Prediction Tsunami

2.Data Understanding

Collect Initial Data

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
import numpy as np
```

```
eq=pd.read_csv('/content/drive/My Drive/Data_science/Project/earthquake_data.csv')
eq
```

	title	magnitude	date_time	cdi	mmi	alert	tsunami	sig	net	nst	dmin
0	M 7.0 - 18 km SW of Malango, Solomon Islands	7.0	22-11-2022 02:03	8	7	green	1	768	us	117	0.509
1	M 6.9 - 204 km SW of Bengkulu, Indonesia	6.9	18-11-2022 13:37	4	4	green	0	735	us	99	2.229
2	M 7.0 -	7.0	12-11-2022 07:09	3	3	green	1	755	us	147	3.125
3	M 7.3 - 205 km ESE of Neiafu, Tonga	7.3	11-11-2022 10:48	5	5	green	1	833	us	149	1.865
4	M 6.6 -	6.6	09-11-2022 10:14	0	2	green	1	670	us	131	4.998
...	...	...	...	...	...	...	...	...	...	...	...
777	M 7.7 - 28 km SSW of Puerto El Triunfo, El Sal...	7.7	13-01-2001 17:33	0	8	NaN	0	912	us	427	0.000
778	M 6.9 - 47 km S of Old Harbor, Alaska	6.9	10-01-2001 16:02	5	7	NaN	0	745	ak	0	0.000
779	M 7.1 - 16 km NE of Port-Olry, Vanuatu	7.1	09-01-2001 16:49	0	7	NaN	0	776	us	372	0.000
780	M 6.8 - Mindanao, Philippines	6.8	01-01-2001 08:54	0	5	NaN	0	711	us	64	0.000
781	M 7.5 - 21 km SE of Lukatan, Philippines	7.5	01-01-2001 06:57	0	7	NaN	0	865	us	324	0.000

782 rows × 19 columns

Describe Data

```
#รายละเอียดข้อมูล
eq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 782 entries, 0 to 781
Data columns (total 19 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   title      782 non-null    object
 1   magnitude  782 non-null    float64
 2   date_time  782 non-null    object
 3   cdi        782 non-null    int64
 4   mmi        782 non-null    int64
 5   alert      415 non-null    object
 6   tsunami    782 non-null    int64
 7   sig        782 non-null    int64
 8   net        782 non-null    object
 9   nst        782 non-null    int64
 10  dmin       782 non-null    float64
 11  gap        782 non-null    float64
 12  magType    782 non-null    object
 13  depth      782 non-null    float64
 14  latitude   782 non-null    float64
 15  longitude  782 non-null    float64
 16  location   777 non-null    object
 17  continent  206 non-null    object
 18  country    484 non-null    object
dtypes: float64(6), int64(5), object(8)
memory usage: 116.2+ KB
```

```
#Target
tsunami = eq['tsunami'].unique()
tsunami
```

```
array([1, 0])
```

```
eq.isnull().sum()/eq.shape[0]*100
#continent,country,alert have 73%,38%,46% null values
#we can drop continenet(too much null values)
#we have latitude and longitude so we can drop location
#alert is a unnecessary data
```

```
title         0.000000
magnitude     0.000000
date_time     0.000000
cdi           0.000000
mmi           0.000000
alert        46.930946
tsunami       0.000000
sig           0.000000
net           0.000000
nst           0.000000
dmin          0.000000
gap           0.000000
magType       0.000000
depth         0.000000
latitude      0.000000
longitude     0.000000
location      0.639386
continent    73.657289
country      38.107417
dtype: float64
```

```
eq.columns
```

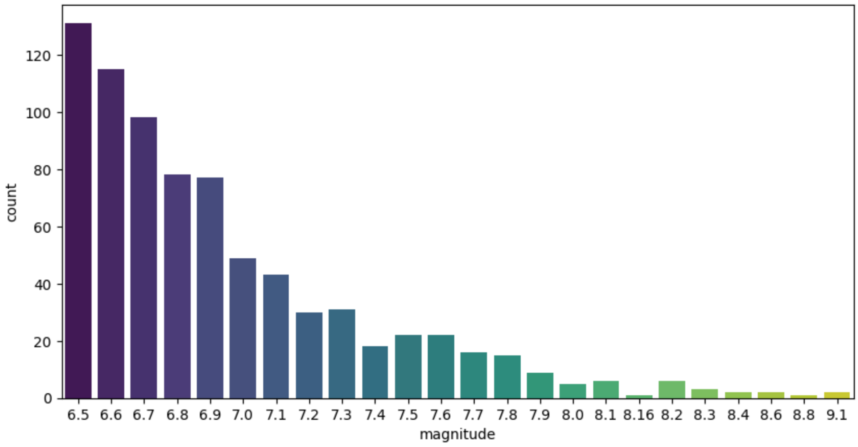
```
Index(['title', 'magnitude', 'date_time', 'cdi', 'mmi', 'alert', 'tsunami',
       'sig', 'net', 'nst', 'dmin', 'gap', 'magType', 'depth', 'latitude',
       'longitude', 'location', 'continent', 'country'],
      dtype='object')
```

Explore Data

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

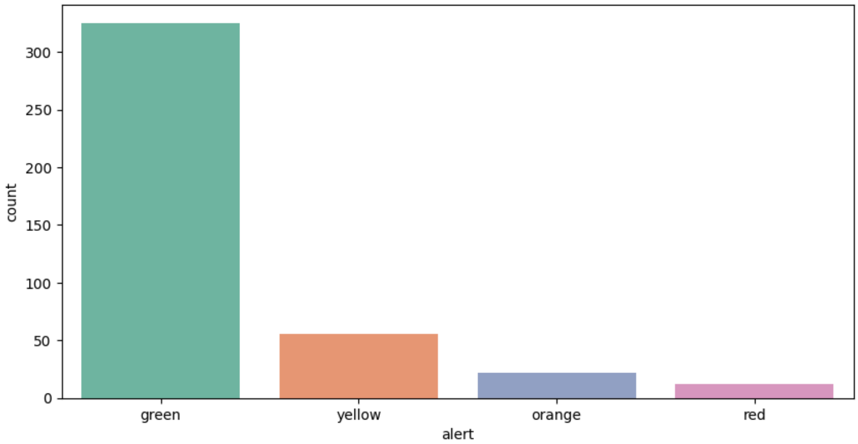
```
# Define a color palette
custom_palette = sns.color_palette("viridis", n_colors=len(eq['magnitude'].unique()))

plt.figure(figsize=(10, 5))
sns.countplot(x='magnitude', data=eq, palette=custom_palette)
plt.show()
```

```
<ipython-input-9-7b3f8b833d6f>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(x='magnitude', data=eq, palette=custom_palette)
```



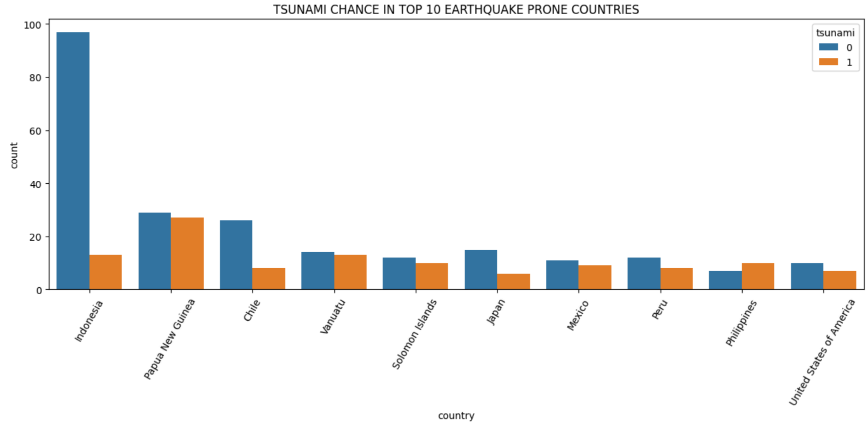

```
# Define a color palette
custom_palette = sns.color_palette("Set2")

plt.figure(figsize=(10, 5))
sns.countplot(x='alert', data=eq, palette=custom_palette)
plt.show()
```

```
<ipython-input-10-70537c2a153f>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(x='alert', data=eq, palette=custom_palette)
<ipython-input-10-70537c2a153f>:5: UserWarning: The palette list has more values (8)
  sns.countplot(x='alert', data=eq, palette=custom_palette)
```



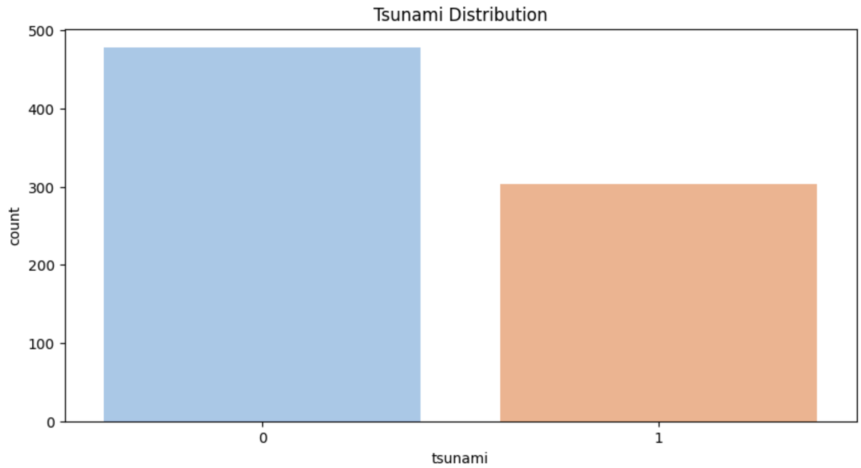

```
o=eq['country'].value_counts().head(10).index
plt.figure(figsize=(15,5))
sns.countplot(x='country',data=eq,order=o,hue='tsunami')
plt.xticks(rotation=60)
plt.title('TSUNAMI CHANCE IN TOP 10 EARTHQUAKE PRONE COUNTRIES')
#Indonesia has the highest number of earthquakes worldwide, but Papua New Guinea and Philippines  has a very high risk of tsunamis follo
```

```
Text(0.5, 1.0, 'TSUNAMI CHANCE IN TOP 10 EARTHQUAKE PRONE COUNTRIES')
```




```
# Define a color palette
custom_palette = sns.color_palette("pastel")

plt.figure(figsize=(10, 5))
sns.countplot(x='tsunami', data=eq, palette=custom_palette)
plt.title('Tsunami Distribution')
plt.show()
```

```
<ipython-input-12-4e78681796d2>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(x='tsunami', data=eq, palette=custom_palette)
<ipython-input-12-4e78681796d2>:5: UserWarning: The palette list has more values (10)
  sns.countplot(x='tsunami', data=eq, palette=custom_palette)
```



2. Data Preparation

```
#DROPING UNNECESSARY DATA
eq.drop(['date_time','title'],axis=1,inplace=True)
```

```
# แทนที่ค่าว่างด้วยค่าที่กำหนด
eq.fillna({'alert': 'No Alert', 'continent': 'Unknown', 'country': 'Unknown', 'location': 'Unknown'}, inplace=True)
```

```
eq.isnull().sum()
```

```
magnitude    0
cdi          0
mmi          0
alert        0
tsunami      0
sig          0
net          0
nst          0
dmin         0
gap          0
magType      0
depth        0
latitude     0
longitude    0
location     0
continent    0
country      0
dtype: int64
```

