EARTHQUAKE PREDICTION MODEL USING PYTHON

PHASE5 PROJECT

PREPARED BY,

M.PRIYANKA,
510521205030,
BHARATHIDASAN ENGINEERING COLLEGE,
PHASE5 PROJECT SUBMISSION.

CONTENTS

CONTENTS

CONTENTS

13 **ADVANTAGE**

14 DIS-
ADVANDAGE

15 BENEFITS

16 CONCLUSION

An earthquake is the shaking of the surface of the Earth resulting from a sudden release of energy in the Earth's lithosphere that creates seismic waves. At the Earth's surface, earthquakes manifest themselves by shaking and displacing or disrupting the ground.Sopredicting the factors of an earthquake is a challenging job as an earthquake does not show specific patterns resulting in inaccurate predictions. Techniques based on machine learning are well known for their capability to find hidden patterns in data. The machine learning model is built based on the past data related to earthquakes where the model can learn the pattern from the data and takes the consideration of the factors. The factors which are taken into consideration are the pre-processed data that is the dependency of the factors are checked in accordance with earthquake. Comparison of machine learning algorithms are done for better prediction and performance metrics are also calculated an evaluated.

02 INTRODUCTION

ADD A SHORT DESCRIPTION

2.1 EARTHQUAKE PREDICTION:

Earthquakes are natural disasters that can result in significant destruction and loss of life. Their sudden and often unpredictable nature makes them a constant concern for people living in seismically active regions around the world. Predicting earthquakes has been a longstanding goal in the field of seismology and geophysics, as early warning and preparedness can significantly mitigate the impact of these catastrophic events.

The study of earthquake prediction involves the analysis of various geological, geophysical, and seismological factors to forecast when and where an earthquake may occur. While the accurate prediction of specific earthquakes remains a formidable challenge, ongoing research and technological advancements have provided valuable insights into the understanding of seismic activity and improved early warning systems.

This introduction explores the complexities and limitations of earthquake prediction, the methods and tools used by scientists to make forecasts, and the potential benefits of advancing our ability to predict earthquakes for the safety and resilience of communities at risk. While we may not yet have the ability to predict earthquakes with pinpoint accuracy, ongoing research and the development of early warning systems bring us closer to reducing the devastating impact of these natural disasters.

2.2 PYTHON:

In this introduction to Python, we will explore its key features, applications, and the reasons behind its popularity in the world of programming. Whether you're a beginner looking to start your programming journey or an experienced developer seeking a powerful and efficient language, Python offers an excellent platform for creating a wide range of software solutions.

Preparing the Dataset:

The data set contains details of all earthquakes that have happened in the last 30 days and is updated every 15 mins in the USGS website. The data dictionary.
Below are the fields included in the CSV format:

Time
Latitude
Longitude
Depth
mag
magType
nst
gap
dmin
rms

Below are the fields included in the CSV format:

    net
    id
    updated
    place
    type
    locationSource
    magSource
    horizontalError
    depthError
    magError
    magNst
    status

3.1 DATASET

ADD A SHORT DESCRIPTION

Dataset Link:

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

Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seis	Magnitude	Magnitude	ID	Source	Location So	Magnitude	Status
1/2/1965	13:44:18	19.246	145.616	Earthquake	131.6			6	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/4/1965	11:29:49	1.863	127.352	Earthquake	80			5.8	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/5/1965	18:05:58	-20.579	-173.972	Earthquake	20			6.2	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/8/1965	18:49:43	-59.076	-23.557	Earthquake	15			5.8	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/9/1965	13:32:50	11.938	126.427	Earthquake	15			5.8	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/10/1965	13:36:32	-13.405	166.629	Earthquake	35			6.7	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/12/1965	13:32:25	27.357	87.867	Earthquake	20			5.9	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/15/1965	23:17:42	-13.309	166.212	Earthquake	35			6	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/16/1965	11:32:37	-56.452	-27.043	Earthquake	95			6	MW	ISCGEMSUP	ISCGEMSUP	ISCGEM	ISCGEM	Automatic
1/17/1965	10:43:17	-24.563	178.487	Earthquake	565			5.8	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/17/1965	20:57:41	-6.807	108.988	Earthquake	227.9			5.9	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/24/1965	0:11:17	-2.608	125.952	Earthquake	20			8.2	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/29/1965	9:35:30	54.636	161.703	Earthquake	55			5.5	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
2/1/1965	5:27:06	-18.697	-177.864	Earthquake	482.9			5.6	MW	ISCGEM859	ISCGEM	ISCGEM	ISCGEM	Automatic
2/2/1965	15:56:51	37.523	73.251	Earthquake	15			6	MW	ISCGEM859	ISCGEM	ISCGEM	ISCGEM	Automatic
2/4/1965	3:25:00	-51.84	139.741	Earthquake	10			6.1	MW	ISCGEM859	ISCGEM	ISCGEM	ISCGEM	Automatic
2/4/1965	5:01:22	51.251	178.715	Earthquake	30.3			8.7	MW	OFFICIAL19	OFFICIAL	ISCGEM	OFFICIAL	Automatic
2/4/1965	6:04:59	51.639	175.055	Earthquake	30			6	MW	ISCGEMSUP	ISCGEMSUP	ISCGEM	ISCGEM	Automatic
2/4/1965	6:37:06	52.528	172.007	Earthquake	25			5.7	MW	ISCGEM859	ISCGEM	ISCGEM	ISCGEM	Automatic
2/4/1965	6:39:32	51.626	175.746	Earthquake	25			5.8	MW	ISCGEM859	ISCGEM	ISCGEM	ISCGEM	Automatic
2/4/1965	7:11:23	51.037	177.848	Earthquake	25			5.9	MW	ISCGEMSUP	ISCGEMSUP	ISCGEM	ISCGEM	Automatic
2/4/1965	7:14:59	51.73	173.975	Earthquake	20			5.9	MW	ISCGEM859	ISCGEM	ISCGEM	ISCGEM	Automatic

04

EARTHQUAKE PREDICTION IN PYTHIN

ADD A SHORT DESCRIPTION

Python is a versatile and powerful programming language that has found extensive use in the field of earthquake prediction and seismology. The study of earthquake prediction, a critical aspect of geophysics, benefits greatly from Python's ease of use, extensive libraries, and strong community support. This introduction will highlight the significance of Python in earthquake prediction and its role in advancing our understanding of seismic activity.

Python's popularity in earthquake prediction can be attributed to several key factors:

Data Analysis and Visualization: Python's data analysis and visualization libraries, such as NumPy, Pandas, and Matplotlib, are invaluable for processing and interpreting seismic data. Scientists use these tools to analyze earthquake patterns, fault movements, and seismic signals.

Machine Learning and AI: Python is at the forefront of machine learning and artificial intelligence (AI) development. In earthquake prediction, machine learning models can be trained to identify earthquake precursors and patterns in seismic data, contributing to early warning systems and forecasting efforts.

Accessibility: Python's simple and readable syntax lowers the entry barrier for researchers, allowing them to focus on the science rather than complex coding. This accessibility has led to a wider adoption of Python in the seismological community.

Numerical and Scientific Computing: Python's libraries like SciPy and scikit-learn enable researchers to perform complex numerical computations and statistical analysis, crucial for understanding seismic behavior and improving prediction models.

