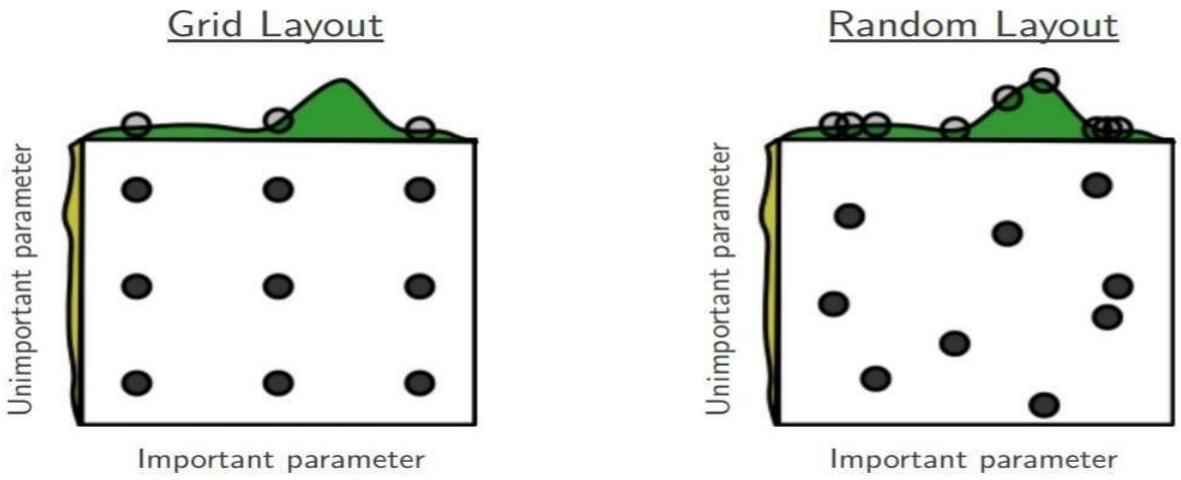
Print(“Best score is {}”.format(logreg_cv.best_score_))

RandomizedSearchCV

RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyperparameter settings. It moves within the grid in a random fashion to find the best set of hyperparameters. This approach reduces unnecessary computation.

Sample program:

```
>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.model_selection import
RandomizedSearchCV
>>> from scipy.stats import uniform
>>> iris = load_iris()
>>> logistic = LogisticRegression(solver='saga', tol=1e-2,
max_iter=200,random_state=0)
>>> distributions = dict(C=uniform(loc=0, scale=4),
penalty=['l2', 'l1'])
>>> clf = RandomizedSearchCV(logistic, distributions,
random_state=0)
>>> search = clf.fit(iris.data, iris.target)
>>> search.best_params_
{'C': 2..., 'penalty': 'l1'}
```



3. MODEL VALIDATION

Validation and testing begins with splitting your training dataset. The **"Valid-Test split"** is a technique to evaluate the performance of your **ML model**. You need to split the data because you don't want your model to over-learn from training data, to not perform well. But, most of all, you want to evaluate **how well your model is generalizing**.

