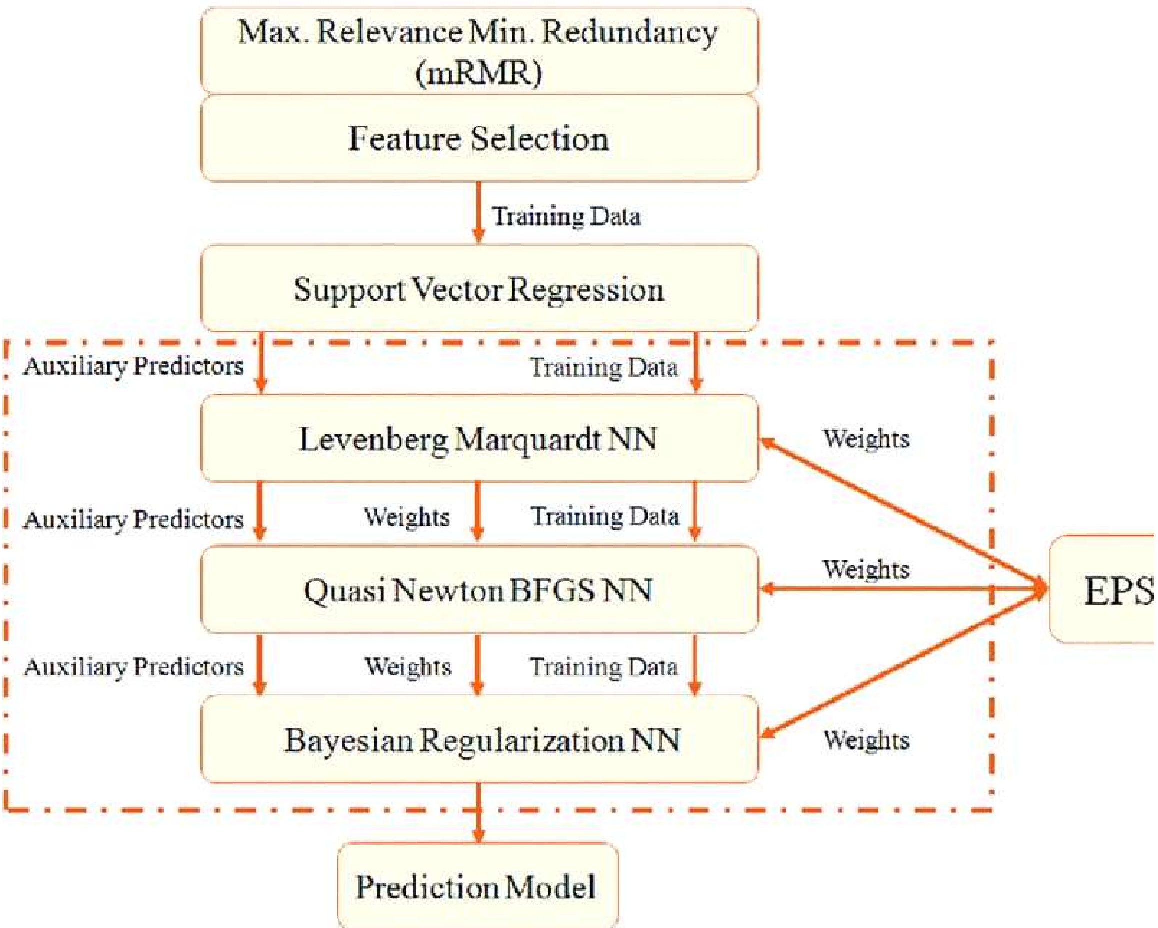
EARTHQUAKE PREDICTION MODEL BY USING PYTHON

PHASE 2:INNOVATION 1

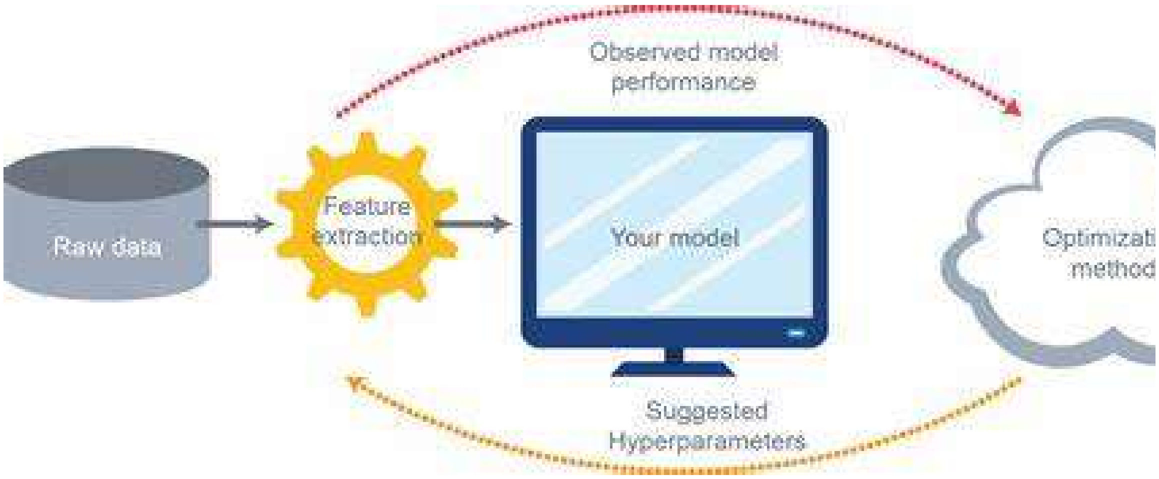
EARTHQUAKE PREDICTION FLOW CHAT



Introduction:

To improve the performance of a prediction model for earthquakes, you can apply advanced techniques like hyperparameter tuning and feature engineering.

Hyperparameter



> Use techniques like grid search or random search to systematically search through different combinations of hyperparameters for your model.

> Focus on hyperparameters specific to the chosen algorithm, such as learning rate, number of hidden layers or units in a neural network, depth of decision trees, etc.

>Utilize tools like cross-validation to assess the performance of different hyperparameter configurations

Random Search

Random search is another valuable technique for hyperparameter tuning in machine learning models, including those used for earthquake prediction.

