In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import pickle as pkl
from numpy import sort
from xgboost import XGBClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import log_loss
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import StackingClassifier
```

In [107]:
```
train_df = pd.read_csv('X_train2.csv')
```

In [108]:
```
# taking a dummy variable to perform the calculation on
X = pd.read_csv('X_train2.csv')
```

In [2]:
```
with open('labels.pkl','rb') as f:
    Y = pkl.load(f)
    print(Y.shape)
```

```
(73812,)
```

In [110]:
```
train_df.head()
```

Out[110]:

	id	age	year	month	day	tfa_year	tfa_month	tfa_day	timediff	gender_-unknown-	...	view
0	d1mm9tcy42	62.0	2014	1	1	2014	1	1	0	0	...	
1	yo8nz8bqcq	-1.0	2014	1	1	2014	1	1	0	1	...	
2	4grx6yxeby	-1.0	2014	1	1	2014	1	1	0	1	...	
3	ncf87guaf0	-1.0	2014	1	1	2014	1	1	0	1	...	
4	4rvqpxoh3h	-1.0	2014	1	1	2014	1	1	0	1	...	

5 rows × 321 columns

In [111]:
```
train_df.shape
```

Out[111]:
```
(73812, 321)
```

In [112]:
```
train_df.set_index('id',inplace=True)
```

In [113]:
```
X.set_index('id',inplace=True)
```

Train set with only important features

In [117]:
```
#using random forest to select the indices of important features
def imp_features(data,keep_num_features):
    clf = RandomForestClassifier(n_estimators=100,n_jobs=-1)
    clf.fit(data,Y)
    important_features_index = np.argsort(clf.feature_importances_)[::-1]
    imp_index_filtered = important_features_index[:keep_num_features]
    return imp_index_filtered
```

In [118]:
```
important_indexes = imp_features(X,160)
```

In [119]:
```
col_names = X.columns
```

In [120]:
```
col_names =col_names.to_list()
```

In [121]:
```
#droping the columns whose indices aren't in the important list
for col in train_df.columns:
    if col not in np.take(col_names,important_indexes):
        train_df.drop(col,axis=1,inplace=True)
```

In [122]:
```
train_df.head()
```

Out[122]:

	age	month	day	tfa_month	tfa_day	gender_-unknown-	gender_FEMALE	gender_MALE	sig
id									
d1mm9tcy42	62.0	1	1	1	1	0	0	1	
yo8nz8bqcq	-1.0	1	1	1	1	1	0	0	
4grx6yxeby	-1.0	1	1	1	1	1	0	0	
ncf87guaf0	-1.0	1	1	1	1	1	0	0	
4rvqpxoh3h	-1.0	1	1	1	1	1	0	0	

5 rows × 160 columns

In [128]:
```
train_df.to_csv("imp_Xtrain.csv")
```

In [123]:
```
test_df = pd.read_csv('X_test2.csv')
```

In [124]:
```
test_df.set_index('id',inplace=True)
```

Test set with only the important features

In [126]:

```
#droping the columns whose indices aren't in the important indices list
for col in test_df.columns:
    if col not in np.take(col_names,important_indexes):
        test_df.drop(col,axis=1,inplace=True)
```

In [127]:

```
test_df.head()
```

Out[127]:

	age	month	day	tfa_month	tfa_day	gender_-unknown-	gender_FEMALE	gender_MALE	sig
id									
5uwns89zht	35.0	7	1	7	1	0	1	0	
jtl0dijy2j	-1.0	7	1	7	1	1	0	0	
xx0ulgorjt	-1.0	7	1	7	1	1	0	0	
6c6puo6ix0	-1.0	7	1	7	1	1	0	0	
czqhjk3yfe	-1.0	7	1	7	1	1	0	0	

5 rows × 160 columns

In [129]:

```
test_df.to_csv('imp_Xtest.csv')
```

In [3]:

```
train_df = pd.read_csv('imp_Xtrain.csv')
```

In [4]:

```
train_df.head()
```

Out[4]:

	id	age	month	day	tfa_month	tfa_day	gender_-unknown-	gender_FEMALE	gender_MALE
0	d1mm9tcy42	62.0	1	1	1	1	0	0	1
1	yo8nz8bqcq	-1.0	1	1	1	1	1	0	0
2	4grx6yxeby	-1.0	1	1	1	1	1	0	0
3	ncf87guaf0	-1.0	1	1	1	1	1	0	0
4	4rvqpxoh3h	-1.0	1	1	1	1	1	0	0

5 rows × 161 columns

In [5]:

