importing the libraries

EarthQuake Classification

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split,GridSearchCV, RandomizedSearchCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report , accuracy_score ,f1_score
from xgboost import XGBClassifier
from  lightgbm import  LGBMClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve ,auc
import mlflow
import mlflow.sklearn
import scipy

import kaggle
from kaggle.api.kaggle_api_extended import KaggleApi
```

importing the data via kaggle API

In [33]:

```
api = KaggleApi()
api.authenticate()

api.dataset_download_files("ahmeduzaki/los-angeles-california-earthquake-dataset",path="
```

```
Dataset URL: https://www.kaggle.com/datasets/ahmeduzaki/los-angeles-california-earthquak
e-dataset
```

logging to mlflow databrics

In [3]:

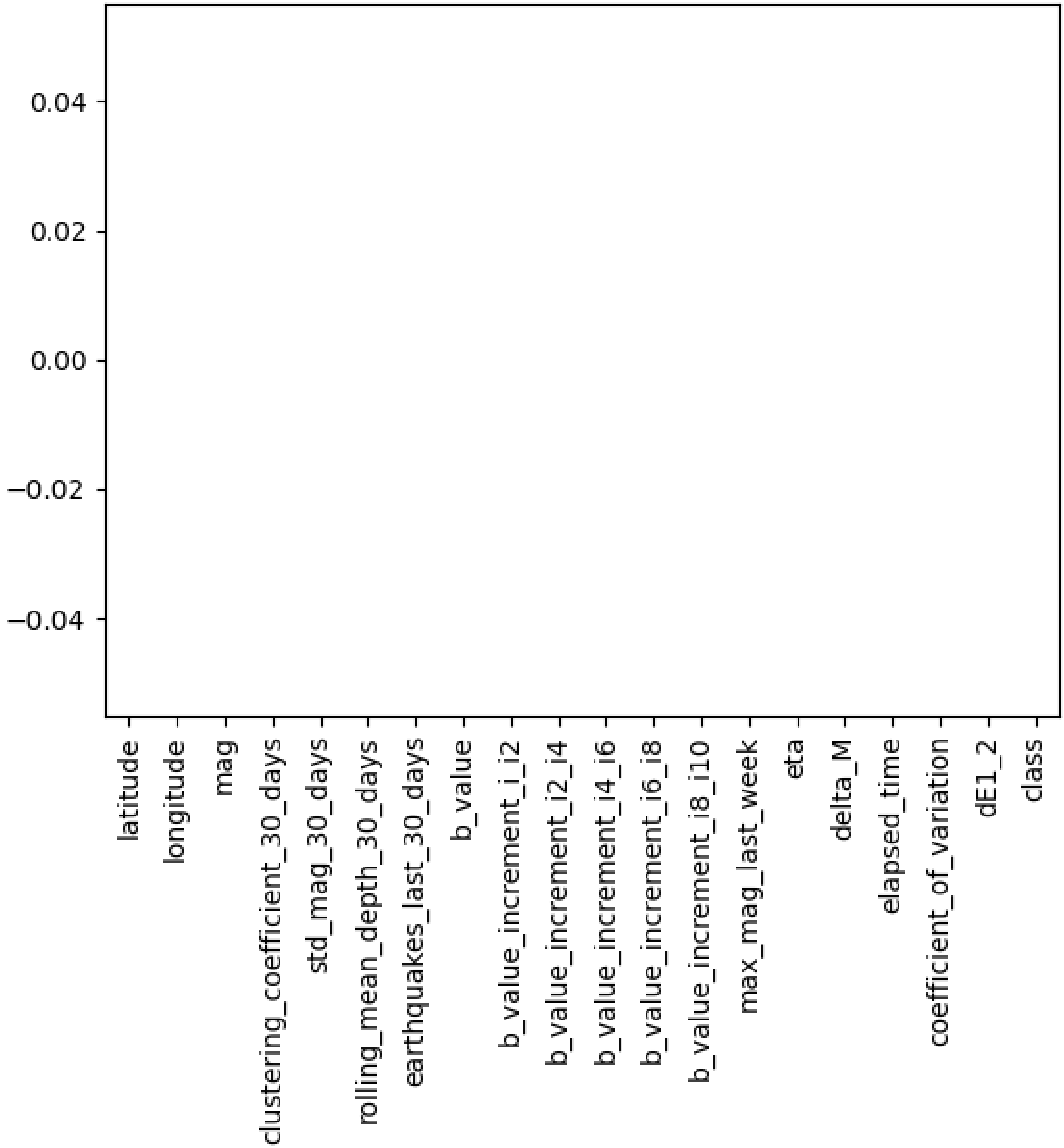
```
mlflow.set_tracking_uri('databricks://DEFAULT')
mlflow.set_experiment("/Users/sadikiyassine743@gmail.com/Ml")
```

Out[3]:

```
<Experiment: artifact_location='dbfs:/databricks/mlflow-tracking/4494213887472809', crea
tion_time=1759140661475, experiment_id='4494213887472809', last_update_time=175915171871
6, lifecycle_stage='active', name='/Users/sadikiyassine743@gmail.com/Ml', tags={'mlflow.
experiment.sourceName': '/Users/sadikiyassine743@gmail.com/Ml',
 'mlflow.experimentKind': 'custom_model_development',
 'mlflow.experimentType': 'MLFLOW_EXPERIMENT',
 'mlflow.ownerEmail': 'sadikiyassine743@gmail.com',
 'mlflow.ownerId': '73175433355009'}>
```

In [5]:

```
df = pd.read_csv(r"C:\Users\ayman\OneDrive\Documents\Data science projects\project\Earth
```

In [6]:
```
df.isna().sum().plot(kind='bar')
```
Out[6]:
```
<Axes: >
```


In [7]:
```
df.shape
```
Out[7]:
```
(22899, 20)
```
In [8]:
```
df.dtypes
```
Out[8]:
```
latitude                          float64
longitude                         float64
mag                               float64
clustering_coefficient_30_days    float64
std_mag_30_days                   float64
rolling_mean_depth_30_days        float64
earthquakes_last_30_days            int64
b_value                           float64
```

```
b_value_increment_i_i2            float64
b_value_increment_i2_i4           float64
b_value_increment_i4_i6           float64
b_value_increment_i6_i8           float64
b_value_increment_i8_i10          float64
max_mag_last_week                 float64
eta                               float64
delta_M                           float64
elapsed_time                      float64
coefficient_of_variation          float64
dE1_2                             float64
class                               int64
dtype: object
```

In [9]:

```
df.duplicated().sum()
```

Out[9]:

```
np.int64(0)
```

In [10]:

```
df["class"].unique()
```

Out[10]:

```
array([3, 1, 4, 2, 5, 6])
```

about the data:

- in this classification project we are using a dataset with `22899` row and 20 columns , the data does not have any null or duplicated values , our target variable have 6 class .

about the project:

- this project is Multiclassification problem with a target variable of 6 classes , so our main goal is to build a rebust multiclassification model that can capture the trends and the criteria of each type of earchquake , and to do so we will :
  - understand the factors that influences and give birth to each type of earthquake
  - Data Preprocessing (feature selection , scaling the data ...)
  - train and evaluate the models
  - select the best models and fine tunned to optimise the performance and the accuracy

EDA

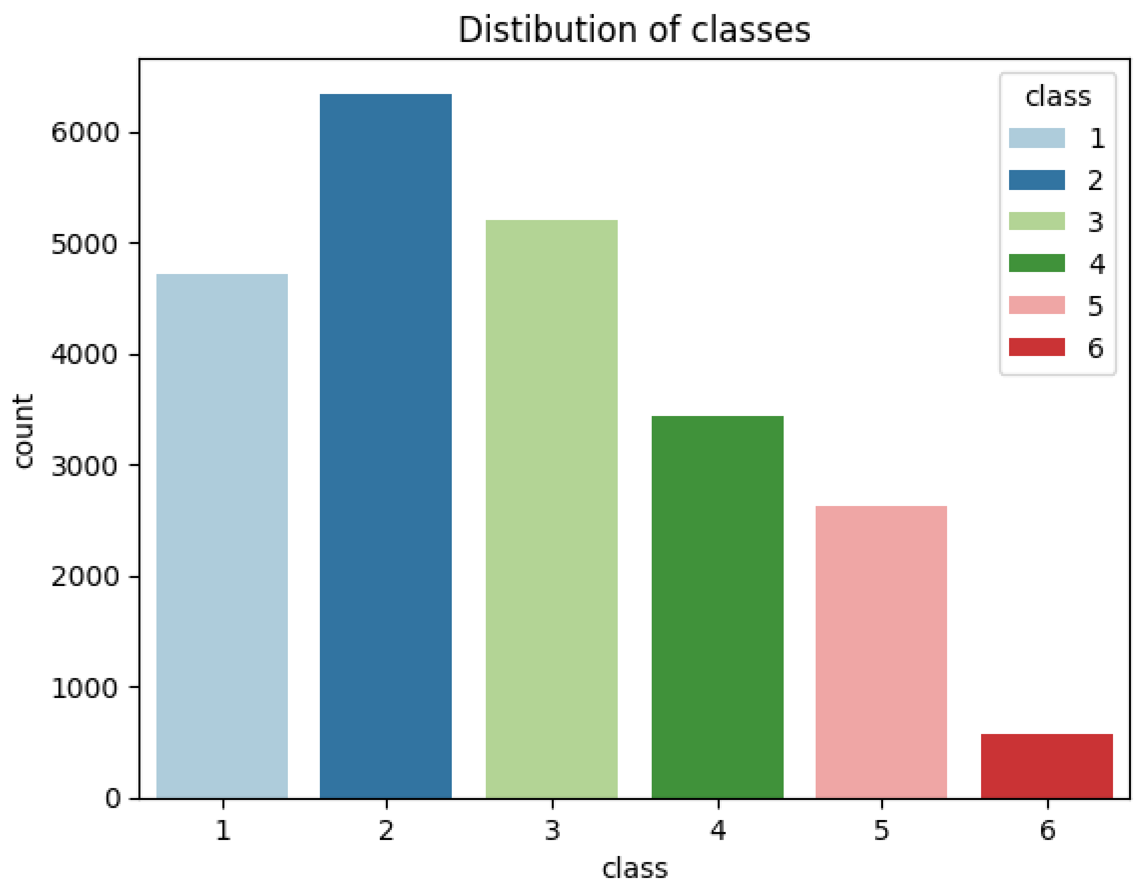
Target data classes

In [11]:

```
sns.countplot(df,x=df['class'],hue=df['class'],palette='Paired',dodge=False)
plt.title('Distibution of classes')
```

Out[11]:

```
Text(0.5, 1.0, 'Distibution of classes')
```



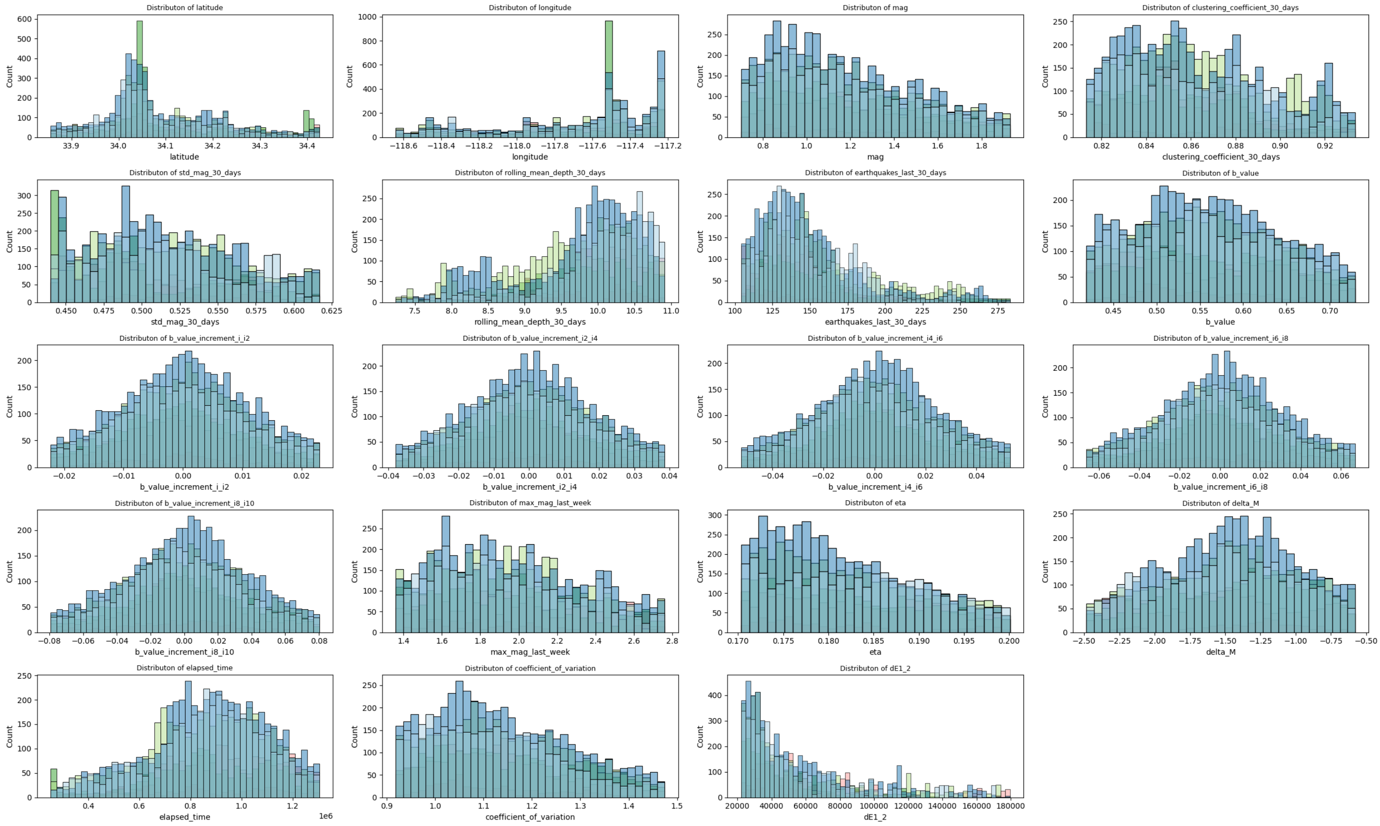
obervation:

Our target variable contains 6 unique classes. Among them, class 2 is the most represented, while class 6 has fewer than 500 rows, making it the least frequent. Overall, the class distribution appears relatively stable, with only a slight imbalance caused by class 6.

Distribution of the independent features

In [12]:

```
import warnings
warnings.filterwarnings('ignore')
fig , ax = plt.subplots(5,4,figsize=(25,15))
ax = ax.flatten()
for i , col in enumerate(df.columns[:-1]):
    upper = np.percentile((df[col]),90)
    lower = np.percentile((df[col]),10)
    sns.histplot(x=df[col],hue=df['class'],palette='Paired',ax=ax[i],legend=False,binran
    ax[i].set_title(f'Distributon of {col}',size=9)
for j in range(i+1, len(ax)):
    fig.delaxes(ax[j])
plt.tight_layout()
```



Distribution Analysis of Features by Class

1. Latitude & Longitude

- Very tight clusters (≈34 latitude, ≈−118 longitude).
- Data is geospatially concentrated in a specific region.
- Strong overlap between classes → not highly discriminative.

2. Magnitude (mag)

- Right-skewed distribution: most events are low-magnitude.
- Heavy class overlap → not a strong standalone feature.

3. Clustering Coefficient (30 days)

- Narrow range (0.82–0.92).
- Almost identical across classes → weak feature for classification.

4. Rolling Mean Depth & Std of Magnitude

- Spread over wider ranges.
- Some variation by class, but separation is limited.

5. Earthquakes in Last 30 Days

- Bell-shaped, ~120–250 range.

