Upload New File
khalil hennara authored 2 weeks ago

f8307599

Code owners Assign users and groups as approvers for specific file changes. Learn more.

Training_Random_forest.ipynb 666.92 KiB

```
import tensorflow as tf

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

from sklearn import svm
from sklearn import ensemble

from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.compose import ColumnTransformer

from sklearn.metrics import accuracy_score,f1_score,recall_score,confusion_matrix,classification_report, roc_curve,roc
_auc_score,auc,make_scorer

from sklearn.model_selection import train_test_split

from sklearn.decomposition import PCA
import joblib
```

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

```
Mounted at content/
```

```
%cd /content/content/MyDrive/ML_DATA_2023
```

```
/content/content/MyDrive/ML_DATA_2023
```

```
!ls
```

```
'Copy of ECO_code protocole avec code condition.xlsx'
'Copy of ECO_code protocole avec liste des codes étapes.xlsx'
'Copy of ECO_formule avec code protocole.xlsx'
'Copy of ECO_formule avec element tech.xlsx'
'Copy of ECO_formule avec liste INCI.xlsx'
'Copy of ECO_formule avec liste MP.xlsx'
'Copy of ECO_liste des codes étapes avec les conditions.xlsx'
'Copy of ECO_liste des codes opérations.xlsx'
'Copy of ECO_liste des conditions.xlsx'
'Copy of ECO_liste des formules avec désignation.xlsx'
'Copy of ECO_MP avec liste INCI.xlsx'
'Copy of MCD ML_Data.gslides'
 Formula_last_version_with_all_ingredient.csv
 Formula_last_version_without_zero_ingredients.csv
 formula_with_full_information.csv
 formula_with_label.csv
 Label_last_version.csv
 new_data_with_label.csv
 new_formula.csv
 Occitan_raw_materials.csv
 svm_without_weight.joblib
```

```
# this function will make the display of the information much easier.
def train_and_test(classifier, X_train, X_test, y_train, y_test,training=True):
    """
    this function will display the confusion matrix for us with different colors
    params:
      classifier: any classifier from sklearn, where the classifier should have the fit and predict_proba method.
      X_train: the training set.
      X_test :the test set.
      y_train,y_test: the train and test labels respectively.
    """
    if training: classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)
    y_score = classifier.predict_proba(X_test)[:,1]


    accuracy = accuracy_score(y_pred,y_test)
    classification_rep = classification_report(y_pred,y_test)
    con_matrix = confusion_matrix(y_pred,y_test)
    fpr, tpr, _ = roc_curve(y_test,y_score)
    roc_auc = auc(fpr,tpr)



    # ploting confustion matrix
    plt.figure(figsize=(12,12))
    plt.subplot(2,1,1)
    sns.heatmap(con_matrix, annot=True, fmt="d")#, xticklabels=target_names, yticklabels=target_names)
    plt.ylabel("Real value")
    plt.xlabel("Predicted value")
    plt.show()

    # print scores
    print ("accuracy  score: {} %".format(accuracy))
    print ("auc  score: {} ".format(roc_auc))
    print(classification_rep)

    # print ROC curve
    plt.figure()
    plt.plot(fpr, tpr, color='darkorange',
         lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

Training Random Forest without removing zero ingredient.

```
from sklearn.model_selection import GridSearchCV
```

```
forest_1 = ensemble.RandomForestClassifier(n_estimators=1000,n_jobs=-1,verbose=1)
```

```
formulas= pd.read_csv("Formula_last_version_with_all_ingredient.csv",index_col='Formula code')
label=pd.read_csv("Label_last_version.csv",index_col="Formula code")
```

```
train_set,test_set,train_label,test_label=train_test_split(formulas,label,test_size=0.3,random_state=42)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-80c75d53560c> in <cell line: 1>()
----> 1 train_set,test_set,train_label,test_label=train_test_split(formulas,label,test_size=0.3,random_s
```

```
params = {"max_depth":[10,30,50,60,100],"oob_score":[True,False],"min_samples_split":[5,10,20,40,50]}
```

```
scoring = make_scorer(f1_score)
```

```
grid_search = GridSearchCV(forest_1,param_grid=params,scoring=scoring)
```

```
grid_search.fit(train_set,train_label)
```

```
grid_search.best_params_
```

```
{'max_depth': 10, 'min_samples_split': 5, 'oob_score': True}
```

```
best_estimator=grid_search.best_estimator_
```

```
best_estimator
```

```
RandomForestClassifier(max_depth=10, min_samples_split=5, n_estimators=1000,
                       n_jobs=-1, oob_score=True, verbose=1)
```

```
encoder=LabelEncoder()
encoder.fit(label)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_label.py:99: DataConversionWarning: A column-vector y wa
s passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
LabelEncoder()
```

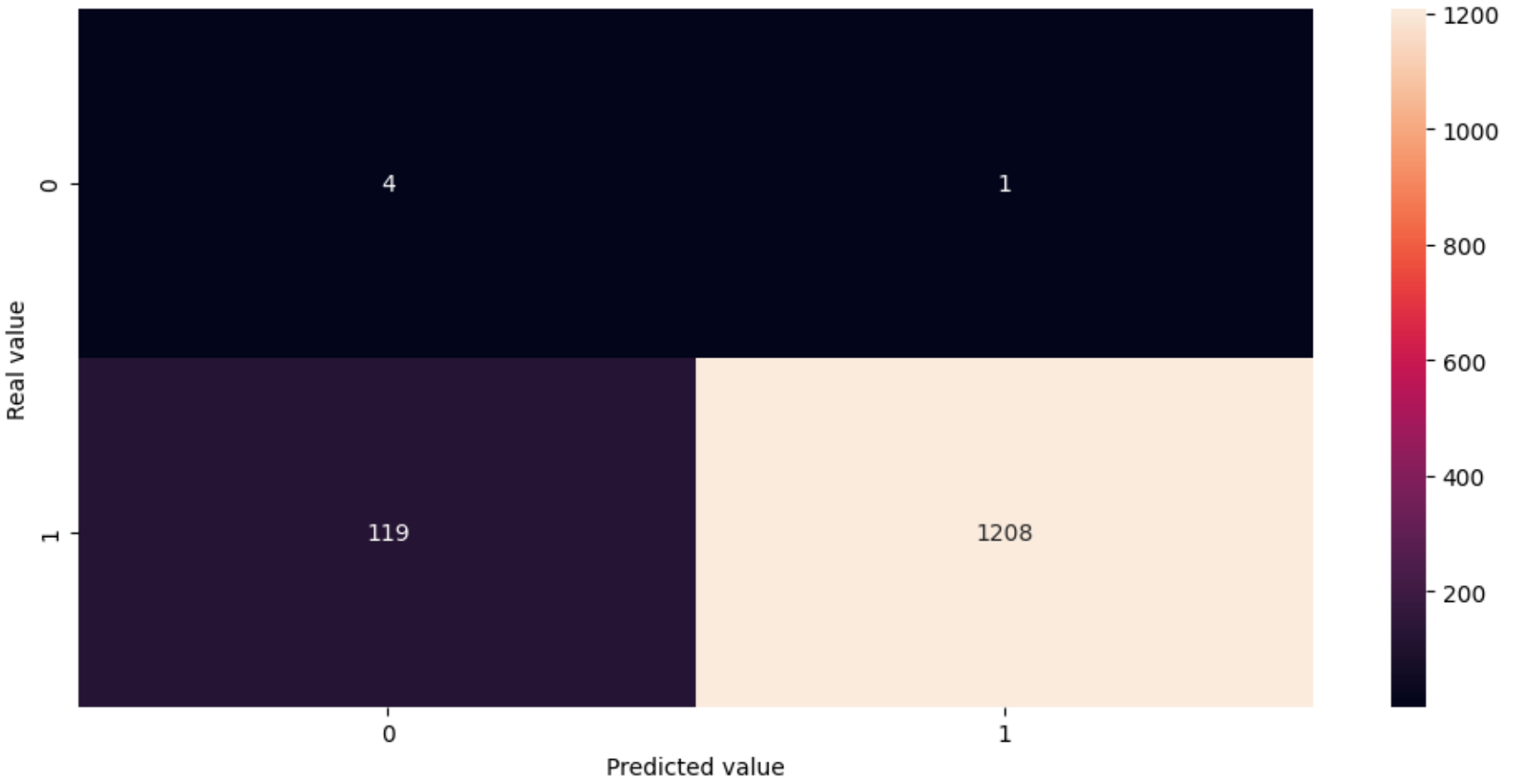
```
en_train_label = encoder.transform(train_label)
en_test_label  = encoder.transform(test_label)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_label.py:134: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_label.py:134: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
```

```
train_and_test(best_estimator,train_set,test_set,en_train_label,en_test_label)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:    0.8s
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed:    2.7s
[Parallel(n_jobs=-1)]: Done 446 tasks      | elapsed:    5.5s
[Parallel(n_jobs=-1)]: Done 796 tasks      | elapsed:    8.6s
```

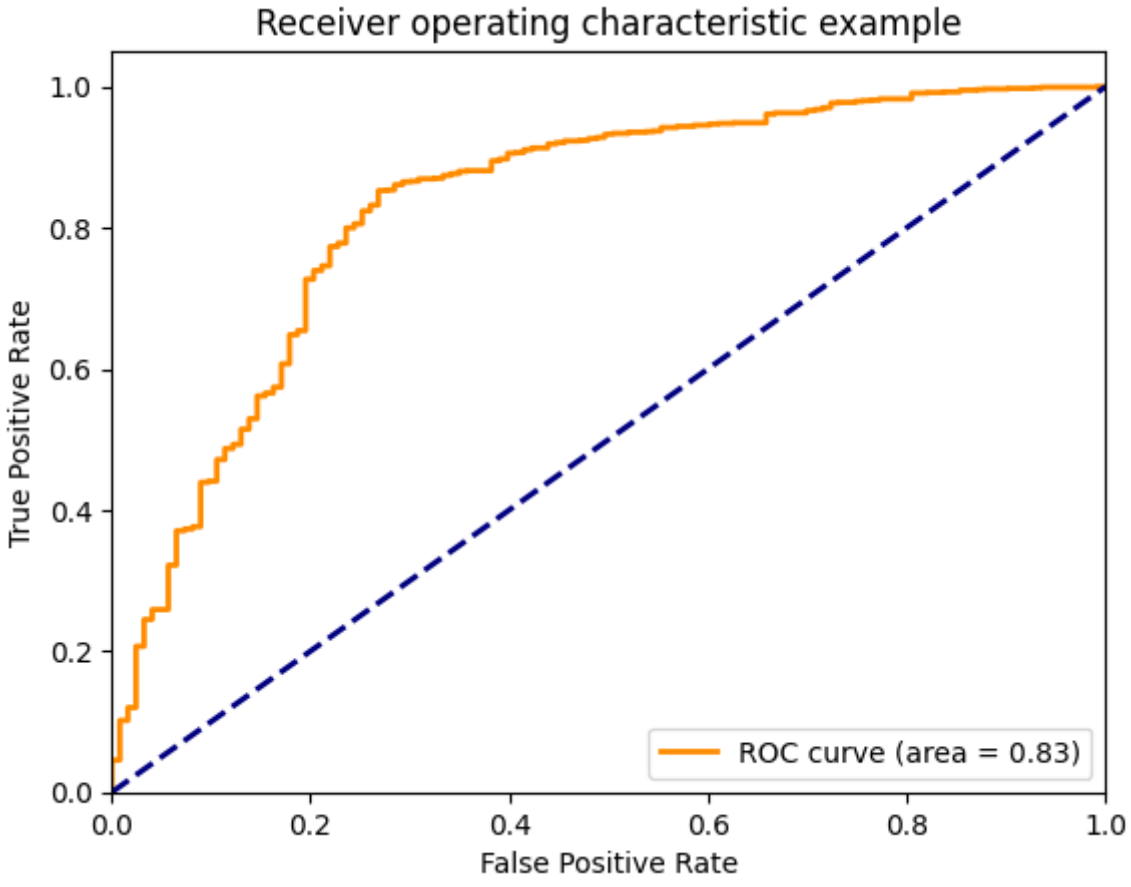
```
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.3s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.4s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.3s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.4s finished
```



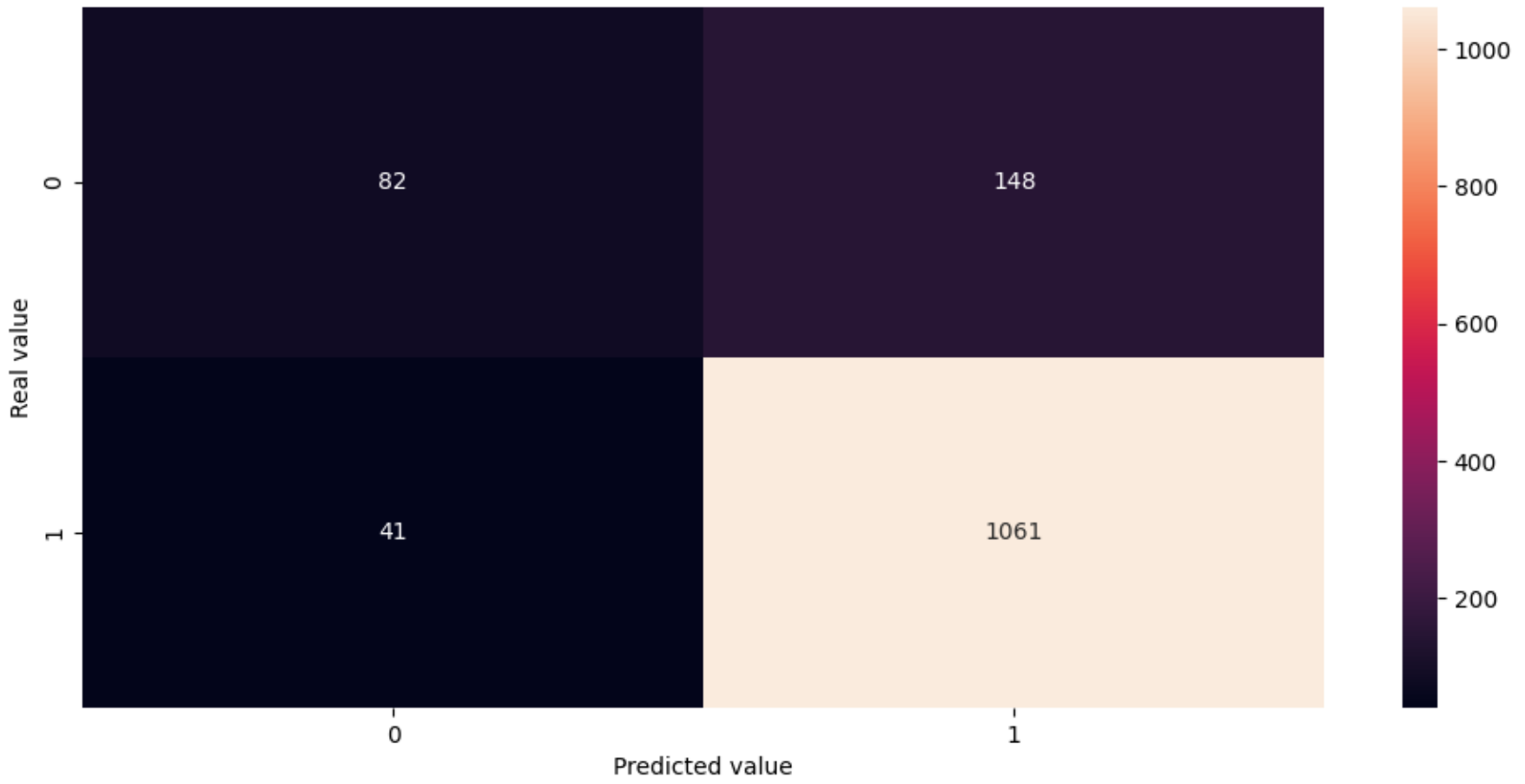

