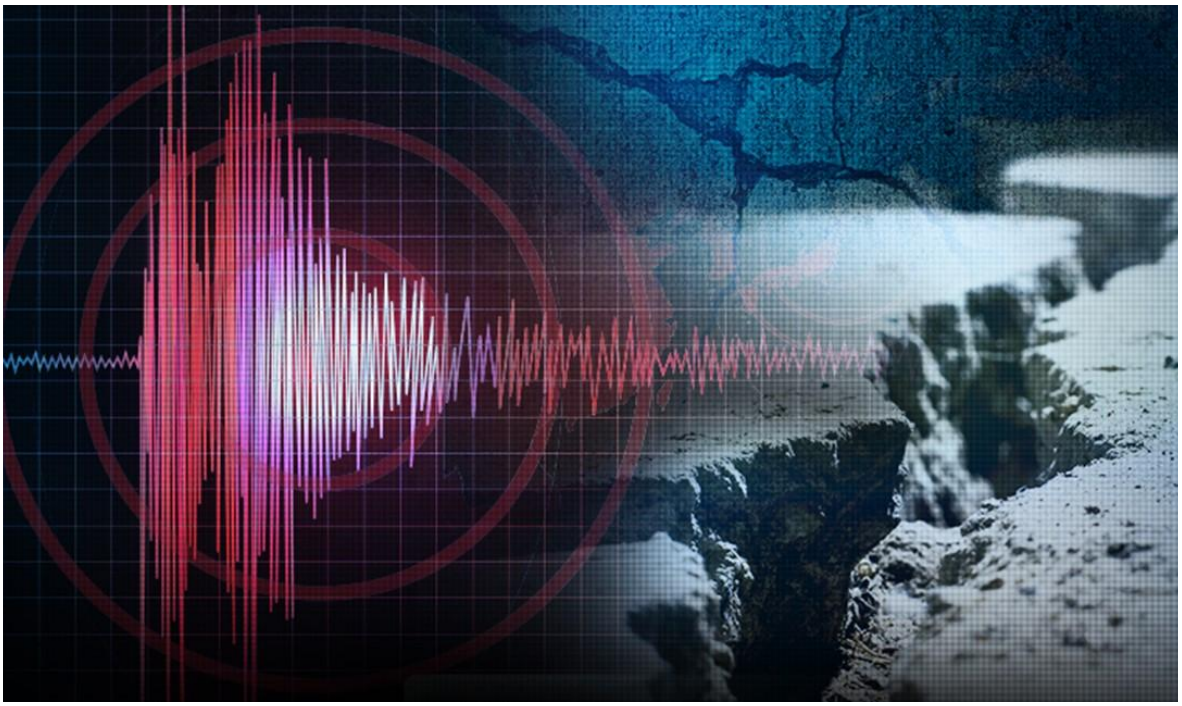
EARTHQUAKE PREDICTION USING MACHINE LEARNING

PHASE-4

Project Title : Earthquake Prediction

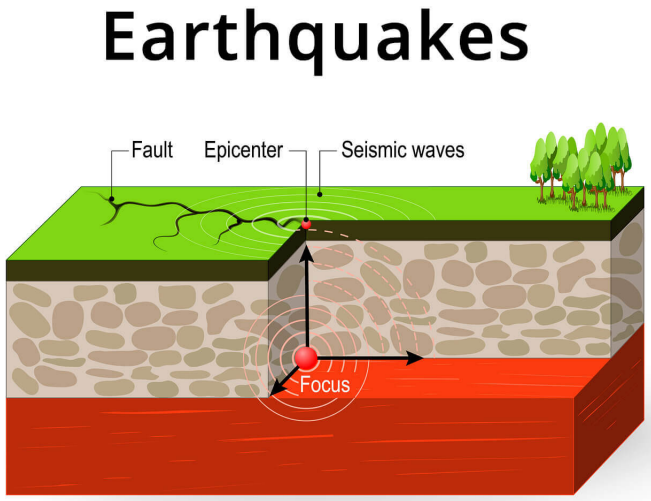
Phase 4 : Development Part 2

Topic : Continue building the earthquake prediction model by Visualizing the data on a world map and splitting it into training and testing sets.



INTRODUCTION :

Predicting earthquakes is no small feat due to their complex and ever-changing nature. But artificial intelligence, specifically a cutting-edge technique Quantum Neural Networks (QNN), could help to foresee these powerful events. Just like how QNN has made its mark in computer games, data management, and understanding social networks, it has found a new mission: predicting earthquakes.



By blending the smarts of classical neural networks with the mysteries of quantum mechanics, QNN offers a fresh way to decode the information hidden in seismic data. This introduction will show you how artificial intelligence, especially Quantum Neural Networks, might hold the key to revolutionizing how we predict earthquakes.

Think Quantum Neural Network like a unique recipe that combines quantum mechanics and clever computer algorithms. This method has worked

wonders in understanding complex data patterns, making it valuable for deciphering the signals that earthquakes send us over time. Just as QNN helps computers learn from examples and manage heaps of information, it can also be our guiding light in understanding the patterns and behaviors of earthquakes. In simpler terms, we're exploring how this new generation AI method could help us predict earthquakes better. The goal? To build smarter systems that can give us a heads-up about potential earthquakes, reducing their impact and keeping us safer.

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

GIVEN DATASET :

date	depth	mag	place	latitude	longitude	depth_avg_22	depth_avg_15	depth_avg_7	mag_avg_22	mag_avg_15	mag_avg_7	mag_outcome
2020-07-14	6.70	1.58	Oklahoma	36.171483	-97.718347	6.717727	6.560000	7.100000	1.352273	1.271333	1.357143	1.338571
2020-07-14	7.55	2.07	Oklahoma	36.171483	-97.718347	6.730000	6.682667	7.132857	1.372727	1.334667	1.527143	1.535714
2020-07-14	7.39	1.89	Oklahoma	36.171483	-97.718347	6.747727	6.708667	6.940000	1.396818	1.377333	1.570000	1.335714
2020-07-15	7.75	1.48	Oklahoma	36.171483	-97.718347	6.834545	6.764000	6.848571	1.383182	1.388667	1.581429	1.251429
2020-07-15	7.81	1.50	Oklahoma	36.171483	-97.718347	6.841364	6.854667	6.964286	1.404545	1.385333	1.602857	1.291429

Overview of the process :

The following is an overview of the process of building a earthquake prediction model by feature selection, model training, and
evaluation:

1. Prepare the data :

This includes cleaning the data, removing outliers, and handling missing values.

2. Perform feature selection :

This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

3. Train the model :

There are many different machine learning algorithms that can be used for earthquake prediction. Some popular choices include linear regression, random forests, and gradient boosting machines.

4. Evaluate the model :

This can be done by calculating the means quared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.

5. Deploy the model :

Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the earthquake.