[Open-Source Community:](#) Python's open-source nature fosters collaboration and the sharing of tools and libraries, enabling seismologists to access a wealth of resources created by experts worldwide.

In this context, Python is not just a programming language but a comprehensive ecosystem for earthquake prediction. Scientists and researchers can develop and refine models, conduct in-depth data analysis, and build early warning systems using Python. As we delve into the intricacies of earthquake prediction with Python, we will discover the numerous ways this programming language empowers seismologists in their efforts to mitigate the impact of earthquakes and protect vulnerable communities.

05

PYTHON TOOLS

ADD A SHORT DESCRIPTION

Creating an earthquake prediction model using Python involves various tools and libraries that enable data processing, feature engineering, model development, and evaluation. Below are some essential tools and libraries commonly used for building earthquake prediction models in Python:

NumPy and Pandas: NumPy and Pandas are fundamental libraries for data manipulation, allowing you to efficiently work with numerical data and structured datasets, respectively.

Matplotlib and Seaborn: These libraries are crucial for data visualization, helping you create meaningful plots and graphs to analyze seismic data and visualize patterns.

SciPy: SciPy provides scientific and statistical functions, including tools for signal processing and statistical analysis, which are vital for earthquake prediction.

[scikit-learn:](#) Scikit-learn is a machine learning library in Python that offers a wide range of tools for building and evaluating machine learning models. You can use this library to implement various algorithms for earthquake prediction.

[TensorFlow or PyTorch:](#) These deep learning frameworks are used for building neural network models for earthquake prediction, particularly for more complex and data-intensive applications.

[Keras:](#) Keras is a high-level neural networks API that runs on top of TensorFlow or other backends. It simplifies the process of building and training neural networks for seismic data.

XGBoost or LightGBM: These gradient boosting libraries are powerful for building ensemble models and making predictions based on various features.

Feature Engineering Libraries: Tools like tsfresh can help with automated feature extraction from time series data, which is common in earthquake prediction.

Jupyter Notebook: Jupyter Notebook is a popular interactive environment for data analysis and model development, allowing you to document your work step by step.

Geospatial Libraries: Libraries like Shapely and GeoPandas are essential for working with geospatial data, which is often crucial in understanding earthquake location and movement.

Hypothesis Testing Libraries: Libraries like Statsmodels can be useful for hypothesis testing and statistical analysis of earthquake-related data.

Seismic Data Libraries: Some specialized libraries like ObsPy provide tools and data formats specific to seismology and seismological research.

Database Management: Libraries like SQLAlchemy can be used for managing and querying large seismic databases efficiently.

Data Collection and Web Scraping: Libraries like Requests and BeautifulSoup can help you gather relevant seismic data from various sources, including government agencies and research institutions.

Geospatial Visualization Tools: Tools like Folium can help you create interactive maps to visualize earthquake data geospatially.

Time Series Analysis Libraries: Libraries like Statsmodels and Prophet can be helpful for analyzing time series data and forecasting.

Building an earthquake prediction model in Python often involves a combination of these tools and libraries, depending on the specific goals and requirements of the project. Keep in mind that earthquake prediction is a complex and challenging task, and models should be developed with a deep understanding of seismology and the limitations of current predictive capabilities.

06

PROBLEM
DEFINITION AND
DESIGN THINKING

Design thinking is an innovative approach to problem-solving that prioritizes empathy, creativity, and collaboration. When applying design thinking to the problem of earthquake prediction, the focus is on understanding the needs of the stakeholders, developing innovative solutions, and fostering a multidisciplinary approach to tackle this complex issue.

1. Empathize (Understand the Problem):

Identify Stakeholders: Begin by identifying the key stakeholders involved in earthquake prediction, including scientists, government agencies, emergency responders, and the general public. Understand their perspectives, needs, and concerns.

Gather Data: Collect and analyze relevant data related to past earthquakes, seismic activity, and predictive models. This data will serve as the foundation for problem understanding.

2. Define (Define the Problem):

Problem Statement: Clearly define the problem based on insights gathered during the empathize stage. For example, "The challenge is to enhance the accuracy and timeliness of earthquake prediction to reduce the potential impact on human lives and infrastructure."

Success Criteria: Establish specific criteria for a successful earthquake prediction system, such as the accuracy of predictions and the lead time for warnings.

3. Ideate (Generate Solutions):

Brainstorming: Bring together a multidisciplinary team, including seismologists, data scientists, engineers, and social scientists, to brainstorm innovative solutions. Encourage the team to think beyond traditional approaches.

Technology Exploration: Investigate emerging technologies like machine learning, deep learning, IoT, and geospatial analysis that could enhance earthquake prediction capabilities.

4. Prototype (Create a Prototype Solution):

Develop a prototype solution that integrates the innovative ideas generated during the ideation stage. This may include creating a model or system that incorporates data analysis, early warning systems, or novel sensor networks.
Utilize simulation tools to test the prototype and refine it based on the feedback received from experts and stakeholders.

5. Test (Test the Prototype):

Deploy the prototype in a controlled environment or use historical earthquake data to assess its effectiveness. Gather feedback and refine the solution based on the results.
Collaborate with the seismology community to validate the model's accuracy and reliability.

6. Implement (Implement the Solution):

Collaborate with government agencies, research institutions, and technology companies to implement the solution on a broader scale.

Develop a plan for integrating the solution into existing earthquake prediction infrastructure.

7. Evaluate (Evaluate and Improve):

Continuously monitor the performance of the earthquake prediction system and gather user feedback.

Make iterative improvements to the system based on real-world data and insights from end-users.

8. Communicate (Communicate Findings):

Share the results and findings with the public, government agencies, and other stakeholders to build trust and awareness about the capabilities and limitations of earthquake prediction.

Engage in public education and outreach to inform communities about earthquake preparedness.

Design thinking for earthquake prediction involves a dynamic, collaborative, and iterative process that strives to create innovative and practical solutions. It acknowledges the complexity of the problem and encourages cross-disciplinary collaboration to develop more effective ways of mitigating the impact of earthquakes.

07

EARTHQUAKE
PREDICTION
TECNIQUES

Earthquake prediction is a challenging and complex field, and while it is not possible to predict specific earthquakes with pinpoint accuracy, there are several techniques and approaches used for earthquake forecasting and early warning systems. These techniques aim to provide advance notice of seismic activity and reduce the potential impact on human lives and infrastructure. Here are some key earthquake prediction techniques:

Seismic Monitoring:

Seismometers and accelerometers are deployed in seismically active regions to monitor ground motion and detect seismic waves.

Earthquake early warning systems use real-time seismic data to issue alerts before strong shaking reaches populated areas.

Historical Data Analysis:

Analysis of historical earthquake records and geological data can help identify patterns and recurrence intervals of earthquakes in specific regions.

This information is used to estimate the likelihood of future seismic events.

GPS and Geodetic Measurements:

Continuous GPS and geodetic monitoring can detect tectonic plate movements, strain buildup, and deformations in the Earth's crust, which may indicate increased seismic risk.

Stress and Strain Analysis:

Researchers analyze stress and strain changes in fault zones to assess the likelihood of an impending earthquake. Coulomb stress transfer models are used to understand how one earthquake can influence the likelihood of another in nearby fault areas.