```
accuracy  score: 0.9099099099099099 %
auc  score: 0.8288883509182484
              precision    recall  f1-score   support

           0       0.03      0.80      0.06         5
           1       1.00      0.91      0.95      1327

    accuracy                           0.91      1332
   macro avg       0.52      0.86      0.51      1332
weighted avg       1.00      0.91      0.95      1332
```



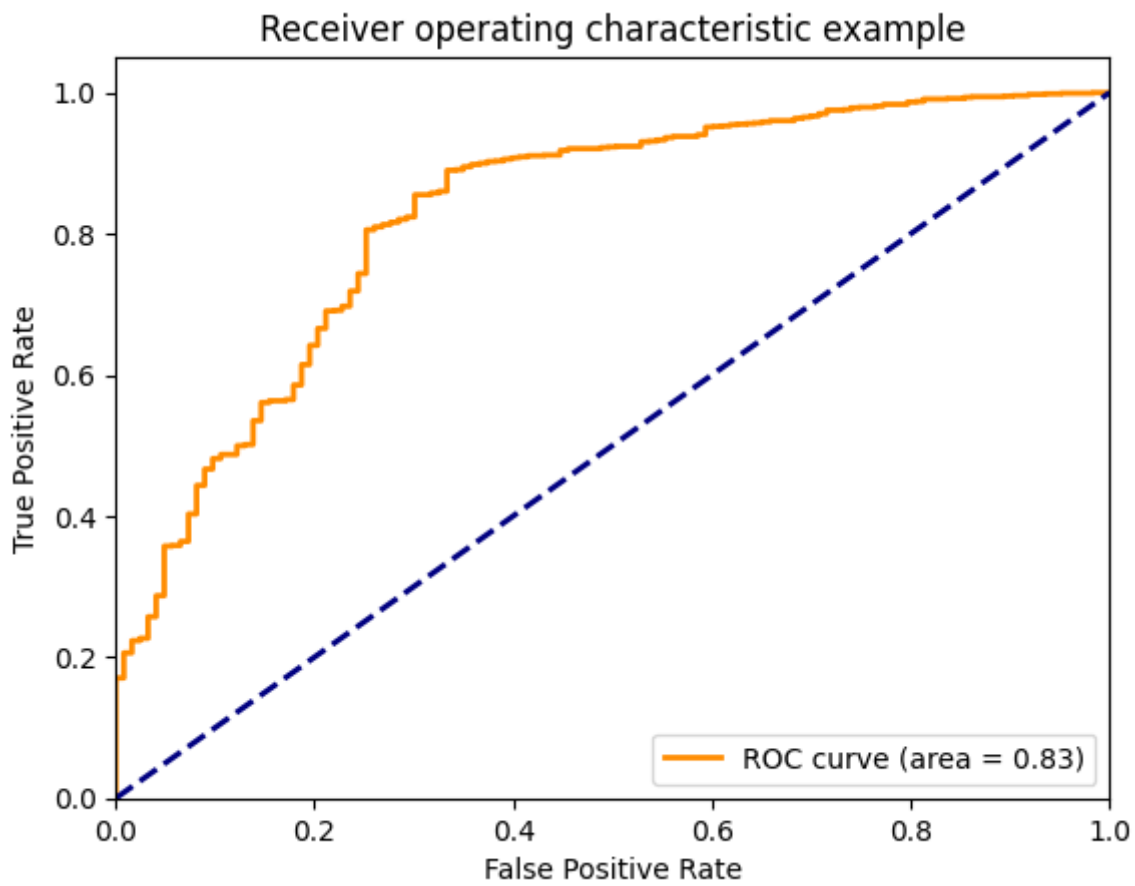
```
train_and_test(balanced_forest,train_set,test_set,en_train_label,en_test_label)
```



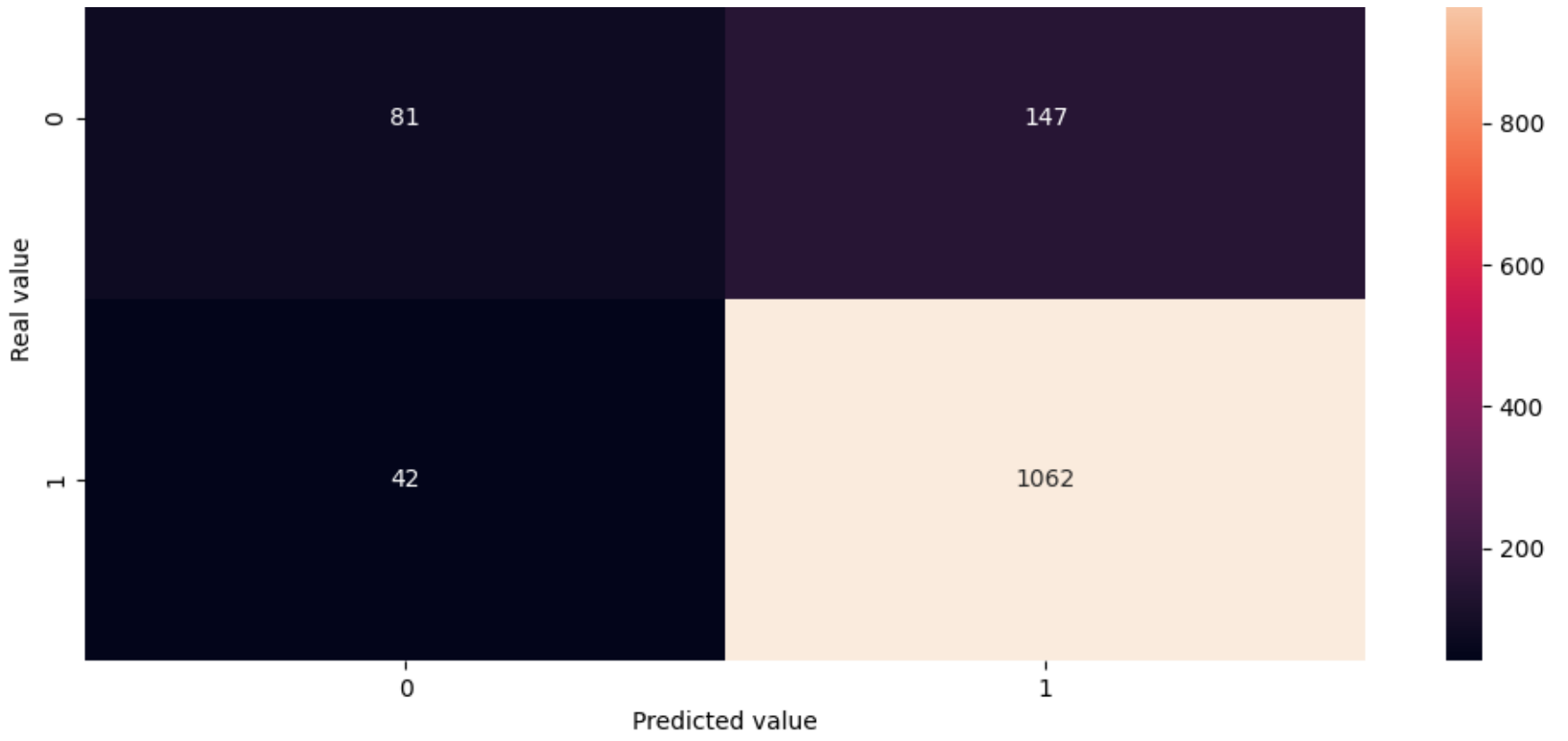

```
accuracy  score: 0.8581081081081081 %
auc  score: 0.8274694533545831
              precision    recall  f1-score   support