```
#รายละเอียดข้อมูลหลังลบค่าว่างใน column 'alert', 'continent', 'country','location'
eq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 782 entries, 0 to 781
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   magnitude  782 non-null    float64
 1   cdi        782 non-null    int64  
 2   mmi        782 non-null    int64  
 3   alert      782 non-null    object 
 4   tsunami    782 non-null    int64  
 5   sig        782 non-null    int64  
 6   net        782 non-null    object 
 7   nst        782 non-null    int64  
 8   dmin       782 non-null    float64
 9   gap        782 non-null    float64
 10  magType    782 non-null    object 
 11  depth      782 non-null    float64
 12  latitude   782 non-null    float64
 13  longitude  782 non-null    float64
 14  location   782 non-null    object 
 15  continent  782 non-null    object 
 16  country    782 non-null    object 
dtypes: float64(6), int64(5), object(6)
memory usage: 104.0+ KB
```

```
eq
```

	magnitude	cdi	mmi	alert	tsunami	sig	net	nst	dmin	gap	magType	depth
0	7.0	8	7	green	1	768	us	117	0.509	17.0	mww	14.000
1	6.9	4	4	green	0	735	us	99	2.229	34.0	mww	25.000
2	7.0	3	3	green	1	755	us	147	3.125	18.0	mww	579.000
3	7.3	5	5	green	1	833	us	149	1.865	21.0	mww	37.000
4	6.6	0	2	green	1	670	us	131	4.998	27.0	mww	624.464
...	...	...	...	...	...	...	...	...	...	...	...	...
777	7.7	0	8	No Alert	0	912	us	427	0.000	0.0	mwc	60.000
778	6.9	5	7	No Alert	0	745	ak	0	0.000	0.0	mw	36.400

```
# แสดงจำนวนข้อมูลก่อนการทำ balancing
print("จำนวนข้อมูลก่อนการทำ balancing:")
print(eq['tsunami'].value_counts())
```

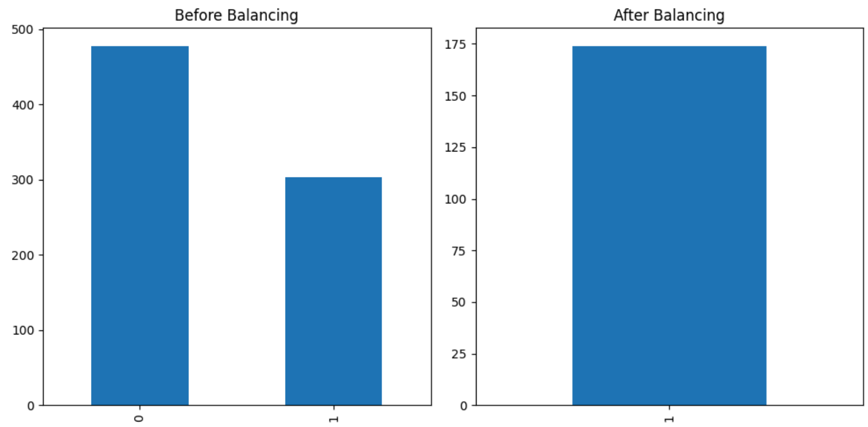
```
จำนวนข้อมูลก่อนการทำ balancing:
0    478
```

```
1    304
Name: tsunami, dtype: int64
```

```
# หาจำนวนตัวอย่างที่มากที่สุดในคอลัมน์ 'tsunami'
max_samples = eq['tsunami'].value_counts().max()

# สร้าง DataFrame เพื่อเก็บข้อมูลที่สมดุล
balanced_data = pd.DataFrame(columns=['tsunami'])

# วนลูปผ่านแต่ละกลุ่มข้อมูล
for index, count in eq['tsunami'].value_counts().items():
    tsunami_value = index

    # ถ้าจำนวนตัวอย่างในแต่ละกลุ่มน้อยกว่า max_samples
    if count < max_samples:
        # สุ่มเพิ่มข้อมูลเพื่อทำให้มีจำนวนเท่ากับ max_samples
        additional_samples = max_samples - count
        additional_data = pd.DataFrame({'tsunami': [tsunami_value] * additional_samples})

        # เพิ่มข้อมูลเพิ่มเติมเข้ากับ DataFrame ที่สมดุล
        balanced_data = pd.concat([balanced_data, additional_data], ignore_index=True)

# แสดง DataFrame หลังการทำ balancing
print("จำนวนข้อมูลหลังการทำ balancing:")
print(balanced_data['tsunami'].value_counts())
```

```
จำนวนข้อมูลหลังการทำ balancing:
1    174
Name: tsunami, dtype: int64
```

```
# สร้างกราฟเปรียบเทียบก่อนและหลังทำ balancing
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
eq['tsunami'].value_counts().plot(kind='bar', title='Before Balancing')

plt.subplot(1, 2, 2)
balanced_data['tsunami'].value_counts().plot(kind='bar', title='After Balancing')

plt.tight_layout()
plt.show()
```




```
eq.head()
```

	magnitude	cdi	mmi	alert	tsunami	sig	net	nst	dmin	gap	magType	depth	la
0	7.0	8	7	green	1	768	us	117	0.509	17.0	mww	14.000	
1	6.9	4	4	green	0	735	us	99	2.229	34.0	mww	25.000	
2	7.0	3	3	green	1	755	us	147	3.125	18.0	mww	579.000	-

Feature Engineering

```
from sklearn.preprocessing import LabelEncoder
```

```
le_alert=LabelEncoder()
le_net=LabelEncoder()
le_magType=LabelEncoder()
le_location=LabelEncoder()
le_continent=LabelEncoder()
le_country=LabelEncoder()
```