Foreshock and Aftershock Analysis:

Foreshocks are smaller earthquakes that precede a larger mainshock. Monitoring foreshock sequences can provide an early indication of increased seismic activity. Aftershocks, which occur after a major earthquake, are also studied to assess the ongoing seismic risk.

Animal Behavior and Anecdotal Reports:

Some studies have explored the potential for unusual animal behavior or anecdotal reports from residents to serve as precursors to earthquakes, though these methods are less reliable.

Machine Learning and Data Analytics:

Advanced machine learning algorithms are used to analyze vast amounts of seismic and geospatial data to identify patterns and make predictions.
Deep learning models can be trained on historical data to recognize seismic precursor signals.

Earthquake Early Warning Systems:

These systems use real-time data from seismometers and accelerometers to provide warnings seconds to minutes before strong shaking from an earthquake reaches a specific location.
They are particularly valuable for critical infrastructure, such as transportation and utilities.

Probabilistic Seismic Hazard Assessment (PSHA):

PSHA combines geological and seismic data to estimate the probability of ground shaking at different levels of intensity over a specific time frame (e.g., 50 years).
It helps in assessing seismic risk for urban planning and building design.

Remote Sensing and Satellite Technology:

Satellite imagery and remote sensing can be used to monitor surface changes and crustal deformations, providing valuable data for assessing earthquake risk.

It's important to note that while these techniques are valuable for understanding seismic activity and reducing risk, earthquake prediction remains an ongoing area of research. The ability to accurately predict the timing, location, and magnitude of specific earthquakes is still a significant challenge, and earthquake preparedness, early warning systems, and building resilience to seismic events continue to be crucial components of earthquake risk reduction.

08

LOADING AND PROCESSING DATASET

ADD A SHORT DESCRIPTION

Loading and processing earthquake prediction datasets involves several steps, from acquiring the data to cleaning, preparing, and analyzing it. Here's a general guide on how to handle earthquake prediction datasets using Python as an example:

Data Acquisition:

Obtain earthquake-related datasets from reliable sources. These sources may include seismological agencies, research institutions, or open data repositories.

Data Loading:

Use Python libraries like Pandas to load the dataset into your environment. You can typically load data from CSV, Excel, JSON, or other common formats.

python

import pandas as pd

# Load earthquake dataset from a CSV file
earthquake_data = pd.read_csv('earthquake_data.csv')

Data Exploration:

Start by exploring the dataset to understand its structure and content. Use functions like head(), info(), and describe() to get a sense of the data.
python

```python
# Display the first few rows of the dataset
print(earthquake_data.head())

# Get general information about the dataset
print(earthquake_data.info())

# Summary statistics
print(earthquake_data.describe())
```

Data Cleaning:

Address missing values, outliers, and inconsistencies in the dataset. You may choose to remove or impute missing data, and correct any data quality issues.
python
Copy code

```python
# Remove rows with missing values
earthquake_data = earthquake_data.dropna()

# Handle outliers
# ...

# Data type conversion (if necessary)
# ...
```

Feature Engineering:

Create relevant features for earthquake prediction. This may involve extracting information from timestamps, geospatial data, or transforming data to make it suitable for modeling.

```
# Example: Extract year, month, and day from a timestamp column
earthquake_data['year'] = pd.to_datetime(earthquake_data['timestamp']).dt.year
earthquake_data['month'] = pd.to_datetime(earthquake_data['timestamp']).dt.month
earthquake_data['day'] = pd.to_datetime(earthquake_data['timestamp']).dt.day
```

Data Visualization:

Utilize libraries like Matplotlib and Seaborn to create visualizations that help in understanding the data's distribution, patterns, and correlations.
python
Copy code

```python
import matplotlib.pyplot as plt
import seaborn as sns


# Create a scatter plot of earthquake magnitudes vs. depths
plt.figure(figsize=(8, 6))
sns.scatterplot(x='magnitude', y='depth', data=earthquake_data)
plt.title('Magnitude vs. Depth')
plt.xlabel('Magnitude')
plt.ylabel('Depth')
plt.show()
```

Data Preprocessing:

Prepare the data for machine learning by encoding categorical variables, normalizing or standardizing numerical features, and splitting it into training and testing sets.

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Encode categorical variables
label_encoder = LabelEncoder()
earthquake_data['location_encoded'] =
label_encoder.fit_transform(earthquake_data['location'])

# Normalize numerical features
scaler = StandardScaler()
earthquake_data[['magnitude', 'depth']] =
scaler.fit_transform(earthquake_data[['magnitude', 'depth']])
```

Model Training:

Apply machine learning or statistical modeling techniques to build an earthquake prediction model. Various algorithms such as decision trees, random forests, support vector machines, and neural networks may be used.
python

```python
from sklearn.ensemble import RandomForestClassifier

# Create and train a Random Forest classifier
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)
```

Model Evaluation:

Evaluate the model's performance using appropriate metrics such as accuracy, precision, recall, and F1-score.

```python
from sklearn.metrics import accuracy_score, classification_report

y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(report)
```

Deployment and Monitoring:

If the model performs well, deploy it in a production environment for real-time predictions. Monitor the model's performance and retrain it as needed.

The specific steps and techniques will vary depending on the dataset, the problem's complexity, and the chosen modeling approach. This guide provides a general framework for working with earthquake prediction datasets using Python.

09

FEATURE ENGINEERING

ADD A SHORT DESCRIPTION

Feature engineering is a crucial aspect of earthquake prediction, as it involves creating meaningful and informative features from the available data to improve the predictive capabilities of models. In the context of earthquake prediction, feature engineering involves extracting relevant information from seismic and geospatial data, as well as other related datasets. Here are some feature engineering techniques you can use for earthquake prediction:

Time-Based Features:

Extract temporal information from earthquake data, such as the time of day, day of the week, and month of the year. Certain time patterns may be associated with increased seismic activity.

```
# Example: Extract the hour of the day from a timestamp
earthquake_data['hour'] = pd.to_datetime(earthquake_data['timestamp']).dt.hour
```

Geospatial Features:

Incorporate location-specific information. Features like distance to fault lines, proximity to tectonic plate boundaries, and geological characteristics of the region can be valuable.

```
# Example: Calculate the distance to the nearest fault line
earthquake_data['distance_to_fault'] =
calculate_distance_to_fault(earthquake_data['latitude'],
earthquake_data['longitude'])
```

Magnitude-Related Features:

Magnitude-related features can provide insight into seismic activity. You can compute statistical measures of magnitudes, such as the maximum, minimum, or average magnitude over a specific period.

```
# Example: Calculate the average magnitude in the last 30 days
earthquake_data['avg_magnitude_30_days'] = earthquake_data['magnitude'].rolling('30D').mean()
```

Seismic Activity Patterns:

Create features that capture patterns in seismic activity, such as the number of foreshocks and aftershocks, seismic swarm indicators, or seismicity rates over time.

```
# Example: Count the number of foreshocks in a given time window
earthquake_data['foreshocks_count'] = count_foreshocks(earthquake_data, time_window=7)
```

Strain and Stress Features:

Features related to stress and strain in the Earth's crust, such as Coulomb stress, can be useful in understanding earthquake risk. These features may involve complex geophysical calculations.

```
# Example: Calculate Coulomb stress change in the region
earthquake_data['coulomb_stress'] = calculate_coulomb_stress(earthquake_data['latitude'], earthquake_data['longitude'])
```

Historical Earthquake Data:

Features based on historical earthquake data, such as the number of earthquakes in the same region over a specific time frame, can help in identifying earthquake-prone areas.