           0       0.67      0.36      0.46       230
           1       0.88      0.96      0.92      1102

    accuracy                           0.86      1332
   macro avg       0.77      0.66      0.69      1332
weighted avg       0.84      0.86      0.84      1332
```



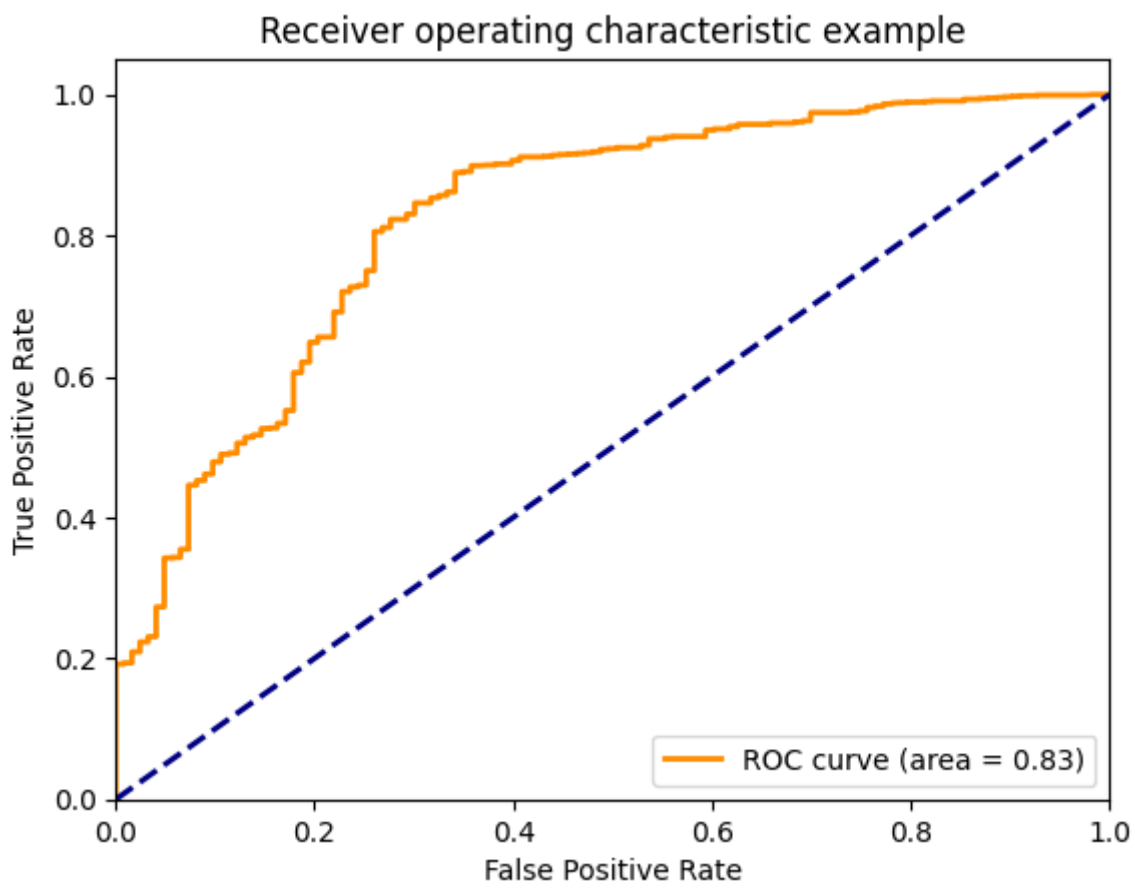

```
train_and_test(balanced_forest,train_set,test_set,en_train_label,en_test_label)
```




```
accuracy  score: 0.8581081081081081 %
auc  score: 0.8256605270767348
              precision    recall  f1-score   support

           0       0.66      0.36      0.46       228
           1       0.88      0.96      0.92      1104

    accuracy                           0.86      1332
   macro avg       0.77      0.66      0.69      1332
weighted avg       0.84      0.86      0.84      1332
```



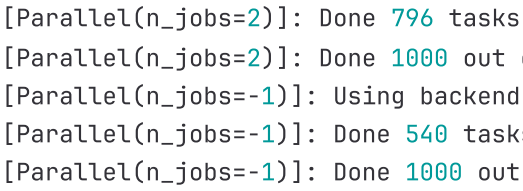

```
params['class_weight']=[{0:0.7,1:0.3},{0:0.8,1:0.2}]
```

```
params
```

```
{'max_depth': [10, 30, 50, 60, 100],
 'oob_score': [True, False],
 'min_samples_split': [5, 10, 20, 40, 50],
 'class_weight': [{0: 0.7, 1: 0.3}, {0: 0.8, 1: 0.2}]}
```

```
grid_search_2 = GridSearchCV(forest_1,cv=3,param_grid=params,scoring=scoring)
```

```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.3s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 540 tasks      | elapsed:    2.9s
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:    5.2s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.4s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.5s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 540 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:    5.2s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.3s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 540 tasks      | elapsed:    5.1s
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:    7.4s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.3s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 540 tasks      | elapsed:    3.0s
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:    5.4s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.4s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 540 tasks      | elapsed:    5.1s
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:    7.3s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.3s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:    0.5s
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed:    2.2s
[Parallel(n_jobs=-1)]: Done 446 tasks      | elapsed:    4.6s
[Parallel(n_jobs=-1)]: Done 796 tasks      | elapsed:    9.1s
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:   12.4s finished
```

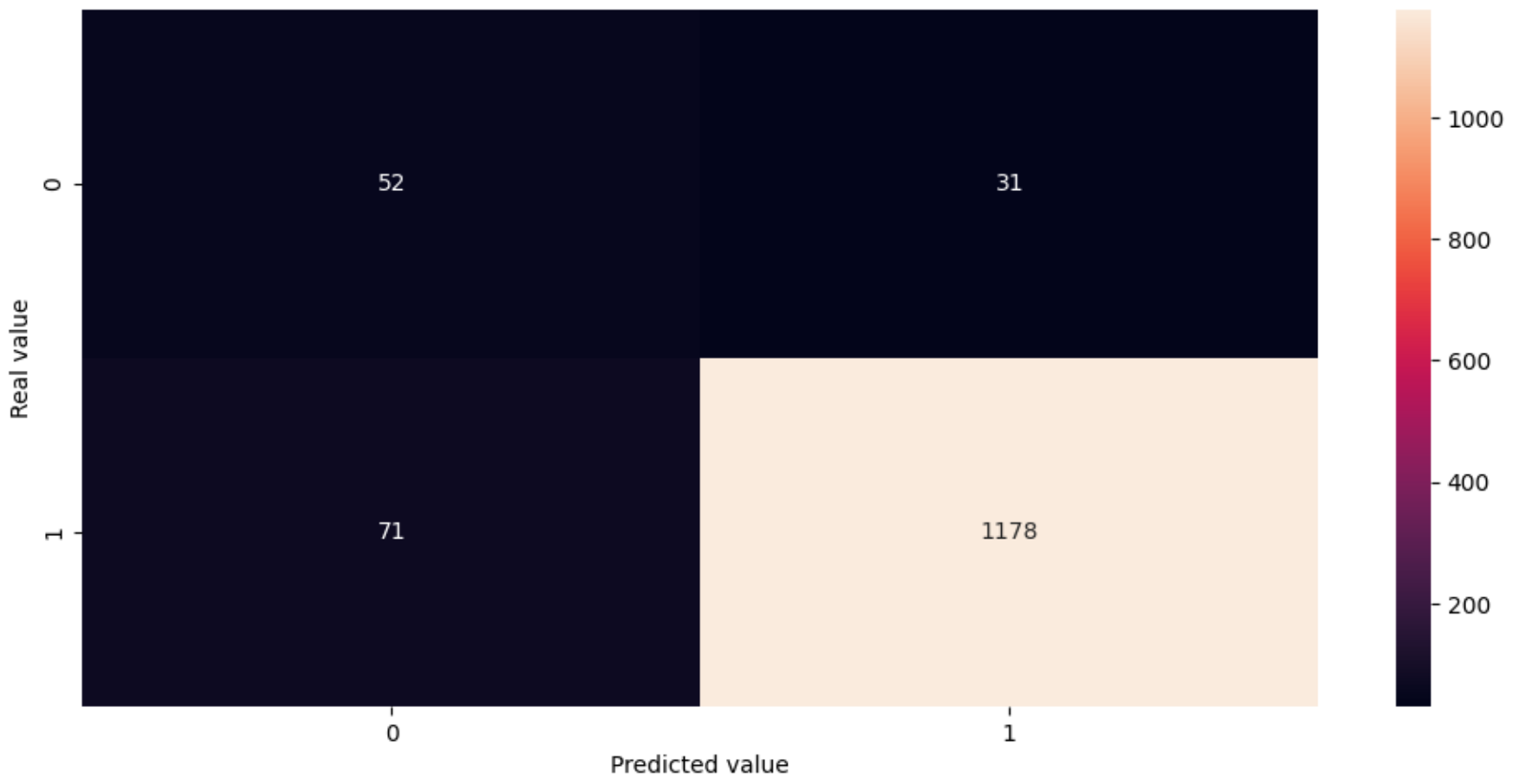