```
#แปลงข้อมูล
alert_en=le_alert.fit_transform(eq['alert'])
net_en=le_net.fit_transform(eq['net'])
magType_en=le_magType.fit_transform(eq['magType'])
location_en=le_location.fit_transform(eq['location'])
continent_en=le_continent.fit_transform(eq['continent'])
country_en=le_country.fit_transform(eq['country'])
```

```
#ใส่ข้อมูลเข้าไปใหม่ในตาราง
eq['alert_en']=alert_en
eq['net_en']=net_en
eq['magType_en']=magType_en
eq['location_en']=location_en
eq['continent_en']=continent_en
eq['country_en']=country_en
```

```
#แสดงข้อมูลหลังแปลงตัวอักษรเป็นตัวเลข
eq.head()
```

	magnitude	cdi	mmi	alert	tsunami	sig	net	nst	dmin	gap	...	longitude	loc
0	7.0	8	7	green	1	768	us	117	0.509	17.0	...	159.596	Ma Sc I
1	6.9	4	4	green	0	735	us	99	2.229	34.0	...	100.738	Ber Ind
2	7.0	3	3	green	1	755	us	147	3.125	18.0	...	-178.346	Un
3	7.3	5	5	green	1	833	us	149	1.865	21.0	...	-172.129	N
4	6.6	0	2	green	1	670	us	131	4.998	27.0	...	178.278	Un

5 rows × 23 columns

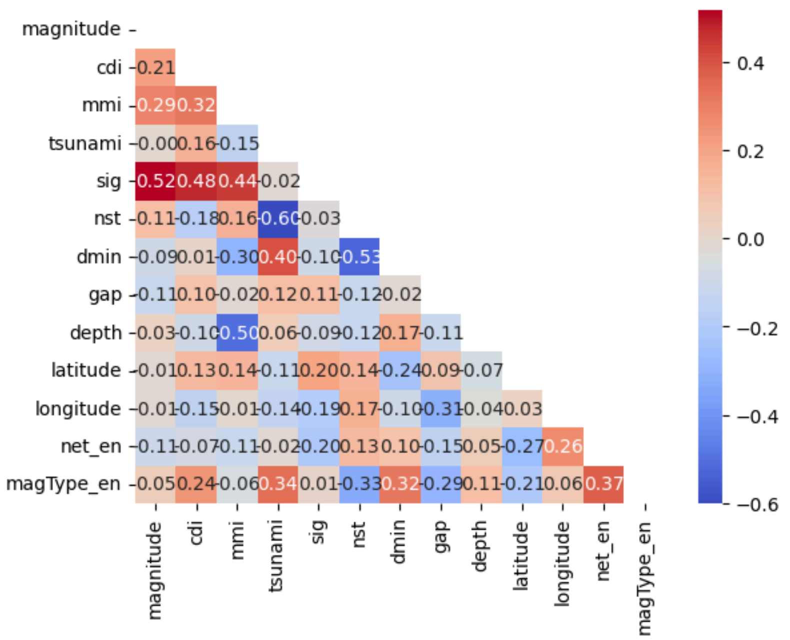
```
#ลบคอลัมน์ที่ไม่ใช้งาน
eq.drop(['alert','net','magType','location','continent','country','location_en','country_en','continent_en','alert_en'], axis=1, inplac
```

```
eq
```

	magnitude	cdi	mmi	tsunami	sig	nst	dmin	gap	depth	latitude	longitude
0	7.0	8	7	1	768	117	0.509	17.0	14.000	-9.7963	159.596
1	6.9	4	4	0	735	99	2.229	34.0	25.000	-4.9559	100.738
2	7.0	3	3	1	755	147	3.125	18.0	579.000	-20.0508	-178.346
3	7.3	5	5	1	833	149	1.865	21.0	37.000	-19.2918	-172.129
4	6.6	0	2	1	670	131	4.998	27.0	624.464	-25.5948	178.278
...	...	...	...	...	...	...	...	...	...	...	...
777	7.7	0	8	0	912	427	0.000	0.0	60.000	13.0490	-88.660
778	6.9	5	7	0	745	0	0.000	0.0	36.400	56.7744	-153.281
779	7.1	0	7	0	776	372	0.000	0.0	103.000	-14.9280	167.170
780	6.8	0	5	0	711	64	0.000	0.0	33.000	6.6310	126.899
781	7.5	0	7	0	865	324	0.000	0.0	33.000	6.8980	126.579

```
#correlation matrix ของข้อมูล tsunami

# Check data correlation
correlation = eq.corr()

# Get correlations related to 'tsunami' column
tsunami_correlation = correlation['tsunami']

# Exclude self-correlation and sort by absolute value
top_5_correlation = tsunami_correlation.drop('tsunami').abs().nlargest(5)

print("Top 5 highest correlations with 'tsunami':")
print(top_5_correlation)

# Create a mask for the upper triangle
mask = np.triu(np.ones_like(correlation, dtype=bool))

# Generate the heatmap
sns.heatmap(correlation, mask=mask, cmap='coolwarm', annot=True, fmt=".2f")
```

```
Top 5 highest correlations with 'tsunami':
nst          0.600231
dmin         0.400752
magType_en   0.340445
cdi          0.160266
mmi          0.147363
Name: tsunami, dtype: float64
<Axes: >
```



3. Modeling