PROCEDURE:

FEATURE SELECTION :

1. Identify the target variable. This is the variable that you want to predict, such as house price.

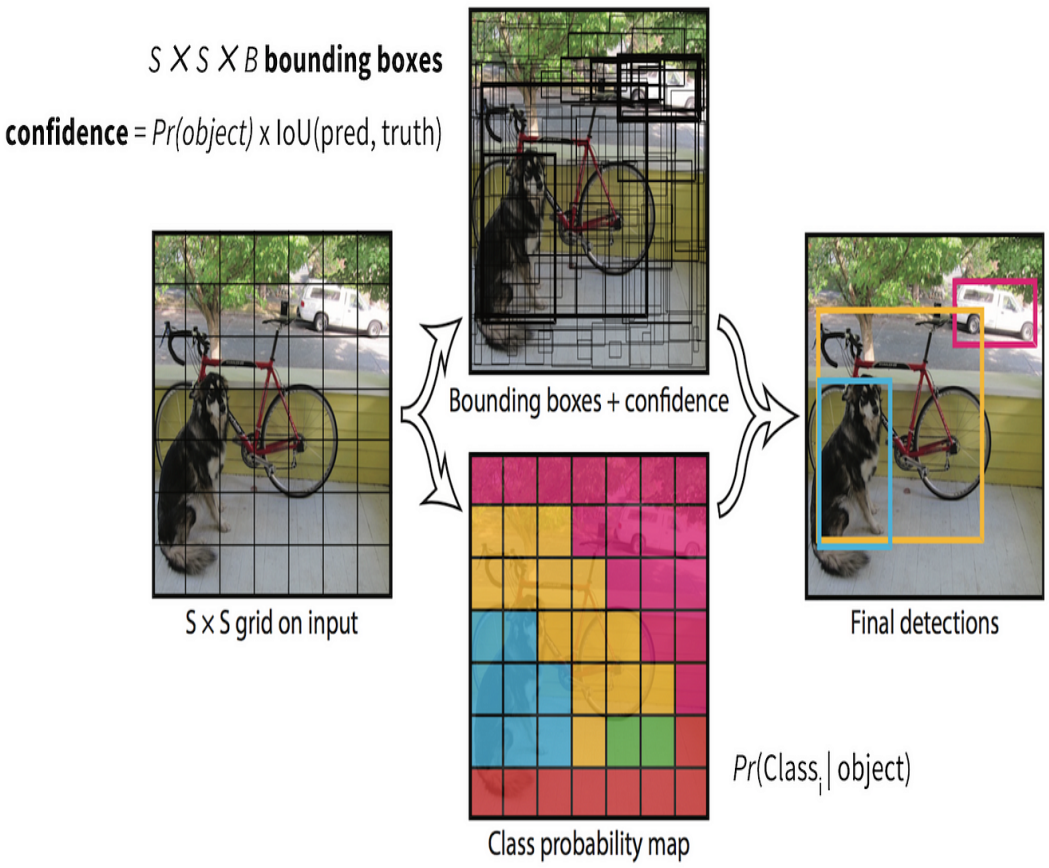
2. Explore the data. This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.

3. Remove redundant features. If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.

4. Remove irrelevant features. If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

OBJECT DETECTION :

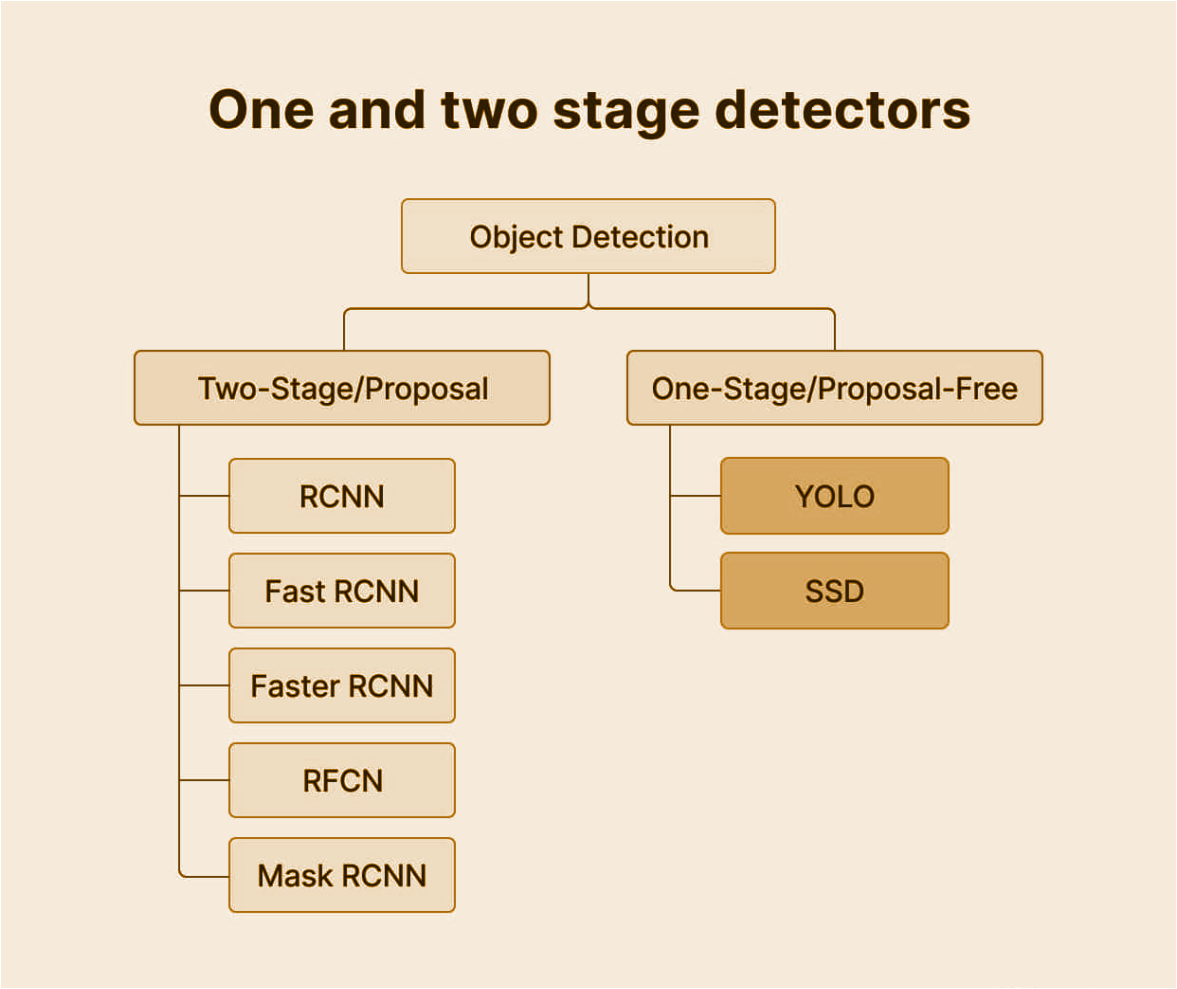
Object detection is a computer vision task that involves identifying and locating objects in images or videos. It is an important part of many applications, such as surveillance, self-driving cars, or robotics. Object detection algorithms can be divided into two main categories: single-shot detectors and two-stage detectors.



One of the earliest successful attempts to address the object detection problem using deep learning was the R-CNN (Regions with CNN features) model, developed by Ross Girshick and his team at Microsoft Research in 2014. This model used a combination of region proposal algorithms and convolutional neural networks (CNNs) to detect and localize objects in images.

Object detection algorithms are broadly classified into two categories based on how many times the same input image is passed through a network.

Object detection models performance evaluation metrics :



To determine and compare the predictive performance of different object detection models, we need standard quantitative metrics.

The two most common evaluation metrics are Intersection over Union (IoU) and the Average Precision (AP) metrics.

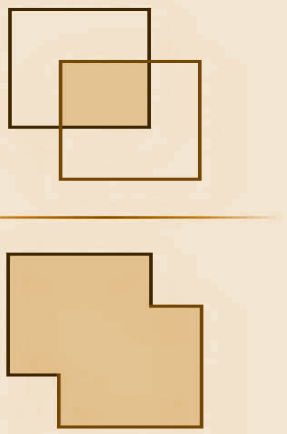
Intersection over Union (IoU) :

Intersection over Union (IoU) is a popular metric to measure localization accuracy and calculate localization errors in object detection models.

To calculate the IoU between the predicted and the ground truth bounding boxes, we first take the intersecting area between the two corresponding bounding boxes for the same object. Following this, we calculate the total area covered by the two bounding boxes— also known as the "Union" and the area of overlap between them called the "Intersection."

The intersection divided by the Union gives us the ratio of the overlap to the total area, providing a good estimate of how close the prediction bounding box is to the original bounding box.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} =$$



Average Precision (AP) or (mean AP) :

Average Precision (AP) is a metric used to evaluate object detection models such as Fast R-CNN, YOLO, Mask R-CNN, etc. The mean of average precision values are calculated over recall values from 0 to 1.

Using different thresholds, a precision-recall curve is created. From that curve, the average precision (AP) is measured. For an object detection model, the threshold is the intersection over union (IoU) that scores the detected objects. Once the AP is measured for each class in the dataset, the mAP is calculated.

Average precision ranges from the frequency of positive examples (0.5 for balanced data) to 1.0 (perfect model). If the model makes "balanced" predictions that don't tend towards being wrong or being right, then we have a random model with 0.5 AUROC and 0.5 average precision (for frequency of positives = 0.5).

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k)$$

$$Recalls(n) = 0, Precisions(n) = 1$$

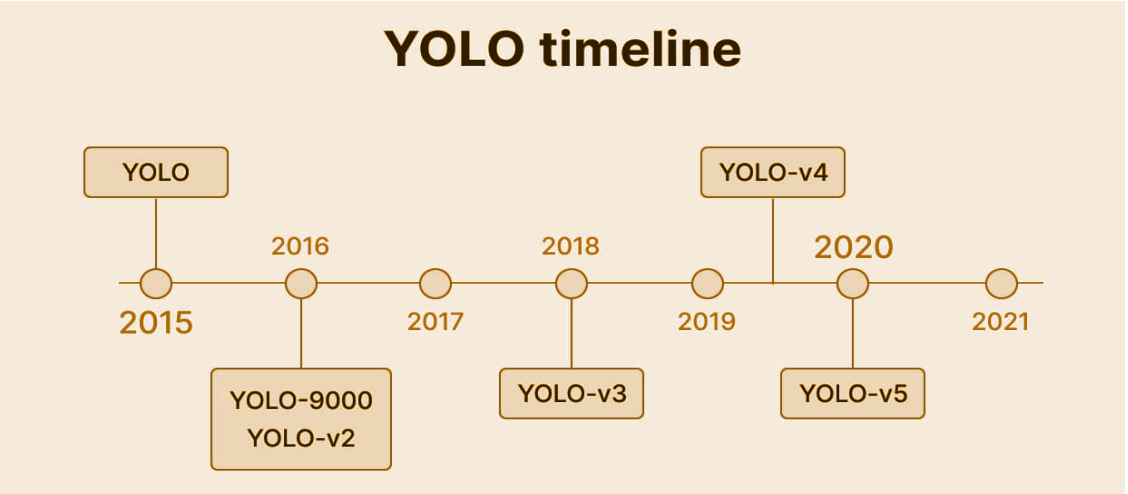
$$n = Number\ of\ thresholds.$$

What is YOLO? :

You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection.

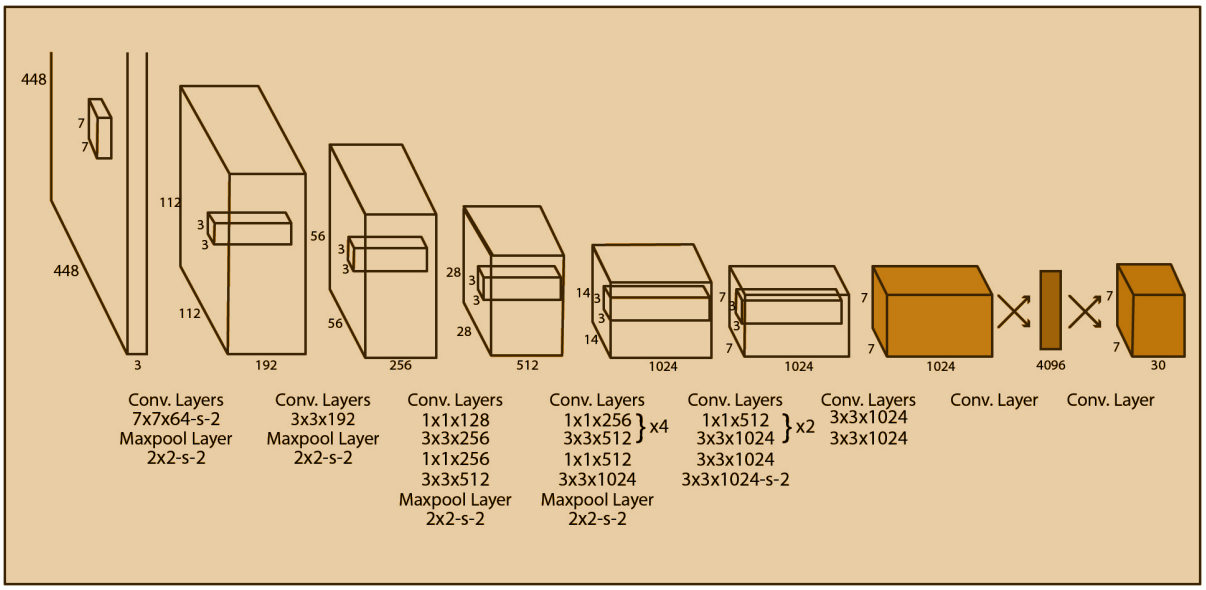
Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

Methods that use Region Proposal Networks perform multiple iterations for the same image, while YOLO gets away with a single iteration. Several new versions of the same model have been proposed since the initial release of YOLO in 2015, each building on and improving its predecessor. Here's a timeline showcasing YOLO's development in recent years.



How does YOLO work? YOLO Architecture :

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.