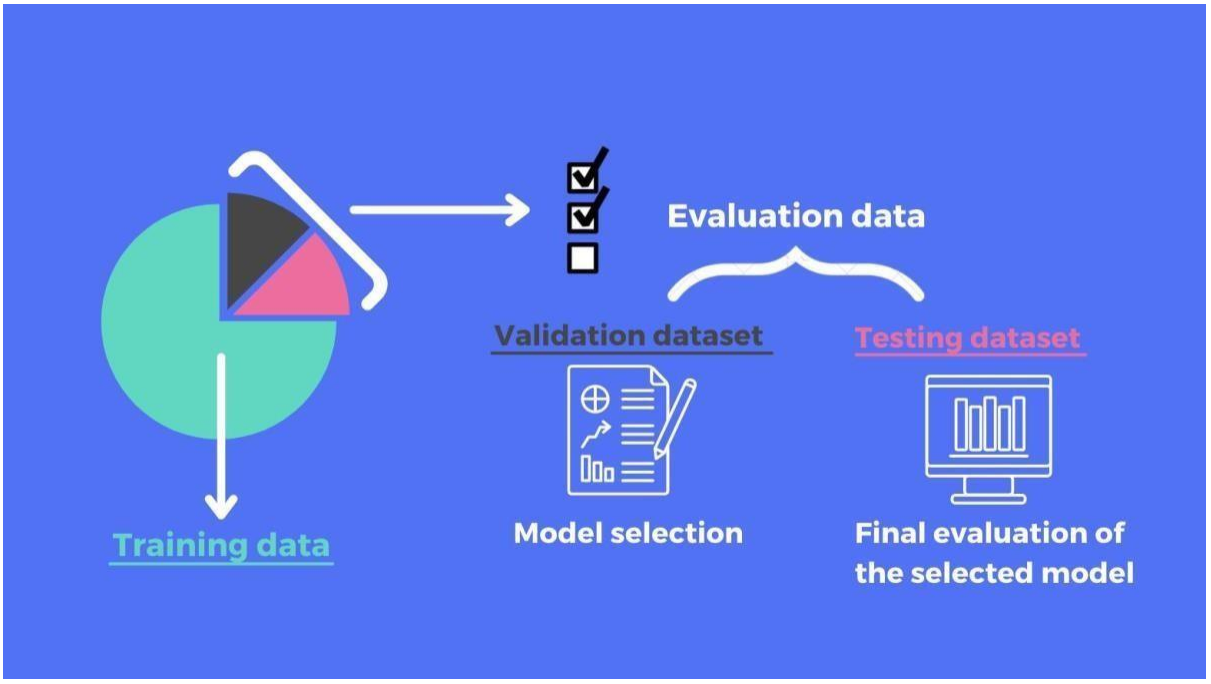
Hence, you held back from training dataset, **validation and testing** subsets for assessing your model in a meaningful way. Notice that a **typical split ratio** of data, between training, validation and testing sets is around . A brief explanation of the role of each of these dataset is below.

- **Validation dataset**: it is useful when it comes to **model selection**. The data included in this set will be used to find the optimal values for the parameters of the model under consideration. When you work with ML models, you typically need to **test multiple models** with different parameters values for finding the optimal values that will

give the best possible performance. Therefore, in order to **pick the best model** you must evaluate each of them.

- **Testing dataset**: when you have tuned the model by performing parameters optimisation, you should end up with the **final model**. The testing set is used to provide an **unbiased evaluation** of the performance of this model and ensure that it can generalise well to new, unseen data.

```
# Importing numpy & scikit-learn
```