```
!pip install pycaret
```

```
Requirement already satisfied: pycaret in /usr/local/lib/python3.10/dist-packages (3.3.0)
Requirement already satisfied: ipython>=5.5.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (7.34.0)
Requirement already satisfied: ipywidgets>=7.6.5 in /usr/local/lib/python3.10/dist-packages (from pycaret) (7.7.1)
Requirement already satisfied: tqdm>=4.62.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (4.66.2)
Requirement already satisfied: numpy<1.27,>=1.21 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.25.2)
Requirement already satisfied: pandas<2.2.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.5.3)
Requirement already satisfied: jinja2>=3 in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.1.3)
Requirement already satisfied: scipy<=1.11.4,>=1.6.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.11.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.3.2)
Requirement already satisfied: scikit-learn>1.4.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.4.1.post1)
Requirement already satisfied: pyod>=1.1.3 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.1.3)
Requirement already satisfied: imbalanced-learn>=0.12.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.12.0)
Requirement already satisfied: category-encoders>=2.4.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.6.3)
Requirement already satisfied: lightgbm>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (4.1.0)
Requirement already satisfied: numba>=0.55.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.58.1)
Requirement already satisfied: requests>=2.27.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.31.0)
Requirement already satisfied: psutil>=5.9.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.9.5)
Requirement already satisfied: markupsafe>=2.0.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.1.5)
Requirement already satisfied: importlib-metadata>=4.12.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (7.1.0)
Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.10.3)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.2.1)
Requirement already satisfied: deprecation>=2.1.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.1.0)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.4.1)
Requirement already satisfied: matplotlib<3.8.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.7.1)
Requirement already satisfied: scikit-plot>=0.3.7 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.3.7)
Requirement already satisfied: yellowbrick>=1.4 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.5)
Requirement already satisfied: plotly>=5.14.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.15.0)
Requirement already satisfied: kaleido>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.2.1)
Requirement already satisfied: schemdraw==0.15 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.15)
Requirement already satisfied: plotly-resampler>=0.8.3.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.10.0)
Requirement already satisfied: statsmodels>=0.12.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.14.1)
Requirement already satisfied: sktime>=0.26.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.28.0)
Requirement already satisfied: tbats>=1.1.3 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.1.3)
Requirement already satisfied: pmdarima>=2.0.4 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.0.4)
Requirement already satisfied: wurlitzer in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.0.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category-encoders>=2.4.0->pycaret)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from deprecation>=2.1.0->pycaret) (24.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn>=0.12.0->p
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata>=4.12.0->pycaret) (
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (67.7.
Requirement already satisfied: jedi>=0.16 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (0.19.1)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (5.7.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from ipy
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (2.16.1)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (0.2.0)
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (0.1.
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (4.9.0)
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret) (5.
Requirement already satisfied: ipython-genutils~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycar
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pyc
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pyc
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret) (1.2
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret) (4.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret) (1.
```

```
# import pycaret classification and init setup
from pycaret.classification import *
py_eq = setup(eq, target = 'tsunami', session_id = 123 ,fold = 5)
```

	Description	Value
0	Session id	123
1	Target	tsunami
2	Target type	Binary
3	Original data shape	(782, 13)
4	Transformed data shape	(782, 13)
5	Transformed train set shape	(547, 13)
6	Transformed test set shape	(235, 13)
7	Numeric features	12
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	StratifiedKFold
13	Fold Number	5
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	clf-default-name
18	USI	a2fd

```
from pycaret.classification import *
exp_reg101 = setup(data = eq, target = 'tsunami', session_id=123,fold=5)
```

	Description	Value
0	Session id	123
1	Target	tsunami
2	Target type	Binary
3	Original data shape	(782, 13)
4	Transformed data shape	(782, 13)
5	Transformed train set shape	(547, 13)
6	Transformed test set shape	(235, 13)
7	Numeric features	12
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	StratifiedKFold
13	Fold Number	5
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	clf-default-name
18	USI	a84b

```
compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.8903	0.9456	0.8681	0.8537	0.8599	0.7698	0.7711	0.2820
rf	Random Forest Classifier	0.8885	0.9474	0.8588	0.8575	0.8570	0.7657	0.7671	0.5300
et	Extra Trees Classifier	0.8885	0.9406	0.8685	0.8491	0.8585	0.7665	0.7670	0.4340
gbc	Gradient Boosting Classifier	0.8867	0.9416	0.8590	0.8534	0.8548	0.7620	0.7637	0.4700
lightgbm	Light Gradient Boosting Machine	0.8848	0.9466	0.8493	0.8532	0.8500	0.7566	0.7580	1.1920
ada	Ada Boost Classifier	0.8739	0.9478	0.8216	0.8516	0.8355	0.7334	0.7346	0.5800
dt	Decision Tree Classifier	0.8556	0.8460	0.8027	0.8228	0.8116	0.6947	0.6959	0.0840
lda	Linear Discriminant Analysis	0.8245	0.8653	0.8734	0.7291	0.7944	0.6434	0.6521	0.0700
knn	K Neighbors Classifier	0.8227	0.8798	0.7934	0.7625	0.7760	0.6294	0.6317	0.1180
ridge	Ridge Classifier	0.8227	0.0000	0.8688	0.7280	0.7918	0.6393	0.6477	0.0820
lr	Logistic Regression	0.8135	0.8532	0.8501	0.7207	0.7797	0.6199	0.6269	2.2540
nb	Naive Bayes	0.8117	0.8272	0.8544	0.7170	0.7789	0.6170	0.6254	0.1060
qda	Quadratic Discriminant	0.7952	0.8411	0.8071	0.7075	0.7532	0.5793	0.5844	0.0860

Extreme Gradient Boosting

```
xgboost_model = create_model('xgboost')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8818	0.9337	0.8605	0.8409	0.8506	0.7529	0.7530
1	0.8909	0.9427	0.9070	0.8298	0.8667	0.7747	0.7769
2	0.8807	0.9453	0.8095	0.8718	0.8395	0.7448	0.7461
3	0.8899	0.9460	0.8333	0.8750	0.8537	0.7655	0.7661
4	0.9083	0.9605	0.9302	0.8511	0.8889	0.8110	0.8134
Mean	0.8903	0.9456	0.8681	0.8537	0.8599	0.7698	0.7711
Std	0.0099	0.0086	0.0449	0.0175	0.0169	0.0230	0.0236

```
#tune เพื่อปรับค่าพารามิเตอร์ให้ดีที่สุด
tuned_xgboost_model = tune_model(xgboost_model)
```

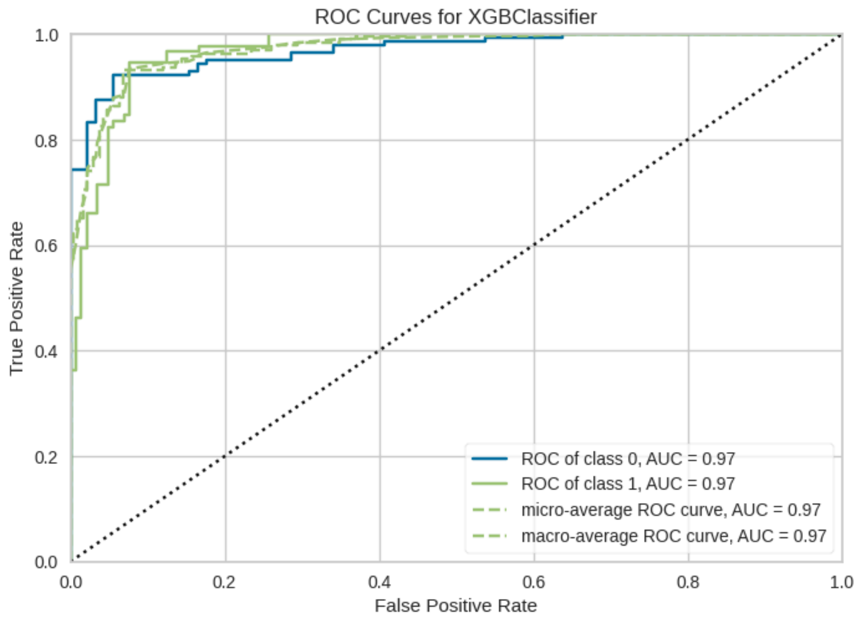
Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8818	0.9351	0.8837	0.8261	0.8539	0.7549	0.7561
1	0.8818	0.9434	0.9302	0.8000	0.8602	0.7589	0.7653
2	0.8899	0.9471	0.9762	0.7885	0.8723	0.7775	0.7912
3	0.8440	0.9382	0.8810	0.7551	0.8132	0.6807	0.6866
4	0.8532	0.9602	1.0000	0.7288	0.8431	0.7114	0.7431
Mean	0.8702	0.9448	0.9342	0.7797	0.8486	0.7367	0.7485
Std	0.0181	0.0087	0.0479	0.0342	0.0201	0.0354	0.0347

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

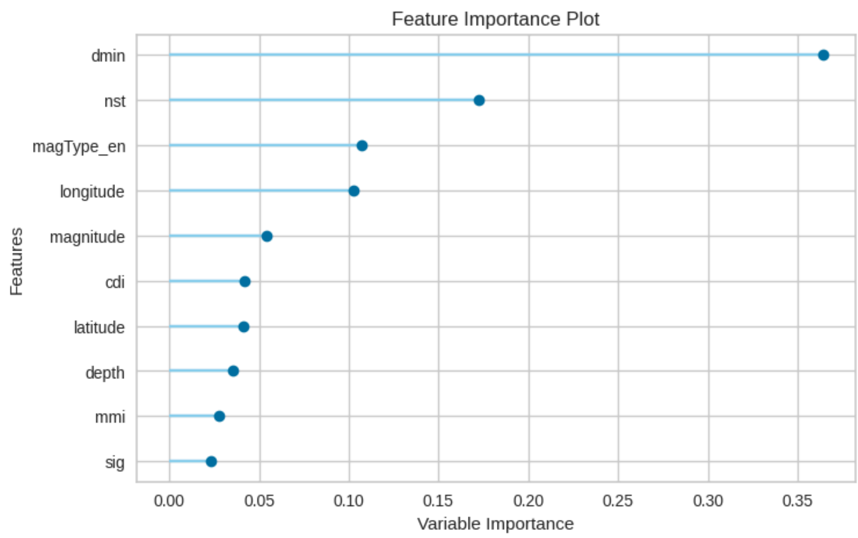
```
#tuned model object is stored in the variable 'tuned_model'.
print(tuned_xgboost_model)
```

```
XGBClassifier(base_score=None, booster='gbtree', callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device='cpu', early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=-1,
              num_parallel_tree=None, objective='binary:logistic', ...)
```

```
plot_model(tuned_xgboost_model)
```




```
plot_model(tuned_xgboost_model, plot='feature')
```




```
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
```

```
# กำหนดค่าพารามิเตอร์ที่ถูกปรับค่า
tuned_xgboost_params = {
    'n_estimators': 100,
    'max_depth': 6,
    'learning_rate': 0.1,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'gamma': 0,
    'random_state': 123
}


# Split Data into Train and Test Sets
X_train, X_test, y_train, y_test = train_test_split(eq[['dmin','nst','magType_en', 'longitude', 'magnitude']], eq['tsunami'], test_size=

# สร้างโมเดล XGBClassifier ด้วยพารามิเตอร์ที่ถูกปรับค่า
tuned_xgboost_model = XGBClassifier(**tuned_xgb_params)

# เทรนโมเดลด้วยชุดข้อมูลการฝึก
tuned_xgboost_model.fit(X_train, y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=6, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, objective='binary:logistic', ...)
```

```
# ทำนายข้อมูลทดสอบ
y_pred = tuned_xgboost_model.predict(X_test)
y_pred
```

```
array([1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0])
```

Random Forest Classifier