The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialisation between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is non-maximum suppression (NMS). NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

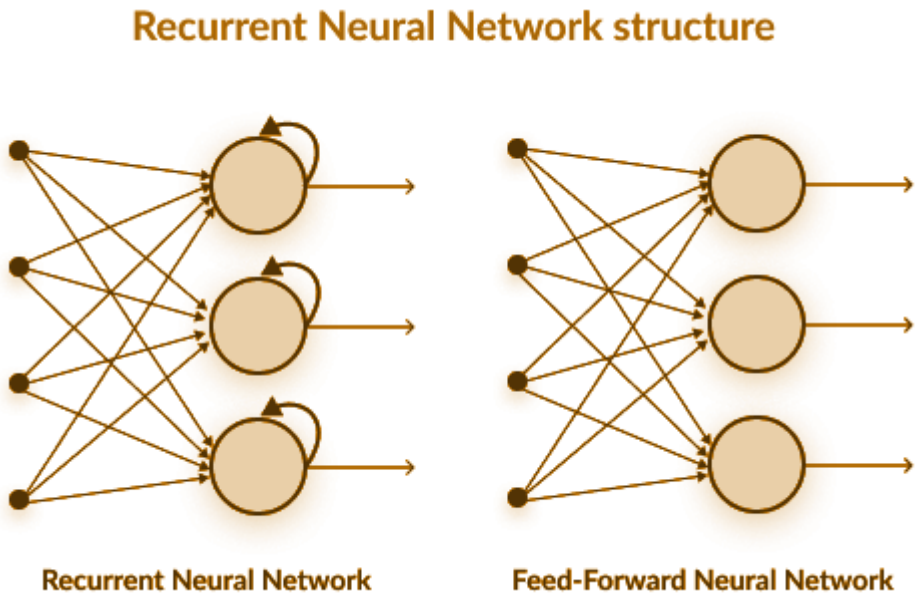
Now, let us look into the improvements that the later versions of YOLO have brought to the parent model.

RECURRENT NEURAL NETWORKS :

Recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate.

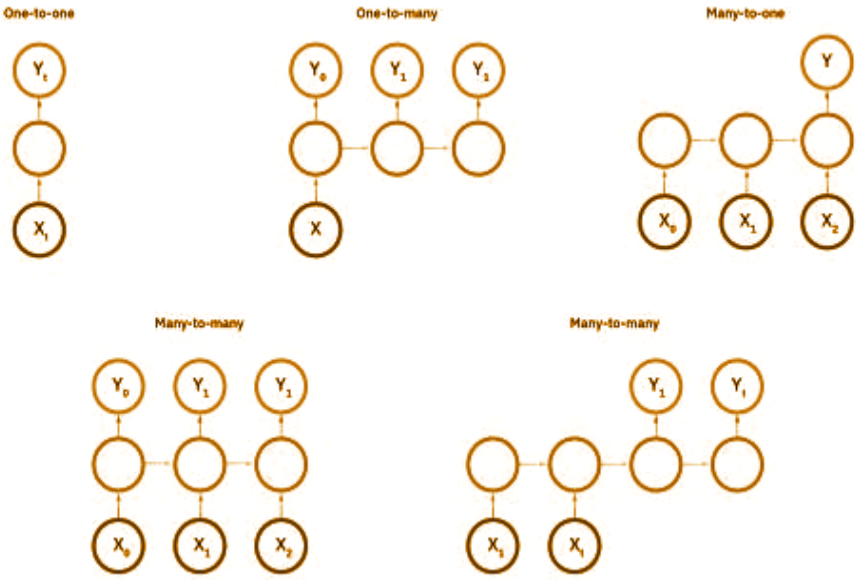
Like feedforward and convolutional neural networks (CNNs), recurrent neural networks utilize training data to learn.While future events would

also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.



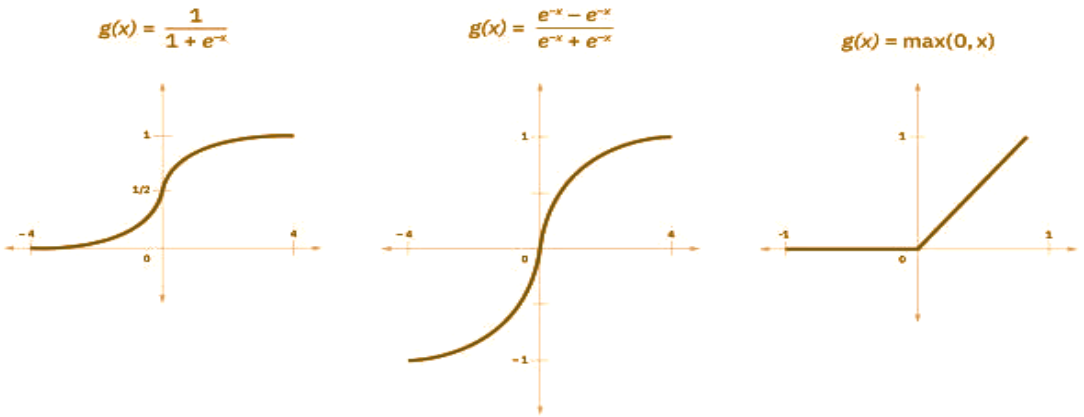
Types of recurrent neural networks :

Feedforward networks map one input to one output, and while we've visualized recurrent neural networks in this way in the above diagrams, they do not actually have this constraint. Instead, their inputs and outputs can vary in length, and different types of RNNs are used for different use cases, such as music generation, sentiment classification, and machine translation.



Common activation functions :

As discussed in the Learn article on Neural Networks, an activation function determines whether a neuron should be activated. The nonlinear functions typically convert the output of a given neuron to a value between 0 and 1 or -1 and 1.



NATURAL LANGUAGE PROCESSING :

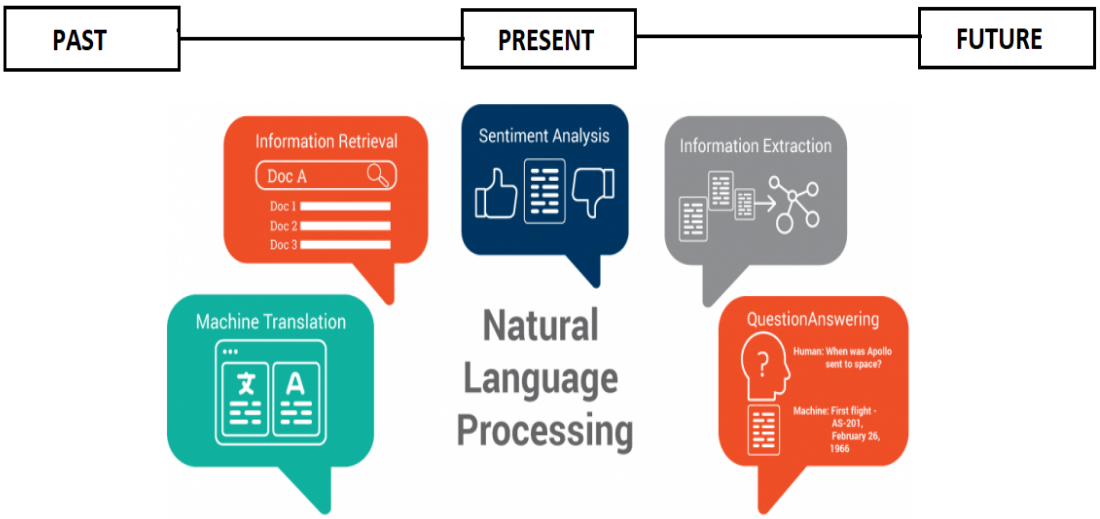
Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