```
# Example: Count the number of earthquakes in the same
region in the past year
earthquake_data['earthquakes_in_same_region_1_year'] =
count_earthquakes_in_region(earthquake_data,
time_window='365D')
```

Geological and Tectonic Features:

Features describing geological properties, tectonic plate boundaries, and fault types can be incorporated to capture the geological context.

```
# Example: Encode tectonic plate type as a categorical feature
earthquake_data['tectonic_plate_type'] =
encode_tectonic_plate_type(earthquake_data['latitude'],
earthquake_data['longitude'])
```

Feature Aggregation:

Create aggregated features over specific time windows, such as moving averages, rolling sums, or statistical measures like standard deviation.

```
# Example: Calculate the rolling standard deviation of earthquake magnitudes over 30 days
earthquake_data['magnitude_std_30_days'] = earthquake_data['magnitude'].rolling('30D').std()
```

Frequency Domain Features:

Transform seismic signals into the frequency domain and extract spectral features, which can provide insights into the characteristics of seismic waves.

```
# Example: Compute the dominant frequency of seismic signals
earthquake_data['dominant_frequency'] = compute_dominant_frequency(earthquake_data['seismic_signals'])
```

Feature engineering in earthquake prediction is highly domain-specific, and the choice of features depends on the available data and the insights provided by domain experts. It's important to continuously refine and experiment with feature engineering to improve the performance of earthquake prediction models. Additionally, machine learning techniques can be used to automate feature selection and extraction for large and complex datasets.

Earthquake prediction involves using various machine learning and statistical modeling techniques to develop models that can forecast the occurrence of earthquakes. While it is important to note that predicting specific earthquakes with precise accuracy remains a significant challenge, the focus is often on assessing seismic risk and providing early warnings. Here are some model training techniques commonly used in earthquake prediction:

Logistic Regression:

Logistic regression can be used for binary classification tasks in earthquake prediction, such as determining whether a region is at high or low risk of experiencing a significant earthquake based on various features.

python

```python
from sklearn.linear_model import LogisticRegression

# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

Random Forest:

Random Forest is an ensemble learning method that combines multiple decision trees. It can capture complex relationships in the data and is often used for earthquake risk assessment.

python

```
from sklearn.ensemble import RandomForestClassifier

# Train a random forest classifier
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

Support Vector Machines (SVM):

SVM is a powerful algorithm for classification tasks. It can be used to build models that distinguish between areas at high risk and low risk of earthquakes.

python

```python
from sklearn.svm import SVC

# Train a support vector machine classifier
model = SVC(kernel='linear')
model.fit(X_train, y_train)
```

Gradient Boosting:

Gradient boosting methods like XGBoost or LightGBM are often employed to improve model performance by combining the predictions of multiple weak models.

python

```
import xgboost as xgb

# Train an XGBoost classifier
model = xgb.XGBClassifier()
model.fit(X_train, y_train)
```

Neural Networks:

Deep learning techniques, such as neural networks, can be applied to seismic data for earthquake prediction. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are common choices.

```python
python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
# Build and train a neural network model
model = Sequential()
model.add(LSTM(32, input_shape=(X_train.shape[1], 1)))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam')
model.fit(X_train, y_train, epochs=50, batch_size=64)
```

Time Series Forecasting Models:

Time series forecasting models like ARIMA (AutoRegressive Integrated Moving Average) and Prophet can be used to predict future seismic activity based on historical data.

```python
from statsmodels.tsa.arima.model import ARIMA
from fbprophet import Prophet
# Train an ARIMA model
model = ARIMA(earthquake_data, order=(5, 1, 0))
model_fit = model.fit()
# Train a Prophet model
model = Prophet()
model.fit(earthquake_data)
```

Hybrid Models:

Combining multiple models or approaches, such as integrating statistical and machine learning models, can enhance prediction accuracy.

python

```
# Example: Create a hybrid model that combines logistic regression and a decision tree
model1 = LogisticRegression()
model2 = DecisionTreeClassifier()
model1.fit(X_train, y_train)
model2.fit(X_train, y_train)
predictions1 = model1.predict(X_test)
predictions2 = model2.predict(X_test)
```

Ensemble Models:

Ensemble models, like stacking or bagging, combine the predictions of multiple models to improve overall performance. For earthquake prediction, you can create ensemble models to leverage the strengths of different algorithms.

```python
from sklearn.ensemble import StackingClassifier
# Create a stacking ensemble of multiple classifiers
estimators = [('rf', RandomForestClassifier(n_estimators=100)),
         ('svc', SVC(kernel='linear'))]
model = StackingClassifier(estimators=estimators,
final_estimator=LogisticRegression())
model.fit(X_train, y_train)
```

The choice of model depends on the specific problem, available data, and the goals of the earthquake prediction system. Experimentation and rigorous evaluation of different models, along with ongoing validation with new data, are crucial in developing effective earthquake prediction models. Additionally, model performance metrics like accuracy, precision, recall, and F1-score should be used to assess the quality of the predictions.

11

EVALUTION TECHNIQUES

ADD A SHORT DESCRIPTION

Evaluating the performance of earthquake prediction models is crucial to assess their effectiveness and reliability. While predicting specific earthquakes is extremely challenging, evaluation techniques can help measure a model's ability to assess seismic risk, provide early warnings, or predict seismic events. Here are some common evaluation techniques for earthquake prediction models:

Accuracy:

Accuracy is a basic measure of how well the model classifies earthquake events correctly. It calculates the ratio of correctly predicted events to the total number of events.

python

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(true_labels, predicted_labels)

Precision and Recall:

Precision measures the proportion of true positive predictions among all positive predictions, while recall (sensitivity) measures the proportion of true positive predictions among all actual positive instances. These metrics are especially important when dealing with imbalanced datasets.
python

```python
from sklearn.metrics import precision_score, recall_score

precision = precision_score(true_labels, predicted_labels)
recall = recall_score(true_labels, predicted_labels)
```

F1-Score:

The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance, particularly when the dataset has class imbalance.

python

```
from sklearn.metrics import f1_score

f1 = f1_score(true_labels, predicted_labels)
```

Area Under the ROC Curve (AUC-ROC):

AUC-ROC measures the model's ability to distinguish between positive and negative cases, considering various threshold settings. A higher AUC-ROC value indicates better performance.

python

from sklearn.metrics import roc_auc_score

roc_auc = roc_auc_score(true_labels, predicted_probabilities)

Confusion Matrix:

The confusion matrix provides a detailed breakdown of true positives, true negatives, false positives, and false negatives, allowing a deeper understanding of a model's performance.

python

```python
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(true_labels, predicted_labels)
```

Receiver Operating Characteristic (ROC) Curve:

The ROC curve is a graphical representation of the trade-off between true positive rate and false positive rate at various threshold settings. It is particularly useful for binary classification models.