```
rf_model = create_model('rf')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8545	0.9346	0.8372	0.8000	0.8182	0.6971	0.6976
1	0.8909	0.9427	0.8837	0.8444	0.8636	0.7728	0.7734
2	0.8899	0.9636	0.8095	0.8947	0.8500	0.7634	0.7658
3	0.8991	0.9316	0.8333	0.8974	0.8642	0.7841	0.7854
4	0.9083	0.9646	0.9302	0.8511	0.8889	0.8110	0.8134
Mean	0.8885	0.9474	0.8588	0.8575	0.8570	0.7657	0.7671
Std	0.0182	0.0141	0.0431	0.0361	0.0231	0.0378	0.0383

```
#tune เพื่อปรับค่าพารามิเตอร์ให้ดีที่สุด
tuned_rf_model = tune_model(rf_model)
```

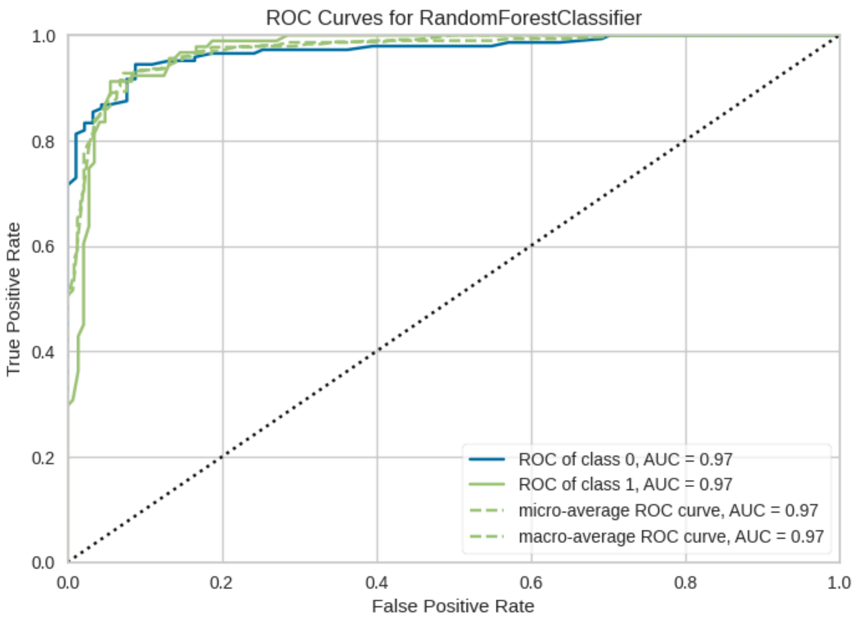
Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8364	0.8820	0.7907	0.7907	0.7907	0.6564	0.6564
1	0.8727	0.9360	0.8837	0.8085	0.8444	0.7371	0.7392
2	0.8624	0.9371	0.8095	0.8293	0.8193	0.7082	0.7083
3	0.8807	0.9138	0.8810	0.8222	0.8506	0.7515	0.7528
4	0.8532	0.9412	0.9302	0.7547	0.8333	0.7047	0.7170
Mean	0.8611	0.9220	0.8590	0.8011	0.8277	0.7116	0.7147
Std	0.0155	0.0222	0.0515	0.0267	0.0213	0.0327	0.0332

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

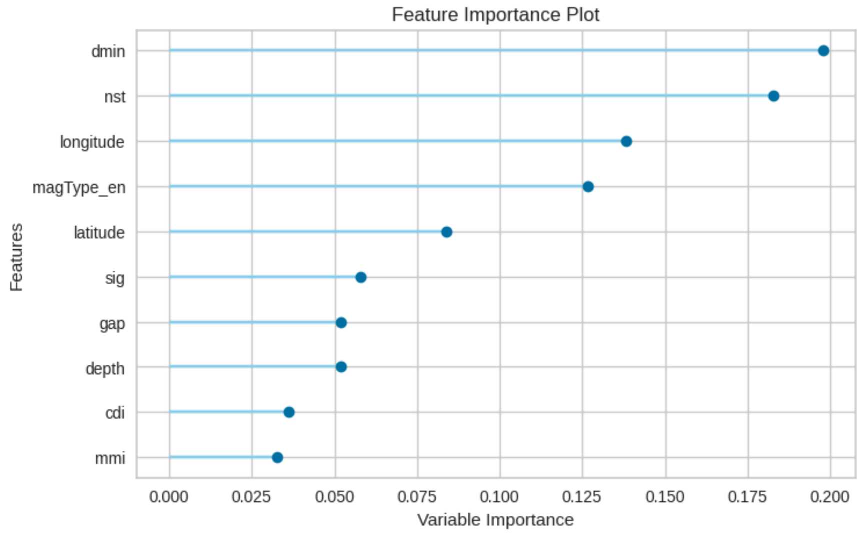
```
#tuned model object is stored in the variable 'tuned_model'.
print(tuned_rf_model)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       monotonic_cst=None, n_estimators=100, n_jobs=-1,
                       oob_score=False, random_state=123, verbose=0,
                       warm_start=False)
```

```
plot_model(tuned_rf_model)
```




```
plot_model(tuned_rf_model, plot='feature')
```




```
from sklearn.ensemble import RandomForestClassifier


# แบ่งข้อมูลเป็น Train และ Test
X_train, X_test, y_train, y_test = train_test_split(eq[['longitude', 'dmin', 'latitude','nst','magType_en']], eq['tsunami'], test_size=(


# กำหนดพารามิเตอร์ที่จะใช้ในการปรับแต่ง Random Forest
tuned_rf_params = {
    'n_estimators': 100,
    'max_depth': None,
    'min_samples_split': 2,
    'min_samples_leaf': 1,
    'bootstrap': True
}


# สร้างโมเดล RandomForestClassifier โดยใช้พารามิเตอร์ที่ถูกปรับค่า
tuned_rf_model = RandomForestClassifier(**tuned_rf_params)

# เทรนโมเดลด้วยข้อมูลการฝึก
tuned_rf_model.fit(X_train, y_train)
```

```
▾                        RandomForestClassifier                      i ?
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       monotonic_cst=None, n_estimators=100, n_jobs=None,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False)
```

```
# ใช้โมเดลที่ฝึกไว้เพื่อทำนายผลลัพธ์ของชุดข้อมูลทดสอบ
# ทำนายบนชุดข้อมูลทดสอบ
rf_y_pred = tuned_rf_model.predict(X_test)
```

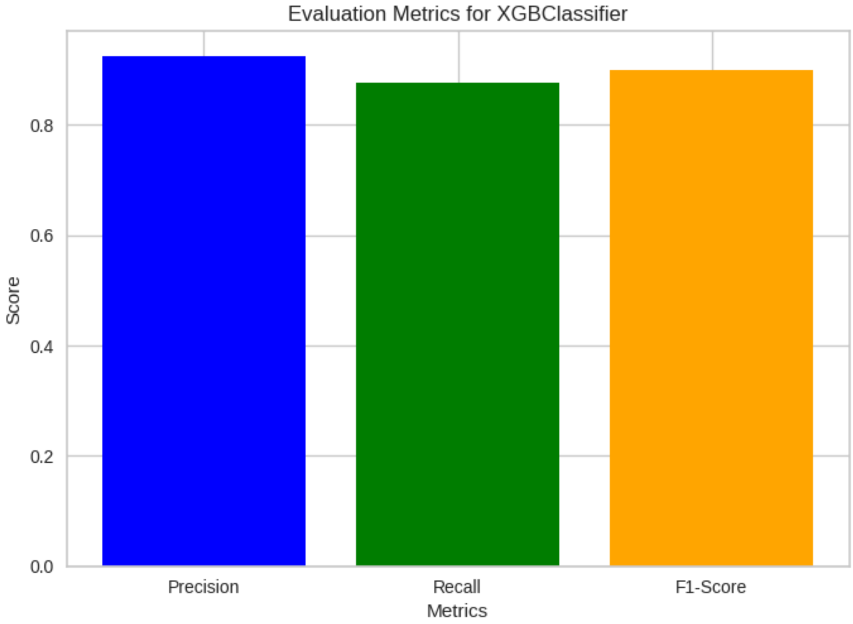
4. Evaluation

```
from sklearn.metrics import precision_score, accuracy_score, recall_score, f1_score, confusion_matrix
```

```
# คำนวณค่าการประเมิน
xgboost_accuracy = accuracy_score(y_test, y_pred)
xgboost_precision = precision_score(y_test, y_pred)
xgboost_recall = recall_score(y_test, y_pred)
xgboost_f1 = f1_score(y_test, y_pred)

# แสดงผลลัพธ์
print("Accuracy:", xgboost_accuracy)
print("Precision:", xgboost_precision)
print("Recall:", xgboost_recall)
print("F1-Score:", xgboost_f1)
```

```
Accuracy: 0.9191489361702128
Precision: 0.9239130434782609
Recall: 0.8762886597938144
F1-Score: 0.8994708994708994
```

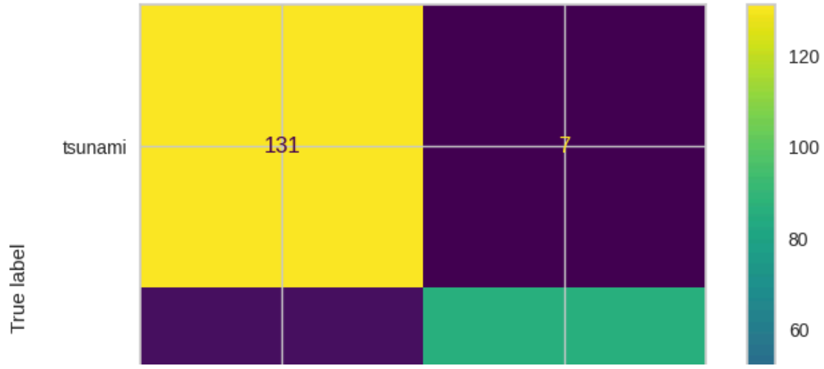
```
# กำหนดค่า labels
labels = ['Precision', 'Recall', 'F1-Score']

# พล็อตกราฟ
plt.bar(labels, [xgboost_precision, xgboost_recall, xgboost_f1], color=['blue', 'green', 'orange'])
plt.xlabel('Metrics')
plt.ylabel('Score')
plt.title('Evaluation Metrics for XGBClassifier')
plt.show()
```




```
# คำนวณ Confusion Matrix สำหรับ XGBoost
conf_matrix_xgboost = confusion_matrix(y_test, y_pred)
conf_matrix_xgboost
```

```
array([[131,   7],
       [ 12,  85]])
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

# พล็อต Confusion Matrix
disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix_xgboost, display_labels=['tsunami', 'no tsunami'])
disp.plot()
plt.show()
```



Random Forest Classifier


```
# คำนวณค่าการประเมินสำหรับ Random Forest
rf_accuracy = accuracy_score(y_test, rf_y_pred)
rf_precision = precision_score(y_test, rf_y_pred)
rf_recall = recall_score(y_test, rf_y_pred)
rf_f1 = f1_score(y_test, rf_y_pred)

print("Evaluation Metrics for Random Forest:")
print("Accuracy:", rf_accuracy)
print("Precision:", rf_precision)
print("Recall:", rf_recall)
print("F1-Score:", rf_f1)
print()
```