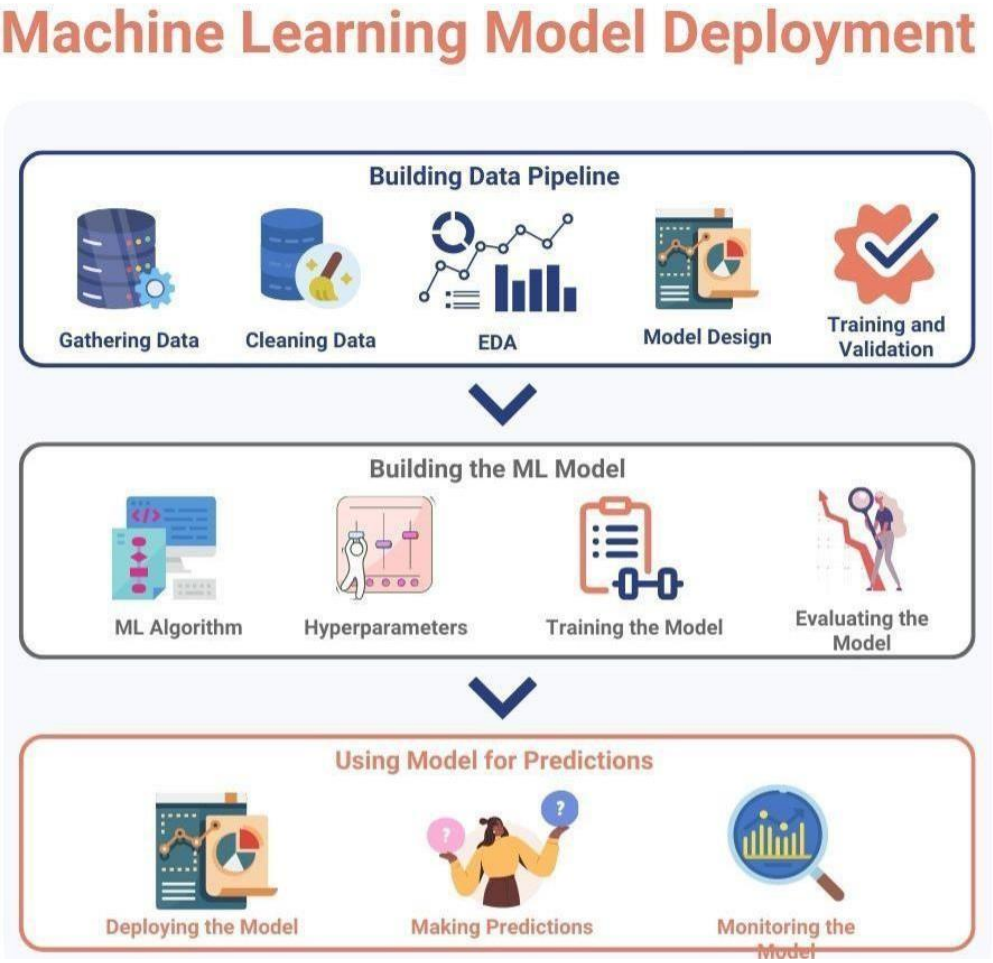
4. DEPLOYMENT

Machine learning model deployment is the process of placing a finished machine learning model into a live environment where it can be used for its intended purpose. Data science

models can be deployed in a wide range of environments, and they are often integrated with apps through an API so they can be accessed by end users.

While model deployment is the third stage of the data science lifecycle (manage, develop, deploy and monitor), every aspect of a model's creation is performed with deployment in mind.

Models are usually developed in an environment with carefully prepared data trained and sets, where they are tested . Most models created during the development stage meet desired objectives. Few models pass their test ose that do represent a sizable investment of ces. So moving a model into a dynamic environment quire a great deal of planning and preparation for the projectcessful.



5. MONITORING

Monitoring earthquake prediction is a critical task that involves the continuous collection and analysis of various data sources to detect potential seismic events. Here are some key aspects of monitoring earthquake prediction:

Seismic Activity Monitoring: Continuous monitoring of seismic activity using a network of seismometers and accelerometers. These instruments detect ground

motion and record seismic events, which are then analyzed to assess the likelihood of an earthquake.

Data Processing and Analysis: Collected seismic data is processed and analyzed to identify patterns and anomalies. Advanced algorithms and machine learning models can help detect precursors or changes in seismic activity that may indicate an impending earthquake.

Early Warning Systems: Some regions implement early warning systems that can provide a few seconds to minutes of advance notice before strong shaking from an earthquake reaches a specific location. These systems rely on real-time seismic data and communication infrastructure to alert the public and authorities.

GPS and Ground Deformation Monitoring: Monitoring the displacement of the Earth's crust through GPS and ground deformation sensors can help identify tectonic stress accumulation that may lead to an earthquake.

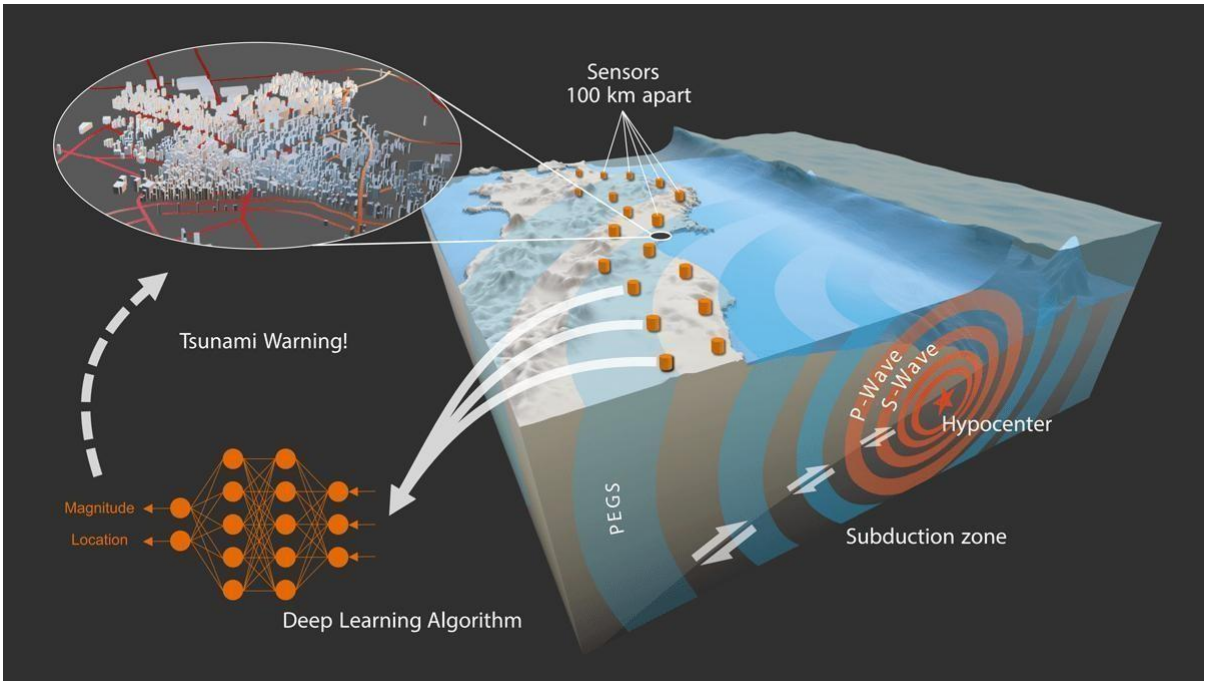
Monitoring Radon Gas and Other Precursors: Some researchers investigate unconventional precursors such as changes in radon gas emissions, groundwater levels, and animal behavior in an attempt to predict earthquakes