```
test_df = pd.read_csv('imp_Xtest.csv')
```

```
In [6]: test_df.head()
```

Out[6]:

	id	age	month	day	tfa_month	tfa_day	gender_-unknown-	gender_FEMALE	gender_MALE	s
0	5uwns89zht	35.0	7	1	7	1	0	1	0	
1	jtl0dijy2j	-1.0	7	1	7	1	1	0	0	
2	xx0ulgorjt	-1.0	7	1	7	1	1	0	0	
3	6c6puo6ix0	-1.0	7	1	7	1	1	0	0	
4	czqhjk3yfe	-1.0	7	1	7	1	1	0	0	

5 rows × 161 columns

```
In [7]: train_df.set_index('id',inplace=True)
```

```
In [8]: train_df.head()
```

Out[8]:

	age	month	day	tfa_month	tfa_day	gender_-unknown-	gender_FEMALE	gender_MALE	sig
id									
d1mm9tcy42	62.0	1	1	1	1	0	0	1	
yo8nz8bqcq	-1.0	1	1	1	1	1	0	0	
4grx6yxeby	-1.0	1	1	1	1	1	0	0	
ncf87guaf0	-1.0	1	1	1	1	1	0	0	
4rvqpxoh3h	-1.0	1	1	1	1	1	0	0	

5 rows × 160 columns

```
In [9]: train_df.shape
```

```
Out[9]: (73812, 160)
```

Xgboost Classifier

In [12]:

```
x_cfl=XGBClassifier()

prams={
    'learning_rate':[0.01,0.03,0.05,0.1,0.2,0.25],
     'n_estimators':[50,100,200,500,1000],
     'max_depth':[3,6,5,10],
    'colsample_bytree':[0.1,0.3,0.5,0.7],
    'subsample':[0.1,0.3,0.5,0.8]
}
random_cfl=RandomizedSearchCV(x_cfl,param_distributions=prams,cv= 3,verbose=10,n_
random_cfl.fit(train_df,Y)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:  3.5min
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed: 10.2min
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed: 15.0min
[Parallel(n_jobs=-1)]: Done  27 out of  30 | elapsed: 54.8min remaining:  6.1mi
n
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 61.2min finished
```

Out[12]:

```
RandomizedSearchCV(cv=3, estimator=XGBClassifier(), n_jobs=-1,
                   param_distributions={'colsample_bytree': [0.1, 0.3, 0.5,
                                                             0.7],
                                        'learning_rate': [0.01, 0.03, 0.05, 0.
1,
                                                          0.2, 0.25],
                                        'max_depth': [3, 6, 5, 10],
                                        'n_estimators': [50, 100, 200, 500,
                                                         1000],
                                        'subsample': [0.1, 0.3, 0.5, 0.8]},
                   verbose=10)
```

In [13]:

```
random_cfl.best_params_
```

Out[13]:

```
{'subsample': 0.1,
 'n_estimators': 500,
 'max_depth': 3,
 'learning_rate': 0.03,
 'colsample_bytree': 0.3}
```

In [14]:

```
x_cfl=XGBClassifier(n_estimators=500,max_depth=3,learning_rate=0.03,colsample_byt
x_cfl.fit(train_df,Y,verbose=True)
```

Out[14]:

```
XGBClassifier(colsample_bytree=0.3, learning_rate=0.03, n_estimators=500,
              nthread=-1, objective='multi:softprob', subsample=0.1)
```

In [15]:

```
import pickle
pickle.dump(x_cfl,open('impxgboost.pickle.dat','wb'))
```

In [17]:

```
classifier = pickle.load(open('impxgboost.pickle.dat','rb'))
```

In [18]:
```
test_df.set_index('id',inplace=True)
```

In [19]:
```
test_df.head()
```

Out[19]:

	age	month	day	tfa_month	tfa_day	gender_-unknown-	gender_FEMALE	gender_MALE	sig
id									
5uwns89zht	35.0	7	1	7	1	0	1	0	
jtl0dijy2j	-1.0	7	1	7	1	1	0	0	
xx0ulgorjt	-1.0	7	1	7	1	1	0	0	
6c6puo6ix0	-1.0	7	1	7	1	1	0	0	
czqhjk3yfe	-1.0	7	1	7	1	1	0	0	

5 rows × 160 columns

In [20]:
```
pred_probab = classifier.predict_proba(test_df)
```

In [21]:
```
pred_probab_df = pd.DataFrame(pred_probab,index=test_df.index)
```

In [22]:
```
pred_probab_df.head()
```

Out[22]:

	0	1	2	3	4	5	6	7	
id									
5uwns89zht	0.001348	0.003388	0.001354	0.006484	0.009293	0.005759	0.004955	0.736903	0.001
jtl0dijy2j	0.000643	0.002042	0.000653	0.003399	0.005007	0.002858	0.004524	0.918438	0.000
xx0ulgorjt	0.000558	0.003694	0.000904	0.003437	0.007690	0.003184	0.005945	0.902449	0.000
6c6puo6ix0	0.000633	0.002954	0.000787	0.005300	0.007852	0.002682	0.005044	0.901289	0.001
czqhjk3yfe	0.002832	0.024966	0.004055	0.018896	0.057158	0.026750	0.038000	0.180890	0.007

In [23]:
```
output_classes = {'AU': 0,
 'CA': 1,
 'DE': 2,
 'ES': 3,
 'FR': 4,
 'GB': 5,
 'IT': 6,
 'NDF': 7,
 'NL': 8,
 'PT': 9,
 'US': 10,
 'other': 11}
```

In [24]:
```
inv_classes = {v:k for k,v in output_classes.items()}
```

In [25]:
```
def top_5_countries(s):
    """
    This function takes the probability values of each id, sorts the top 5 values
    """
    indices = np.arange(0,12)
    pred_dict = dict(zip(indices,s))
    sorted_abc = sorted(pred_dict.items(),key=lambda x:x[1],reverse=True)[:5]
    row_indices = [x[0] for x in sorted_abc]
    top_five = [inv_classes[i] for i in row_indices]
    return top_five
```

In [26]:
```
pred_probab_df['top_five'] = pred_probab_df.apply(top_5_countries,axis=1)
```

In [27]:
```
pred_probab_df.head()
```

Out[27]:

	0	1	2	3	4	5	6	7	
id									
5uwns89zht	0.001348	0.003388	0.001354	0.006484	0.009293	0.005759	0.004955	0.736903	0.001
jtl0dijy2j	0.000643	0.002042	0.000653	0.003399	0.005007	0.002858	0.004524	0.918438	0.000
xx0ulgorjt	0.000558	0.003694	0.000904	0.003437	0.007690	0.003184	0.005945	0.902449	0.000
6c6puo6ix0	0.000633	0.002954	0.000787	0.005300	0.007852	0.002682	0.005044	0.901289	0.001
czqhjk3yfe	0.002832	0.024966	0.004055	0.018896	0.057158	0.026750	0.038000	0.180890	0.007

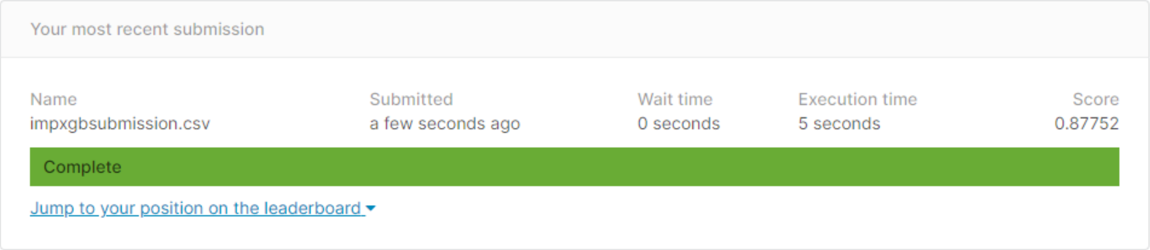
In [28]:
```
s = pred_probab_df.apply(lambda x: pd.Series(x['top_five']),axis=1).stack().reset
s.name = 'country'
```

```
In [29]: submission = pred_probab_df.drop([i for i in range(0,12)] + ['top_five'],axis=1).
         submission.head()
```

Out[29]:

	country
id	
0010k6l0om	NDF
0010k6l0om	US
0010k6l0om	other
0010k6l0om	FR
0010k6l0om	ES

```
In [30]: submission.to_csv('impxgbsubmission.csv')
```

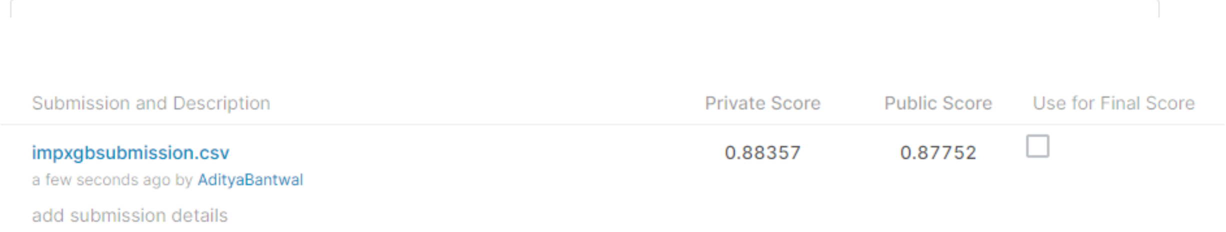


Your most recent submission

Name	Submitted	Wait time	Execution time	Score
impxgbsubmission.csv	a few seconds ago	0 seconds	5 seconds	0.87752

Complete

Jump to your position on the leaderboard ▾



Submission and Description	Private Score	Public Score	Use for Final Score
impxgbsubmission.csv a few seconds ago by AdityaBantwal add submission details	0.88357	0.87752	□

Random Forest

In [33]:
```
x_cfl= RandomForestClassifier()