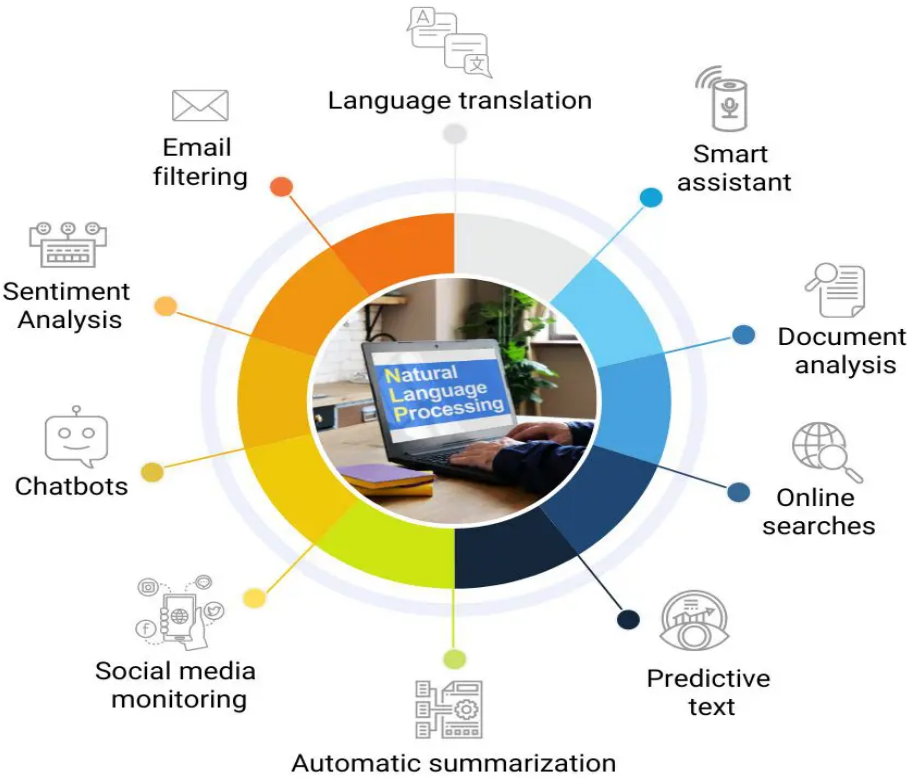
NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There’s a good chance you’ve interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation

software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

Evoloution of NLP



APPLICATIONS OF NLP :



Application for earthquake forecasting :

1. **Disaster preparedness :**

Earthquake forecasts can help communities and governments better prepare through emergency planning, retrofitting infrastructure, stockpiling supplies, etc.

2. **Critical system protection :**

Early warning allows protective actions for critical infrastructure like trains, pipelines, power plants to avoid damage and disruption.

3. **Casualty reduction :**

Warnings to the public before shaking starts enables self-protective actions like moving to safe locations. This saves lives and reduces injuries.

4. **Secondary hazard mitigation :**

With early alerts, steps can be taken to reduce secondary hazards like fires, landslides etc. that follow major earthquakes.

5. **Social stability :**

Accurate warnings and education on risks prevents panic and helps maintain law and order after large quakes.

6. **Financial stability :**

Earthquake forecasts enable better catastrophe risk modeling for insurance and reinsurance industries. This allows pricing models and capital allocation to match risks.

7. **Future resilience :**

Data and insights from earthquake prediction research helps improve building codes, zoning policies, and long-term resilience investments in communities.

Checking the missing values :

```
print("Missing Values by Column")
print("-"*30)
print(df.isna().sum())
print("-"*30)
print("TOTAL MISSING VALUES:",df.isna().sum().sum())
```

OUTPUT :

Missing Values by Column

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

ID 0

Source 0

Location Source 0

Magnitude Source 0

Status 0

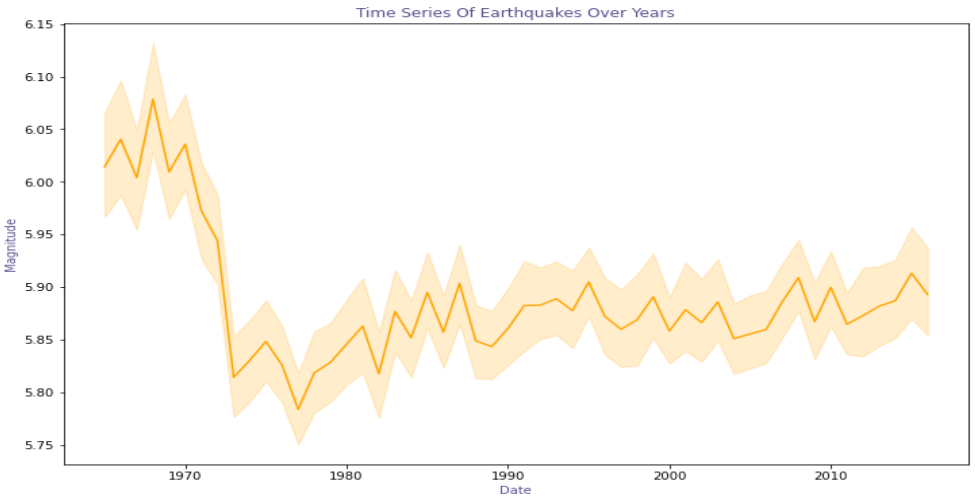
dtype: int64

TOTAL MISSING VALUES: 145439

VISUALISATION :

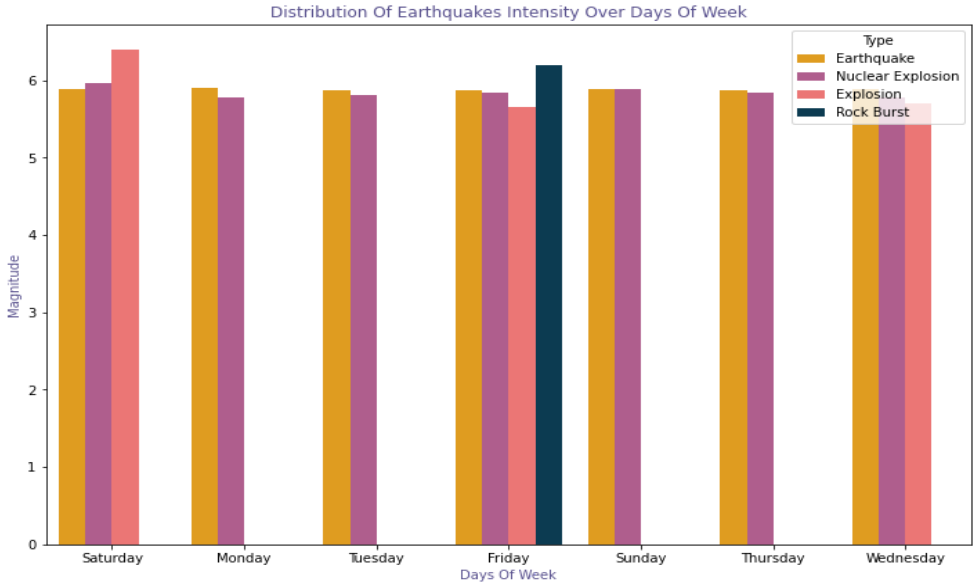
```
plt.figure(figsize=(12,8))
Time_series=sns.lineplot(x=df['Date'].dt.year,
y="Magnitude",df=data, color="#ffa600")
Time_series.set_title("Time Series Of Earthquakes Over
Years", color="#58508d")
```

```
Time_series.set_ylabel("Magnitude", color="#58508d")
Time_series.set_xlabel("Date", color="#58508d")
```



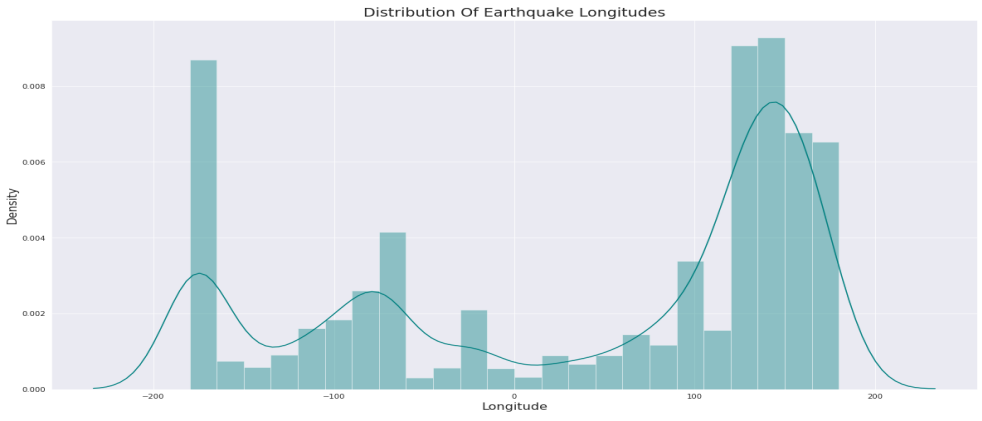
Bar plot :

```
plt.figure(figsize=(12,8))
Days_of_week=sns.barplot(x=df['Days'],y="Magnitude",data
=data, ci =None, hue ="Type",palette = colours)
Days_of_week.set_title("Distribution Of Earthquakes
Intensity Over Days Of Week", color="#58508d")
Days_of_week.set_ylabel("Magnitude", color="#58508d")
Days_of_week.set_xlabel("Days Of Week", color="#58508d")
```



Distplot :

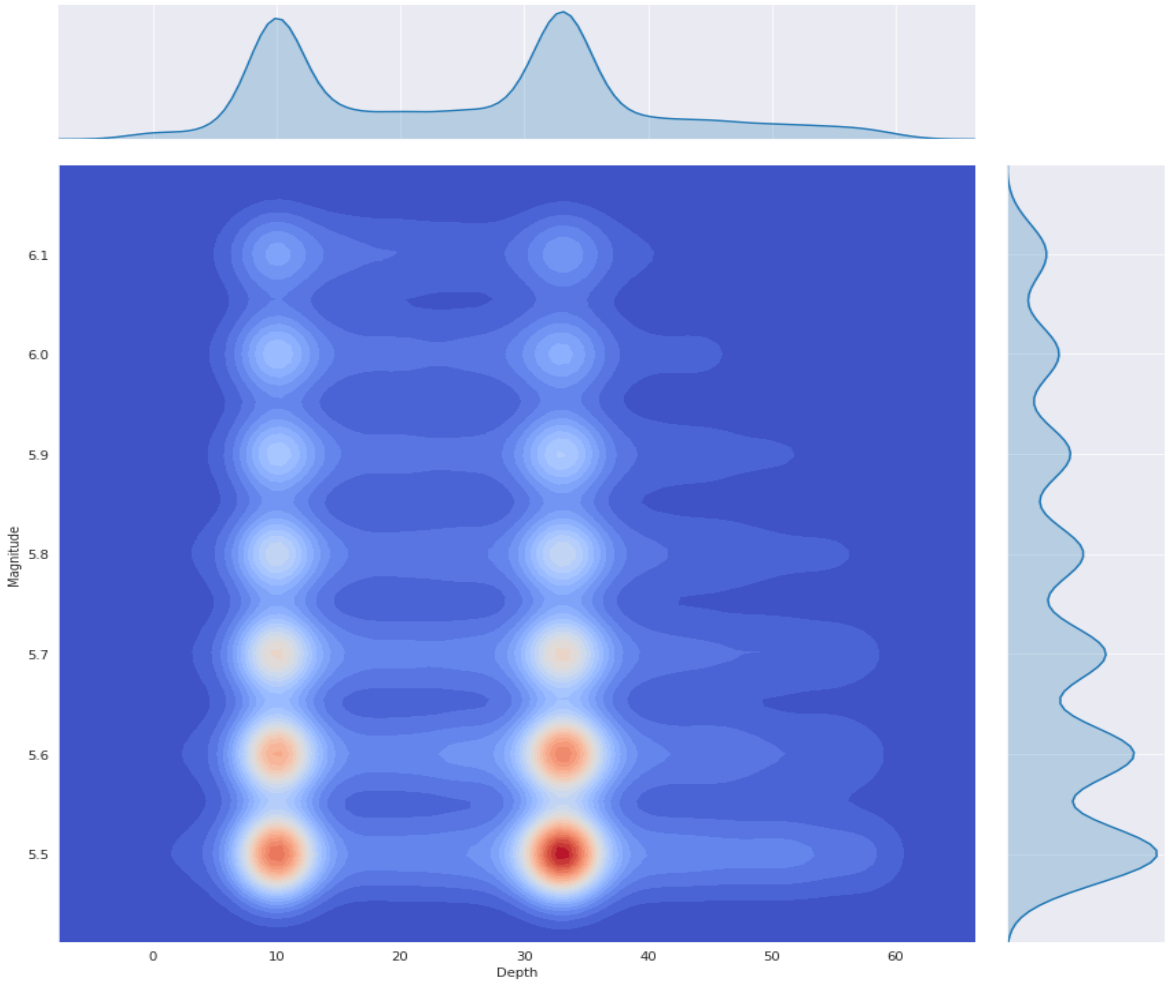
```
plt.figure(figsize=(20,11))
ax =
sns.distplot(e_df['Longitude'],label='Longitude',color='
teal')
ax.set_title('Distribution Of Earthquake
Longitudes',fontsize=19)
ax.set_ylabel('Density',fontsize=16)
ax.set_xlabel('Longitude',fontsize=16)
plt.show()
```



Jointplot :

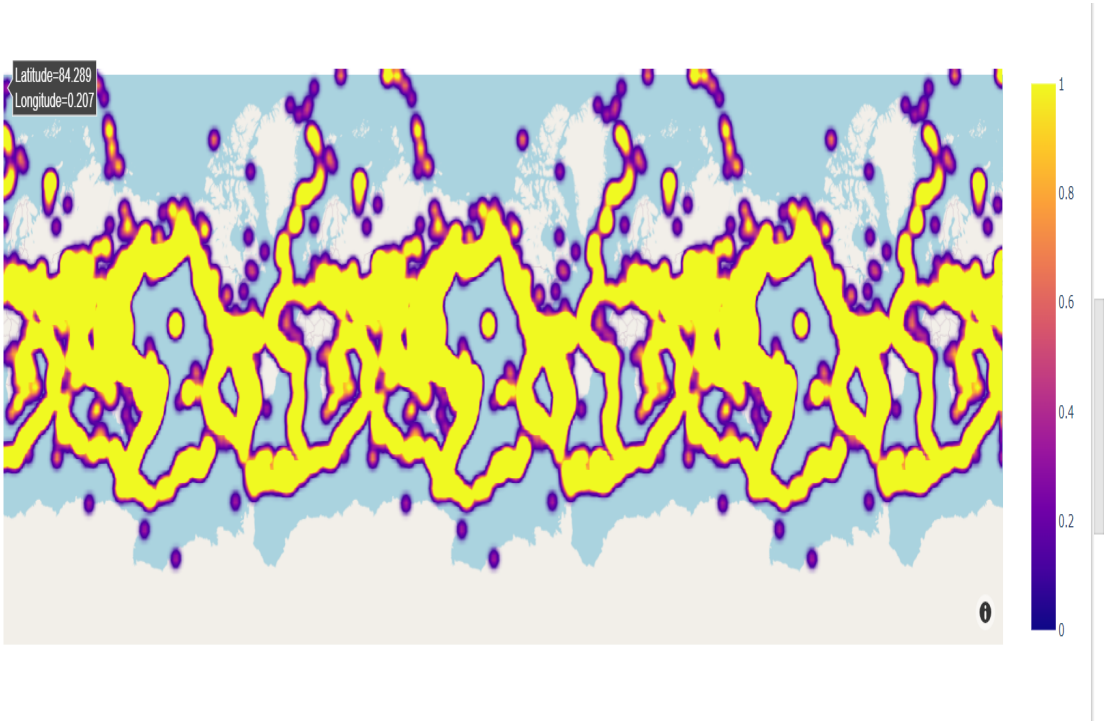
```
tmp = e_data.copy()
tmp = tmp[tmp['Magnitude']<=6.1]
tmp = tmp[tmp['Depth']<60]

sns.jointplot(data=tmp,x='Depth',y='Magnitude',kind='kde',cmap='coolwarm',height=12,levels=30)
```