```
grid_search_2.best_params_
```

```
{'class_weight': {0: 0.7, 1: 0.3},
 'max_depth': 50,
 'min_samples_split': 5,
 'oob_score': False}
```

```
train_and_test(best_balances_estimator,train_set,test_set,en_train_label,en_test_label)
```

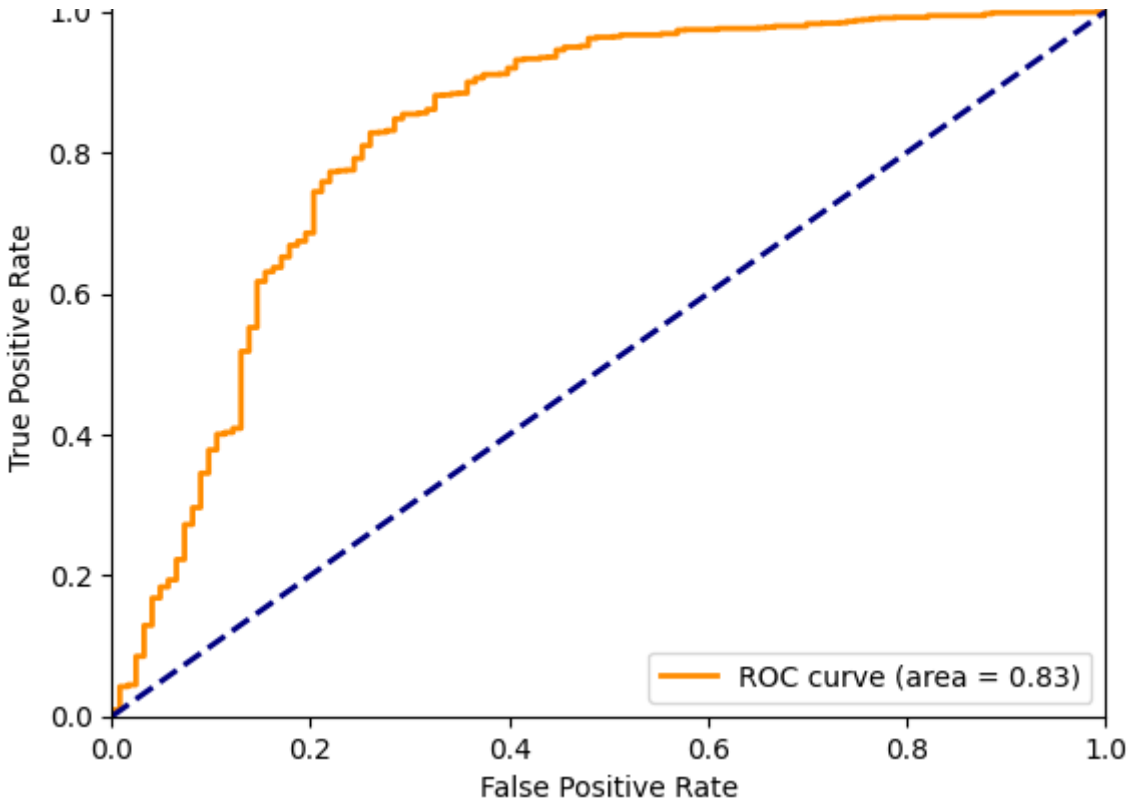
```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed:    6.3s
[Parallel(n_jobs=-1)]: Done 446 tasks      | elapsed:   11.6s
[Parallel(n_jobs=-1)]: Done 796 tasks      | elapsed:   21.4s
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:   24.9s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.4s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.5s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 196 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 446 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 796 tasks      | elapsed:    0.4s
[Parallel(n_jobs=2)]: Done 1000 out of 1000 | elapsed:    0.5s finished
```




```
accuracy  score: 0.9234234234234234 %
auc  score: 0.8281015688568796
              precision    recall  f1-score   support

           0       0.42      0.63      0.50        83
           1       0.97      0.94      0.96      1249

    accuracy                           0.92      1332
   macro avg       0.70      0.78      0.73      1332
weighted avg       0.94      0.92      0.93      1332
```

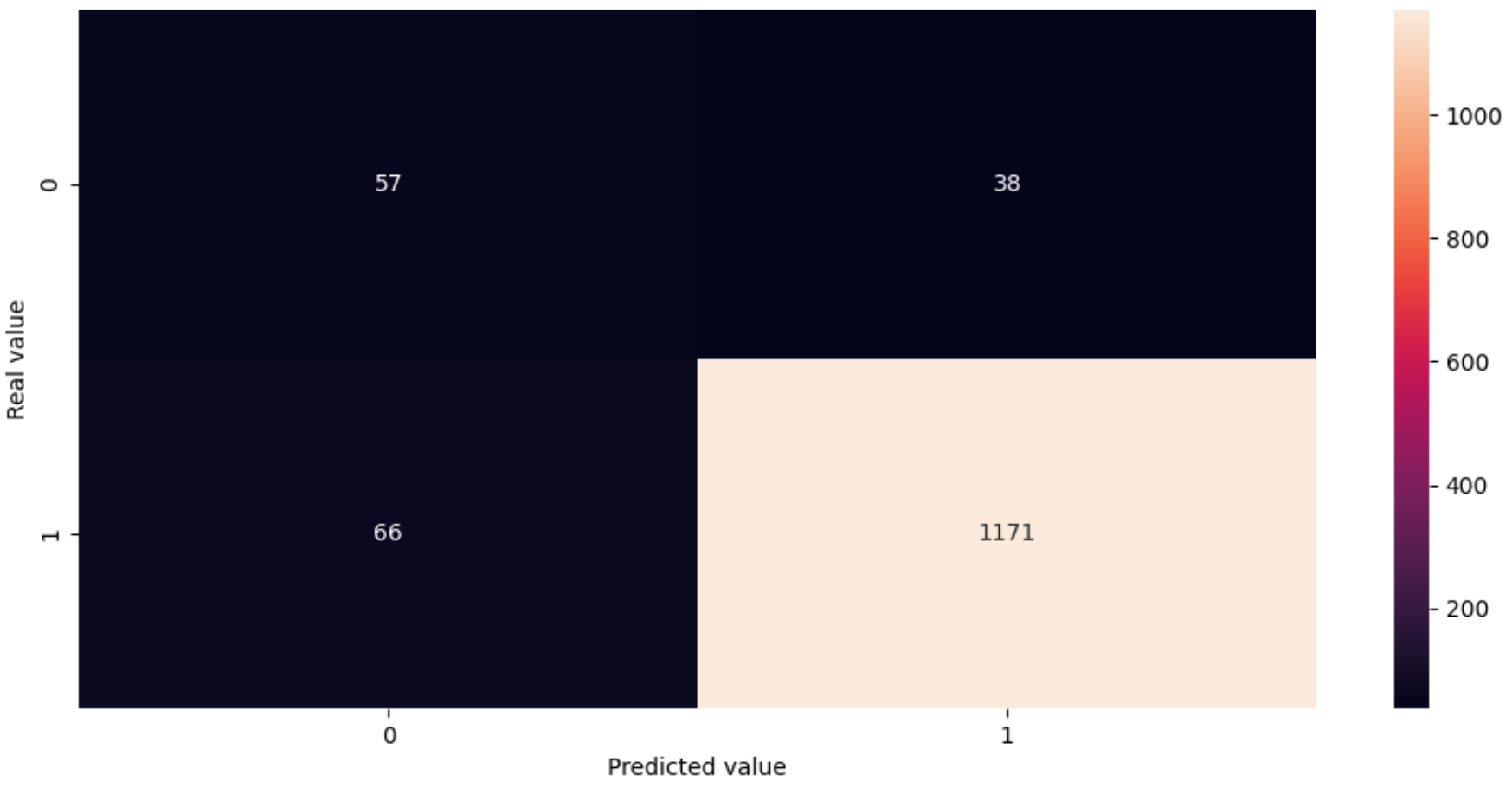