prams={
     'min_samples_split':[2,20],
     'n_estimators':[100,200,500,1000,2000],
     'max_depth':[3,5,10]
}
random_cfl=RandomizedSearchCV(x_cfl,param_distributions=prams,cv= 3,verbose=10,n_
random_cfl.fit(train_df,Y)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:   37.6s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   42.4s
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done  27 out of  30 | elapsed:  1.6min remaining:   10.6
s
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  1.8min finished
```

Out[33]:
```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1,
                   param_distributions={'max_depth': [3, 5, 10],
                                        'min_samples_split': [2, 20],
                                        'n_estimators': [100, 200, 500, 1000,
                                                         2000]},
                   verbose=10)
```

In [34]:
```
random_cfl.best_params_
```

Out[34]:
```
{'n_estimators': 500, 'min_samples_split': 2, 'max_depth': 10}
```

In [35]:
```
#Using the best parameters to train the model
x_cfl=RandomForestClassifier(n_estimators=500,min_samples_split=2,max_depth=10)
x_cfl.fit(train_df,Y)

```

Out[35]:
```
RandomForestClassifier(max_depth=10, n_estimators=500)
```

In [36]:
```
import pickle
pickle.dump(x_cfl,open('impRF.pickle.dat','wb'))
```

In [37]:
```
classifier = pickle.load(open('impRF.pickle.dat','rb'))
```

In [39]:
```
pred_probab = classifier.predict_proba(test_df)
```

In [40]:
```
# storing the predictions of each user_id in a dataframe with user_id as the inde
pred_probab_df = pd.DataFrame(pred_probab,index=test_df.index)
```

In [41]:
```
pred_probab_df.head()
```

Out[41]:

	0	1	2	3	4	5	6	7	
id									
5uwns89zht	0.001591	0.003624	0.002313	0.006600	0.012448	0.006342	0.007307	0.714988	0.002
jtl0dijy2j	0.000664	0.001860	0.000946	0.003721	0.007824	0.003196	0.004988	0.874647	0.001
xx0ulgorjt	0.000768	0.002631	0.001045	0.004568	0.009756	0.004282	0.006852	0.854332	0.001
6c6puo6ix0	0.000703	0.002589	0.001032	0.004657	0.010227	0.004203	0.007036	0.855365	0.001
czqhjk3yfe	0.001767	0.014381	0.004084	0.018438	0.047249	0.019202	0.039174	0.188846	0.00

In [42]:
```
pred_probab_df['top_five'] = pred_probab_df.apply(top_5_countries,axis=1)
```

In [43]:
```
pred_probab_df.head()
```

Out[43]:

	0	1	2	3	4	5	6	7	
id									
5uwns89zht	0.001591	0.003624	0.002313	0.006600	0.012448	0.006342	0.007307	0.714988	0.002
jtl0dijy2j	0.000664	0.001860	0.000946	0.003721	0.007824	0.003196	0.004988	0.874647	0.001
xx0ulgorjt	0.000768	0.002631	0.001045	0.004568	0.009756	0.004282	0.006852	0.854332	0.001
6c6puo6ix0	0.000703	0.002589	0.001032	0.004657	0.010227	0.004203	0.007036	0.855365	0.001
czqhjk3yfe	0.001767	0.014381	0.004084	0.018438	0.047249	0.019202	0.039174	0.188846	0.00

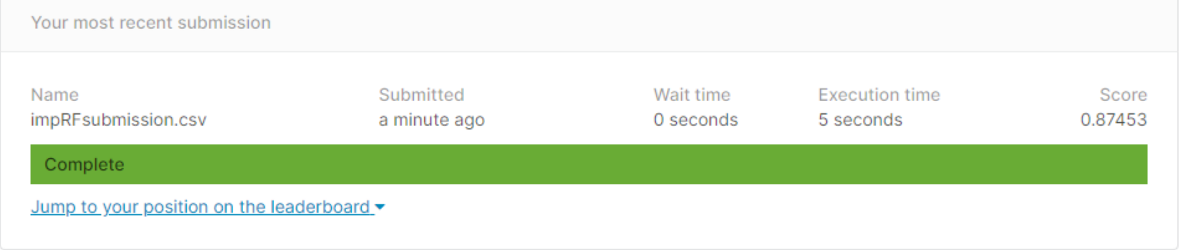
In [44]:
```
# ungrouping the list values of the top_five column
s = pred_probab_df.apply(lambda x: pd.Series(x['top_five']),axis=1).stack().reset
s.name = 'country'
```

In [45]:
```
submission = pred_probab_df.drop([i for i in range(0,12)] + ['top_five'],axis=1).
submission.head()
```

Out[45]:

	country
id	
0010k6l0om	NDF
0010k6l0om	US
0010k6l0om	other
0010k6l0om	FR
0010k6l0om	IT

In [47]:
```
submission.to_csv('impRFsubmission.csv')
```

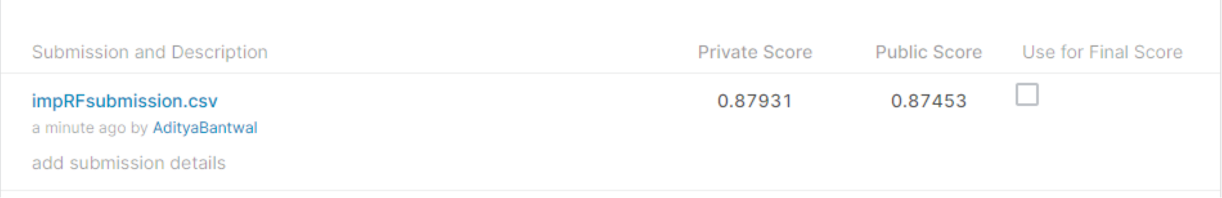


Your most recent submission

Name	Submitted	Wait time	Execution time	Score
impRFsubmission.csv	a minute ago	0 seconds	5 seconds	0.87453

Complete

Jump to your position on the leaderboard ▾



Submission and Description	Private Score	Public Score	Use for Final Score
impRFsubmission.csv a minute ago by AdityaBantwal add submission details	0.87931	0.87453	□

In [13]:
```
# splitting the train data into train and Cross validation data
from sklearn.model_selection import train_test_split
X_train,X_cv,y_train,y_cv = train_test_split(train_df,Y,test_size=0.20)
```

Stacking Classifier

In [21]:
```
#class sklearn.ensemble.StackingClassifier(estimators, final_estimator=None, *, c
# link - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.Stack
alpha = [0.0001,0.001,0.01,0.1,1,10]
estimators = [('xgb',XGBClassifier(n_estimators=500,max_depth=3,learning_rate=0.0
              ('rf',RandomForestClassifier(n_estimators=500,min_samples_split=2,
best_alpha = 999
for i in alpha:
    lr = LogisticRegression(C=i)
    sclf = StackingClassifier(estimators=estimators, final_estimator=lr)
    sclf.fit(X_train,y_train)
    print("Stacking Classifer : for the value of alpha: %f Log Loss: %0.3f" % (i,
    log_error =log_loss(y_cv, sclf.predict_proba(X_cv))
    if best_alpha > log_error:
        best_alpha = log_error
```