1. Select a Model:

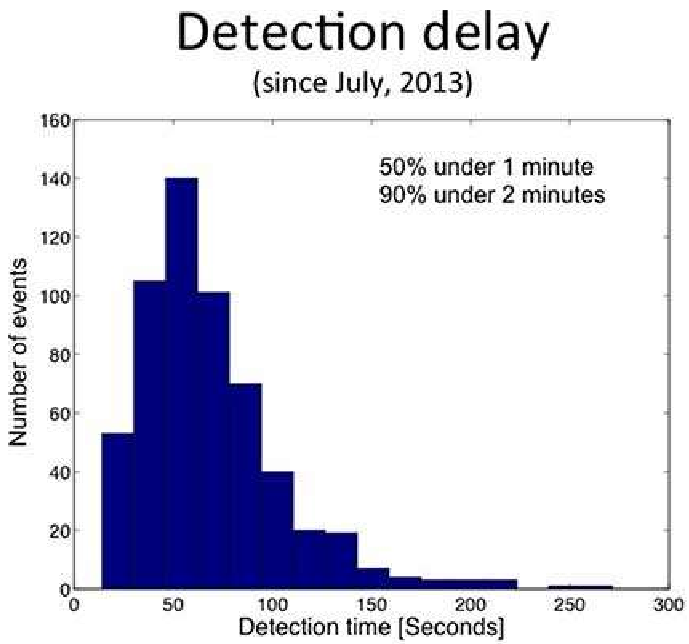


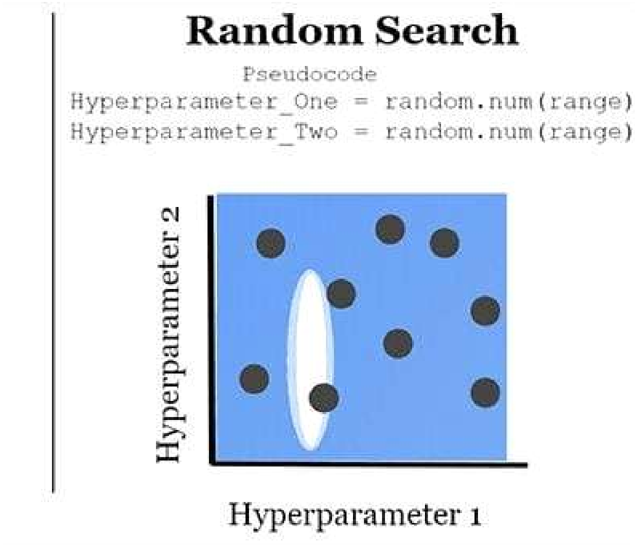
Figure 5. Tweet Based Event Detection Delay.
The histogram was derived from the integrated social media detected and seismically derived events dataset and indicates that half of all the tweet detections occur is less than one minute after earthquake origin time and ninety percent of the tweet based detections occur with in two minutes of the earthquake origin time.

>Choose the machine learning or deep learning model you want to use for earthquake prediction.

>This could be a decision tree, random forest, support vector machine, neural network, or any other suitable model.

2. Define Hyperparameter Ranges:

>Identify the hyperparameters of your chosen model that you want to tune. For earthquake prediction, these hyperparameters might include the learning rate, number of layers and units in a neural network, regularization strength, or kernel parameters for support vect



or machines. Define ranges or distributions for

each hyperparameter that you want to explore.

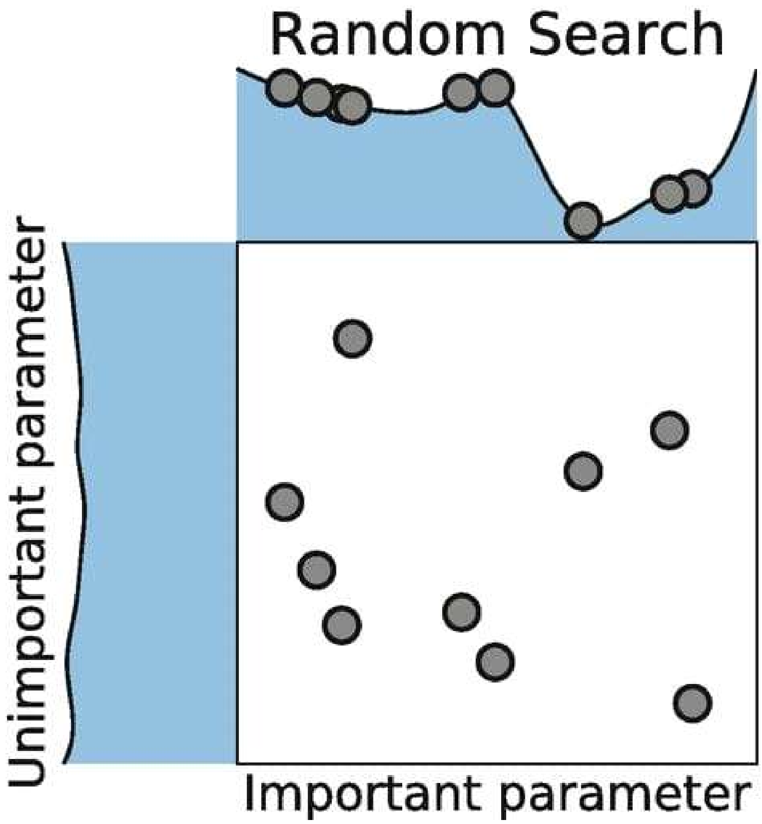
3. Split Data:

>Divide your earthquake dataset into training, validation, and test ets.

