Aruna Kumaraswamy
akumaras@iu.edu

Airbnb First time booking prediction

Aruna Kumaraswamy

Indiana University – Bloomington

Introduction

Airbnb establishes a platform for property owners and travelers across the world to do business. This research project is to analyze the current activity of travelers within Airbnb website. Goal of this project is to predict the country of destination for a new Airbnb customer based on the first time booking characteristics of existing customers. Training data involves customers from USA. Destination of choice for these customers are across the world. User demographics such as age, gender is used in the analysis. User behavior such as when they signed up for Airbnb, when they became active within the website, how much time they spent in the site, their browsing history and the device that they used to browse are some of the features used in this prediction. Prediction model can help Airbnb market the available properties in the predicted destination to the customer. Accuracy of this model will facilitate an effective targeted marketing for Airbnb.

Data

Airbnb has published, users, sessions, country and age-gender data. User and session information are used in the analysis. All data provided are in CSV format.

User

Following sections detail the features and its distribution within the user data:

Date account created feature has no missing values. Account creation in Airbnb has been trending up since 2010. Accounts are created more between May and August.

First time the user became active within the Airbnb web site is captured as timestamp. But this information is not complete. Year and month is available, but day and time is not present in the values. This column is presented as float in the csv and needs transformation.

The date user first booked a destination in Airbnb is not available for all customers. Less than 50% (88K out of 235K records) of customer information has the first booking information. Hence it may not be a good predictor feature.

Marketing motivation for user signup (first affiliate tracked) has nearly 50% as untracked. There is a small percentage of missing information. This feature is not a strong predictor.

User's gender has 4 unique values. 45% is unknown.

Age of the user has missing values and several outliers. It is 50% complete. It needs to be normalized to make use of the available data.

There are 25 unique languages in the dataset. But 97% is English.

8 Affiliate channels drew customers to the booking. More than 40% in the data are direct customers to Airbnb. There are 18 affiliate providers. Nearly 60% customers did not go through any provider such as facebook, google etc. This is evident via the sign up method as well, where basic dominates other affiliates. Mac desktop is the most popular device. Chrome is the popular browser among Airbnb customers. Country destination has 'No destination Found' to be around 50% and next is 'USA'. This data has a high bias for most features.

Session

Session data comprises of the action, type of action, device and the time spent by Airbnb customer. Without the knowledge of what each action/type and how they influence the booking, it is hard to infer the relation to the booking destination. But number of actions performed by the user and how long they spend before booking the destination, could be a good predictor feature.

Methods

Following steps are essential to build our prediction model.

Feature Engineering

As discussed in data exploration section, most features in this data set have a strong bias, missing information and some(age) have outliers. So, it is essential to select relevant features and/or normalize the features to facilitate better prediction. In date related features, we will choose the day and month wherever applicable. Minimum age will be set to 18 and maximum will be 100. Values where user entered years, will be converted to age. This helps establish a near normal distribution for age. One hot encoding is applied to all categorical features to aid binary classification of the relevant feature. Session data is aggregated and joined with user data using the session id as key. As session is a subset of user data a left join is performed. Output classifier is the country of destination. This is a multi-class discrete variable. Label encoder is used to classify the output to be continuous variable. There are about 146 features to perform prediction.

Model Selection

There are 12 destination countries and the model has to classify based on the given input which destination country is more likely be the first time booking. Based on feature selection, we have 146 input features to be used to predict any of the 12 classes.

RandomForestClassifier – 63.3%

GradientBoostClassifier – 63.3%

Neural Network – MLP – 61.4%

Naïve Bayes – 56.6%

Based on the accuracy scores listed above, Random Forest and Gradient Boost classification models perform comparatively better to neural network and naïve bayes.

Aruna Kumaraswamy
akumaras@iu.edu

Results

Following is the classification report. Model has classified only 'NDF' and 'USA'. Even within these 2 classification FP and FN are significantly higher.

Gradient Boost classifier

	Precision	Recall	F1-score	Support
AU	0.00	0.00	0.00	116
CA	0.00	0.00	0.00	285
DE	0.00	0.00	0.00	241
ES	0.00	0.00	0.00	451
FR	0.00	0.00	0.00	1029
GB	0.00	0.00	0.00	474
IT	0.00	0.00	0.00	581
NDF	0.69	0.84	0.76	24866
NL	0.00	0.00	0.00	170
PT	0.00	0.00	0.00	52
US	0.49	0.49	0.49	12439
other	0.00	0.00	0.00	1987
avg / total	0.55	0.63	0.59	42691

Random Forest

	Precision	Recall	F1-score	Support
AU	0.00	0.00	0.00	116
CA	0.00	0.00	0.00	285
DE	0.00	0.00	0.00	241
ES	0.00	0.00	0.00	451
FR	0.00	0.00	0.00	1029
GB	0.00	0.00	0.00	474
IT	0.00	0.00	0.00	581
NDF	0.68	0.86	0.76	24866
NL	0.00	0.00	0.00	170
PT	0.00	0.00	0.00	52
US	0.50	0.44	0.47	12439
other	0.00	0.00	0.00	1987
avg / total	0.54	0.63	0.58	42691

Discussion

Models have to be tuned to improve accuracy. Basic tuning in terms of number of estimators , number of leaf nodes/depth has been tried on the models. These do not significantly increase the model accuracy. These models are too generalized for solving this classification. We need a multi-layer pipeline where the model learns from the previous errors.

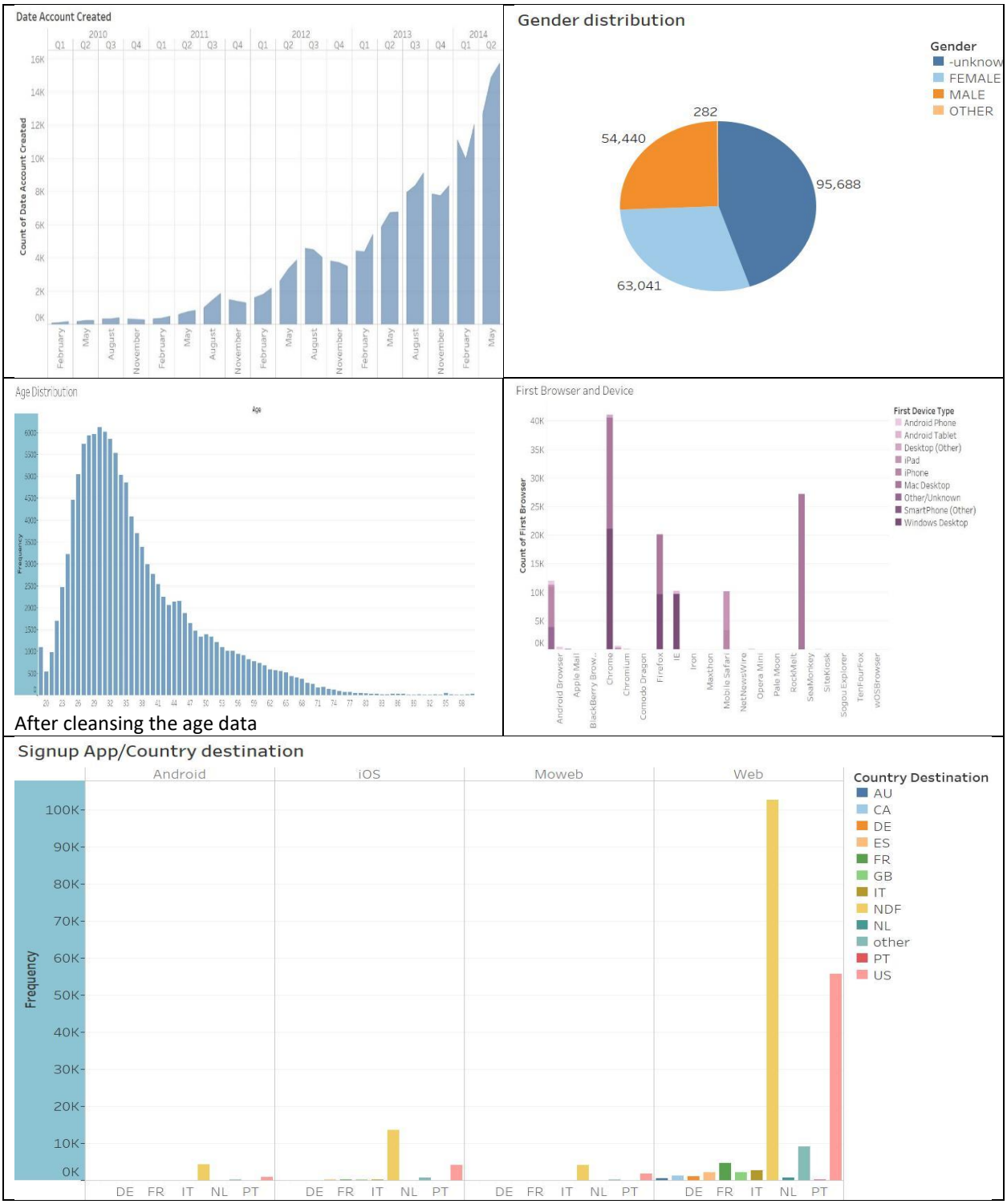
Also, number of features can be increased based on additional data exploration. Machine learning pipeline using feature selection techniques such as 'pca' or 'selectkmethod' can help identify the correct features and use these features on selected estimators for prediction.

Conclusion

In this project, we have tried to predict the country of destination of a new Airbnb customer based on the training data. Model can perform the required prediction with an accuracy of 63%. This performance will not be very effective to perform targeted marketing using this model. Based on the results and the recommendations in this report, if we perform additional feature engineering and model tuning, it is very likely we could improve on the accuracy of the prediction.

Aruna Kumaraswamy
akumaras@iu.edu

Appendix A – Data exploration



After cleansing the age data

Aruna Kumaraswamy
akumaras@iu.edu

Appendix B – Code

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 26 07:17:08 2017
Airbnb First time booking prediction
@author: Aruna Kumaraswamy
"""
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report


def generation_by_age(age_values):
  gen_values = []
  for x in age_values:
    if (x > 70):
      gen_values.append('silent')
    elif (x > 53) and (x <= 70):
      gen_values.append('boomer')
    elif (x > 41) and (x <= 53):
      gen_values.append('genx')
    elif (x > 21) and (x <= 41):
      gen_values.append('geny')
    elif (x <= 21):
      gen_values.append('genz')
  return gen_values


seed = 7
np.random.seed(seed)
```

```
print 'Feature Selection begins'

print 'Processing -  User data'


booking_df = pd.read_csv("train_users_2.csv",parse_dates=True)

target = booking_df['country_destination']

booking_df = booking_df.drop(['country_destination'],axis=1)


# Convert age to numeric and retain age in the range 18:100

booking_df['age'] = booking_df['age'].fillna(0)

age_array = booking_df['age'].values

#Susbtitue the birth year with age

age_array = np.where(age_array > 1900, 2014-age_array, age_array)

#If age more than 100, then assume 100 as maximum

age_array = np.where(age_array > 100, 100, age_array)

#If age less than 18, then assume 18 as minimum

age_array = np.where(age_array <18, 18, age_array)

booking_df['age'] = age_array


age_int_values = age_array.astype(np.int)

gen_values = generation_by_age(age_int_values)

booking_df['generation'] = gen_values


#Time first active

booking_df['timestamp_first_active'] = booking_df['timestamp_first_active'].astype(str)

tfa_array = booking_df['timestamp_first_active'].values

tfa_month_array = [ x[4:6] for x in tfa_array]

tfa_yr_array = [ x[:4] for x in tfa_array]

booking_df['tfa_mon'] = tfa_month_array

booking_df['tfa_yr'] = tfa_yr_array

booking_df = booking_df.drop(['timestamp_first_active'],axis=1)


#Date Account Created

booking_df['date_account_created'] = booking_df['date_account_created'].astype(str)
```

```
dac_array = booking_df['date_account_created'].values
dac_dates = [x.split('-') for x in dac_array]
dac_month_array = [ x[0] for x in dac_dates]
dac_day_array = [ x[1] for x in dac_dates]
booking_df['dac_mon'] = dac_month_array
booking_df['dac_day'] = dac_day_array
booking_df = booking_df.drop(['date_account_created'], axis=1)


#Less than 50% records have this populated, so drop
booking_df = booking_df.drop(['date_first_booking'],axis=1)


#Fill missing values with 'missing'
booking_df['first_affiliate_tracked'] = booking_df['first_affiliate_tracked'].fillna('missing')


# One Hot Encoding of the discrete features
ohe_features = ['generation','gender','language','affiliate_channel','affiliate_provider',
            'first_affiliate_tracked','signup_method','signup_app','first_device_type','first_browser' ]
for ohf in ohe_features:
   ohe_df = pd.get_dummies(booking_df[ohf], prefix=ohf)
   booking_df = booking_df.drop([ohf], axis=1)
   booking_df = pd.concat((booking_df,ohe_df),axis=1)


# Read sessions csv and group by user id to sum the elapsed time
print 'Processing Session data'
session_df = pd.read_csv('sessions.csv')
session_df.loc[session_df['secs_elapsed'].isnull() ,'secs_elapsed'] = 0
session_group_df = session_df.groupby('user_id').agg({'action':np.size,'action_type':np.size,'device_type':np.size,'secs_elapsed': np.sum})


session_group_df['id'] = session_group_df.index
joined_df = pd.merge(booking_df,session_group_df,on='id',how='left')# on='user_id',
joined_df.loc[joined_df['secs_elapsed'].isnull() ,'secs_elapsed'] = 0
joined_df.loc[joined_df['action_type'].isnull() ,'action_type'] = 0
joined_df.loc[joined_df['action'].isnull() ,'action'] = 0
joined_df.loc[joined_df['device_type'].isnull() ,'device_type'] = 0
```

```
#joined_df.to_csv('airbnb_updated_train_2.csv')

joined_df = joined_df.drop(['id'],axis=1)

train_vals = joined_df.values

print 'Feature Selection - Completed'


le = preprocessing.LabelEncoder()

le = le.fit(target)

train_Y = le.transform(target)


trainX,testX,trainY,testY = train_test_split(train_vals,train_Y,test_size=.20)


#Naive Bayes

from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

nb.fit(trainX,trainY)


print("\n\nNaive Bayes Performance")

print accuracy_score(testY,nb.predict(testX))

print '\n RandomForest Classifier'

classes = le.inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11])

rf = RandomForestClassifier(max_depth=12,random_state=seed, n_estimators=100)

clf = rf.fit(trainX, trainY)

predY = clf.predict(testX)

print 'Accuracy: ',accuracy_score(testY,predY)

print(classification_report(testY, predY,target_names=classes))


print '\n Neural Network'

from sklearn.neural_network import MLPClassifier

nn = MLPClassifier(random_state=seed)

nn.fit(trainX,trainY)

print accuracy_score(testY,nn.predict(testX))

#svm = svm.SVC()

#clf = svm.fit(trainX,trainY)
```

```
print '\n Gradient Boost'

grd = GradientBoostingClassifier(n_estimators=50,random_state=seed,max_leaf_nodes=12)

grd.fit(trainX,trainY)

grd_pred_y = grd.predict(testX)

print 'Accuracy: ',accuracy_score(testY,grd_pred_y)

print(classification_report(testY, grd_pred_y,target_names=classes))
```