```
Stacking Classifer : for the value of alpha: 0.000100 Log Loss: 1.038
Stacking Classifer : for the value of alpha: 0.001000 Log Loss: 0.956
Stacking Classifer : for the value of alpha: 0.010000 Log Loss: 0.942

C:\Users\user\Anaconda3\envs\tf-gpu\lib\site-packages\sklearn\linear_model\_log
istic.py:764: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)



Stacking Classifer : for the value of alpha: 0.100000 Log Loss: 0.938

C:\Users\user\Anaconda3\envs\tf-gpu\lib\site-packages\sklearn\linear_model\_log
istic.py:764: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)



Stacking Classifer : for the value of alpha: 1.000000 Log Loss: 0.935

C:\Users\user\Anaconda3\envs\tf-gpu\lib\site-packages\sklearn\linear_model\_log
istic.py:764: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)


Stacking Classifer : for the value of alpha: 10.000000 Log Loss: 0.935
```

In [22]:

```
estimators = [('xgb',XGBClassifier(n_estimators=500,max_depth=3,learning_rate=0.0
               ('rf',RandomForestClassifier(n_estimators=500,min_samples_split=2,
lr = LogisticRegression(C=1.000000)
sclf = StackingClassifier(estimators=estimators, final_estimator=lr)
sclf.fit(X_train,y_train)
```

```
C:\Users\user\Anaconda3\envs\tf-gpu\lib\site-packages\sklearn\linear_model\_log
istic.py:764: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[22]:

```
StackingClassifier(estimators=[('xgb',
                                XGBClassifier(colsample_bytree=0.3,
                                              learning_rate=0.03,
                                              n_estimators=500, nthread=-1,
                                              subsample=0.1)),
                               ('rf',
                                RandomForestClassifier(max_depth=10,
                                                       n_estimators=500))],
                   final_estimator=LogisticRegression())
```

In [24]:

```
import pickle
pickle.dump(sclf,open('stackingclf.pickle.dat','wb'))
```

In [25]:

```
classifier = pickle.load(open('stackingclf.pickle.dat','rb'))
```

In [26]:
```
test_df.head()
```

Out[26]:

	id	age	month	day	tfa_month	tfa_day	gender_-unknown-	gender_FEMALE	gender_MALE	s
0	5uwns89zht	35.0	7	1	7	1	0	1	0	
1	jtl0dijy2j	-1.0	7	1	7	1	1	0	0	
2	xx0ulgorjt	-1.0	7	1	7	1	1	0	0	
3	6c6puo6ix0	-1.0	7	1	7	1	1	0	0	
4	czqhjk3yfe	-1.0	7	1	7	1	1	0	0	

5 rows × 161 columns

In [27]:
```
test_df.set_index('id',inplace=True)
```

In [29]:
```
# since in the problem statement it is mentioned that the we need to predict the
pred_probab = classifier.predict_proba(test_df)
```

In [30]:
```
# storing the predictions of each user_id in a dataframe with user_id as the inde
pred_probab_df = pd.DataFrame(pred_probab,index=test_df.index)
```

In [31]:
```
pred_probab_df.head()
```

Out[31]:

	0	1	2	3	4	5	6	7	
id									
5uwns89zht	0.001139	0.003217	0.001732	0.005167	0.010172	0.005415	0.007185	0.778874	0.001
jtl0dijy2j	0.000551	0.001486	0.000813	0.002568	0.005085	0.002635	0.003777	0.903564	0.000
xx0ulgorjt	0.000621	0.001683	0.000924	0.002939	0.005923	0.002993	0.004398	0.893672	0.000
6c6puo6ix0	0.000626	0.001701	0.000932	0.002966	0.005998	0.003026	0.004446	0.892665	0.000
czqhjk3yfe	0.005169	0.016121	0.008965	0.025866	0.059067	0.025625	0.039220	0.130506	0.009

In [32]:
```
# The dictionary is the label encoding of the countries feature
output_classes = {'AU': 0,
 'CA': 1,
 'DE': 2,
 'ES': 3,
 'FR': 4,
 'GB': 5,
 'IT': 6,
 'NDF': 7,
 'NL': 8,
 'PT': 9,
 'US': 10,
 'other': 11}
```

In [33]:
```
# inverting the dictionary
inv_classes = {v:k for k,v in output_classes.items()}
```