The training set is used to train the model, the validation set helps in

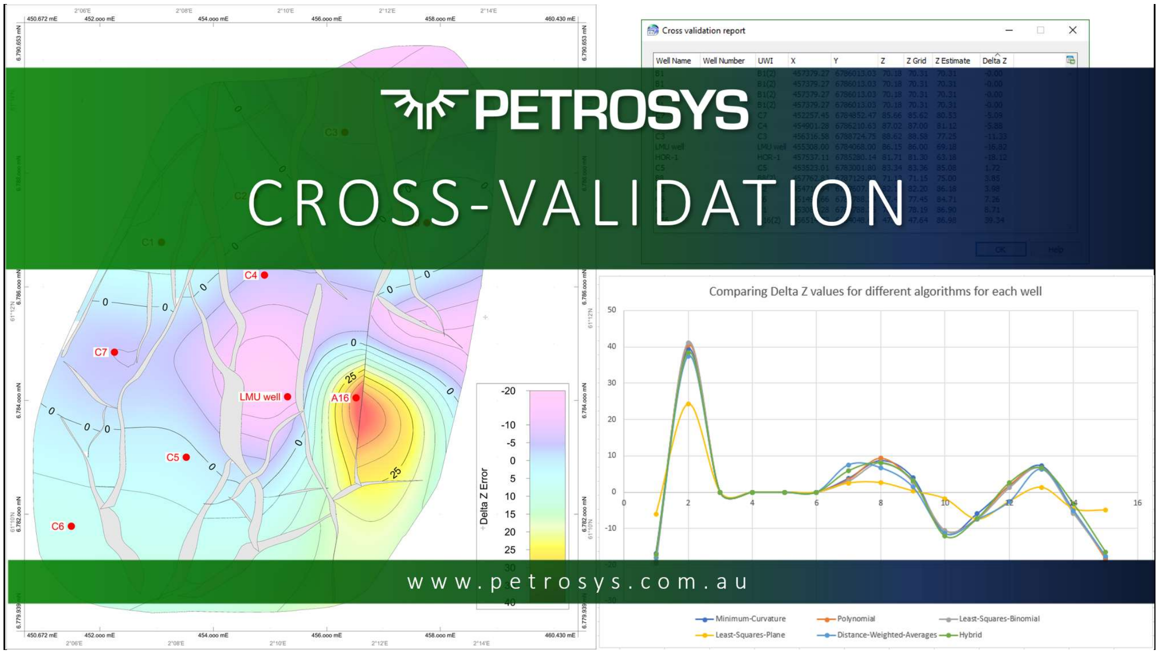
hyperparameter tuning, and the test set is used for final evaluation.

4. Random Search:



> Instead of systematically exploring all possible combinations like grid search, random search randomly selects hyperparameter combinations from the defined ranges. It samples hyperparameters according to a specified distribution for each hyperparameter.