Mapbox :

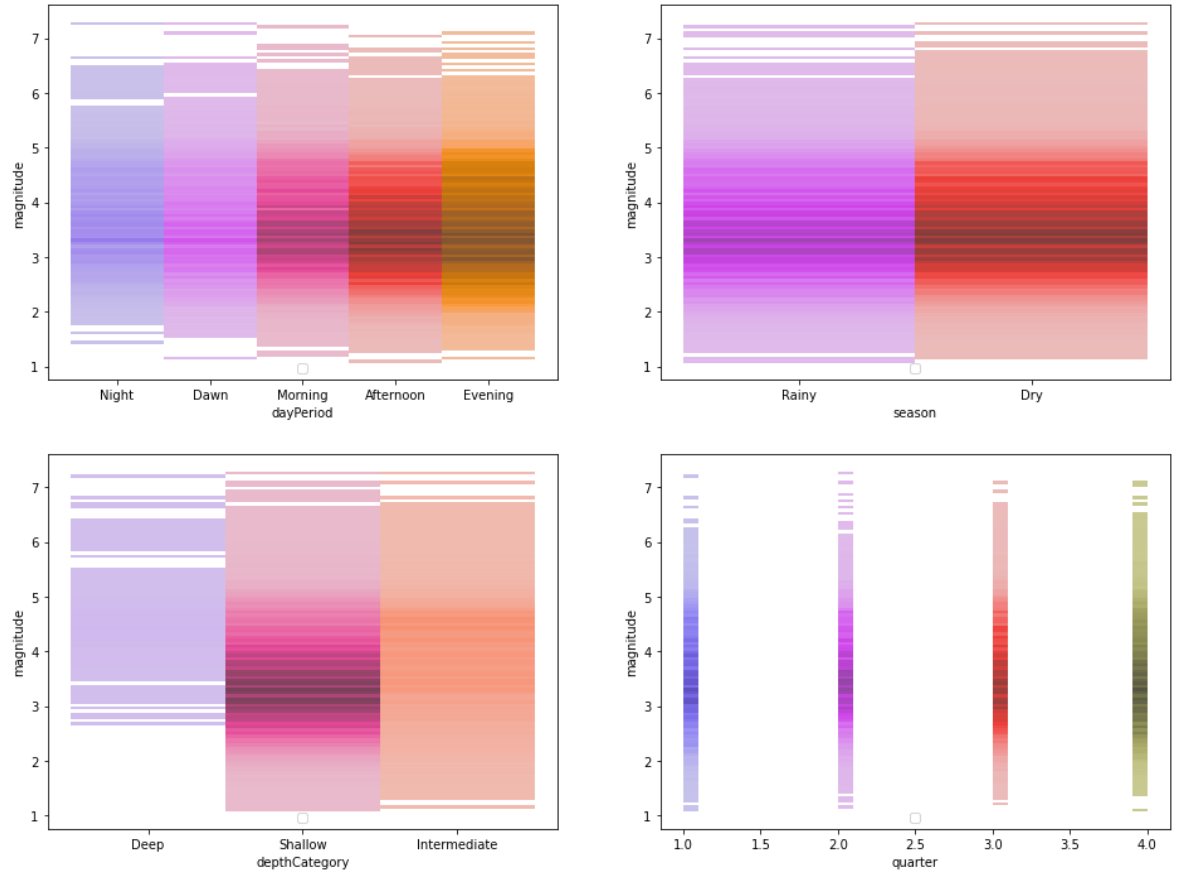
```
fig = px.density_mapbox(df, lat="Latitude",
lon="Longitude", mapbox_style="open-street-map", zoom =
0.5, radius = 10)
fig.show()
```



Subplot :

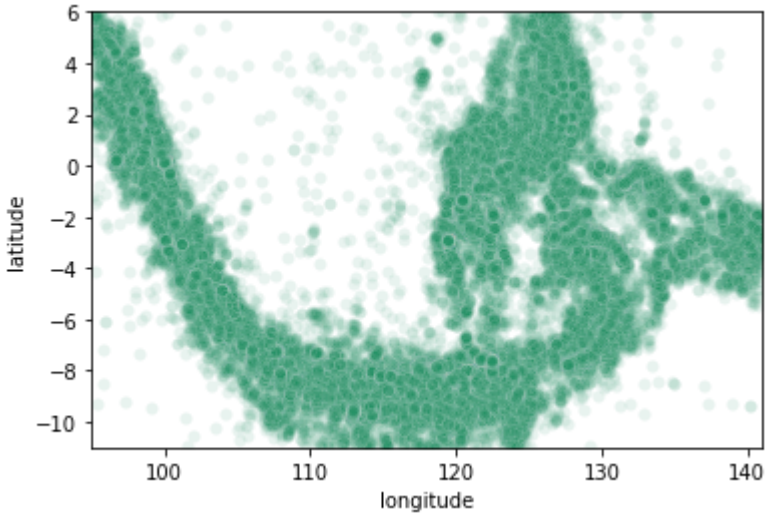
```
subplot = 1
plt.figure(figsize = (16,12))
for i in tempList:
    plt.subplot(2,2,subplot)
    sns.histplot(data = df,y ='magnitude',x =i,hue =
i,palette ='magma')
```

```
    plt.legend(loc=8)
    subplot = subplot + 1
plt.show()
```



Scatterplot :

```
sns.scatterplot(x=df['longitude'], y=df['latitude'],
alpha=0.1)
plt.axis([95, 141, -11, 6])
plt.show()
```

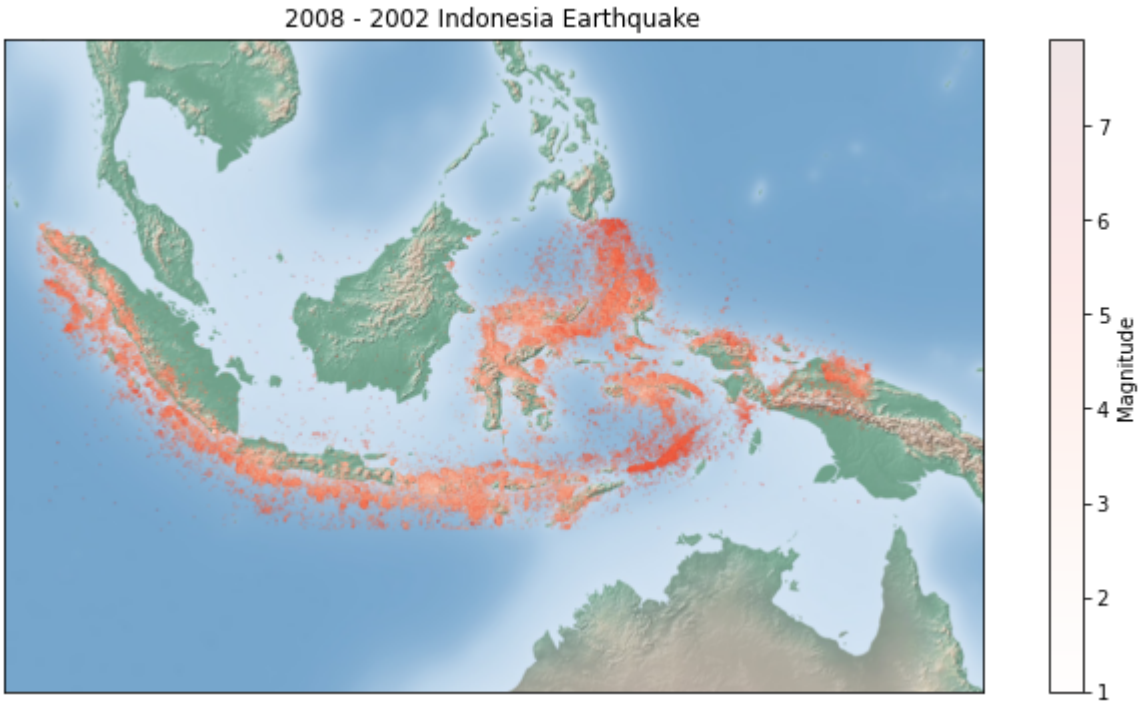


Geographical Visualisation :

```
def background_map():
    plt.figure(figsize=(12,6))
    m = Basemap(projection='lcc',
resolution=None,width=6E6, height=4E6, lat_0=-2,
lon_0=120)
    m.shadedrelief(scale=0.5)
    return m
m = background_map()
```

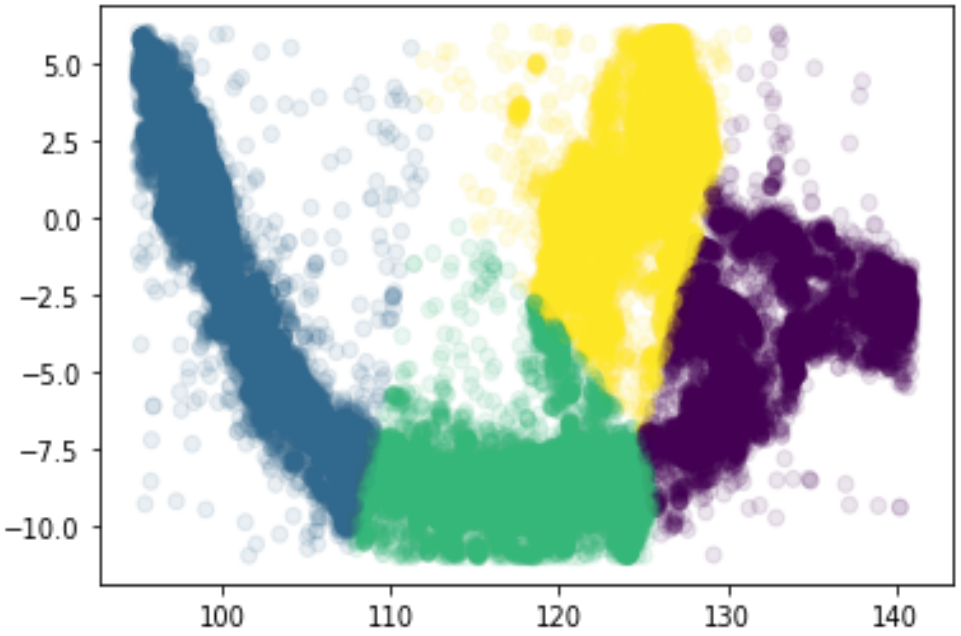
Background Map :

```
m = background_map()
m.scatter(df['longitude'], df['latitude'], s=0.5 ,
c=df['magnitude'], cmap='Reds', latlon=True, alpha=0.1)
plt.colorbar(label='Magnitude')
plt.title('2008 - 2002 Indonesia Earthquake')
Text(0.5, 1.0, '2008 - 2002 Indonesia Earthquake')
```



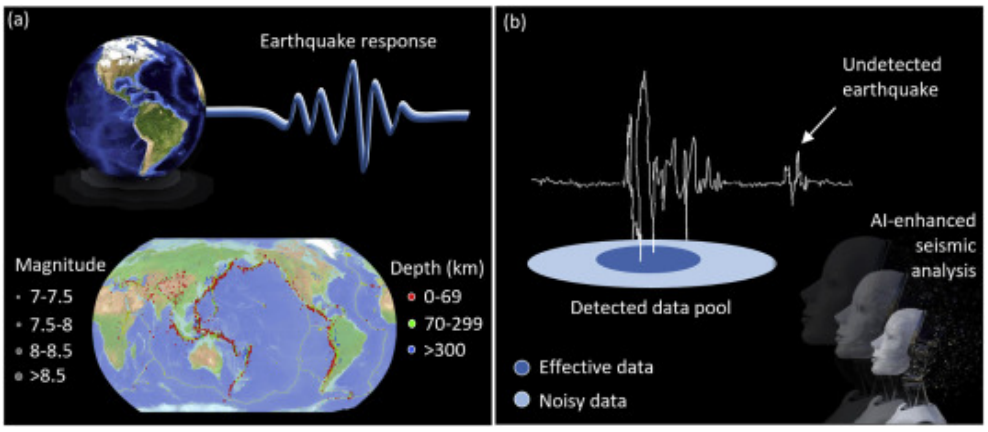
Segmentation Analysis :

```
from sklearn.cluster import KMeans
km_clf = KMeans(4)
km_clf.fit(df[['latitude', 'longitude']])
KMeans(n_clusters=4)
plt.scatter(df['longitude'], df['latitude'], alpha=0.1,
c=km_clf.labels_)
```



CONCLUSION :

Seismic event data exhibits temporal patterns like cycles and clustering that can be modelled by techniques like RNN, NLP ,YOLO .



Suitable performance metrics and baselining against naive models based on known statistical laws is key to evaluate true model skill. Combining geoscience domain expertise to constrain model architecture and inputs would be more fruitful than pure black-box modelling.

REFERENCES :

1. https://github.com/akash-r34/Earthquake-prediction-using-Machine-learning-models
2. https://www.researchgate.net/publication/313858017_Machine_Learning_Predicts_Laboratory_Earthquakes
3. http://earthquake.usgs.gov/data/comcat/data-eventterms.php#time
4. DriveData, Richter's Predictor: Modelling Earthquake Damage (2021).
5. Sameer, Earthquake History (1965–2016): Data Visualization and Model Development.
6. World Health Organization, Earthquakes, Health topics.