```python
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
fpr, tpr, thresholds = roc_curve(true_labels, predicted_probabilities)
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
```

Precision-Recall Curve:

The precision-recall curve visualizes the trade-off between precision and recall at different threshold settings. It is valuable for imbalanced datasets and when positive class identification is more important.
python

```python
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt

precision, recall, thresholds = precision_recall_curve(true_labels, predicted_probabilities)
plt.plot(recall, precision)
plt.xlabel('Recall')
```

Mean Absolute Error (MAE) and Mean Squared Error (MSE):

If the earthquake prediction task involves regression (predicting a continuous value, such as earthquake magnitude), metrics like MAE and MSE can assess the model's accuracy in estimating the target value.

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(true_values, predicted_values)
mse = mean_squared_error(true_values, predicted_values)
```

Root Mean Square Error (RMSE):

RMSE is the square root of the mean squared error, providing a measure of prediction errors in the same unit as the target variable.

python

```python
rmse = np.sqrt(mse)
```

Cross-Validation:

Cross-validation techniques, such as k-fold cross-validation, help assess a model's performance on different subsets of the data. It provides a more robust evaluation by reducing the risk of overfitting.

python

```python
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=5)
```

Evaluating earthquake prediction models is a complex task due to the unique nature of seismic events and the challenge of dealing with imbalanced and highly variable data. It is essential to choose evaluation metrics that are suitable for the specific goals of the model, whether it is assessing seismic risk, providing early warnings, or predicting seismic activity. Additionally, continuous model monitoring and validation with new data are crucial in maintaining model effectiveness over time.

12 CODING

ADD A SHORT DESCRIPTION

12.1 Data-Validation & Pre-processing:

```
importpandasasp
importnumpyasn
importwarnings
warnings.filterwarnings('ignore')
data=p.read_csv('demo1.csv')
data.head()
data.shape
data.columns
data.isnull().sum()
```

Drop the Given Dataset

```
df=data.dropna()
df.shape
df.isnull().sum()
df.describe()
df.columns
df.magType.unique()
df.status.unique()
df.locationSource.unique()
df.magSource.unique()
df.net.unique()
```

```python
p.Categorical(df['locationSource'])
p.Categorical(df['net']).describe()
p.Categorical(df['magSource']).describe()
92
p.Categorical(df['locationSource']).describe()
df.gap.unique()
print("Gap: ",sorted(df['gap'].unique()))
df.info()
df.duplicated()
sum(df.duplicated())
p.crosstab(df.horizontalError,df.depthError)
```

```
df.columns
print("Minimum value of Depth is:",df.depth.min())
print("Maximum value of Depth is:",df.depth.max())
print("Minimum value of Gap is:",df.gap.min())
print("Maximum value of Gap is:",df.gap.max())
print("Minimum value of Latitude is:",df.latitude.min())
print("Maximum value of Latitude is:",df.latitude.max())
print("Minimum value of Longitude is:",df.longitude.min())
print("Maximum value of Longitude is:",df.longitude.max())
print("Minimum value of Gap is:",df.gap.min())
print("Maximum value of Gap is:",df.gap.max())
p.Categorical(df['magType']).describe()
```

```python
p.Categorical(df['status']).describe()
p.Categorical(df['locationSource']).describe()
df['mag'].value_counts()
df.corr()
fromsklearn.preprocessingimportLabelEncoder
var_mod=['magType','status','locationSource']
le=LabelEncoder()
foriinvar_mod:
df[i]=le.fit_transform(df[i]).astype(int)
df.head()
```

OUTPUT:

Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seis	Magnitude	Magnitude	ID	Source	Location So	Magnitude	Status
1/2/1965	13:44:18	19.246	145.616	Earthquake	131.6			6	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/4/1965	11:29:49	1.863	127.352	Earthquake	80			5.8	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/5/1965	18:05:58	-20.579	-173.972	Earthquake	20			6.2	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/8/1965	18:49:43	-59.076	-23.557	Earthquake	15			5.8	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/9/1965	13:32:50	11.938	126.427	Earthquake	15			5.8	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/10/1965	13:36:32	-13.405	166.629	Earthquake	35			6.7	MW	ISCGEM860	ISCGEM	ISCGEM	ISCGEM	Automatic
1/12/1965	13:32:25	27.357	87.867	Earthquake	20			5.9	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/15/1965	23:17:42	-13.309	166.212	Earthquake	35			6	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/16/1965	11:32:37	-56.452	-27.043	Earthquake	95			6	MW	ISCGEMSUP	ISCGEMSUP	ISCGEM	ISCGEM	Automatic
1/17/1965	10:43:17	-24.563	178.487	Earthquake	565			5.8	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/17/1965	20:57:41	-6.807	108.988	Earthquake	227.9			5.9	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic
1/24/1965	0:11:17	-2.608	125.952	Earthquake	20			8.2	MW	ISCGEM861	ISCGEM	ISCGEM	ISCGEM	Automatic

12.2 Visualization:

```python
#import library packages
importpandasasp
importmatplotlib.pyplotasplt
importseabornass
importnumpyasn
importwarnings
warnings.filterwarnings('ignore')
data=p.read_csv('demo1.csv')
df=data.dropna()
fromsklearn.preprocessingimportLabelEncoder
var_mod=['magType','status','locationSource']
le=LabelEncoder()
foriinvar_mod:
df[i]=le.fit_transform(df[i]).astype(str)
```

```python
df.head()
df.columns
df['gap'].hist(figsize=(5,5),color='orange',alpha=1)
plt.xlabel('gap')
plt.ylabel('depth')
plt.title('gap & depth')
df['magError'].hist(figsize=(5,5),color='r',alpha=1)
plt.xlabel('magError')
plt.ylabel('depthError')
plt.title('magError&depthError')
plt.boxplot(df['latitude'])
plt.show()
```

```python
importseabornass
s.boxplot(df['latitude'],color='m')
fig,ax=plt.subplots(figsize=(16,8))
ax.scatter(df['gap'],df['depth'])
ax.set_xlabel('Gap')
ax.set_ylabel('Depth')
plt.show()
#Propagation by variable
defPropByVar(df,variable):
dataframe_pie=df[variable].value_counts()
ax=dataframe_pie.plot.pie(figsize=(10,10),autopct='%1.2f%%',fontsize=12)
ax.set_title(variable+' \n',fontsize=15)
returnn.round(dataframe_pie/df.shape[0]*100,2)
PropByVar(df,'nst')
```

```python
fig,ax=plt.subplots(figsize=(15,10))
s.heatmap(df.corr(),ax=ax,annot=True)
plt.plot(df["gap"],df["depth"],color='g')
plt.xlabel('gap')
plt.ylabel('depth')
plt.title('EarthQuake')
plt.show()
df.columns
#preprocessing, split test and dataset, split response variable
X=df.drop(labels='status',axis=1)
#Response variable
y=df.loc[:,'status']
```

```python
'''We'll use a test size of 14%. We also stratify the split on the
response variable,
which is very important to do because there are so few
fraudulent transactions'''
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.14,random_state=
1,stratify=y)
print("Number of training dataset: ",len(X_train))
print("Number of test dataset: ",len(X_test))
print("Total number of dataset: ",len(X_train)+len(X_test))
```