```
balanced_forest
```

```
                                RandomForestClassifier
RandomForestClassifier(class_weight='balanced', max_depth=10,
                       min_samples_split=5, n_estimators=1000, oob_score=True)
```

```
random_forest=ensemble.RandomForestClassifier(min_samples_split=5,n_estimators=1000,class_weight={0:0.8,1:0.2},max_depth=30)
```

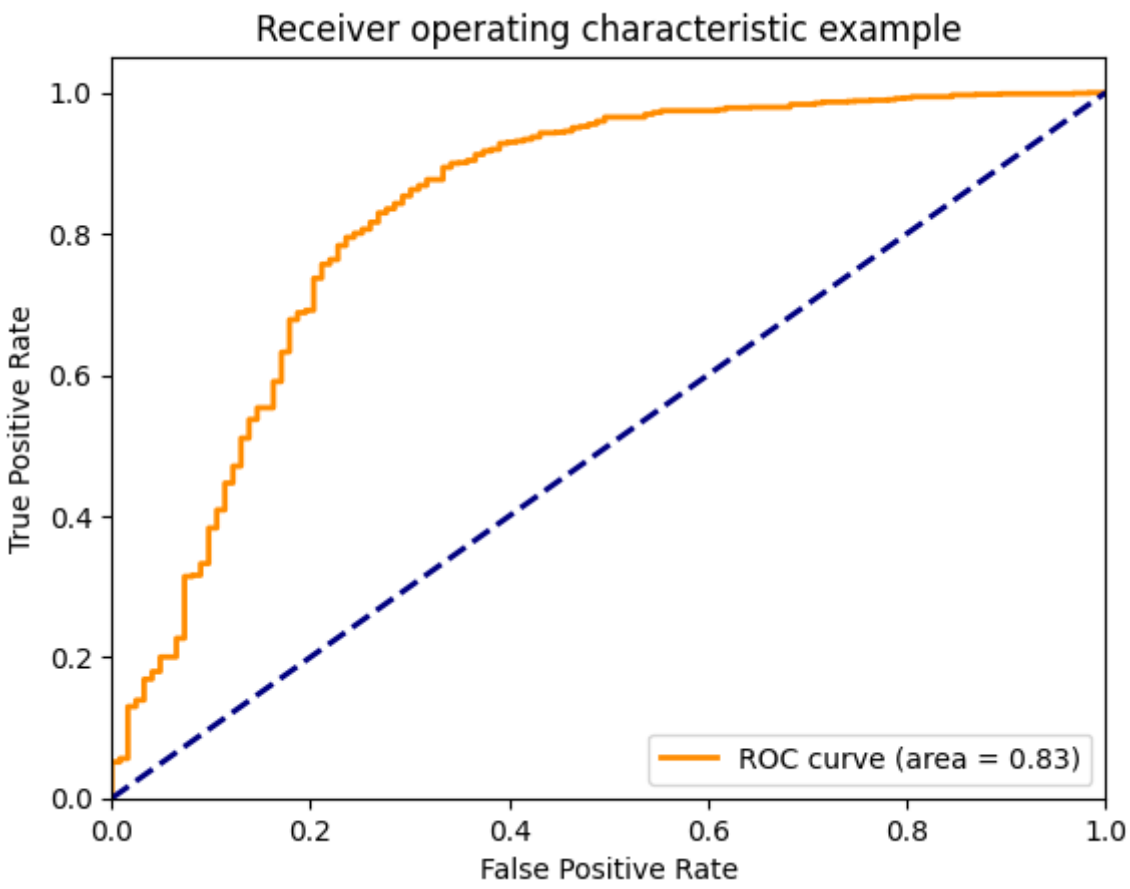
```
train_and_test(random_forest,train_set,test_set,en_train_label,en_test_label)
```




```
accuracy  score: 0.9219219219219219 %
auc  score: 0.8317866677426079
              precision    recall  f1-score   support

           0       0.46      0.60      0.52        95
           1       0.97      0.95      0.96      1237

    accuracy                           0.92      1332
```



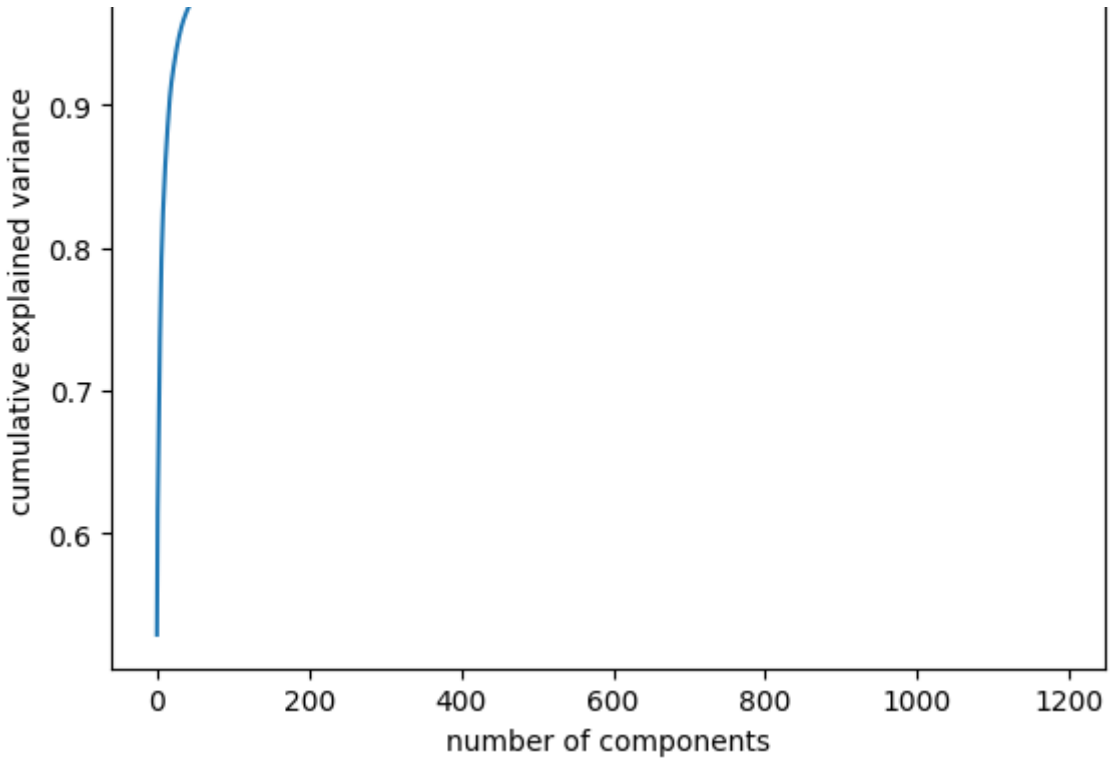

```
best_balances_estimator.get_params()
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': {0: 0.7, 1: 0.3},
 'criterion': 'gini',
 'max_depth': 50,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 1000,
 'n_jobs': -1,
 'oob_score': False,
 'random_state': None,
 'verbose': 1,
 'warm_start': False}
```

```
from sklearn.utils.validation import joblib
joblib.dump(best_balances_estimator,"/content/content/MyDrive/ML_DATA_2023/RandomForestModel.joblib")
```