5. Cross-Validation:

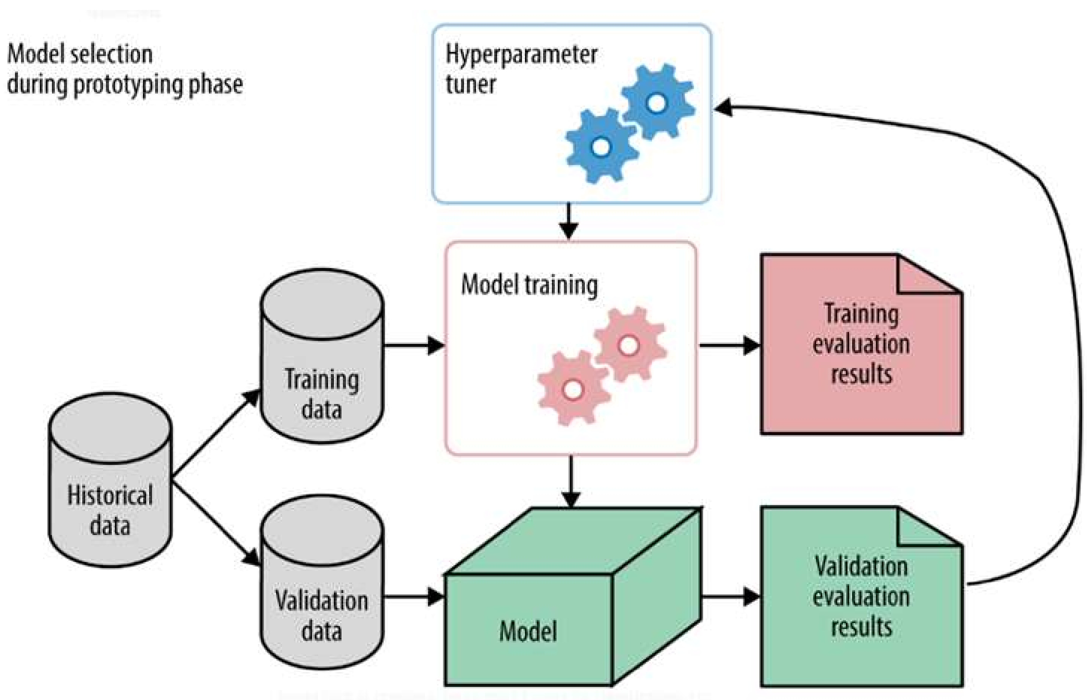


>Perform cross-validation on each randomly selected hyperparameter combination to evaluate model performance and prevent overfitting.

6. Evaluate Performance:

>Use a suitable evaluation metric for earthquake prediction, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or a domain-specific metric if available. The metric should reflect the accuracy and precision of earthquake predictions.

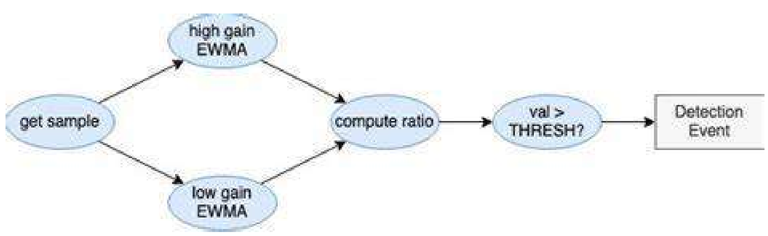
7. Select Best Hyperparameters:



>After running multiple random combinations, choose the combination

that yielded the best performance on the validation set.

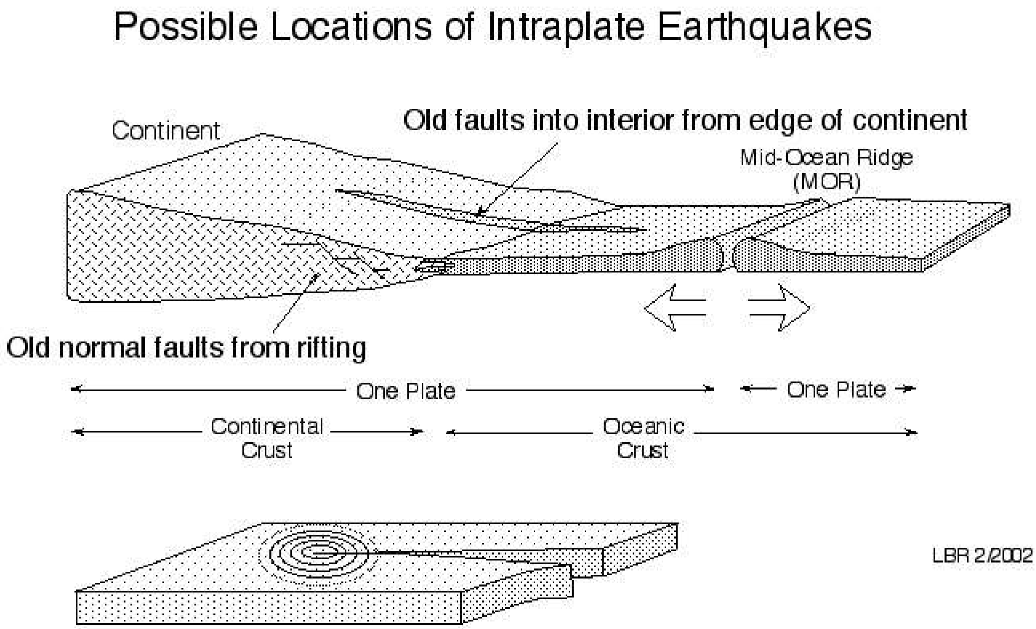
8. Final Model Training:



>Train a final model using the selected hyperparameters on the entire

training dataset (including the validation set).