All affected areas
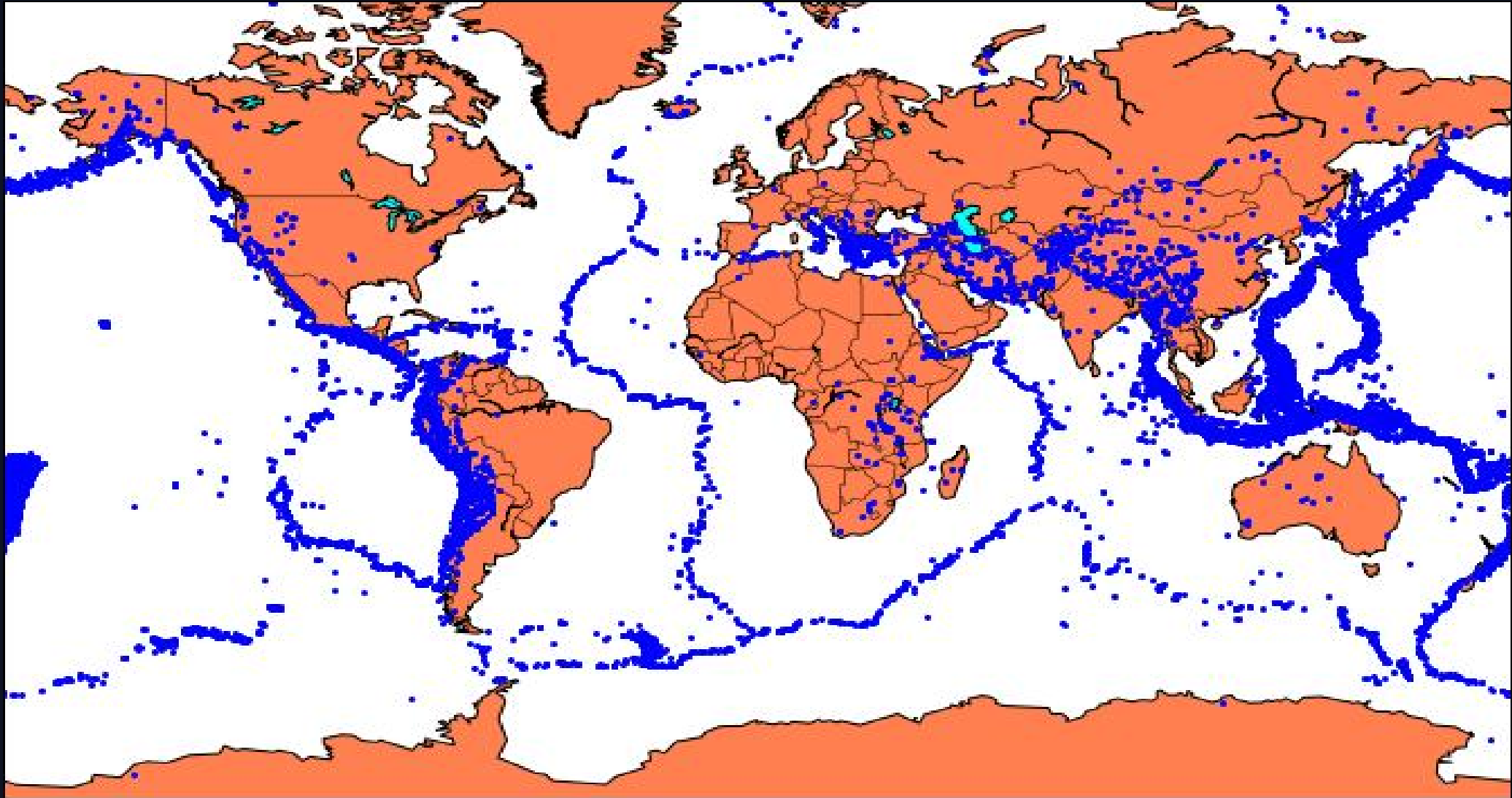
12.3 Logistic Regression

```python
#import library packages
importpandasasp
importmatplotlib.pyplotasplt
importseabornass
importnumpyasn
importwarnings
warnings.filterwarnings('ignore')
data=p.read_csv('demo1.csv')
df=data.dropna()
df.head()
deldf['time']
deldf['updated']
deldf['place']
deldf['net']
deldf['magSource']
```

```python
deldf['type']
deldf['id']
df.columns
fromsklearn.preprocessingimportLabelEncoder
var_mod=['magType','status','locationSource']
le=LabelEncoder()
foriinvar_mod:
df[i]=le.fit_transform(df[i]).astype(str)
#preprocessing, split test and dataset, split response variable
X=df.drop(labels='status',axis=1)
#Response variable
y=df.loc[:,'status']
```

```python
def graph():
    import matplotlib.pyplot as plt
    data=[LR]
    alg="Logistic Regression"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))
    plt.title("Accuracy comparison of Earth Quake",fontsize=15)
    plt.legend(b,data,fontsize=9)
graph()
TP=cm1[0][0]
FP=cm1[1][0]
FN=cm1[1][1]
TN=cm1[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
```

```python
'''We'll use a test size of 20%. We also stratify the split on the
response variable,
which is very important to do because there are so few
fraudulent transactions'''
fromsklearn.model_selectionimporttrain_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=
1,stratify=y)
print("Number of training dataset: ",len(X_train))
print("Number of test dataset: ",len(X_test))
print("Total number of dataset: ",len(X_train)+len(X_test))
#According to the cross-validated MCC scores, the random
forest is the bestperforming model, so now let's evaluate its
performance on the test set.
```

```python
fromsklearn.metricsimportconfusion_matrix,classification_report,matthews_co
rrcoef,cohen_kappa_score,accuracy_score,average_precision_score,roc_
auc_sc
ore
fromsklearn.metricsimportaccuracy_score,confusion_matrix
fromsklearn.linear_modelimportLogisticRegression
fromsklearn.model_selectionimportcross_val_score
logR=LogisticRegression()
logR.fit(X_train,y_train)
predictLR=logR.predict(X_test)
print("")
print('Classification report of Logistic Regression Results:')
print("")
print(classification_report(y_test,predictLR))
print("")
cm1=confusion_matrix(y_test,predictLR)
```

```python
print('Confusion Matrix result of Logistic Regression is:\n',cm1)
print("")
sensitivity1=cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ',sensitivity1)
print("")
specificity1=cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ',specificity1)
print("")
accuracy=cross_val_score(logR,X,y,scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean()*100)
LR=accuracy.mean()*100
```

```python
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR=TP/(TP+FN)
TNR=TN/(TN+FP)
FPR=FP/(FP+TN)
FNR=FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV=TP/(TP+FP)
NPV=TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
```

```python
defplot_confusion_matrix(cm1,title='Confusion
matrixLogistic_Regression',cmap=plt.cm.Blues):
target_names=['Predict','Actual']
plt.imshow(cm1,interpolation='nearest',cmap=cmap)
plt.title(title)
plt.colorbar()
tick_marks=n.arange(len(target_names))
plt.xticks(tick_marks,target_names,rotation=45)
plt.yticks(tick_marks,target_names)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
cm1=confusion_matrix(y_test,predictLR)
print('Confusion matrix-Logistic_Regression:')
print(cm1)
plot_confusion_matrix(cm1)
```