```
['/content/content/MyDrive/ML_DATA_2023/RandomForestModel.joblib']
```

```
pca = PCA().fit(train_set)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```

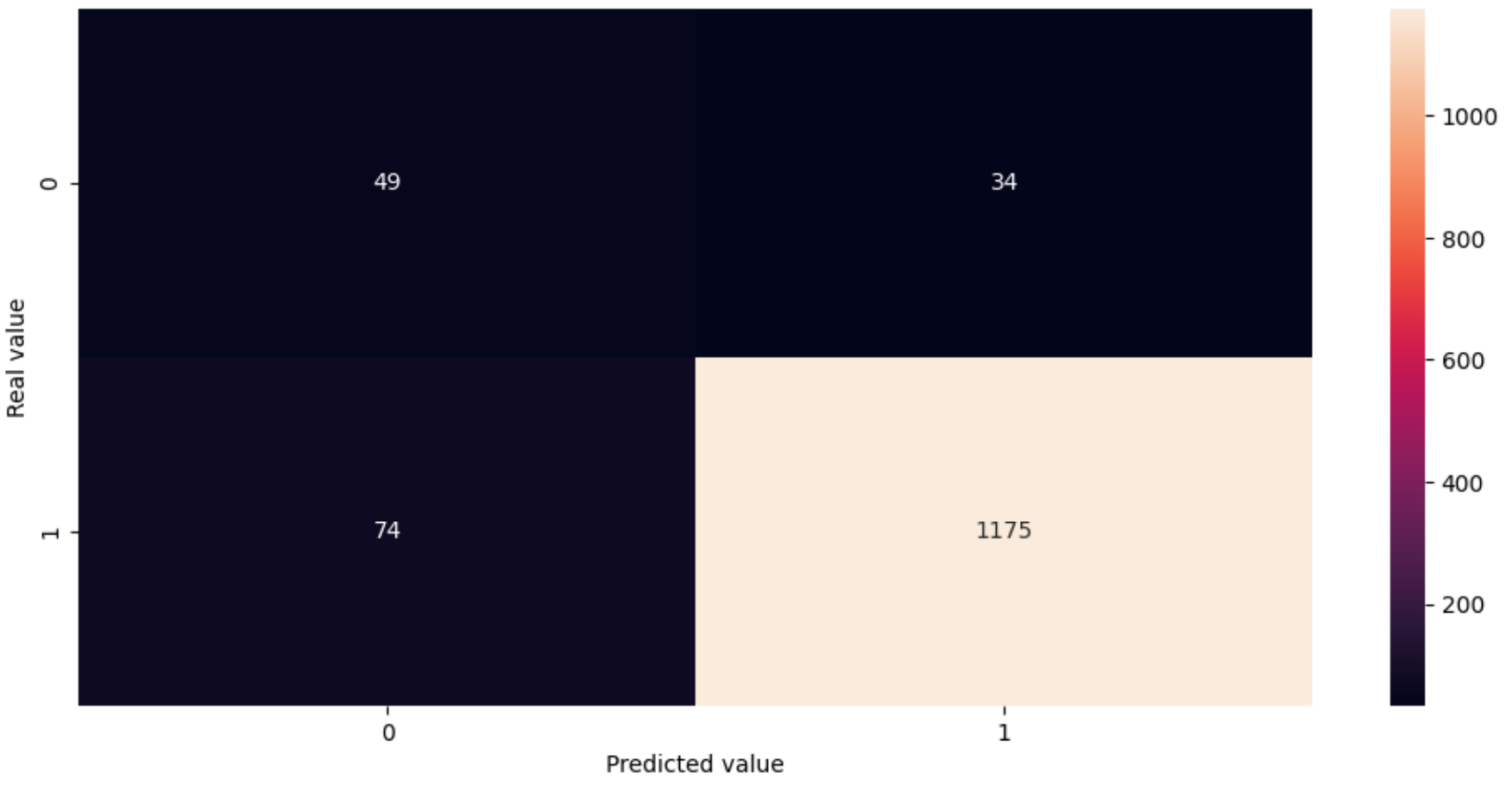



```
pca=PCA(n_components=150)
```

```
en_train_set = pca.fit_transform(train_set)
en_test_set  = pca.transform(test_set)
```

```
random_forest=ensemble.RandomForestClassifier(min_samples_split=5,n_estimators=1000,class_weight={0:0.9,1:0.1},max_depth=15)
```

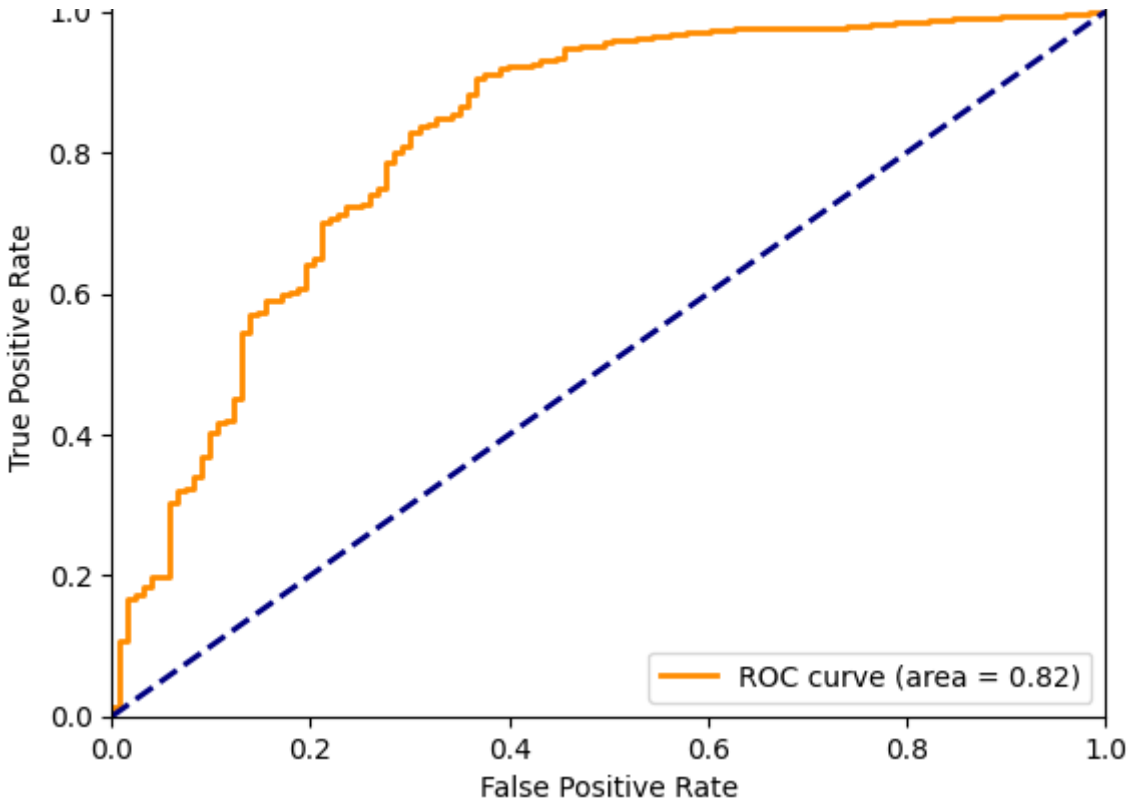
```
train_and_test(random_forest,en_train_set,en_test_set,en_train_label,en_test_label)
```




```
accuracy  score: 0.918918918918919 %
auc  score: 0.8212794286751799
              precision    recall  f1-score   support

           0       0.40      0.59      0.48        83
           1       0.97      0.94      0.96      1249

    accuracy                           0.92      1332
   macro avg       0.69      0.77      0.72      1332
weighted avg       0.94      0.92      0.93      1332
```




```
joblib.dump(random_forest,"/content/content/MyDrive/ML_DATA_2023/RandomForest_150PCA.joblib")
```

```
['/content/content/MyDrive/ML_DATA_2023/RandomForest_150PCA.joblib']
```

```
model=joblib.load("/content/content/MyDrive/ML_DATA_2023/Model/RandomForestModel.joblib")
```