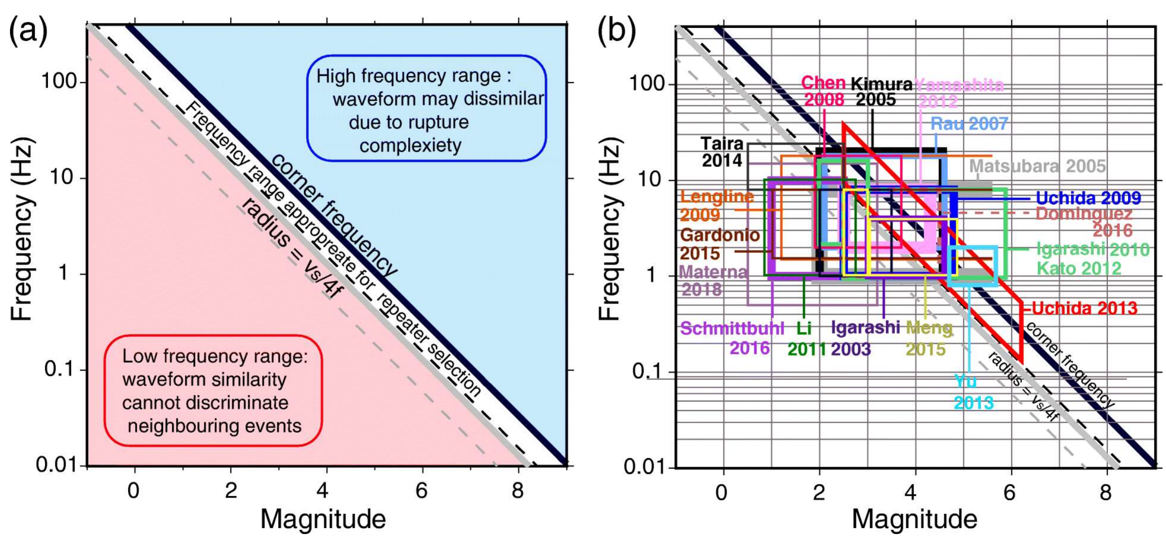
9.Model Evaluation:



>Assess the final model's performance on the test dataset to get an

unbiased estimate of its predictive accuracy.

10. Iterate if Necessary:



> If the model's performance is not satisfactory, you can repeat the random search process with a larger number of trials or explore additional hyperparameters. Continue iterating until you achieve the desired level of prediction accuracy.

For the instance,consider the following python source program

CODING

Python program:

```
import numpy as np

from sklearn.datasets import load_iris

from sklearn.ensemble import RandomForestRegressor

iris = load_iris()

rf=RandomForestRegressor(random_state=35)

X = iris.data

y = iris.target

n_estimators = [int(x) for x in np.linspace(start = 1, stop = 20, num = 20)]

max_features=['auto','sqrt']

max_depth = [int(x) for x in np.linspace(10, 120, num = 12)]

min_samples_split = [2, 6, 10]

min_samples_leaf = [1, 3, 4]

bootstrap = [True, False]

random_grid = {'n_estimators': n_estimators,

'max_features': max_features,

'max_depth': max_depth,

'min_samples_split': min_samples_split,
```

```
'min_samples_leaf': min_samples_leaf,
'bootstrap': bootstrap}
rf_random = RandomizedSearchCV(estimator = rf,
param_distributions = random_grid,
 n_iter = 100, cv = 5, verbose=2, random_state=35, n_jobs = -1)
rf_random.fit(X,y)
print ('Random grid: ', random_grid, '\n')
print('Best parameters:',rf_random.best_params_,'\n')
```

Output:

[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed: 3.6s finished

Random grid:

{'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20], 'max_features': ['auto', 'sqrt'], 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120], 'min_samples_split': [2, 6, 10], 'min_samples_leaf': [1, 3, 4], 'bootstrap': [True, False]}

Best Parameters:

{'n_estimators': 10, 'min_samples_split': 10, 'min_samples_leaf': 4, 'max_features': 'auto', 'max_depth': 70, 'bootstrap': True}

FeatureEngineering:

>Gather relevant domain knowledge or consult with experts to identify potentially important features related to earthquake prediction.