12.4 Random Forest Algorithm:

```
#import library packages
importpandasasp
importmatplotlib.pyplotasplt
importseabornass
importnumpyasn
importwarnings
warnings.filterwarnings('ignore')
data=p.read_csv('demo1.csv')
df=data.dropna()
df.tail(10)
deldf['time']
deldf['updated']
deldf['place']
```

```python
deldf['net']
deldf['magSource']
deldf['type']
deldf['id']
df.status.unique()
fromsklearn.preprocessingimportLabelEncoder
var_mod=['magType','status','locationSource']
le=LabelEncoder()
foriinvar_mod:
df[i]=le.fit_transform(df[i]).astype(int)
df.status.unique()
#preprocessing, split test and dataset, split response variable
X=df.drop(labels='status',axis=1)
#Response variable
y=df.loc[:,'status']
```

```python
'''We'll use a test size of 20%. We also stratify the split on the
response variable,
which is very important to do because there are so few
fraudulent transactions'''
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,
random_state=
1,stratify=y)
print("Number of training dataset: ",len(X_train))
print("Number of test dataset: ",len(X_test))
print("Total number of dataset: ",len(X_train)+len(X_test))
#According to the cross-validated MCC scores, the random
forest is the bestperforming model, so now let's evaluate its
performance on the test set.
```

```python
from sklearn.metrics import confusion_matrix,classification_report,matthews_corrcoef,cohen_kappa_score,accuracy_score,average_precision_score,roc_auc_score
from sklearn.metrics import accuracy_score,confusion_matrix
101
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
predictRF=rfc.predict(X_test)
print("")
print('Classification report of Random Forest Results:')
print("")
print(classification_report(y_test,predictRF))
```

```python
print("")
cm1=confusion_matrix(y_test,predictRF)
print('Confusion Matrix result of Random Forest Classifier is:\n',cm1)
print("")
sensitivity1=cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ',sensitivity1)
print("")
specificity1=cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ',specificity1)
print("")
accuracy=cross_val_score(rfc,X,y,scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
```

```python
print("Accuracy result of Random Forest Classifier
is:",accuracy.mean()*100)
RF=accuracy.mean()*100
defgraph():
importmatplotlib.pyplotasplt
data=[RF]
alg="Random Forest Classifier"
plt.figure(figsize=(5,5))
b=plt.bar(alg,data,color=("m"))
plt.title("Accuracy comparison of Earth Quake",fontsize=15)
plt.legend(b,data,fontsize=9)
graph()
```

```python
TP=cm1[0][0]
FP=cm1[1][0]
FN=cm1[1][1]
TN=cm1[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR=TP/(TP+FN)
TNR=TN/(TN+FP)
FPR=FP/(FP+TN)
FNR=FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
```

```python
print("False Negative Rate :",FNR)
print("")
PPV=TP/(TP+FP)
NPV=TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
defplot_confusion_matrix(cm1,title='Confusion
matrixRandomForestClassifier',cmap=plt.cm.Blues):
target_names=['Predict','Actual']
plt.imshow(cm1,interpolation='nearest',cmap=cmap)
plt.title(title)
plt.colorbar()
tick_marks=n.arange(len(target_names))
plt.xticks(tick_marks,target_names,rotation=45)
plt.yticks(tick_marks,target_names)
plt.tight_layout()
```

```python
plt.ylabel('True label')
plt.xlabel('Predicted label')
cm1=confusion_matrix(y_test,predictRF)
print('Confusion matrix-RandomForestClassifier:')
print(cm1)
103
plot_confusion_matrix(cm1)
```

Creating pkl File

```python
importjoblib
joblib.dump(rfc,"rf.pkl")
```

Import the necessary libraries required for buidling the model and data analysis of the earthquakes.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import os
print(os.listdir("../input"))
```

OUTPUT:

```
['database.csv']
```

Figure out the main features from earthquake data and create a object of that features, namely, Date, Time, Latitude, Longitude, Depth, Magnitude.

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

	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

Here, the data is random we need to scale according to inputs to the model. In this, we convert given Date and Time to Unix time which is in seconds and a numeral. This can be easily used as input for the network we built.

```python
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)
```

```python
data['Timestamp'] = timeStamp.values
final_data = data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

OUTPUT:

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-1.57631e+08
1	1.863	127.352	80.0	5.8	-1.57466e+08
2	-20.579	-173.972	20.0	6.2	-1.57356e+08
3	-59.076	-23.557	15.0	5.8	-1.57094e+08
4	11.938	126.427	15.0	5.8	-1.57026e+08

Here, we used the RandomForestRegressor model to predict the outputs, we see the strange prediction from this with score above 80% which can be assumed to be best fit but not due to its predicted values.

```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
reg.predict(X_test)
```
/opt/conda/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning: numpy.core.umath_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d

```
array([[ 5.96, 50.97],
       [ 5.88, 37.8 ],
       [ 5.97, 37.6 ],

       ...,
       [ 6.42, 19.9 ],
       [ 5.73, 591.55],
       [ 5.68, 33.61]])
```

Neural Network model
 So, Now, we build the neural network to fit the data for training set. Neural Network consists of three Dense layer with each 16, 16, 2 nodes and relu, relu and softmax as activation function.

```python
from keras.models import Sequential
from keras.layers import Dense
def create_model(neurons, activation, optimizer, loss):
    model = Sequential()
    model.add(Dense(neurons, activation=activation, input_shape=(3,)))
    model.add(Dense(neurons, activation=activation))
    model.add(Dense(2, activation='softmax'))
    model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

```python
from keras.wrappers.scikit_learn import KerasClassifier
model = KerasClassifier(build_fn=create_model, verbose=0)
# neurons = [16, 64, 128, 256]
neurons = [16]
# batch_size = [10, 20, 50, 100]
batch_size = [10]
epochs = [10]
# activation = ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear',
'exponential']
activation = ['sigmoid', 'relu']
# optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam',
'Adamax', 'Nadam']
optimizer = ['SGD', 'Adadelta']
loss = ['squared_hinge']
imizer=optimizer, loss=loss)
```

```python
param_grid = dict(neurons=neurons, batch_size=batch_size,
epochs=epochs, activation=activation, optimizer=optimizer,
loss=loss)
```
Here, we find the best fit of the above model and get the mean test score and standard deviation of the best fit model.

```python
grid = GridSearchCV(estimator=model,
param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X_train, y_train)
print("Best: %f using %s" % (grid_result.best_score_,
grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

Best: 0.957655 using {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.333316 (0.471398) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.000000 (0.000000) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}
0.957655 (0.029957) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.645111 (0.456960) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}

The best fit parameters are used for same model to compute the score with training data and testing data.

```
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(3,)))
model.add(Dense(16, activation='relu'))
model.add(Dense(2, activation='softmax'))

model.compile(optimizer='SGD', loss='squared_hinge',
metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=10, epochs=20,
verbose=1, validation_data=(X_test, y_test))
```

```
Train on 18727 samples, validate on 4682 samples
Epoch 1/20
18727/18727 [==============================] - 6s
330us/step - loss: 0.5038 - acc: 0.9182 - val_loss: 0.5038 -
val_acc: 0.9242
Epoch 2/20
18727/18727 [==============================] - 6s
320us/step - loss: 0.5038 - acc: 0.9182 - val_loss: 0.5038 -
val_acc: 0.9242
Epoch 3/20
18727/18727 [==============================] - 6s
320us/step - loss: 0.5038 - acc: 0.9182 - val_loss: 0.5038 -
val_acc: 0.9242
Epoch 4/20
18727/18727 [==============================] - 6s
322us/step - loss: 0.5038 - acc: 0.9182 - val_loss: 0.5038 -
```

```
Epoch 5/20
18727/18727 [==============================] - 6s
321us/step - loss: 0.5038 - acc: 0.9182 - val_loss: 0.5038 -
val_acc: 0.9242
Epoch 6/20
18727/18727 [==============================] - 6s
323us/step - loss: 0.5038 - acc: 0.9182 - val_loss: 0.5038 -
val_acc: 0.9242
Epoch 7/20
18727/18727 [==============================] - 6s
322us/step - loss: 0.5038 - acc: 0.9182 - val_loss: 0.5038 -
val_acc: 0.9242
Epoch 8/20
18727/18727 [==============================] - 6s
321us/step - loss: 0.5038 - acc: 0.9182 - val_loss: 0.5038 -
val_acc: 0.9242
```

13 ADVANTAGE

Early Warning and Risk Reduction:
One of the most significant advantages is the ability to provide early warnings to at-risk communities and critical infrastructure. These warnings can give people valuable seconds or minutes to take protective actions, such as seeking shelter or shutting down critical systems.