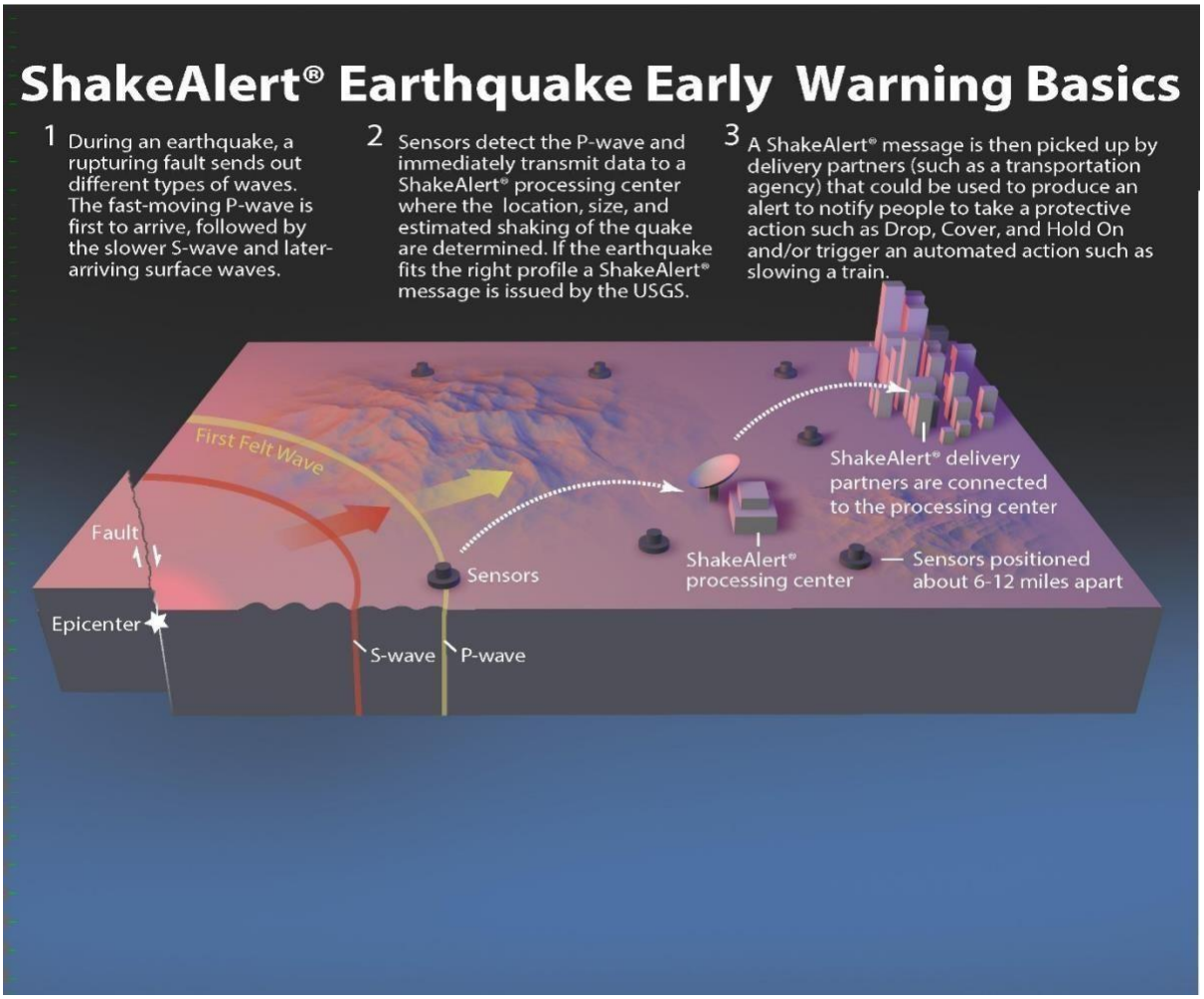
Satellite Imagery: Remote sensing techniques using satellites can be employed to monitor surface changes,

fault movements, and other geological features that might be associated with seismic activity.

Global Seismic Networks: International cooperation through global seismic networks allows the sharing of seismic data and information about earthquakes in different regions, contributing to a better understanding of seismic patterns.

Public Awareness and Education: In earthquake-prone regions, public awareness and education programs are essential to inform and prepare the population for potential seismic events.





6. MAINTENANCE

Maintaining earthquake prediction and monitoring systems is essential to ensure their continued effectiveness in providing early warnings and risk assessments. Here are key aspects of maintaining earthquake prediction systems

Instrument Maintenance: Regularly calibrate and maintain seismometers, accelerometers, GPS devices, and other monitoring instruments. This

ensures that data collection remains accurate and reliable.

Data Quality Assurance: Implement data quality control procedures to identify and address issues such as sensor malfunctions or data gaps promptly.

Network Reliability: Maintain a robust and redundant communication network for transmitting seismic data in real time. Redundancy is crucial to ensure data continuity during network failures.

Sensor Deployment and Upgrades: Periodically assess the location and density of monitoring sensors. If necessary, deploy additional sensors or upgrade existing ones to improve coverage and sensitivity.

Maintenance Scheduling: ML and DO Work Best Together