> Create new features based on existing data, like temporal features (time of day, day of the week, seasonality) or spatial features (distance to fault lines, geological characteristics).

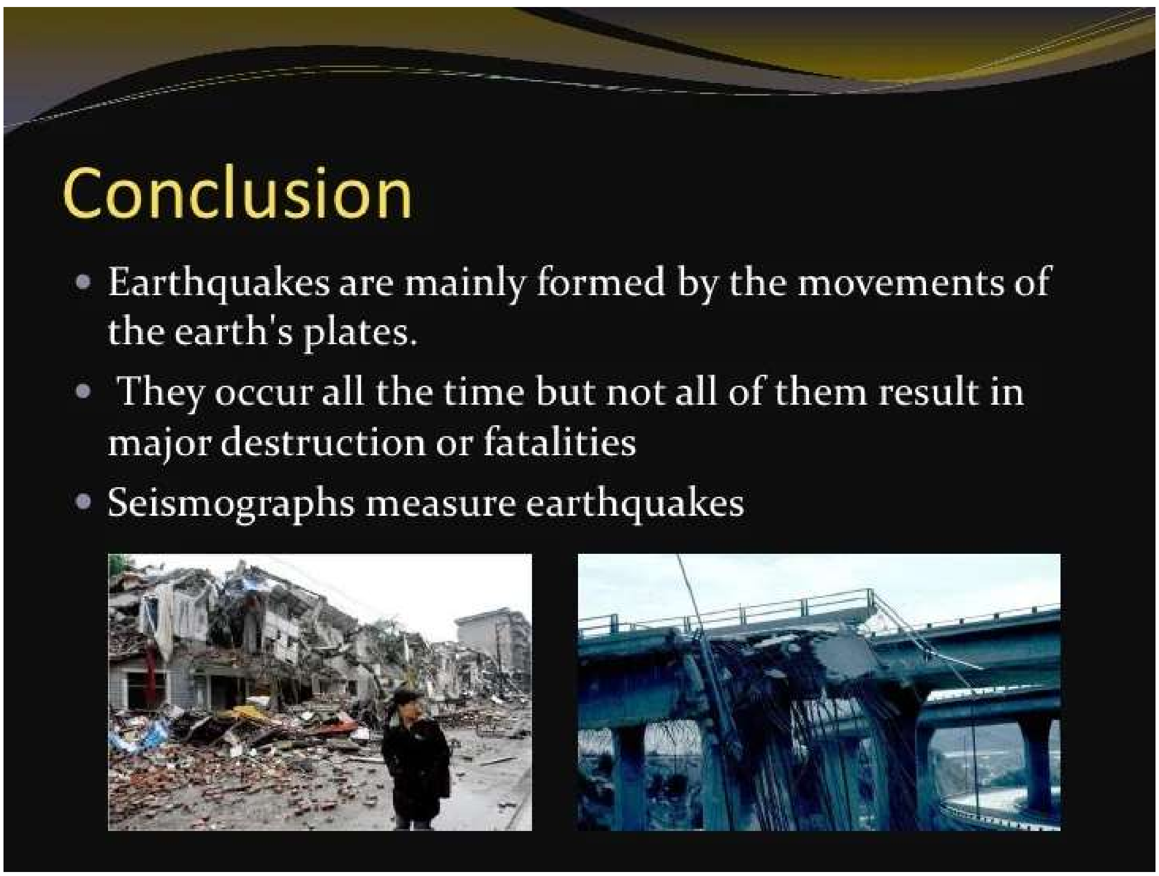
> Experiment with various feature selection techniques to identify the most informative attributes and reduce dimensionality if needed.

>Consider using domain-specific feature engineering methods tailored to seismology or geophysics.

Deployment:

> Once you have a well-tuned model, you can deploy it for real-time earthquake prediction, continuously monitoring incoming seismic data and providing alerts when necessary.

CONCLUSION:



>These innovations are part of ongoing efforts to improve earthquake prediction,

enhance early warning systems, and ultimately reduce the impact of seismic events on communities and infrastructure. While predicting earthquakes with high precision remains a complex challenge, these approaches aim to provide valuable insights and advance our understanding of seismic activity