In [34]:
```
inv_classes
```

Out[34]:
```
{0: 'AU',
 1: 'CA',
 2: 'DE',
 3: 'ES',
 4: 'FR',
 5: 'GB',
 6: 'IT',
 7: 'NDF',
 8: 'NL',
 9: 'PT',
 10: 'US',
 11: 'other'}
```

In [35]:
```
def top_5_countries(s):
    """
    This function takes the probability values of each id, sorts the top 5 values
    """
    indices = np.arange(0,12)
    pred_dict = dict(zip(indices,s))
    sorted_abc = sorted(pred_dict.items(),key=lambda x:x[1],reverse=True)[:5]
    row_indices = [x[0] for x in sorted_abc]
    top_five = [inv_classes[i] for i in row_indices]
    return top_five
```

In [36]:
```
# here we apply the above function on each row of the dataframe to get the top 5
pred_probab_df['top_five'] = pred_probab_df.apply(top_5_countries,axis=1)
```

In [37]:
```
pred_probab_df.head()
```

Out[37]:

	0	1	2	3	4	5	6	7	
id									
5uwns89zht	0.001139	0.003217	0.001732	0.005167	0.010172	0.005415	0.007185	0.778874	0.001
jtl0dijy2j	0.000551	0.001486	0.000813	0.002568	0.005085	0.002635	0.003777	0.903564	0.000
xx0ulgorjt	0.000621	0.001683	0.000924	0.002939	0.005923	0.002993	0.004398	0.893672	0.000
6c6puo6ix0	0.000626	0.001701	0.000932	0.002966	0.005998	0.003026	0.004446	0.892665	0.000
czqhjk3yfe	0.005169	0.016121	0.008965	0.025866	0.059067	0.025625	0.039220	0.130506	0.009

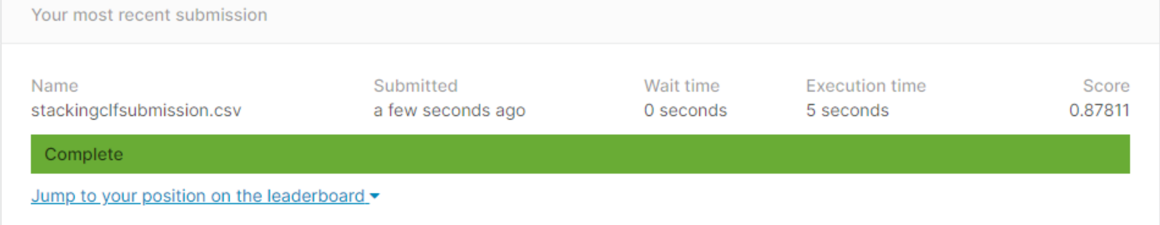
In [38]:
```
# ungrouping the list values of the top_five column
s = pred_probab_df.apply(lambda x: pd.Series(x['top_five']),axis=1).stack().reset
s.name = 'country'
```

In [39]:
```
submission = pred_probab_df.drop([i for i in range(0,12)] + ['top_five'],axis=1).
submission.head()
```

Out[39]:

	country
id	
0010k6l0om	NDF
0010k6l0om	US
0010k6l0om	other
0010k6l0om	FR
0010k6l0om	IT

In [40]:
```
submission.to_csv('stackingclfsubmission.csv')
```

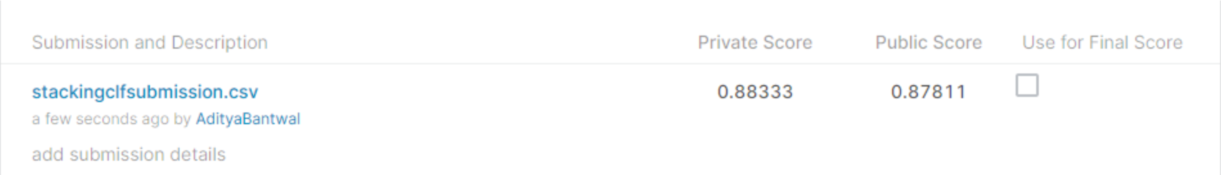


Your most recent submission

Name	Submitted	Wait time	Execution time	Score
stackingclfsubmission.csv	a few seconds ago	0 seconds	5 seconds	0.87811

Complete

Jump to your position on the leaderboard



Submission and Description	Private Score	Public Score	Use for Final Score
stackingclfsubmission.csv a few seconds ago by AdityaBantwal add submission details	0.88333	0.87811	□