Millsop_Ventura_Ye_Final_Project_Baseline

November 18, 2018

1 W207 Final Project Baseline

- Authors: Christian Millsop, Chris Ventura, Stanley Ye

- Kaggle Project: Airbnb New User Bookings
- GitHub Repo: w207_final_airbnb

1.1 Table of Contents

2 Introduction

For this paper, we will use various machine learning models to complete the Airbnb new user bookings Kaggle competition. This competition was originally designed as a recruiting tool for Airbnb. The stated goal of this competition is to predict in which country a new user will make their first booking[1]. To complete this challenge, Airbnb and Kaggle have provided several datasets consisting of user demographics, summary statistics, and web session information to try to predict a target variable consisting of the country in which a user made their first booking. Given that our core problem is to predict where users will make their first booking, we will attempt to answer the following research question: What is the relationship of a user profile and online activity within the Airbnb website to the travel destination?

To answer this question we will optimize and combine several machine learning models to generate accurate destination predictions. We will be using both derived and raw features from the datasets as inputs to these models. Using known best practices[2][3] to examine the effect of the various features on the end predictions. This will show both the effectiveness of various models as well as the relative importance of each feature at predicting accurate destinations. After examination, we will be able to answer our initial research question by identifying which aspects of a user's profile and online activity have a relationship to the selection of one or more travel destinations.

While we will be optimizing our models to accurately predict the destination country, the Kaggle competition allows for up to five predictions per user. The competition uses the following Normalized Discounted Cumulative Gain (NDCG) formula to score results[4]:

3 Dataset Descriptions

3.1 Dataset: age_gender_bkts:

This dataset contains the population, in thousands, aggregated at the following dimensions: year, age bucket, destination country, and gender.

Column	Description
age_bucket	five-year age interval buckets, with a single bucket for 100+
country_destination	the destination country dimension, can be US', 'FR', 'CA', 'GB', 'ES', 'IT', 'PT', 'NL','DE', or 'AU'
gender	the gender dimension, can be 'male' or 'female'
population_in_thousands	the population for the specific set of dimensions
year	the year dimension

3.2 Dataset: countries

This dataset lists summary information for each possible destination country.

Column	Description
country_destination	the destination country dimension, can be US', 'FR', 'CA', 'GB', 'ES', 'IT', 'PT', 'NL','DE', or 'AU'
lat_destination	the latitude of the destination country
lng_destination	the longitude of the destination country
distance_km	the distance in kilometers of the country from the United States
distance_km_2	unknown
destination_language	the primary spoken language of the destination country
language_levenshtein_distance	a measure of similarity between the destination_language and English, with a lower number representing a closer match

3.3 Dataset: sample_submission_NDF

This dataset contains the Kaggle submission format, consisting of the user ID and destination country. According to the competition documentation, there can be up to five entries per user ID, with the lower index rank being the higher choice, which impacts the ultimate score returned in the scoring formula.

Column	Description
id	the user id
country	the predicted destination country

3.4 Dataset: sessions

This dataset contains user session data, broken out by user action. This data will be used to examine if user behavior patterns can be indicative of the user's end destination choice.

Column	Description
user_id	the user id, used to connect to other datasets
action	the action taken by the user
action_type	the category of the action
action_detail	further detail of the action
device_type	the device on which the user performed the action
secs_elapsed	the time elapsed between this action and the prior action

3.5 Dataset: train_users_2

This dataset contains the user id, target variable of destination country, as well as demographic and account detail information.

Column	Description
id	the user id, used to connect to other datasets
date_account_created	the date on which the user first created the account
timestamp_first_active	the timestamp of when the user was first active, can predate account creation
date_first_booking	the date of the user's first booking, will be 'NaN' in event the user did not book
gender	the gender of the user, if known
age	the age of the user, if known
signup_method	the user's method for account signup
signup_flow	
language	the user's selected language
affiliate_channel	indicates the channel through which the user was directed to the website
affiliate_provider	indicates the provider of the channel, if applicable
first_affiliate_tracked	indicates the first affiliate through which the user was tracked, if applicable
signup_app	indicates which app the user used to create an account
first_device_type	indicates the first device identified for the user
first_browser	indicates the first browser identified for the user
country_destination	the target variable; the ultimate destination selection by the user

4 Description of Approach

4.1 Initial Pipeline

To start, we plan to construct an initial, bare-bones pipeline that will use only the (users) training dataset, while skipping all other datasets (e.g., sessions), to train a single classifier (e.g., KNN) and then generate an accuracy score on the test dataset. In this initial pipeline, we will omit more complex operations, such as data processing and feature engineering, with the goal of generating results that can be submitted to Kaggle for evaluation using the private test data. In addition, the bare-bones pipeline will establish baseline metrics against which to benchmark more complex models[6].

4.2 Feature Engineering

For feature engineering, we will start by incorporating the remaining datasets (age_gender_bkts, countries, and sessions) into the training data for our classifiers. Inspired by the 2nd-place winner's strategy of creating 1312 features[7], we will look to derive significantly more features from the couple dozen in the raw data. As noted by the 3rd-place winner, there is room for creativity in feature engineering, since the training data is highly unbalanced with the 12 classes having overlap[9]. On the other hand, we will also look to discard unwanted features, since such features can degrade the performance of prediction models[8].

4.3 Classifiers

Being novice students of machine learning, we will build simple classifiers that we have experienced in class, such as KNN, decision tree, and NB. Time permitting, we will also explore more complex models, such as random forest and neural network. Since our objective is to practice machine learning, as opposed to competing in Kaggle, we will value breadth over depth in choosing to explore more classifiers rather than fine-tuning a select few.

4.4 Ensemble Learning

To improve on the accuracy of single classifiers, we will use the ensemble learning technique, Stacking, to combine the predictions from multiple classifiers using a meta-classifier. Stacking has been shown to be a popular approach for winning Kaggle competitions[5]. We plan to start by using a simple meta-classifier, such as Logistic Regression, and time permitting, we will evaluate more complex ones, such as XGBoost[12].

4.5 Validation

As mentioned in the article, Machine Learning: Validation Techniques[10], there are many validation techniques for estimating the population error rate. For this project, we will explore and utilize several of these techniques. To start, we will use the Holdout technique, where we isolate the test dataset from the training one; however, with this technique, these is risk of an uneven distribution of classes in the training and test dataset[10]. Other validation techniques being considered are random subsampling and bootstrapping. In general, using validation techniques will help both avoid and identify potential data leakage, which would undermine the quality of the models[11].

5 Setup and Data Cleaning

5.1 Import Libraries

```
In [1]: %matplotlib inline

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

5.2 Data files from Kaggle

We will be exploring each of the data files in turn. The below code will load all of the data files as data frames into a dictionary and then made a copy of that dictionary. One of the dictionaries will be used as our raw representation of the data whereas the other will be the final, cleaned representation. This is to prevent mutation of the data and allow error-free, partial re-execution of this notebook.

```
In [2]: DATA_PATH = './data/extracted'
        dfs_raw = {}
        dfs = {}
        for root, dirs, files in os.walk(DATA_PATH):
            for file in files:
                dfs_raw[file.split('.')[0]] = pd.read_csv(f'{DATA_PATH}/{file}')
                dfs = dfs_raw.copy()
                print(file)
```

```
age_gender_bkts.csv
countries.csv
sample_submission_NDF.csv
sessions.csv
test_users.csv
train_users_2.csv
```

5.3 Dataset: age_gender_bkts

Description: This file contains demographic information for each of the possible destination countries. The demographics are bucketed into age ranges with gender and population count by year.

Relevance: Demographic information of the destination of the destination country might be correlated to the gender and age of the user.

```
In [3]: dfs_raw['age_gender_bkts'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 420 entries, 0 to 419
Data columns (total 5 columns):
```

```
age_bucket                  420 non-null object
country_destination         420 non-null object
gender                      420 non-null object
population_in_thousands     420 non-null float64
year                        420 non-null float64
dtypes: float64(2), object(3)
memory usage: 16.5+ KB
```

Change the coding on 'year' to ensure that it is represented properly in our analyses.

```
In [4]: dfs['age_gender_bkts'].year = dfs_raw['age_gender_bkts'].year.astype(int)
```

```
In [5]: dfs['age_gender_bkts'].head()
```

```
Out[5]:   age_bucket country_destination gender  population_in_thousands  year
        0       100+                  AU   male                      1.0  2015
        1      95-99                  AU   male                      9.0  2015
        2      90-94                  AU   male                     47.0  2015
        3      85-89                  AU   male                    118.0  2015
        4      80-84                  AU   male                    199.0  2015
```

5.4 Dataset: countries

Description: Information about the destination countries, including the location and language spoken at each of the countries as well as the distance of the location and language compared to the USA (origin country). The language codes need to be modified in order to match with the language codes used in the user datasets.

Relevance: + Countries may be clustered by similarity to each other and by dissimilarity to the origin country. + Specific characteristics of the traveler and destination might have a correlation.

```
In [6]: dfs_raw['countries'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 7 columns):
country_destination              10 non-null object
lat_destination                  10 non-null float64
lng_destination                  10 non-null float64
distance_km                      10 non-null float64
destination_km2                  10 non-null float64
destination_language             10 non-null object
language_levenshtein_distance    10 non-null float64
dtypes: float64(5), object(2)
memory usage: 640.0+ bytes
```

```
In [7]: dfs['countries']
```

```
Out[7]:   country_destination  lat_destination  lng_destination  distance_km  \
        0                  AU       -26.853388       133.275160   15297.7440
        1                  CA        62.393303       -96.818146    2828.1333
        2                  DE        51.165707        10.452764    7879.5680
        3                  ES        39.896027        -2.487694    7730.7240
        4                  FR        46.232193         2.209667    7682.9450
        5                  GB        54.633220        -3.432277    6883.6590
        6                  IT        41.873990        12.564167    8636.6310
        7                  NL        52.133057         5.295250    7524.3203
        8                  PT        39.553444        -7.839319    7355.2534
        9                  US        36.966427       -95.844030       0.0000

           destination_km2 destination_language   language_levenshtein_distance
        0        7741220.0                  eng                            0.00
        1        9984670.0                  eng                            0.00
        2         357022.0                  deu                           72.61
        3         505370.0                  spa                           92.25
        4         643801.0                  fra                           92.06
        5         243610.0                  eng                            0.00
        6         301340.0                  ita                           89.40
        7          41543.0                  nld                           63.22
        8          92090.0                  por                           95.45
        9        9826675.0                  eng                            0.00
```

Modify the language codes so that they match with the user datasets.

```
In [8]: dfs['countries']['destination_language '] = pd.Series(['en', 'en', 'de', 'es', 'fr', '
```

5.5 Dataset: sample_submission_NDF

Description: The results of our analysis should match the format of this file.
Relevance: This is not relevant to the analysis.

```
In [9]: dfs['sample_submission_NDF'].head()
```

```
Out[9]:            id country
        0  5uwns89zht     NDF
        1  jtl0dijy2j     NDF
        2  xx0ulgorjt     NDF
        3  6c6puo6ix0     NDF
        4  czqhjk3yfe     NDF
```

5.6 Dataset: sessions

Description: User session data on the Airbnb website. A session is a sequence of actions performed on the website. + secs_elapsed = The amount of time between that action and the prior action. + There is no session_id column and some of the secs_elapsed columns are extremely long. We will assume that all visits to Airbnb are aggregated into a single session per user. The large secs_elapsed are the intervals between user visits to Airbnb. + The dataset does not tell us what searches the user performed (ie. related to a destination), only that a user was searching or interacting with the Airbnb platform in some way. + device_type = Device that the user performed the action from + If the device changes over time is the user more engaged in making a booking through AirBnB?

Relevance: + It's hard to pinpoint the "hard" relevance of this dataset. It could be used to develop an understanding of user interest/engagement or to identify whether a user has specific concerns/requirements related to their destination. + Some actions appear interesting: "view_ghosting_reasons", "special_offer_field", "airbnb_picks_wishlists"

```
In [10]: dfs_raw['sessions'].info(null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10567737 entries, 0 to 10567736
Data columns (total 6 columns):
user_id          10533241 non-null object
action           10488111 non-null object
action_type      9441533 non-null object
action_detail    9441533 non-null object
device_type      10567737 non-null object
secs_elapsed     10431706 non-null float64
dtypes: float64(1), object(5)
memory usage: 483.8+ MB
```

Clean the 34,496 null user_id's since the sessions data can't be joined to the users data without it.

```
In [11]: dfs['sessions'] = dfs_raw['sessions'].dropna(subset=['user_id'])

In [12]: dfs['sessions'].head()
```

```
Out[12]:       user_id          action action_type        action_detail  \
         0  d1mm9tcy42          lookup         NaN                  NaN
         1  d1mm9tcy42  search_results       click  view_search_results
         2  d1mm9tcy42          lookup         NaN                  NaN
         3  d1mm9tcy42  search_results       click  view_search_results
         4  d1mm9tcy42          lookup         NaN                  NaN

                device_type  secs_elapsed
         0  Windows Desktop         319.0
         1  Windows Desktop       67753.0
         2  Windows Desktop         301.0
         3  Windows Desktop       22141.0
         4  Windows Desktop         435.0
```

5.7 Dataset: train_users_2

Description: This dataset contains the main training data. Each row is a user profile and contains basic information as well the chosen destination. + date_first_booking has NaN values. We'll accept these into the dataset since they correspond to NDF destinations. + first_affiliate_tracked also has a significant number of NaN. + Age has bad values as well as NaN. The NaN we will leave in since they comprise a significant portion of our training set and we expect that to be representative of real data that we encounter. Our classifier should be capable of predicting users with NaN age.

Relevance: We will join the other datasets into this one on id and country.

```
In [13]: dfs_raw['train_users_2'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 213451 entries, 0 to 213450
Data columns (total 16 columns):
id                         213451 non-null object
date_account_created       213451 non-null object
timestamp_first_active     213451 non-null int64
date_first_booking         88908 non-null object
gender                     213451 non-null object
age                        125461 non-null float64
signup_method              213451 non-null object
signup_flow                213451 non-null int64
language                   213451 non-null object
affiliate_channel          213451 non-null object
affiliate_provider         213451 non-null object
first_affiliate_tracked    207386 non-null object
signup_app                 213451 non-null object
first_device_type          213451 non-null object
first_browser              213451 non-null object
country_destination        213451 non-null object
dtypes: float64(1), int64(2), object(13)
memory usage: 26.1+ MB
```

```
In [14]: dfs['train_users_2'].head()
```

```
Out[14]:            id date_account_created  timestamp_first_active date_first_booking  \
         0  gxn3p5htnn           2010-06-28          20090319043255                NaN
         1  820tgsjxq7           2011-05-25          20090523174809                NaN
         2  4ft3gnwmtx           2010-09-28          20090609231247         2010-08-02
         3  bjjt8pjhuk           2011-12-05          20091031060129         2012-09-08
         4  87mebub9p4           2010-09-14          20091208061105         2010-02-18

               gender   age signup_method  signup_flow language affiliate_channel  \
         0  -unknown-   NaN      facebook            0       en            direct
         1       MALE  38.0      facebook            0       en               seo
         2     FEMALE  56.0         basic            3       en            direct
         3     FEMALE  42.0      facebook            0       en            direct
         4  -unknown-  41.0         basic            0       en            direct

           affiliate_provider first_affiliate_tracked signup_app first_device_type  \
```

```
0           direct                untracked       Web       Mac Desktop
1           google                untracked       Web       Mac Desktop
2           direct                untracked       Web   Windows Desktop
3           direct                untracked       Web       Mac Desktop
4           direct                untracked       Web       Mac Desktop

  first_browser country_destination
0        Chrome                 NDF
1        Chrome                 NDF
2            IE                  US
3       Firefox               other
4        Chrome                  US
```

5.8 Cleaning: age

The oldest person in the world was 122 year old. Our dataset contains 781 entries where the age is >122 years old. Realistically we don't expect there to be many people anywhere near 122 years old since this is a travel and ecommerce dataset, but we can't rule out the possibility.

Likewise, the minimum age to use Airbnb is 18. There are 158 users in our dataset with an age less than that.

```
In [15]: dfs_raw['train_users_2'].query('age > 122').age.count()

Out[15]: 781

In [16]: dfs_raw['train_users_2'].query('age <18').age.count()

Out[16]: 158

In [17]: dfs['train_users_2'] = dfs['train_users_2'][(dfs['train_users_2'].age.isnull()) | ((d
```

5.9 Cleaning: Date Account Created

29 accounts have the date_first_booking before the date_account_created. These should be excluded.

```
In [18]: dfs_raw['train_users_2'].query('date_account_created > date_first_booking').id.count(

Out[18]: 29

In [19]: dfs['train_users_2'].loc[:,'date_first_booking'] = pd.to_datetime(dfs_raw['train_user
         dfs['train_users_2'].loc[:,'date_account_created'] = pd.to_datetime(dfs_raw['train_us

C:\Users\sye\AppData\Local\
Continuum\anaconda3\envs\mids-w203\lib
\site-
packages\pandas\core\indexing.py:543:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

```
In [20]: dfs['train_users_2'] = dfs['train_users_2'][(dfs['train_users_2'].date_first_booking.i
```

5.10 Cleaning: timestamp_first_active

178 rows have a timestamp_first_active > date_account_created. Since timestamp_first_active seems to indicate the first time that a user was tracked on Airbnb (before creating an account), we'll omit the relatively few rows where this doesn't hold true.

```
In [21]: (pd.to_datetime(dfs_raw['train_users_2'].date_account_created, format='%Y-%m-%d', erro
```

```
Out[21]: True     213273
         False       178
         dtype: int64
```

```
In [22]: dfs['train_users_2']['delta_creation_active'] = dfs['train_users_2']['date_account_cre
```

```
In [23]: dfs['train_users_2'] = dfs['train_users_2'][dfs['train_users_2'].delta_creation_active
```

6 Exploratory Data Analysis

6.1 Univariate exploration

Observations + age-gender-bkts + The age_bucket is top-coded at 100+ + All data is for the year 2015. If we use this data for analysis then we need to assume that the demographic trends hold for all years in our user dataset. + train_user_2 + A significant number of gender values are non-binary. We should be careful to transform the gender column into separate categorical features for each of the options. + ~half of the dataset did not choose a destination, most of the rest went to the US. This will make it difficult to identify if people are going to specific foreign countries since the data related to those outcomes is relatively limited.

```
In [24]: dfs['age_gender_bkts'].describe(include='all')
```

Out[24]:

	age_bucket	country_destination	gender	population_in_thousands	year
count	420	420	420	420.000000	420.0
unique	21	10	2	NaN	NaN
top	90-94	US	female	NaN	NaN
freq	20	42	210	NaN	NaN
mean	NaN	NaN	NaN	1743.133333	2015.0
std	NaN	NaN	NaN	2509.843202	0.0
min	NaN	NaN	NaN	0.000000	2015.0
25%	NaN	NaN	NaN	396.500000	2015.0
50%	NaN	NaN	NaN	1090.500000	2015.0
75%	NaN	NaN	NaN	1968.000000	2015.0
max	NaN	NaN	NaN	11601.000000	2015.0

In [25]:
```python
dfs['age_gender_bkts'].describe(include='all')
```

Out[25]:

	age_bucket	country_destination	gender	population_in_thousands	year
count	420	420	420	420.000000	420.0
unique	21	10	2	NaN	NaN
top	90-94	US	female	NaN	NaN
freq	20	42	210	NaN	NaN
mean	NaN	NaN	NaN	1743.133333	2015.0
std	NaN	NaN	NaN	2509.843202	0.0
min	NaN	NaN	NaN	0.000000	2015.0
25%	NaN	NaN	NaN	396.500000	2015.0
50%	NaN	NaN	NaN	1090.500000	2015.0
75%	NaN	NaN	NaN	1968.000000	2015.0
max	NaN	NaN	NaN	11601.000000	2015.0

In [26]:
```python
dfs['sessions'].describe(include='all')
```

Out[26]:

	user_id	action	action_type	action_detail	device_type \
count	10533241	10453761	9410284	9410284	10533241
unique	135483	359	10	155	14
top	mxqbh3ykxl	show	view	view_search_results	Mac Desktop
freq	2722	2758985	3549375	1771026	3585886
mean	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN

	secs_elapsed
count	1.039776e+07
unique	NaN
top	NaN
freq	NaN
mean	1.941124e+04
std	8.890920e+04
min	0.000000e+00
25%	2.290000e+02
50%	1.146000e+03
75%	8.442000e+03
max	1.799977e+06

In [27]:
```python
dfs['train_users_2'].describe(include='all')
```

Out[27]:

	id	date_account_created	timestamp_first_active \
count	212336	212336	2.123360e+05
unique	212336	1633	NaN
top	2vxljiqxjt	2014-05-13 00:00:00	NaN

```
freq            1                 671                   NaN
first         NaN 2010-01-01 00:00:00                   NaN
last          NaN 2014-06-30 00:00:00                   NaN
mean          NaN                 NaN          2.013088e+13
std           NaN                 NaN          9.231791e+09
min           NaN                 NaN          2.010010e+13
25%           NaN                 NaN          2.012123e+13
50%           NaN                 NaN          2.013091e+13
75%           NaN                 NaN          2.014031e+13
max           NaN                 NaN          2.014063e+13

         date_first_booking    gender            age signup_method  \
count                 88387    212336  124367.000000        212336
unique                 1976         4            NaN             3
top     2014-05-22 00:00:00 -unknown-            NaN         basic
freq                    247     95573            NaN        152068
first   2010-01-02 00:00:00       NaN            NaN           NaN
last    2015-06-29 00:00:00       NaN            NaN           NaN
mean                    NaN       NaN      37.440261           NaN
std                     NaN       NaN      13.933221           NaN
min                     NaN       NaN      18.000000           NaN
25%                     NaN       NaN      28.000000           NaN
50%                     NaN       NaN      34.000000           NaN
75%                     NaN       NaN      43.000000           NaN
max                     NaN       NaN     115.000000           NaN

          signup_flow language affiliate_channel affiliate_provider  \
count   212336.000000   212336            212336             212336
unique            NaN       25                 8                 18
top               NaN       en            direct             direct
freq              NaN   205261            137041             136745
first             NaN      NaN               NaN                NaN
last              NaN      NaN               NaN                NaN
mean         3.273722      NaN               NaN                NaN
std          7.646294      NaN               NaN                NaN
min          0.000000      NaN               NaN                NaN
25%          0.000000      NaN               NaN                NaN
50%          0.000000      NaN               NaN                NaN
75%          0.000000      NaN               NaN                NaN
max         25.000000      NaN               NaN                NaN

       first_affiliate_tracked signup_app first_device_type first_browser  \
count                   206331     212336            212336        212336
unique                       7          4                 9            52
top                  untracked        Web       Mac Desktop        Chrome
freq                    108706     181692             89147         63526
first                      NaN        NaN               NaN           NaN
last                       NaN        NaN               NaN           NaN
```

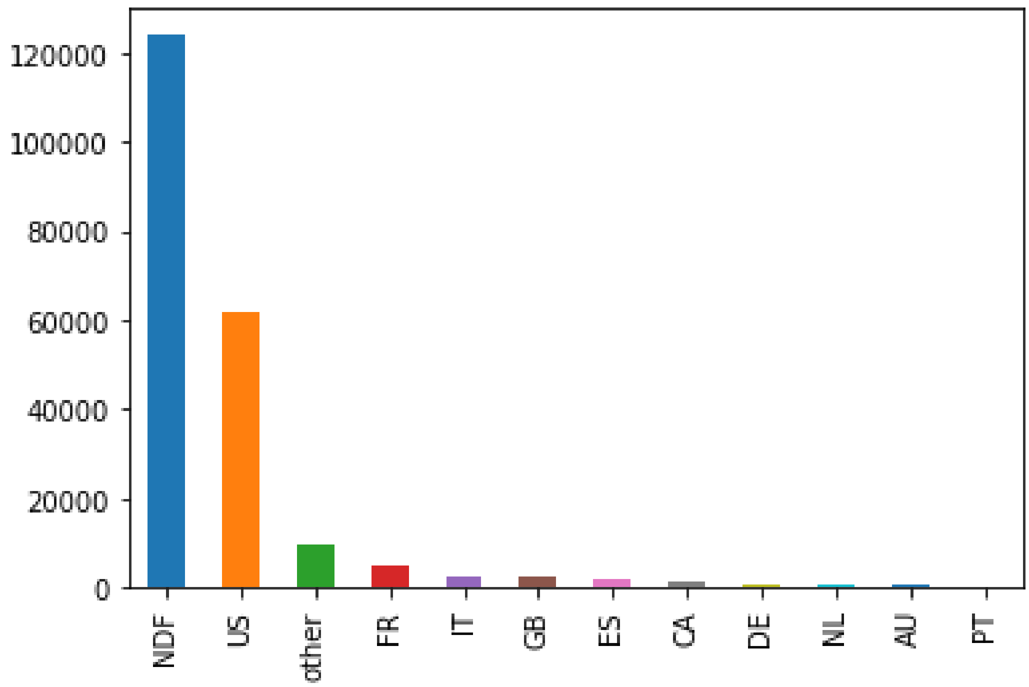
```
mean                          NaN        NaN                NaN            NaN
std                           NaN        NaN                NaN            NaN
min                           NaN        NaN                NaN            NaN
25%                           NaN        NaN                NaN            NaN
50%                           NaN        NaN                NaN            NaN
75%                           NaN        NaN                NaN            NaN
max                           NaN        NaN                NaN            NaN

       country_destination     delta_creation_active
count               212336                    212336
unique                  12                       NaN
top                    NDF                       NaN
freq                123949                       NaN
first                  NaN                       NaN
last                   NaN                       NaN
mean                   NaN  -1 days +11:15:26.392533
std                    NaN    0 days 08:04:01.427727
min                    NaN         -1 days +00:00:01
25%                    NaN         -1 days +03:50:46
50%                    NaN         -1 days +08:16:32
75%                    NaN         -1 days +19:21:05
max                    NaN         -1 days +23:59:59
```

```
In [28]: dfs['train_users_2']['gender'].value_counts()
```

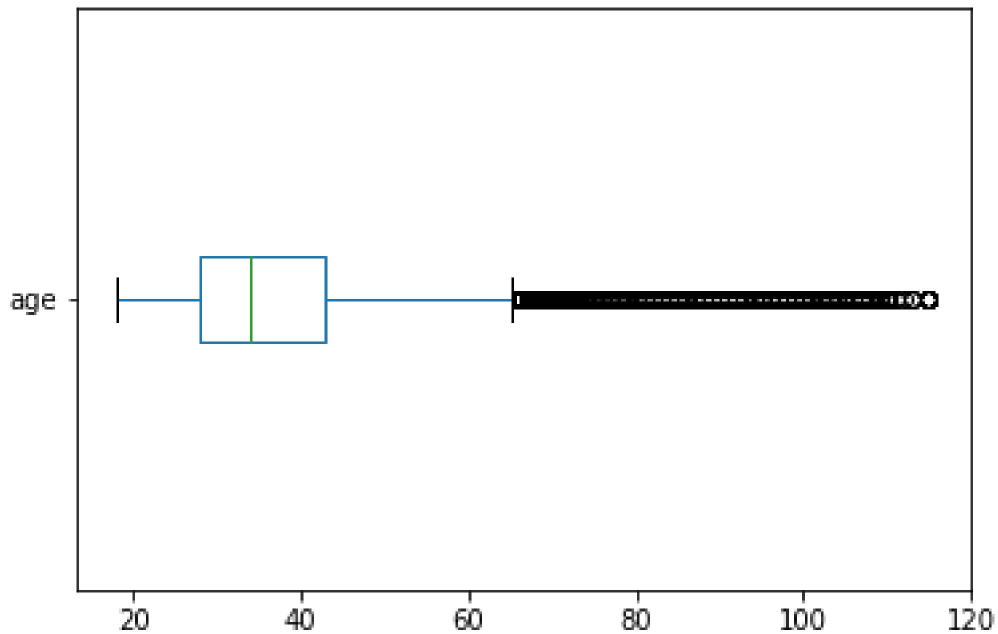
```
Out[28]: -unknown-    95573
         FEMALE       62417
         MALE         54070
         OTHER          276
         Name: gender, dtype: int64
```

```
In [29]: dfs['train_users_2'].country_destination.value_counts().plot.bar()
```

```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x156025f4a90>
```




```
In [30]: dfs['train_users_2'].age.describe()

Out[30]: count    124367.000000
         mean         37.440261
         std          13.933221
         min          18.000000
         25%          28.000000
         50%          34.000000
         75%          43.000000
         max         115.000000
         Name: age, dtype: float64

In [31]: dfs['train_users_2'].age.plot.box(vert=False)

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x156016cee48>
```



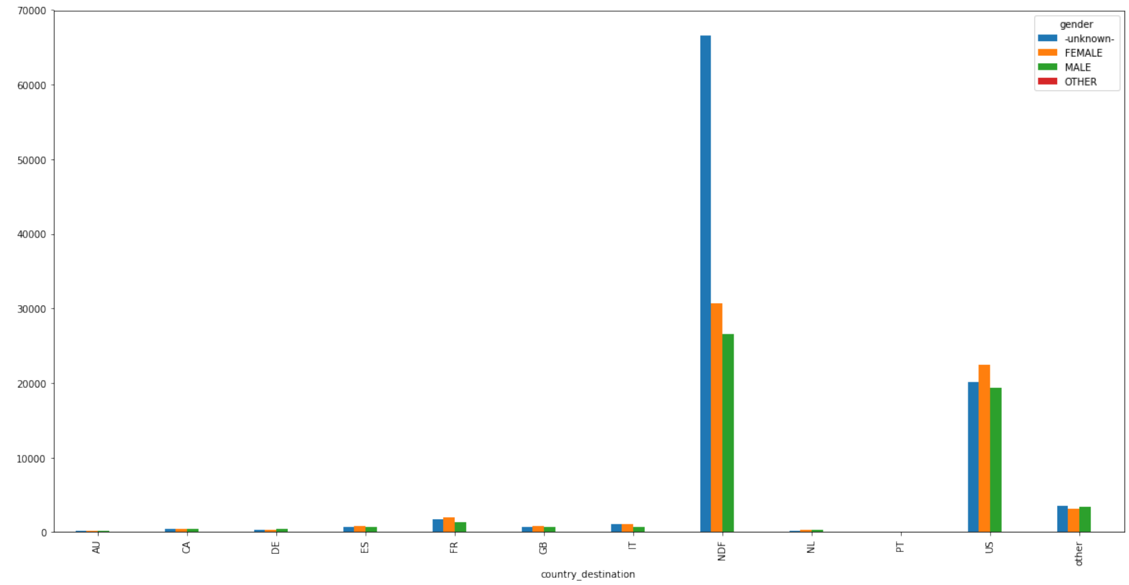
6.2 Multivariate exploration

Observations: + unknown gender and NDF are highly correlated + When a destination_country speaks the same language as the user, there is an increased probability that the user will go to that country. However, users overwhelming are going to English speaking destinations, regardless of their chosen language. + When a user is young, they are more likely to be female, whereas when they are older, there is equal probability of male or female.

Gender and country destination

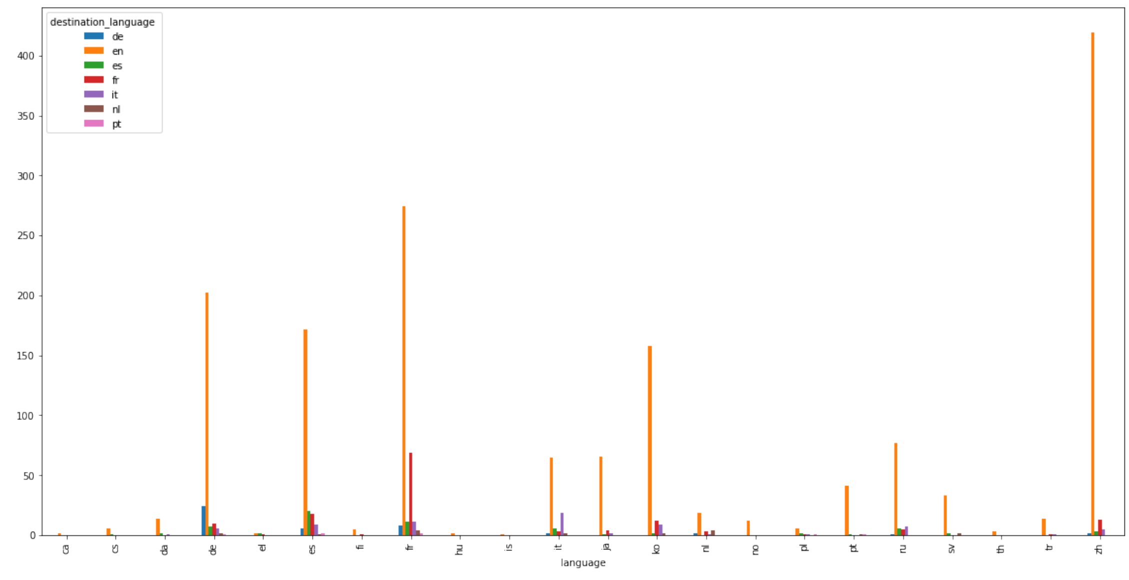
```
In [32]: plt.rcParams['figure.figsize'] = [20, 10]
         pd.crosstab(dfs['train_users_2']['country_destination'],dfs['train_users_2']['gender']
```

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1560173fbe0>
```



User language and language at destination (excluding language)

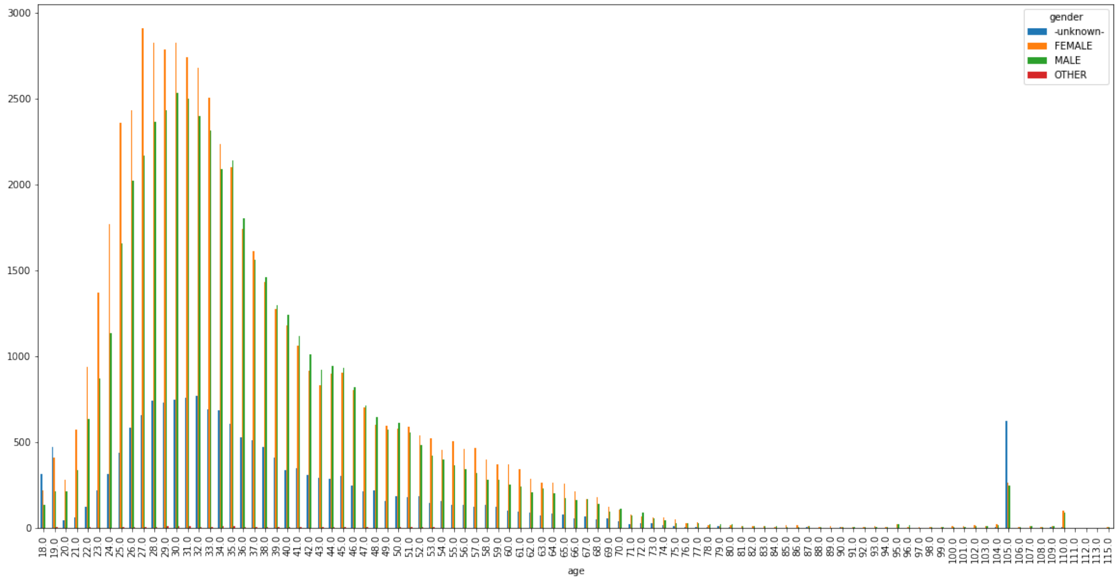
```
In [33]: joined_country = dfs['train_users_2'].join(dfs['countries'].set_index('country_destina
         joined_country_no_english = joined_country[joined_country['language'] != 'en']
         plt.rcParams['figure.figsize'] = [20, 10]
         ct = pd.crosstab(joined_country_no_english['language'],joined_country_no_english['des
```



Age and gender among users

```
In [34]: pd.crosstab(dfs['train_users_2']['age'],dfs['train_users_2']['gender']).plot.bar()
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x15601cea438>



7 Full Structure of Solution

- Code below is commented out due to being work-in-progress and not functional at the moment.

7.1 Initial Pipeline

In [1]:
```
# Step 1
# Load necessary Python packages.
# import pandas as pd
# from sklearn.neighbors import KNeighborsClassifier

# Step 2
# Read in training and test data from csv files.
# train_user = pd.read_csv('unzipped_data/train_users_2.csv')
# test_user = pd.read_csv('unzipped_data/test_users.csv')

# Step 3
# Split into data and labels (panda dataframes).
# train_data   = train_user.iloc[:, 0:-1]
# train_labels = train_user.iloc[:, -1:]
# test_data    = test_user.iloc[:, 0:-1]
# test_labels  = test_user.iloc[:, -1:]

# Step 4
```

```
# Clean and pre-process training data (e.g., filter missing/invalid entries).
# Use OneHotEncoding and StandardScaler on features in train_users_2 dataset.

# Step 5
# Train a single classifier (KNN) using only features in train_users_2 dataset.
# k = 20
# clf = KNeighborsClassifier(n_neighbors=k)
# clf.fit(train_data, train_labels)

# Step 6
# Get accuracy using test data.
# accuracy = clf.score(test_data, test_labels)

# Step 7
# Generate predictions for test data to submit to Kaggle for scoring.
# predictions = clf.predict(test_data)
```

7.2 Final Pipeline

In [2]:
```
# Step 1
# Load necessary Python packages.

# Step 2
# Read in all given csv files for training and test.

# Step 3
# Split into data and labels (panda dataframes).
# train_data   = train_user.iloc[:, 0:-1]
# train_labels = train_user.iloc[:, -1:]
# test_data    = test_user.iloc[:, 0:-1]
# test_labels  = test_user.iloc[:, -1:]

# Step 4
# Clean training and test data (e.g., filter missing/invalid entries).

# Step 5
# Explore alternate encoding strategies compared to OneHotEncoder and StandardScaler u
# Create pipelines that can test strategies and be applied with multiple classifiers.
# Pipelines also simplify usage of test data by ensuring that it receives equal proces

# Step 6
# Use feature engineering to create new features from all training data: users, sessio
# Apply feature engineering to create same set of new features for both training and t
# Examples of new features: transform age features into buckets, calculate lag feature

# Step 7
# Train several basic classifiers: KNN, decision tree, NB, etc.
# Time permitting, train more complex classifiers: random forest, neural network, etc.
```

```
# Step 8
# Ensemble Learning - add meta-classifier to combine predictions of all classifiers fr

# Step 9
# Validate prediction model using techniques: random subampling and bootstrapping.
# Resolve any issues found (e.g., data leakage).

# Step 10
# Get accuracy using test data.

# Step 11
# Generate predictions for test data to submit to Kaggle for scoring.
```

8 References

[1] "Airbnb New User Bookings - Description" Kaggle, 24 Nov. 2015, https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings#description.

[2] Molnar, Christoph "Feature Importance" Interpretable Machine Learning, https://christophm.github.io/interpretable-ml-book/feature-importance.html.

[3] Fisher, Rudin, and Dominici "All Models are Wrong but many are Useful: Variable Importance for Black-Box, Proprietary, or Misspecified Prediction Models, using Model Class Reliance" arXiv, https://arxiv.org/abs/1801.01489v3.

[4] "Airbnb New User Bookings - Evaluation" Kaggle, 24 Nov. 2015, https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings#evaluation.

[5] Smolyakov, Vadim. "Ensemble Learning to Improve Machine Learning Results." Stats and Bots, 22 Aug. 2017, blog.statsbot.co/ensemble-learning-d1dcd548e936.

[6] Zinkevich, M. 2017. Rules of Machine Learning: Best Practices for ML Engineering. http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf.

[7] "Airbnb New User Bookings, Winner's Interview: 2nd Place, Keiichi Kuroyanagi (@Keiku)." No Free Hunch, 15 Mar. 2016, blog.kaggle.com/2016/03/17/airbnb-new-user-bookings-winners-interview-2nd-place-keiichi-kuroyanagi-keiku/.

[8] Shekhar, Amit. "What Is Feature Engineering for Machine Learning? – MindOrks – Medium." Medium.com, Medium, 14 Feb. 2018, medium.com/mindorks/what-is-feature-engineering-for-machine-learning-d8ba3158d97a.

[9] "Airbnb New User Bookings, Winners Interview: 3rd Place: Sandro Vega Pons." No Free Hunch, 15 Mar. 2016, blog.kaggle.com/2016/03/07/airbnb-new-user-bookings-winners-interview-3rd-place-sandro-vega-pons/.

[10] Kumar, Ajitesh. "Machine Learning: Validation Techniques - DZone AI." Dzone.com, 12 Feb. 2018, dzone.com/articles/machine-learning-validation-techniques.

[11] Brownlee, Jason. "Data Leakage in Machine Learning." Machine Learning Mastery, 31 July 2018, machinelearningmastery.com/data-leakage-machine-learning/.

[12] Brownlee, Jason. "A Gentle Introduction to XGBoost for Applied Machine Learning." Machine Learning Mastery, 21 Sept. 2016, machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.