- Slight class-level shifts, but no clear separation.

6. b-value & b-value Increments

b_value: centered ~0.5–0.6.

- Increments: centered near 0, normally distributed.
- Overlaps across classes, but may provide subtle signals.

7. Max Magnitude Last Week

- Range ~1.5–2.7.
- More informative than other features, potential class-level separation.

8. Eta & Delta M

- eta: very narrow (0.17–0.20), not useful.
- delta_M: wider spread (−2.5 to −0.5), potentially discriminative.

9. Elapsed Time

- Wide range, slightly skewed.
- Likely needs normalization/log scaling.

10. Coefficient of Variation & ΔE1_2

- ΔE1_2: heavy-tailed with extreme outliers.
- Could dominate training without scaling.
- Coefficient of variation: more stable, but still skewed.

Insights for Modeling

- Imbalance: Class 6 is underrepresented; imbalance will impact learning.
- Feature Scaling: Needed for variables like elapsed_time and ΔE1_2.
- Strong features: max_mag_last_week, delta_M, elapsed_time, ΔE1_2.
- Weak features: eta, clustering_coefficient_30_days, latitude, longitude.

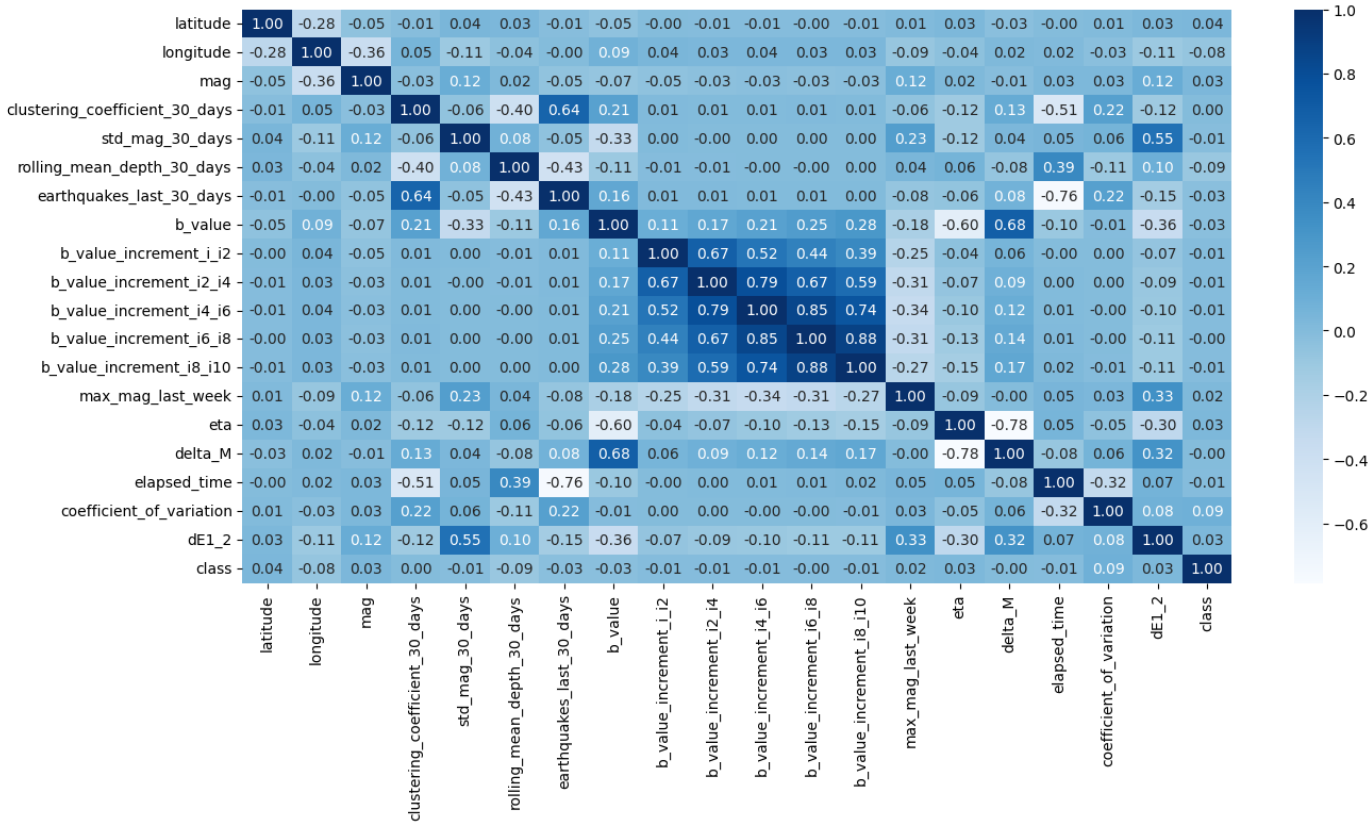
Correlation betwean the features

In [13]:

```
plt.figure(figsize=(15,7))
sns.heatmap(df.corr(method='spearman'),annot=True,fmt='.2f',cmap='Blues')
```

Out[13]:

```
<Axes: >
```



the plot above is a heapmap populated with a matrix of correlation between all the variables , this plot inform us about how the variables of the dataset relate to each others:

- first we can see a high correlation accross the b_value_increments this suggests a existence of multicoliniarity across this varaibles
- the correlation between the target variable `class` and the independent features is very low less than 1%

Building the model

In [14]:

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

In [15]:

```
# we need our target variable to start from 0 not 1 due to the model expected input
encoder = LabelEncoder()
df['class'] = encoder.fit_transform(df['class'])
```

In [16]:

```
df['class'].unique()
```

Out[16]:

```
array([2, 0, 3, 1, 4, 5])
```

In [17]:

```
X ,y = df.iloc[:,:-1] , df.iloc[:,-1]
print('shape of X',X.shape)
print('Shape of y ',y.shape)
```

```
shape of X (22899, 19)
Shape of y  (22899,)
```

In [18]:

```
x_train , x_test , y_train , y_test = train_test_split(X,y,random_state=42,test_size=0.7
```

In [19]:

```
#setting the models
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve ,auc
models = {
    "Rf":RandomForestClassifier(),
    "xgb":XGBClassifier(),
    "lgb":LGBMClassifier()
}
```

In [20]:

```
input_example = x_train[:5]
input_example
```

Out[20]:

	latitude	longitude	mag	clustering_coefficient_30_days	std_mag_30_days	rolling_mean_dep
6906	33.984500	-117.225500	1.21	0.818193	0.449442	
11472	34.501000	-118.682167	1.70	0.782197	0.519339	
12224	34.035500	-117.269000	0.93	0.909526	0.509348	
6720	33.947833	-117.215333	2.34	0.916105	0.586051	
1235	34.492333	-118.063333	1.08	0.873197	0.579047	

In [ ]:

```
accuracy_list =[]
for model_name,model in models.items():
    with mlflow.start_run(run_name=model_name):
        mlflow.set_tag('training_strategy','default models+scaling')

        #defining the model pipeline
        pipeline = Pipeline(steps=[('scaler',StandardScaler()),
                        (model_name,models[model_name])
        ])
        pipeline.fit(x_train,y_train)

        # model evaluation
        y_hat = pipeline.predict(x_test)
        y_prob = pipeline.predict_proba(x_test)
        class_report = classification_report(y_test,y_hat)
        acc = accuracy_score(y_test, y_hat)
        f1 = f1_score(y_test,y_hat,average='macro')
        accuracy_list.append({
            'model':model_name,
            'accuracy_score':acc,
        })


        #login the metrics and the models
        mlflow.log_param("model_type",model_name)
```

```
        mlflow.log_metrics({"accuracy":acc,
                            "f1_score":f1})
        mlflow.sklearn.log_model(sk_model=pipeline,
                                name =model_name,
                                input_example=input_example,
                                metadata={"description":"iniale models"})

        print(model_name)
        print('model accuracy :',acc)
        print(class_report)
        print('-'*25)
```

In [ ]:

```
accuracy = pd.DataFrame(accuracy_list)
accuracy
```

In [ ]:

```
sns.barplot(accuracy,x='model',y="accuracy_score",width=.6)
```

we can see that our models performing very well the with a high accuracy more than 90% and a good precision and recall, but the random forest outperformance the other models with an accuracy of 93% and a good micro avg and weighed avg, so our next goal is to tune the hyperparams of the random forst model such as depth, n-jobs,number of splits .....

Hyperparams tunning

In [25]:

```
from sklearn.model_selection import GridSearchCV , RandomizedSearchCV

pipeline = Pipeline(steps=[('scaler',StandardScaler()),
                    ('Rf',RandomForestClassifier())
                           ])


params = {
    'Rf__n_estimators':[150,200,250],# Number of trees in the forest
    'Rf__max_depth':[10,15,20], # controle the depth of each tree
    'Rf__min_samples_split':[2,5], #Specifies the minimum number of samples required to
    'Rf__min_samples_leaf':[1,2],
    'Rf__max_features':['sqrt','log2']# controles the umber of features to consider when
}

grid_search =RandomizedSearchCV(pipeline,params)
grid_search.fit(x_train,y_train)

print("best Params",grid_search.best_params_)
print("best Models",grid_search.best_estimator_)
```

```
best Params {'Rf__n_estimators': 150, 'Rf__min_samples_split': 2, 'Rf__min_samples_lea
f': 1, 'Rf__max_features': 'sqrt', 'Rf__max_depth': 20}
best Models Pipeline(steps=[('scaler', StandardScaler()),
                ('Rf', RandomForestClassifier(max_depth=20, n_estimators=150))])
```

Testing the model

In [26]:

```
pipeline_v1 = grid_search.best_estimator_
pipeline_v1
```

Out[26]:

```
▸ Pipeline
    ▸ StandardScaler
    ▸ RandomForestClassifier
```

In [ ]:

```
with mlflow.start_run():
    #Training and Predicting
    pipeline_v1.fit(x_train,y_train)
    preds = pipeline_v1.predict(x_test)
    #Performance metrics
    acc = accuracy_score(y_test,preds)
    f1 = f1_score(y_test,preds,average='macro')