In [1]:

```
!pip install boto3
```

Out [1]:

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting boto3
  Downloading boto3-1.26.148-py3-none-any.whl (135 kB)
 [2K      [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ [0m  [32m135.6/135.6 kB [0m  [31m4.4 MB/s [0m eta  [36m0:00:0
0 [0m
 [?25hCollecting botocore<1.30.0,>=1.29.148 (from boto3)
  Downloading botocore-1.29.148-py3-none-any.whl (10.8 MB)
 [2K      [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ [0m  [32m10.8/10.8 MB [0m  [31m73.3 MB/s [0m eta  [36m0:00:0
0 [0m
 [?25hCollecting jmespath<2.0.0,>=0.7.1 (from boto3)
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting s3transfer<0.7.0,>=0.6.0 (from boto3)
  Downloading s3transfer-0.6.1-py3-none-any.whl (79 kB)
 [2K      [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ [0m  [32m79.8/79.8 kB [0m  [31m5.8 MB/s [0m eta  [36m0:00:00 [0
m
 [?25hRequirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.10/dist-packages (from boto
core<1.30.0,>=1.29.148->boto3) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/local/lib/python3.10/dist-packages (from botocore<1.30.0,
>=1.29.148->boto3) (1.26.15)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil<3.0.0,>=2.1->
botocore<1.30.0,>=1.29.148->boto3) (1.16.0)
Installing collected packages: jmespath, botocore, s3transfer, boto3
Successfully installed boto3-1.26.148 botocore-1.29.148 jmespath-1.0.1 s3transfer-0.6.1
```

In [2]:

```
import boto3

# Create a session using your AWS credentials
session = boto3.Session(
    aws_access_key_id='AKIA37GH64LDTCUMKQGA',
    aws_secret_access_key='urts3DMgowMAgX1n7tGpLM2SjxvZ/G8xBPsMBvdc'
)

# Create an S3 client
s3 = session.client('s3')

s3_resource =session.resource('s3')
```

```
file_name = '/content/content/MyDrive/ML_DATA_2023/Model/RandomForestModel.joblib'
object_key = 'RandomForestModel.joblib'

# s3.upload_file(file_name, bucket_name, object_key)

# # Download a file from S3
# s3.download_file(bucket_name, object_key, 'local_file.txt')

# # Delete a file from S3
# s3.delete_object(Bucket=bucket_name, Key=object_key)
```

```
obj = s3_resource.Object(bucket_name, 'RandomForestModel.joblib')
model = joblib.load(obj.get()['Body'].read())
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-99298cd7ac1c> in <cell line: 2>()
      1 obj = s3_resource.Object(bucket_name, 'RandomForestModel.joblib')
----> 2 model = joblib.load(obj.get()['Body'].read())

/usr/local/lib/python3.10/dist-packages/joblib/numpy_pickle.py in load(filename, mmap_mode)
    648             obj = _unpickle(fobj)
    649     else:
--> 650         with open(filename, 'rb') as f:
    651             with _read_fileobject(f, filename, mmap_mode) as fobj:
    652                 if isinstance(fobj, str):

ValueError: embedded null byte
```

```
%cd content/MyDrive/ML_DATA_2023
```

```
/content/content/MyDrive/ML_DATA_2023
```

```
Model= model.decode()
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-10-1499010354b3> in <cell line: 1>()
----> 1 Model= model.decode()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0x80 in position 0: invalid start byte
```

```
!ls
```

```
'Copy of ECO_code protocole avec code condition.xlsx'
'Copy of ECO_code protocole avec liste des codes étapes.xlsx'
'Copy of ECO_formule avec code protocole.xlsx'
'Copy of ECO_formule avec element tech.xlsx'
'Copy of ECO_formule avec liste INCI.xlsx'
'Copy of ECO_formule avec liste MP.xlsx'
'Copy of ECO_liste des codes étapes avec les conditions.xlsx'
'Copy of ECO_liste des codes opérations.xlsx'
'Copy of ECO_liste des conditions.xlsx'
'Copy of ECO_liste des formules avec désignation.xlsx'
'Copy of ECO_MP avec liste INCI.xlsx'
'Copy of MCD ML_Data.gslides'
 Formula_last_version_with_all_ingredient.csv
 Formula_last_version_without_zero_ingredients.csv
 formula_with_full_information.csv
 formula_with_label.csv
 inci_list.csv
 Label_last_version.csv
 logs
 Model
```

```
random_forest.joblib
svm_without_weight.joblib
```

```
!pip install smart_open
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: smart_open in /usr/local/lib/python3.10/dist-packages (6.3.0)
```

```
import smart_open
def read_joblib_file_from_s3(bucket_name, file_name,session):
    s3_path = f"s3://{bucket_name}/{file_name}"
    with smart_open.open(s3_path, 'rb') as s3_file:
        return joblib.load(s3_file)
```

```
model= read_joblib_file_from_s3("ecomundo-ai-challenge","RandomForestModel.joblib",session)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-be489f0d8e2b> in <cell line: 1>()
----> 1 model= read_joblib_file_from_s3("ecomundo-ai-challenge","RandomForestModel.joblib",session)

<ipython-input-18-931671bec8ac> in read_joblib_file_from_s3(bucket_name, file_name, session)
      2 def read_joblib_file_from_s3(bucket_name, file_name,session):
      3     s3_path = f"s3://{bucket_name}/{file_name}"
----> 4     with smart_open.open(s3_path, 'rb', s3=session.client('s3')) as s3_file:
      5         return joblib.load(s3_file)

TypeError: open() got an unexpected keyword argument 's3'
```

```
!pip install boto3
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting boto3
  Downloading boto3-1.26.147-py3-none-any.whl (135 kB)
 [2K     [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ [0m [32m135.6/135.6 kB [0m [31m5.9 MB/s [0m eta [36m0:00:0
0 [0m
 [?25hCollecting botocore<1.30.0,>=1.29.147 (from boto3)
  Downloading botocore-1.29.147-py3-none-any.whl (10.8 MB)
 [2K     [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ [0m [32m10.8/10.8 MB [0m [31m65.7 MB/s [0m eta [36m0:00:0
0 [0m
 [?25hCollecting jmespath<2.0.0,>=0.7.1 (from boto3)
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting s3transfer<0.7.0,>=0.6.0 (from boto3)
  Downloading s3transfer-0.6.1-py3-none-any.whl (79 kB)
 [2K     [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ [0m [32m79.8/79.8 kB [0m [31m10.3 MB/s [0m eta [36m0:00:0
0 [0m
 [?25hRequirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.10/dist-packages (from boto
core<1.30.0,>=1.29.147->boto3) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/local/lib/python3.10/dist-packages (from botocore<1.30.0,
>=1.29.147->boto3) (1.26.15)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil<3.0.0,>=2.1->
botocore<1.30.0,>=1.29.147->boto3) (1.16.0)
Installing collected packages: jmespath, botocore, s3transfer, boto3
Successfully installed boto3-1.26.147 botocore-1.29.147 jmespath-1.0.1 s3transfer-0.6.1
```

```
import boto3
import logging
from io import BytesIO
import joblib
```

```
def get_s3_client():
```

```
    return session.client('s3')



def load_model_from_s3(bucket, key):
    s3_client = get_s3_client()
    try:
        with BytesIO() as data:
            s3_client.download_fileobj(bucket,key, data)
            data.seek(0)  # move back to the beginning after writing
            return joblib.load(data)
    except Exception as e:
        raise logging.exception(e)
```

```
model=load_model_from_s3('ecomundo-ai-challenge','RandomForestModel.joblib')
```

```
model.estimator_params
```

```
('criterion',
 'max_depth',
 'min_samples_split',
 'min_samples_leaf',
 'min_weight_fraction_leaf',
 'max_features',
 'max_leaf_nodes',
 'min_impurity_decrease',
 'random_state',
 'ccp_alpha')
```

```
s3=get_s3_client()
```

```
s3.do
```