Reduced Loss of Life:
Effective earthquake prediction can lead to improved preparedness and reduce the loss of human lives during seismic events. Early warnings can provide the opportunity for people to evacuate or take safety measures.

Infrastructure Resilience:
Early warning systems can trigger automatic shutdown procedures in critical infrastructure, such as power plants, transportation systems, and water supply facilities. This minimizes the risk of damage and operational disruptions.

Reduced Economic Impact:
Earthquakes can cause extensive damage to buildings, roads, and utilities. Early warnings and preparedness measures can minimize damage and reduce the economic impact of seismic events, including repair and recovery costs.

Improved Emergency Response:
Earthquake prediction can enhance the ability of emergency response teams to prepare for and respond to disasters. It allows them to allocate resources more effectively and coordinate response efforts.

Community Preparedness:
Knowing the likelihood of an impending earthquake can motivate communities and individuals to prepare by creating emergency plans, securing heavy objects, and stockpiling essential supplies.

Long-Term Risk Mitigation:
Accurate earthquake prediction can influence urban planning and building codes to ensure that construction is resilient to seismic hazards. It also helps identify high-risk areas for land-use planning and zoning regulations.

Psychological Benefits:
Having access to earthquake prediction and early warning systems can reduce the psychological stress and anxiety associated with living in seismically active regions. It provides a sense of control and preparedness.

International Collaboration:
Earthquake prediction research fosters international collaboration and data sharing, enabling countries and regions to work together to monitor and respond to seismic threats.

14

DIS
ADVANTAGE

Ethical and Psychological Implications:
False alarms can have ethical and psychological implications, causing stress, anxiety, and emotional distress for individuals and communities. Balancing the need for public safety with the potential psychological harm from false alarms is a significant challenge.
Inadequate Data and Data Gaps:
Incomplete or unreliable seismic and geospatial data, especially in regions with limited monitoring infrastructure, can hinder the development of accurate prediction models.
Variable Earthquake Patterns:
Earthquakes are highly variable in terms of their behavior and characteristics, making it difficult to develop one-size-fits-all prediction models. Each seismic region may require unique approaches.

Public Complacency:
The mere presence of an earthquake prediction system might lead individuals and communities to rely solely on predictions rather than maintaining general earthquake preparedness measures.

Rapid-Onset Events:
Some earthquakes, like "blind thrust" earthquakes, occur suddenly without recognizable precursors, making prediction extremely challenging.

Legal and Liability Issues:
Legal and liability issues can arise when making predictions that affect public safety, particularly if predictions are inaccurate and result in economic or personal losses.

Early Warning and Risk Reduction:
One of the most significant advantages is the ability to provide early warnings to at-risk communities and critical infrastructure. These warnings can give people valuable seconds or minutes to take protective actions, such as seeking shelter or shutting down critical systems.

Reduced Loss of Life:
Effective earthquake prediction can lead to improved preparedness and reduce the loss of human lives during seismic events. Early warnings can provide the opportunity for people to evacuate or take safety measures.

Infrastructure Resilience:
Early warning systems can trigger automatic shutdown procedures in critical infrastructure, such as power plants, transportation systems, and water supply facilities. This minimizes the risk of damage and operational disruptions.

Reduced Economic Impact:
Earthquakes can cause extensive damage to buildings, roads, and utilities. Early warnings and preparedness measures can minimize damage and reduce the economic impact of seismic events, including repair and recovery costs.

Improved Emergency Response:
Earthquake prediction can enhance the ability of emergency response teams to prepare for and respond to disasters. It allows them to allocate resources more effectively and coordinate response efforts.

Community Preparedness:
Knowing the likelihood of an impending earthquake can motivate communities and individuals to prepare by creating emergency plans, securing heavy objects, and stockpiling essential supplies.

Long-Term Risk Mitigation:
Accurate earthquake prediction can influence urban planning and building codes to ensure that construction is resilient to seismic hazards. It also helps identify high-risk areas for land-use planning and zoning regulations.

Psychological Benefits:
Having access to earthquake prediction and early warning systems can reduce the psychological stress and anxiety associated with living in seismically active regions. It provides a sense of control and preparedness.

International Collaboration:
Earthquake prediction research fosters international collaboration and data sharing, enabling countries and regions to work together to monitor and respond to seismic threats.

16 CONCLUSION

In conclusion, earthquake prediction is a complex and challenging field that plays a crucial role in mitigating the devastating impacts of seismic events on human lives, infrastructure, and communities. While precise and accurate prediction of specific earthquakes remains elusive, the pursuit of earthquake prediction offers a range of valuable benefits and opportunities:

Early Warning and Preparedness: Earthquake prediction provides the potential for early warnings, giving at-risk communities and critical infrastructure valuable seconds or minutes to take protective actions, reduce loss of life, and minimize damage.

Risk Assessment and Reduction: Through the assessment of seismic risk, earthquake prediction allows for the identification of high-risk areas, influencing land-use planning, building codes, and disaster preparedness measures to enhance community safety.

Infrastructure Resilience: Early warning systems trigger automatic shutdown procedures in critical infrastructure, ensuring the resilience of power plants, transportation systems, and water supply facilities.

Economic Protection: By reducing the economic impact of seismic events, earthquake prediction minimizes repair and recovery costs, protecting businesses and local economies.

Scientific Insights: The pursuit of earthquake prediction advances our understanding of the geophysical processes underlying seismic events, contributing to improved earthquake modeling and risk assessment.

Psychological Benefits: Access to early warning systems and preparedness initiatives can alleviate the psychological stress and anxiety associated with living in seismically active regions, providing peace of mind to residents.

International Collaboration: Collaboration among countries, research institutions, and international organizations fosters data sharing and resource pooling, enabling global efforts to monitor and respond to seismic threats.

While the field of earthquake prediction faces limitations, such as unpredictable earthquake patterns and false alarms, its potential benefits underscore the importance of ongoing research, development, and international cooperation. By focusing on assessing and mitigating seismic risk, providing early warnings, and enhancing preparedness measures, the goal is to reduce the impact of seismic events and safeguard human lives, communities, and infrastructure.

In summary, while earthquake prediction faces limitations and challenges, its potential benefits in terms of public safety, disaster resilience, and economic protection make it a vital area of study and application. The focus is on reducing risk, providing early warnings, and improving disaster preparedness to mitigate the impact of seismic events on communities and infrastructure.

THANK YOU