    #logging the model hyperparams
    mlflow.log_params(pipeline_v1.get_params())

    #logging the performance metrics
    mlflow.log_metrics({"accuracy":acc,
                        "f1_score":f1})

    #logging the model
    mlflow.sklearn.log_model(sk_model=pipeline_v1,
                             name ="Rf_model_tunned",
                             input_example=input_example,
                             metadata={"description":"fine tunned  models"})


    class_report_v1 = classification_report(preds,y_test)
    print(class_report_v1)
```

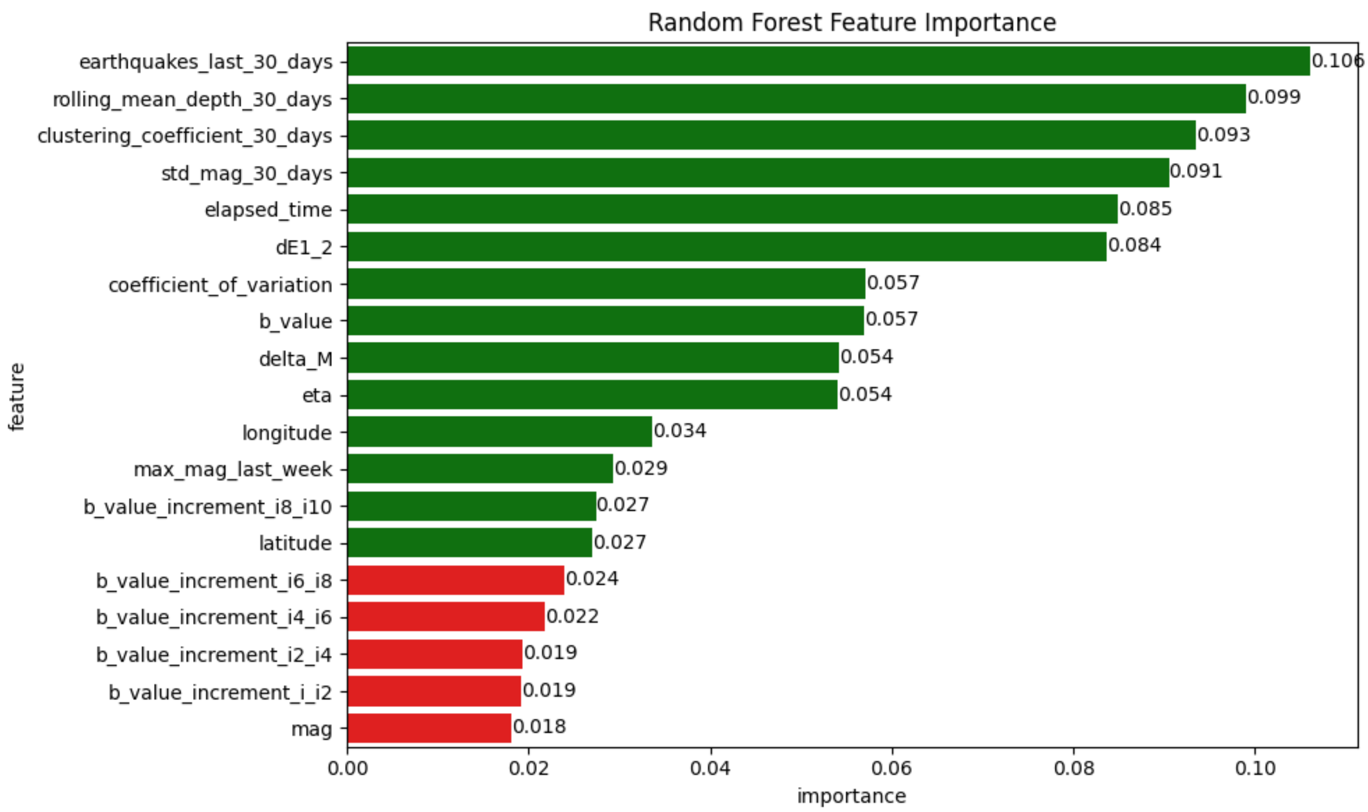
```
Downloading artifacts: 100%|██████████| 7/7 [00:00<00:00, 388.87it/s]
              precision    recall  f1-score   support

           0       0.93      0.90      0.91      3362
           1       0.96      0.86      0.91      4965
           2       0.93      0.95      0.94      3582
           3       0.85      0.97      0.91      2114
           4       0.85      0.95      0.90      1658
           5       0.85      1.00      0.92       349

    accuracy                           0.92     16030
   macro avg       0.90      0.94      0.91     16030
weighted avg       0.92      0.92      0.92     16030

🏃 View run bold-elk-510 at: https://dbc-5270f342-fa02.cloud.databricks.com/ml/experimen
ts/4494213887472809/runs/a6eb862021a143338ad11b91f5d65020
🧪 View experiment at: https://dbc-5270f342-fa02.cloud.databricks.com/ml/experiments/449
4213887472809
```

after hyperparams tunning the Random forest Parameters we got a F1 score above 91% across all the classes this suggests our model have reached a good balance between good Quality (Precision) and a good Covrage of real cases (Recall)

Feature importance

In [64]:

```
importances = pipeline_v1.named_steps['Rf'].feature_importances_
columns = df.columns[:-1]
feature_impor = pd.DataFrame({
    "feature":columns,
    "importance":importances
}).sort_values(by="importance",ascending=False)
color = ['r' if x <= 0.025 else 'g' for x in feature_impor['importance']]
plt.figure(figsize=(10,6))
ax = sns.barplot(x="importance", y="feature", data=feature_impor, palette=color)
[ax.bar_label(c,fmt='%.3f') for c in ax.containers]
plt.title("Random Forest Feature Importance")
plt.tight_layout()
plt.show()
```



this plot shows the importance of our data features or independent varaiables and we can see that the mag and the b_value_increment have a low importance so we will drop those variable to see if the model performance will drop down or get better

In [76]:

```
x_train_1 = x_train.drop(feature_impor["feature"][14:],axis=1)
x_test_1 = x_test.drop(feature_impor["feature"][14:],axis=1)
x_test_1.shape , x_train_1.shape
```

Out[76]:

```
((16030, 14), (6869, 14))
```

In [ ]:

```
input_example.drop(feature_impor["feature"][14:],axis=1,inplace=True)
```

In [84]:

```
with mlflow.start_run():
    #Training and Predicting
    pipeline_v2 = Pipeline(steps=[('scaler', StandardScaler()),
                ('Rf', RandomForestClassifier(max_depth=20, n_estimators=150))])


    pipeline_v2.fit(x_train_1,y_train)
    preds = pipeline_v2.predict(x_test_1)
    #Performance metrics
    acc = accuracy_score(y_test,preds)
    f1 = f1_score(y_test,preds,average='macro')

    #logging the model hyperparams
    mlflow.log_params(pipeline_v2.get_params())

    #logging the performance metrics
    mlflow.log_metrics({"accuracy":acc,
                        "f1_score":f1})

    #logging the model
    mlflow.sklearn.log_model(sk_model=pipeline_v2,
                             artifact_path ="Rf_model_tunned_v2",
                             input_example=input_example,
                             metadata={"description":"fine tunned  models with only t

    class_report_v1 = classification_report(preds,y_test)
    print(class_report_v1)
```

```
2025/09/29 15:35:29 WARNING mlflow.models.model: `artifact_path` is deprecated. Please u
se `name` instead.
Downloading artifacts: 100%|██████████| 7/7 [00:00<00:00, 393.08it/s]
              precision    recall  f1-score   support

           0       0.94      0.92      0.93      3321
           1       0.96      0.88      0.92      4874
           2       0.94      0.95      0.94      3603
           3       0.87      0.98      0.92      2130
           4       0.89      0.96      0.92      1731
           5       0.90      0.99      0.95       371

    accuracy                           0.93     16030
   macro avg       0.92      0.95      0.93     16030
weighted avg       0.93      0.93      0.93     16030

🏃 View run grandiose-croc-97 at: https://dbc-5270f342-fa02.cloud.databricks.com/ml/expe
riments/4494213887472809/runs/e8f4c2ea742c43b7abc2ba791c38f7c0
🧪 View experiment at: https://dbc-5270f342-fa02.cloud.databricks.com/ml/experiments/449
4213887472809
```

In [88]:

```
! jupyter nbconvert la-and-california-earthquake-classification.ipynb --to markdown
```

```
[NbConvertApp] Converting notebook la-and-california-earthquake-classification.ipynb to
markdown
```

```
[NbConvertApp] Support files will be in la-and-california-earthquake-classification_file
s\
[NbConvertApp] Making directory la-and-california-earthquake-classification_files
[NbConvertApp] Writing 46800 bytes to la-and-california-earthquake-classification.md
```