In [23]:
```
import numpy as np
import pandas as pd
import pickle
from sklearn.preprocessing import LabelEncoder
import numpy as np
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import make_scorer
```

In [32]:
```
def final_fun_1(X):
    X.set_index('id',inplace=True)
    #Time features
    date_stack = np.vstack(X.date_account_created.astype(str).apply(lambda x: lis
    X['year'] = date_stack[:,0]
    X['month'] = date_stack[:,1]
    X['day']  = date_stack[:,2]

    X['timestamp_first_active'] = X.timestamp_first_active//1000000

    tfa = np.vstack(X.timestamp_first_active.astype(str).apply(lambda x: list(map

    X['tfa_year'] = tfa[:,0]
    X['tfa_month']  = tfa[:,1]
    X['tfa_day']   = tfa[:,2]

    X['date_account_created'] = X.date_account_created.astype(str).apply(lambda x

    X['timediff'] = X.date_account_created - X.timestamp_first_active

    X.drop(['timestamp_first_active','date_account_created','date_first_booking']

    ## Age feature

    X['age'] = X['age'].apply(lambda x: 2015 - x if x >1900 else x)
    X['age'].fillna(-1,inplace=True)

    feat_onehot = ['gender','signup_method','signup_flow','language','affiliate_c

    for f in feat_onehot:
        df_dummy = pd.get_dummies(X[f],prefix=f)
        X = X.drop([f],axis=1)
        X = pd.concat((X,df_dummy),axis=1)

    #reading the important column names which are recognized by our model
    with open('imp_colnames.pkl','rb') as f:
        imp_col_names = pickle.load(f)


    X = X[X.columns[X.columns.isin(imp_col_names)]]

    extra_features = list(set(imp_col_names)-set(X.columns.to_list()))


    for i in extra_features:
         X[i] = np.nan

    X.reindex(columns=imp_col_names)
    X.fillna(0,inplace=True)

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

    pred_probab = classifier.predict_proba(X[imp_col_names])

    # storing the predictions of each user_id in a dataframe with user_id as the
    pred_probab_df = pd.DataFrame(pred_probab,index=X.index)
```

```
    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}

    # inverting the dictionary
    inv_classes = {v:k for k,v in output_classes.items()}

    def top_5_countries(s):
        """
        This function takes the probability values of each id, sorts the top 5 va
        """
        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




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

    submission = pred_probab_df.drop([i for i in range(0,12)],axis=1)

    return submission.head()
```

In [26]:
```
X = pd.read_csv('Enter Your Train/Test set csv file path')
```

In [33]:
```
output = final_fun_1(X)
```

In [34]:
```
output
```

Out[34]:

id	top_five
5uwns89zht	[NDF, US, other, FR, IT]
jtl0dijy2j	[NDF, US, other, FR, IT]
xx0ulgorjt	[NDF, US, other, FR, IT]
6c6puo6ix0	[NDF, US, other, FR, IT]
czqhjk3yfe	[NDF, US, other, FR, IT]

In [35]:
```
with open('labelsfull.pkl','rb') as f:
    Y = pickle.load(f)
```

In [36]:
```
def final_fun_2(X,Y):
    X.set_index('id',inplace=True)
    #Time features
    date_stack = np.vstack(X.date_account_created.astype(str).apply(lambda x: lis
    X['year'] = date_stack[:,0]
    X['month'] = date_stack[:,1]
    X['day']  = date_stack[:,2]

    X['timestamp_first_active'] = X.timestamp_first_active//1000000

    tfa = np.vstack(X.timestamp_first_active.astype(str).apply(lambda x: list(map

    X['tfa_year'] = tfa[:,0]
    X['tfa_month']  = tfa[:,1]
    X['tfa_day']   = tfa[:,2]

    X['date_account_created'] = X.date_account_created.astype(str).apply(lambda x

    X['timediff'] = X.date_account_created - X.timestamp_first_active

    X.drop(['timestamp_first_active','date_account_created','date_first_booking']

    ## Age feature

    X['age'] = X['age'].apply(lambda x: 2015 - x if x >1900 else x)
    X['age'].fillna(-1,inplace=True)

    feat_onehot = ['gender','signup_method','signup_flow','language','affiliate_c

    for f in feat_onehot:
        df_dummy = pd.get_dummies(X[f],prefix=f)
        X = X.drop([f],axis=1)
        X = pd.concat((X,df_dummy),axis=1)

    #reading the important column names which are recognized by our model
    with open('imp_colnames.pkl','rb') as f:
        imp_col_names = pickle.load(f)


    X = X[X.columns[X.columns.isin(imp_col_names)]]

    extra_features = list(set(imp_col_names)-set(X.columns.to_list()))


    for i in extra_features:
         X[i] = np.nan

    X.reindex(columns=imp_col_names)
    X.fillna(0,inplace=True)

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

    pred_probab = classifier.predict_proba(X[imp_col_names])

    # storing the predictions of each user_id in a dataframe with user_id as the
```

```
    def dcg_score(y_true, y_score, k=5):
        order = np.argsort(y_score)[::-1]
        y_true = np.take(y_true, order[:k])
        gain = 2 ** y_true - 1
        discounts = np.log2(np.arange(len(y_true)) + 2)
        return np.sum(gain / discounts)

    def ndcg_score(ground_truth, predictions, k=5):
        lb = LabelBinarizer()
        lb.fit(range(len(predictions[0]) + 1))
        T = lb.transform(ground_truth)
        scores = []

        for y_true, y_score in zip(T, predictions):
            actual = dcg_score(y_true, y_score, k)
            best = dcg_score(y_true, y_true, k)

            if best <= 0:
                 score = 0.0
            else:
                 score = float(actual) / float(best)
            scores.append(score)

        return np.mean(scores)

    metric_value = ndcg_score(Y,pred_probab)

    return metric_value
```

In [ ]:
```
X = pd.read_csv('Enter your (Train or Test) csv file path')
```

In [ ]:
```
#labels
with open('labelsfull.pkl','rb') as f:
    Y = pickle.load(f)
```

In [39]:
```
value = final_fun_2(X,Y)
```

In [40]:
```
print("The value is:",value)
```

```
The value is: 0.7683478952898508
```

In [ ]: