Model_SessionsData_new

August 25, 2020

1 Final Project - MSCA 31008 Data Mining Principles

Topic: Kaggle Challenge: Airbnb New User Bookings **Goal**: Where will a new guest book their first travel experience? **Link**:https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/overview/description **Instructor**: Anil Chaturvedi

1.0.1 Topics Covered

1.1 Importing the files and loading the dataset 1.2 EDA 1.3 Feature Engineering 1.4 Evalutation Metric by Kaggle 1.5 Understanding what makes a new user books a place 1.6 Models 1.6.1 Random Forest 1.6.2 KNN 1.6.3 XGBOOST 1.6.4 Ensemble Model 1.7 Preparing data for submission 1.8 Challenges 1.9 Future Work

NOTE: The XGBoost model gives a private score of 0.88138 and public score of 0.87503

1.0.2 1.1 Importing the files and loading the dataset

[3]:
```
#loading the libraries
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```
#Importing all the datasets
#Training Set of users
train_users=pd.read_csv('/Users/devanshiverma/Desktop/UChicago/02-Quarter/Data/
 ↪train_users_2.csv')
#Testing Set of users
test_users=pd.read_csv('/Users/devanshiverma/Desktop/UChicago/02-Quarter/Data/
 ↪test_users.csv')
#web sessions log for users
sessions=pd.read_csv('/Users/devanshiverma/Desktop/UChicago/02-Quarter/Data/
 ↪sessions.csv')
#Loading the summary stats of countries
countries=pd.read_csv('/Users/devanshiverma/Desktop/UChicago/02-Quarter/Data/
 ↪countries.csv')
#Summary stats of user age
age_gender_bkts=pd.read_csv('/Users/devanshiverma/Desktop/UChicago/02-Quarter/
 ↪Data/age_gender_bkts.csv')
#Sample submission
sample_submission=pd.read_csv('/Users/devanshiverma/Desktop/UChicago/02-Quarter/
 ↪Data/sample_submission_NDF.csv')
```

[5]:
```
#converting the date values to datetime
train_users['date_account_created'] = pd.
 ↪to_datetime(train_users['date_account_created'])
train_users['date_first_booking'] = pd.
 ↪to_datetime(train_users['date_first_booking'])
train_users['timestamp_first_active'] = pd.
 ↪to_datetime(train_users['timestamp_first_active'], format='%Y%m%d%H%M%S')
test_users['date_first_booking'] = pd.
 ↪to_datetime(test_users['date_first_booking'])
test_users['date_account_created'] = pd.
 ↪to_datetime(test_users['date_account_created'])
test_users['timestamp_first_active'] = pd.
 ↪to_datetime(test_users['timestamp_first_active'], format='%Y%m%d%H%M%S')
train_users.head()
test_users.head()
```

[5]:

	id	date_account_created	timestamp_first_active	date_first_booking	\
0	5uwns89zht	2014-07-01	2014-07-01 00:00:06	NaT	
1	jtl0dijy2j	2014-07-01	2014-07-01 00:00:51	NaT	
2	xx0ulgorjt	2014-07-01	2014-07-01 00:01:48	NaT	
3	6c6puo6ix0	2014-07-01	2014-07-01 00:02:15	NaT	
4	czqhjk3yfe	2014-07-01	2014-07-01 00:03:05	NaT	

```
      gender   age signup_method  signup_flow language affiliate_channel  \
0     FEMALE  35.0      facebook            0       en            direct
1  -unknown-   NaN         basic            0       en            direct
2  -unknown-   NaN         basic            0       en            direct
3  -unknown-   NaN         basic            0       en            direct
4  -unknown-   NaN         basic            0       en            direct

  affiliate_provider first_affiliate_tracked signup_app first_device_type  \
0             direct               untracked      Moweb            iPhone
1             direct               untracked      Moweb            iPhone
2             direct                  linked        Web   Windows Desktop
3             direct                  linked        Web   Windows Desktop
4             direct               untracked        Web       Mac Desktop

   first_browser
0  Mobile Safari
1  Mobile Safari
2         Chrome
3             IE
4         Safari
```

[6]:
```
sessions.head()
```

[6]:
```
      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
```

[7]:
```
countries.head()
```

[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

   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
```

[8]:
```
age_gender_bkts.head()
```

[8]:
```
  age_bucket country_destination gender  population_in_thousands    year
0       100+                  AU   male                     1.0  2015.0
1      95-99                  AU   male                     9.0  2015.0
2      90-94                  AU   male                    47.0  2015.0
3      85-89                  AU   male                   118.0  2015.0
4      80-84                  AU   male                   199.0  2015.0
```

[9]:
```
sample_submission.head()
```

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

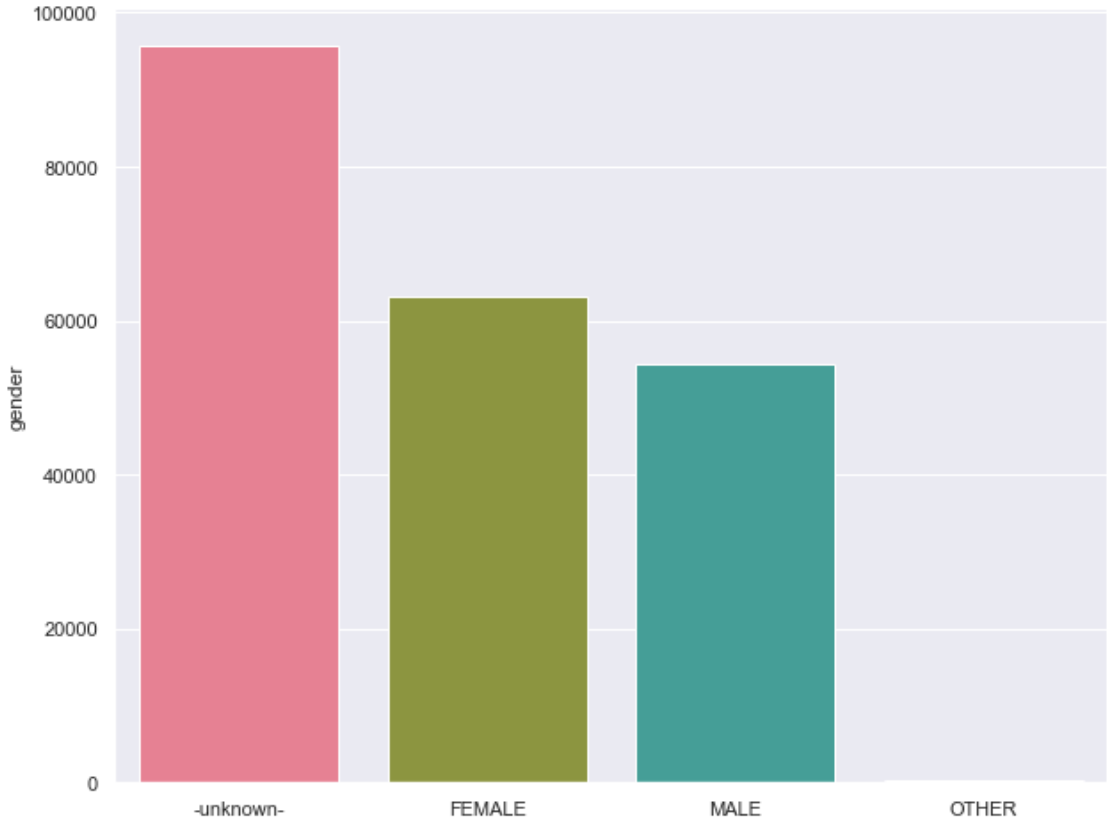
1.0.3 1.2 EDA

[10]:
```
#Function to get distribution graphs

def get_distr(feature):
    sns.set()
    plt.figure(figsize=(10,8))
    sns.barplot(train_users[feature].value_counts().index,train_users[feature].
 ↪value_counts(),palette="husl")
```

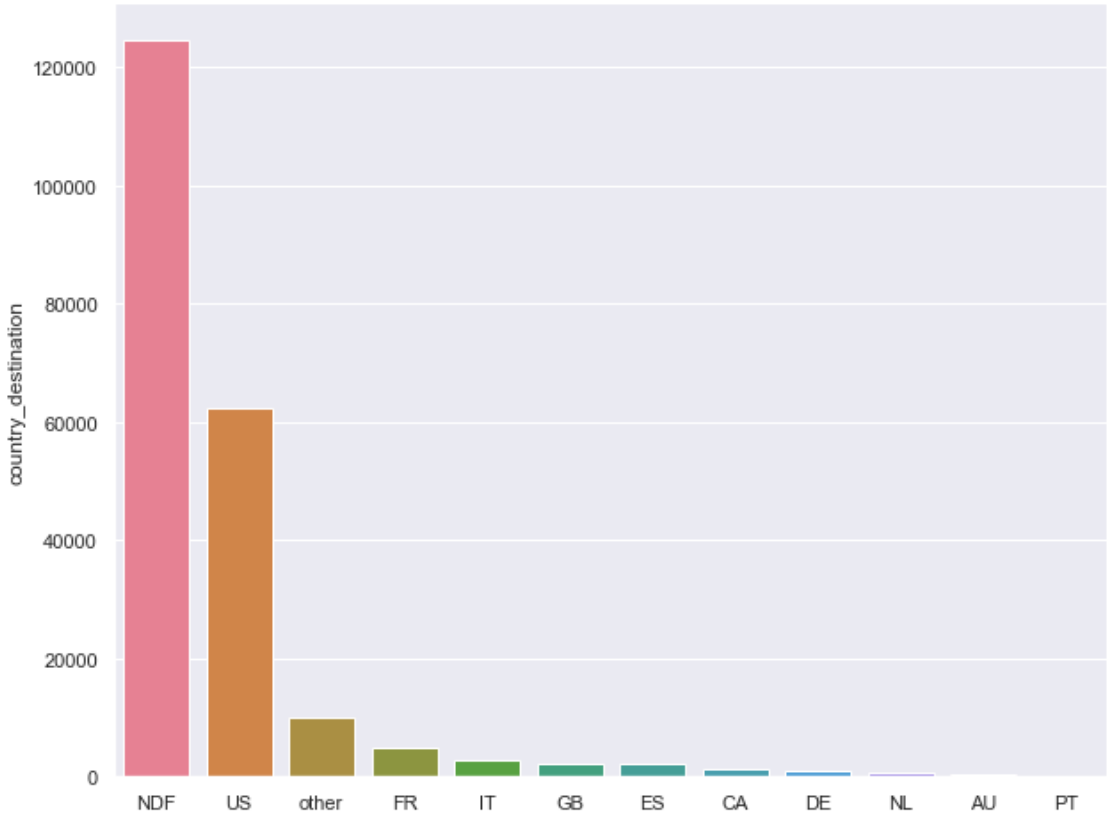
Gender

[11]:
```
get_distr('gender')
```



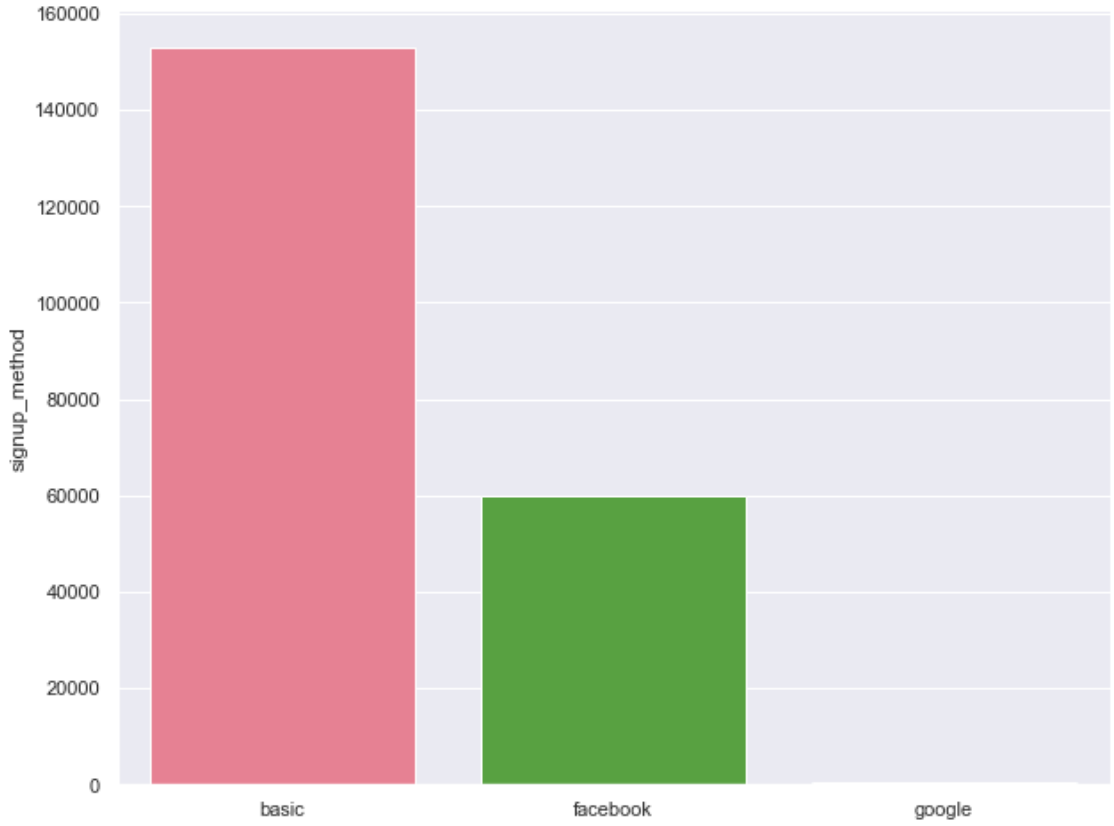
Desination Distribution

[12]:
```
get_distr('country_destination')
```



Signup Method

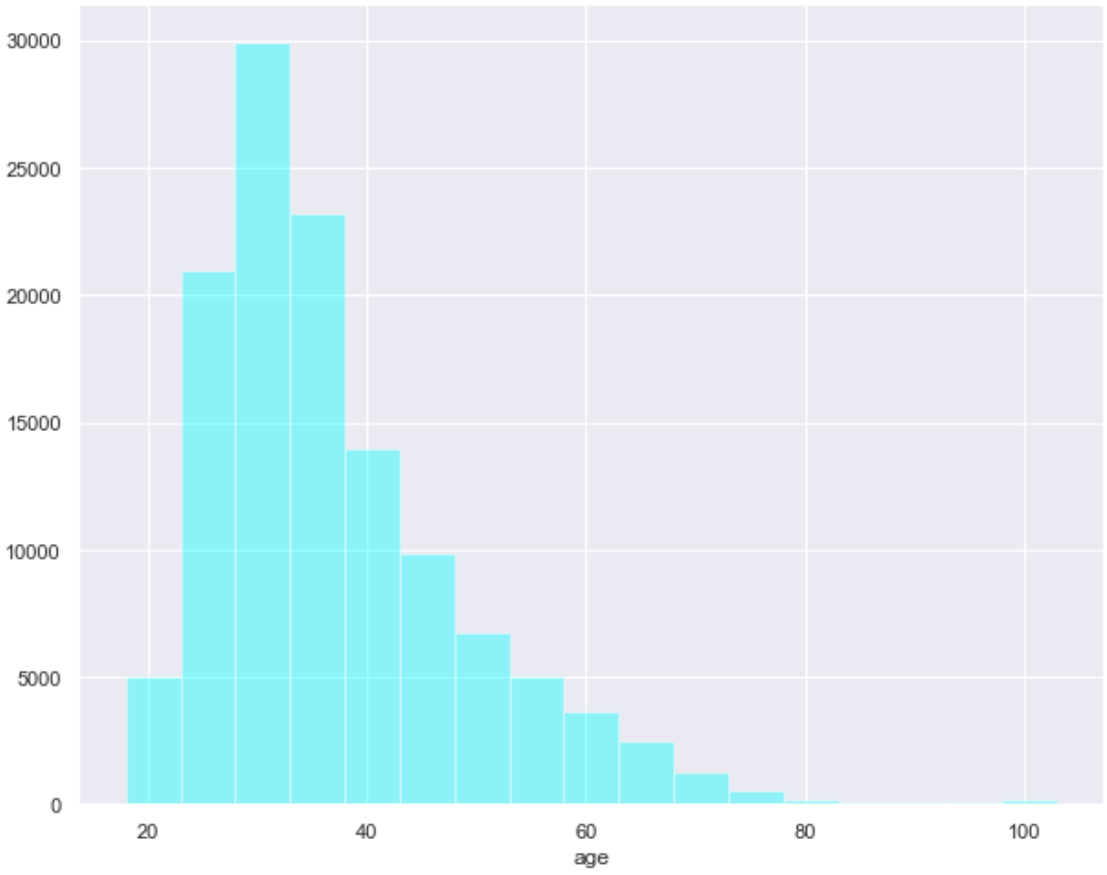
[13]:
```
get_distr('signup_method')
```



age distribution

[14]:
```
#get_distr('age')
sns.set()
plt.figure(figsize=(10,8))
sns.distplot(train_users['age'],color="cyan",bins=np.
 ↪arange(18,100+5,5),kde=False)
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8469b49890>
```

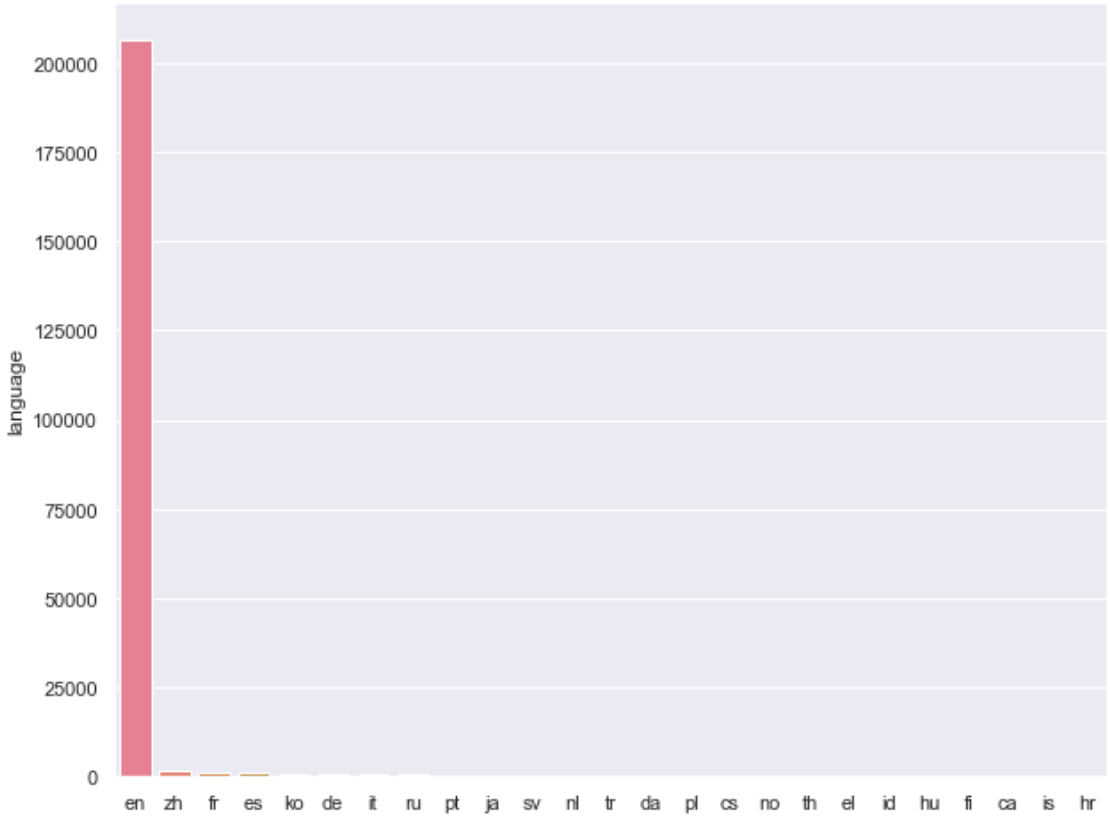


[15]:
```
train_users['age'].describe()
```

[15]:
```
count    125461.000000
mean         49.668335
std         155.666612
min           1.000000
25%          28.000000
50%          34.000000
75%          43.000000
max        2014.000000
Name: age, dtype: float64
```

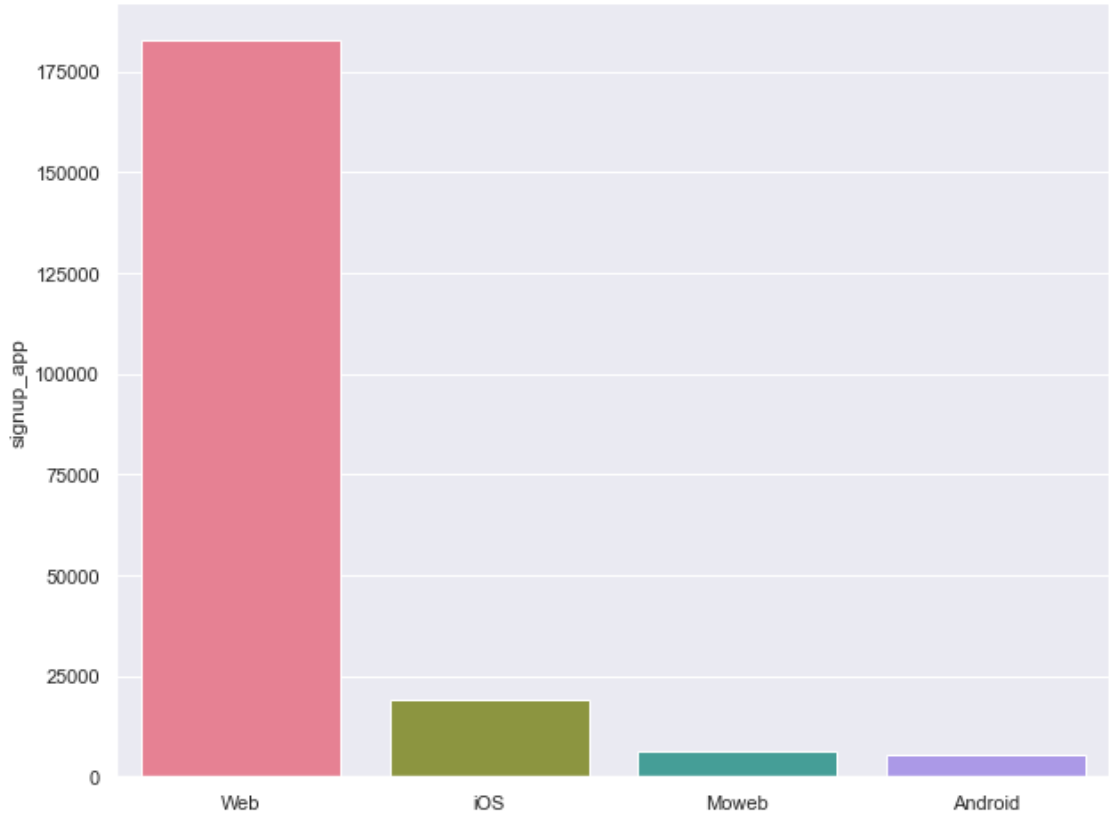
language distribution

[16]:
```
get_distr('language')
```



Signup app

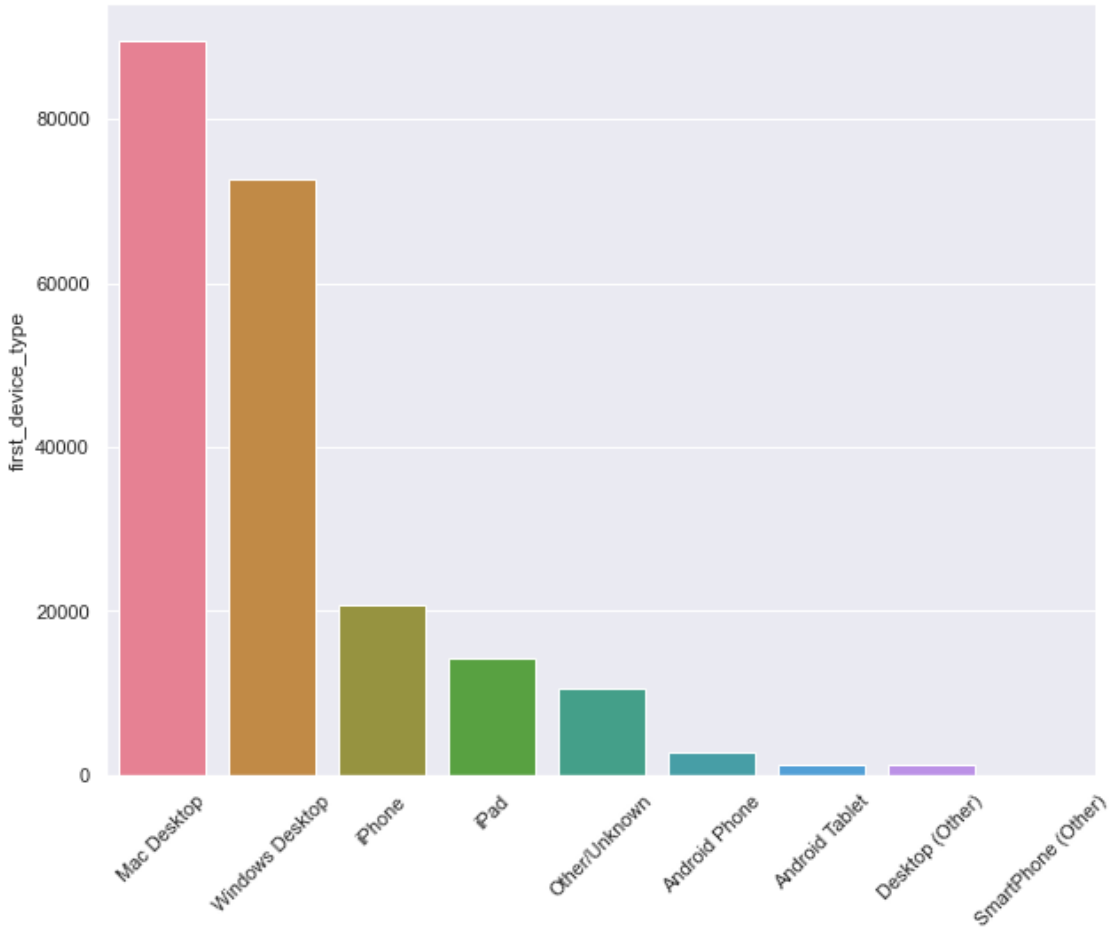
[17]:
```
get_distr('signup_app')
```



first device type

[18]:
```
sns.set()
plt.figure(figsize=(10,8))
sns.barplot(train_users['first_device_type'].value_counts().
 ↪index,train_users['first_device_type'].value_counts(),palette="husl")
plt.xticks(rotation=45)
```

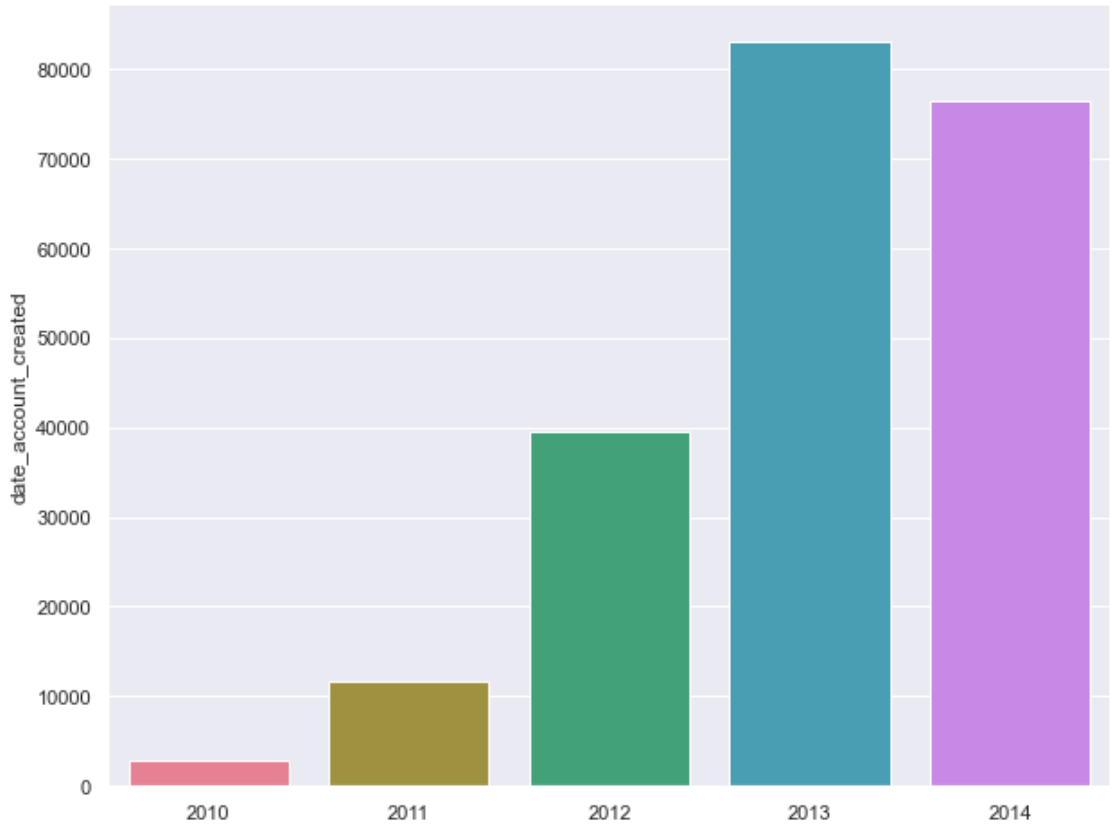
[18]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 <a list of 9 Text major ticklabel objects>)
```



When did the customers joined AirBnB?

[19]:
```
sns.set()
plt.figure(figsize=(10,8))
sns.barplot(train_users['date_account_created'].dt.year.value_counts().
 ↪index,train_users['date_account_created'].dt.year.
 ↪value_counts(),palette="husl")
```

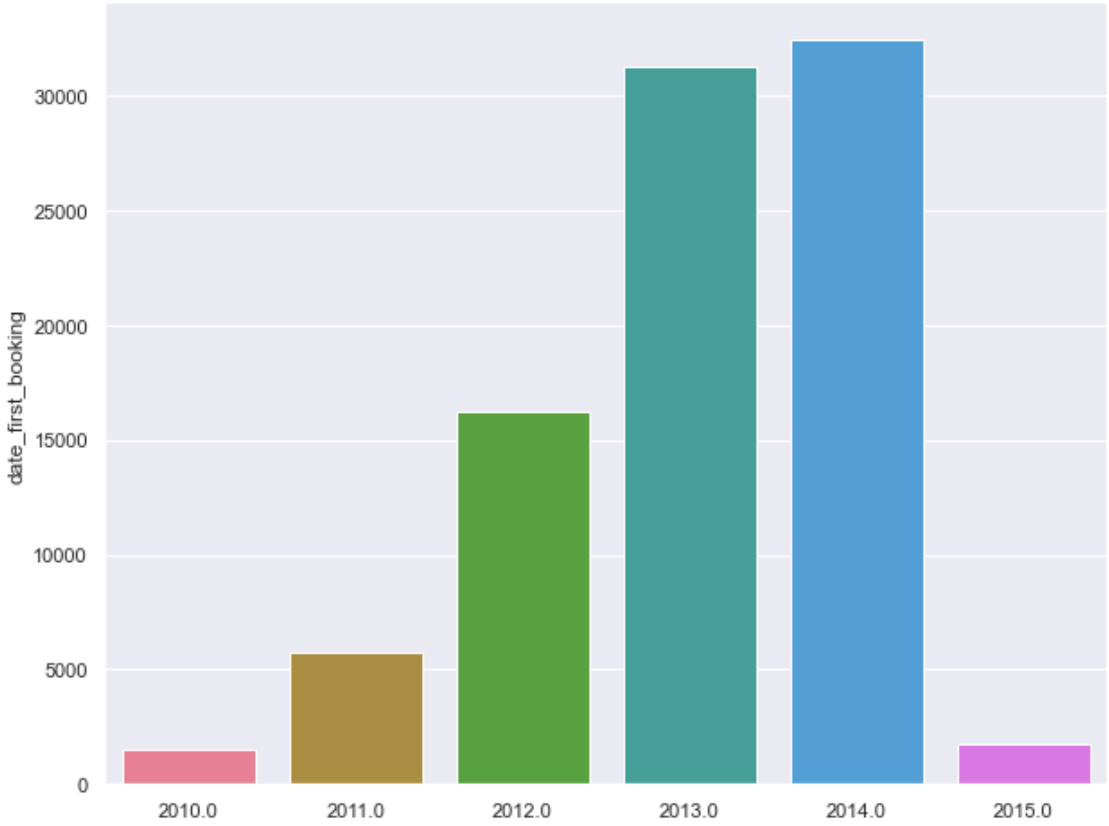
[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f846946cb90>
```



When did customers do there first booking?

[20]:
```
sns.set()
plt.figure(figsize=(10,8))
sns.barplot(train_users['date_first_booking'].dt.year.value_counts().
 ↪index,train_users['date_first_booking'].dt.year.
 ↪value_counts(),palette="husl")
```

[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f84695e49d0>
```



1.0.4 Inference from the data given above

- In terms of gender, females are more active on AirBnB then main. And majority of the users havent specfied their gender
- A major chunk of people just join the website but don't book a destination(NDF). US is the most popular destination followed by other locations, france and italy.
- Majority of our users join via basic then via facebook
- In terms of users, we see high values in the range 20-40 peaking at ~30
- In terms of language major users english, followed by chinese and french
- Majority of the users signed up by Web followed by iOS
- An interesting trend is that major users signed up in 2013 whereas major bookings are done in 2014

1.0.5 1.3 Feature Engineering

- **Extracting features from sessions data**

[26]:
```
sessions.head()
```

[26]:
```
      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
```

[27]:
```
#finding the nuber of actions/sessions by a user by action
action_count = sessions.groupby(['user_id'])['action'].nunique()
#time given to each action type by a user
action_type_count = sessions.groupby(['user_id',␣
↪'action_type'])['secs_elapsed'].agg(['mean','count','max']).unstack()
#rename the columns
action_type_count.columns=action_type_count.columns.map(lambda x: str(x) +␣
↪'_count')
#concatinating and creating a new dataset
df_secondelapsed=pd.concat([action_count,action_type_count],axis=1)
df_secondelapsed=df_secondelapsed.reset_index()
df_secondelapsed.rename(columns={'index':'id'},inplace=True)
#imputing 0 for NaN as we don't have that data for the user
df_secondelapsed.fillna(0,inplace=True)
df_secondelapsed.head()
```

[27]:
```
           id  action  ('mean', '-unknown-')_count  \
0  00023iyk9l      13                     0.000000
1  0010k6l0om      11                 24606.600000
2  001wyh0pz8      10                  3696.833333
3  0028jgx1x1       5                   489.000000
4  002qnbzfs5      25                  4011.788043

   ('mean', 'booking_request')_count  ('mean', 'booking_response')_count  \
0                                0.0                                 0.0
1                                0.0                                 0.0
2                                0.0                                 0.0
3                                0.0                                 0.0
4                            70986.0                                 0.0

   ('mean', 'click')_count  ('mean', 'data')_count  \
0            147672.750000              782.555556
1              4122.125000            15138.444444
2              2406.000000              775.000000
3             22200.222222             1720.600000
```

```
4              4602.307143             2098.107143

   ('mean', 'message_post')_count  ('mean', 'modify')_count  \
0                    129817.0000                       0.0
1                         0.0000                       0.0
2                         0.0000                       0.0
3                         0.0000                       0.0
4                     20327.1875                       0.0

   ('mean', 'partner_callback')_count  …  ('max', '-unknown-')_count  \
0                                 0.0  …                         0.0
1                                 0.0  …                     93024.0
2                                 0.0  …                     17595.0
3                                 0.0  …                       489.0
4                                 0.0  …                     90588.0

   ('max', 'booking_request')_count  ('max', 'booking_response')_count  \
0                               0.0                                0.0
1                               0.0                                0.0
2                               0.0                                0.0
3                               0.0                                0.0
4                           70986.0                                0.0

   ('max', 'click')_count  ('max', 'data')_count  \
0                567165.0                 2348.0
1                 16568.0                127898.0
2                 18299.0                   927.0
3                 84636.0                  8290.0
4                 59316.0                 21613.0

   ('max', 'message_post')_count  ('max', 'modify')_count  \
0                       129817.0                      0.0
1                            0.0                      0.0
2                            0.0                      0.0
3                            0.0                      0.0
4                       105876.0                      0.0

   ('max', 'partner_callback')_count  ('max', 'submit')_count  \
0                                0.0                      0.0
1                                0.0                      0.0
2                                0.0                  50548.0
3                                0.0                      0.0
4                                0.0                1418284.0

   ('max', 'view')_count
0                43110.0
1                65407.0
```

```
2                20501.0
3                47248.0
4                47587.0

[5 rows x 32 columns]
```

- **Dealing with test and train data together to get same number of features after one hot encoding**

[28]:
```
completedata=pd.concat((train_users.drop(['country_destination'], axis=1),␣
↪test_users), axis = 0, ignore_index= True)
completedata.head()
```

[28]:
```
           id date_account_created timestamp_first_active date_first_booking  \
0  gxn3p5htnn           2010-06-28    2009-03-19 04:32:55                NaT
1  820tgsjxq7           2011-05-25    2009-05-23 17:48:09                NaT
2  4ft3gnwmtx           2010-09-28    2009-06-09 23:12:47         2010-08-02
3  bjjt8pjhuk           2011-12-05    2009-10-31 06:01:29         2012-09-08
4  87mebub9p4           2010-09-14    2009-12-08 06:11:05         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
0        Chrome
1        Chrome
2            IE
3       Firefox
4        Chrome
```

[29]:
```
#Adding sessions data to it
completedata=completedata.
↪merge(df_secondelapsed,how="left",left_on="id",right_on="id")
completedata
```

[29]:
```
                id date_account_created timestamp_first_active  \
0       gxn3p5htnn           2010-06-28    2009-03-19 04:32:55
1       820tgsjxq7           2011-05-25    2009-05-23 17:48:09
2       4ft3gnwmtx           2010-09-28    2009-06-09 23:12:47
3       bjjt8pjhuk           2011-12-05    2009-10-31 06:01:29
4       87mebub9p4           2010-09-14    2009-12-08 06:11:05
…              …                    …                    …
275542  cv0na2lf5a           2014-09-30    2014-09-30 23:52:32
275543  zp8xfonng8           2014-09-30    2014-09-30 23:53:06
275544  fa6260ziny           2014-09-30    2014-09-30 23:54:08
275545  87k0fy4ugm           2014-09-30    2014-09-30 23:54:30
275546  9uqfg8txu3           2014-09-30    2014-09-30 23:59:01

       date_first_booking     gender   age signup_method  signup_flow  \
0                     NaT  -unknown-   NaN      facebook            0
1                     NaT       MALE  38.0      facebook            0
2              2010-08-02     FEMALE  56.0         basic            3
3              2012-09-08     FEMALE  42.0      facebook            0
4              2010-02-18  -unknown-  41.0         basic            0
…                     …        …   …           …           …
275542                NaT  -unknown-  31.0         basic            0
275543                NaT  -unknown-   NaN         basic           23
275544                NaT  -unknown-   NaN         basic            0
275545                NaT  -unknown-   NaN         basic            0
275546                NaT     FEMALE  49.0         basic            0

       language affiliate_channel  … ('max', '-unknown-')_count  \
0            en            direct  …                        NaN
1            en               seo  …                        NaN
2            en            direct  …                        NaN
3            en            direct  …                        NaN
4            en            direct  …                        NaN
…          …                 …  …                        …
275542       en            direct  …                    18262.0
275543       ko            direct  …                     1601.0
275544       de            direct  …                     6086.0
275545       en         sem-brand  …                    26491.0
275546       en             other  …                   575816.0

       ('max', 'booking_request')_count ('max', 'booking_response')_count  \
0                                   NaN                                NaN
1                                   NaN                                NaN
2                                   NaN                                NaN
3                                   NaN                                NaN
4                                   NaN                                NaN
…                                     …                                  …
275542                              0.0                                0.0
```

```
275543                            0.0                                0.0
275544                            0.0                                0.0
275545                            0.0                                0.0
275546                            0.0                                0.0

       ('max', 'click')_count ('max', 'data')_count  \
0                         NaN                   NaN
1                         NaN                   NaN
2                         NaN                   NaN
3                         NaN                   NaN
4                         NaN                   NaN
…                           …                     …
275542              1551558.0               50839.0
275543                    0.0                   0.0
275544                60438.0               32319.0
275545               253296.0                 897.0
275546                    0.0                1939.0

       ('max', 'message_post')_count  ('max', 'modify')_count  \
0                                NaN                      NaN
1                                NaN                      NaN
2                                NaN                      NaN
3                                NaN                      NaN
4                                NaN                      NaN
…                                  …                        …
275542                           0.0                      0.0
275543                           0.0                      0.0
275544                           0.0                      0.0
275545                           0.0                      0.0
275546                           0.0                      0.0

       ('max', 'partner_callback')_count  ('max', 'submit')_count  \
0                                    NaN                      NaN
1                                    NaN                      NaN
2                                    NaN                      NaN
3                                    NaN                      NaN
4                                    NaN                      NaN
…                                      …                        …
275542                               0.0                  59185.0
275543                               0.0                      8.0
275544                               0.0                  30390.0
275545                               0.0                  26581.0
275546                               0.0                 124020.0

       ('max', 'view')_count
0                         NaN
1                         NaN
```

```
2                          NaN
3                          NaN
4                          NaN
…                            …
275542                 50836.0
275543                 14307.0
275544                204591.0
275545                 21062.0
275546               1752436.0

[275547 rows x 46 columns]
```

[30]:
```
#Age of 2014 isn't possible and 1 is also not possible so capping the age and converting them to Nan'S
completedata.loc[completedata.age > 100, 'age'] = np.nan
completedata.loc[completedata.age < 13, 'age'] = np.nan
```

[31]:
```
#Finding the Nan's by a user
completedata['nans'] = np.sum([
    (completedata['age'] == np.nan),
    (completedata['gender'] == '-unknown-'),
    (completedata['language'] == '-unknown-'),
    (completedata['first_affiliate_tracked'] == 'untracked'),
    (completedata['first_browser'] == '-unknown-')
], axis=0)
```

[32]:
```
def extract_elements_date(data,feature):
    y=feature+'_year'
    data[y]=data[feature].dt.year
    m=feature+'_month'
    data[m]=data[feature].dt.month
    d=feature+'_date'
    data[d]=data[feature].dt.day
```

[33]:
```
#Extracting columns
extract_elements_date(completedata,'date_account_created')
extract_elements_date(completedata,'date_first_booking')
extract_elements_date(completedata,'timestamp_first_active')
completedata['timestamp_first_active_hour']=completedata['timestamp_first_active'].dt.hour

#Dropping columns
completedata=completedata.drop(['id'],axis=1)
completedata=completedata.drop(['date_account_created'],axis=1)
completedata=completedata.drop(['date_first_booking'],axis=1)
completedata=completedata.drop(['timestamp_first_active'],axis=1)
```

[34]:
```
completedata.head()
```

[34]:

```
      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

   … date_account_created_year  date_account_created_month  \
0  …                      2010                           6
1  …                      2011                           5
2  …                      2010                           9
3  …                      2011                          12
4  …                      2010                           9

   date_account_created_date  date_first_booking_year  \
0                         28                      NaN
1                         25                      NaN
2                         28                   2010.0
3                          5                   2012.0
4                         14                   2010.0

   date_first_booking_month  date_first_booking_date  \
0                       NaN                     NaN
1                       NaN                     NaN
2                       8.0                     2.0
3                       9.0                     8.0
4                       2.0                    18.0

   timestamp_first_active_year  timestamp_first_active_month  \
0                         2009                             3
1                         2009                             5
2                         2009                             6
3                         2009                            10
4                         2009                            12

   timestamp_first_active_date  timestamp_first_active_hour
0                           19                            4
1                           23                           17
```

```
2                            9                           23
3                           31                            6
4                            8                            6

[5 rows x 53 columns]
```

[35]:
```
categorical_features=['gender','signup_method','signup_flow','language','affiliate_channel','a
                      'first_device_type','first_browser']
numerical_features=[i for i in completedata.columns if i not in␣
 ↪categorical_features ]
from sklearn.preprocessing import StandardScaler
scaled_features = completedata.copy()
features = completedata[numerical_features]
features = StandardScaler().fit_transform(features.values)
scaled_features[numerical_features] = features
completedata=scaled_features
completedata.head()
```

[35]:
```
      gender       age signup_method  signup_flow language affiliate_channel  \
0  -unknown-       NaN      facebook            0       en            direct
1       MALE  0.166754      facebook            0       en               seo
2     FEMALE  1.710013         basic            3       en            direct
3     FEMALE  0.509701      facebook            0       en            direct
4  -unknown-  0.423964         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

   … date_account_created_year  date_account_created_month  \
0  …                 -3.521289                   -0.153251
1  …                 -2.435754                   -0.488222
2  …                 -3.521289                    0.851662
3  …                 -2.435754                    1.856575
4  …                 -3.521289                    0.851662

   date_account_created_date  date_first_booking_year  \
0                   1.374944                      NaN
1                   1.032379                      NaN
2                   1.374944                -2.993749
3                  -1.251385                -1.026017
4                  -0.223691                -2.993749

   date_first_booking_month  date_first_booking_date  \
```

```
0                       NaN                       NaN
1                       NaN                       NaN
2                  0.593988                 -1.563733
3                  0.908368                 -0.876101
4                 -1.292286                  0.269952

   timestamp_first_active_year  timestamp_first_active_month  \
0                    -4.603547                     -1.158202
1                    -4.603547                     -0.488227
2                    -4.603547                     -0.153239
3                    -4.603547                      1.186712
4                    -4.603547                      1.856688

   timestamp_first_active_date  timestamp_first_active_hour
0                     0.347280                    -1.019259
1                     0.804054                     0.597322
2                    -0.794655                     1.343437
3                     1.717602                    -0.770554
4                    -0.908848                    -0.770554

[5 rows x 53 columns]
```

[36]:
```
#Making singup flow catgorical
completedata['signup_flow']=completedata['signup_flow'].astype('str')

def create_dummyvariables(data,columname):
    col=pd.get_dummies(data[columname], prefix=columname).iloc[:, 1:] #Taking␣
 ↪all columns after the 0th column for k-1 dummy variables
    data=pd.concat([data, col], axis=1)
    data=data.drop([columname],axis=1)
    return data

categorical_features=['gender','signup_method','signup_flow','language','affiliate_channel','a
                      'first_device_type','first_browser']

for i in categorical_features:
    completedata=create_dummyvariables(completedata,i)

completedata.head()
```

[36]:
```
        age  action  ('mean', '-unknown-')_count  \
0       NaN     NaN                          NaN
1  0.166754     NaN                          NaN
2  1.710013     NaN                          NaN
3  0.509701     NaN                          NaN
4  0.423964     NaN                          NaN
```

```
   ('mean', 'booking_request')_count  ('mean', 'booking_response')_count  \
0                                NaN                                 NaN
1                                NaN                                 NaN
2                                NaN                                 NaN
3                                NaN                                 NaN
4                                NaN                                 NaN

   ('mean', 'click')_count  ('mean', 'data')_count  \
0                      NaN                     NaN
1                      NaN                     NaN
2                      NaN                     NaN
3                      NaN                     NaN
4                      NaN                     NaN

   ('mean', 'message_post')_count  ('mean', 'modify')_count  \
0                             NaN                       NaN
1                             NaN                       NaN
2                             NaN                       NaN
3                             NaN                       NaN
4                             NaN                       NaN

   ('mean', 'partner_callback')_count  …  first_browser_Silk  \
0                                 NaN  …                   0
1                                 NaN  …                   0
2                                 NaN  …                   0
3                                 NaN  …                   0
4                                 NaN  …                   0

   first_browser_SiteKiosk  first_browser_SlimBrowser  \
0                        0                          0
1                        0                          0
2                        0                          0
3                        0                          0
4                        0                          0

   first_browser_Sogou Explorer  first_browser_Stainless  \
0                             0                        0
1                             0                        0
2                             0                        0
3                             0                        0
4                             0                        0

   first_browser_TenFourFox  first_browser_TheWorld Browser  \
0                         0                               0
1                         0                               0
2                         0                               0
3                         0                               0
```

```
4                          0                                   0

   first_browser_UC Browser  first_browser_Yandex.Browser  \
0                          0                                 0
1                          0                                 0
2                          0                                 0
3                          0                                 0
4                          0                                 0

   first_browser_wOSBrowser
0                          0
1                          0
2                          0
3                          0
4                          0

[5 rows x 186 columns]
```

[37]:
```
#Finding Null Entries
print("The null values in the dataset are:")
df=pd.DataFrame(np.sum(completedata.isnull(),axis=0))
df[[0]]=(df[[0]]/completedata.shape[0])*100
df
```

```
The null values in the dataset are:
```

[37]:

	0
age	43.410017
action	50.831256
('mean', '-unknown-')_count	50.831256
('mean', 'booking_request')_count	50.831256
('mean', 'booking_response')_count	50.831256
…	…
first_browser_TenFourFox	0.000000
first_browser_TheWorld Browser	0.000000
first_browser_UC Browser	0.000000
first_browser_Yandex.Browser	0.000000
first_browser_wOSBrowser	0.000000

```
[186 rows x 1 columns]
```

- It can be seen from the above analysis that ~58% of the entries for date first booked are empty, hence, for the time being we will be dropping these values.
- Imputing mean age for NaN
- Since we have a lot of NaN's in sec_elapsed_mean we are currently imputing it with 0

[38]:
```
#imputing mean age
completedata['age'].fillna(completedata['age'].mean(),inplace=True)
```

[39]:
```
completedata.fillna(0,inplace=True)
```

[40]:
```
completedata=completedata.drop(['date_first_booking_year'],axis=1)
completedata=completedata.drop(['date_first_booking_month'],axis=1)
completedata=completedata.drop(['date_first_booking_date'],axis=1)
```

[41]:
```
completedata = completedata.loc[:,~completedata.columns.duplicated()]
completedata.head()
```

[41]:

	age	action	('mean', '-unknown-')_count	\
0	-7.739727e-15	0.0	0.0	
1	1.667540e-01	0.0	0.0	
2	1.710013e+00	0.0	0.0	
3	5.097006e-01	0.0	0.0	
4	4.239639e-01	0.0	0.0	

	('mean', 'booking_request')_count	('mean', 'booking_response')_count	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	('mean', 'click')_count	('mean', 'data')_count	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	('mean', 'message_post')_count	('mean', 'modify')_count	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	('mean', 'partner_callback')_count	…	first_browser_Silk	\
0	0.0	…	0	
1	0.0	…	0	
2	0.0	…	0	
3	0.0	…	0	
4	0.0	…	0	

```
   first_browser_SiteKiosk  first_browser_SlimBrowser  \
0                        0                         0
1                        0                         0
2                        0                         0
3                        0                         0
4                        0                         0

   first_browser_Sogou Explorer  first_browser_Stainless  \
0                             0                        0
1                             0                        0
2                             0                        0
3                             0                        0
4                             0                        0

   first_browser_TenFourFox  first_browser_TheWorld Browser  \
0                         0                               0
1                         0                               0
2                         0                               0
3                         0                               0
4                         0                               0

   first_browser_UC Browser  first_browser_Yandex.Browser  \
0                         0                             0
1                         0                             0
2                         0                             0
3                         0                             0
4                         0                             0

   first_browser_wOSBrowser
0                         0
1                         0
2                         0
3                         0
4                         0

[5 rows x 183 columns]
```

[42]:
```
X=completedata.loc[0:train_users.shape[0]-1,:]
test_df=completedata.loc[train_users.shape[0]:,:]
```

[43]:
```
#getting label for countries
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y=le.fit_transform(train_users['country_destination'])
y.shape
```

[43]:
```
(213451,)
```

[44]:
```
#The number of rows and columns for X
print("The shape of X is {}".format(X.shape))
```

```
The shape of X is (213451, 183)
```

1.0.6 1.4 Evalutation Metric by Kaggle

[24]:
```
"""Metrics to compute the model performance."""

import numpy as np
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import make_scorer


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) + 1))
    T = lb.transform(ground_truth)

    scores = []

    # Iterate over each y_true and compute the DCG score
    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)
        score = float(actual) / float(best)
        scores.append(score)

    return np.mean(scores)


ndcg_scorer = make_scorer(ndcg_score, needs_proba=True, k=5)
```

1.0.7 1.5 Understanding What makes a new user books a place

y=7 implies NDF - y=0 if y =7 i.e Not Booked - y=1 if y !=7 i.e Booked

[45]:
```
y_new=np.array([0 if i==7 else 1 for i in y])
#dividing the data into train and test
from sklearn.model_selection import train_test_split
X_train_new, X_test_new, y_train_new, y_test_new = train_test_split( X, y_new,␣
 ↪test_size=0.3, random_state=42,stratify=y)
```

Fitting a logistic model to find out what makes a user book

[46]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=42,max_iter=12000)
logreg.fit(X_train_new, y_train_new)

y_pred = logreg.predict(X_test_new)
print('Accuracy of Test logistic regression classifier on test set: {:.2f}'.
 ↪format(logreg.score(X_test_new, y_test_new)))
```

```
Accuracy of Test logistic regression classifier on test set: 0.67
```

Tuning the model

[47]:
```
from sklearn.model_selection import GridSearchCV

# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter space
C = [0,1,2,3]

# Create hyperparameter options
hyperparameters = dict( C=C, penalty=penalty)

logreg_cv = GridSearchCV(logreg, hyperparameters, cv=5,␣
 ↪verbose=0,n_jobs=4,scoring='accuracy')
best_model=logreg_cv.fit(X_train_new, y_train_new)

print("tuned hpyerparameters :(best parameters) ",best_model.best_params_)
print("accuracy :",best_model.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1, 'penalty': 'l2'}
accuracy : 0.6778034333902219
```

Fitting the best model

[48]:
```
best_model = LogisticRegression(C=1,penalty='l2',max_iter=12000)
best_model.fit(X_train_new, y_train_new)
```

[48]:
```
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=12000,
                   multi_class='auto', n_jobs=None, penalty='l2',
```

```
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

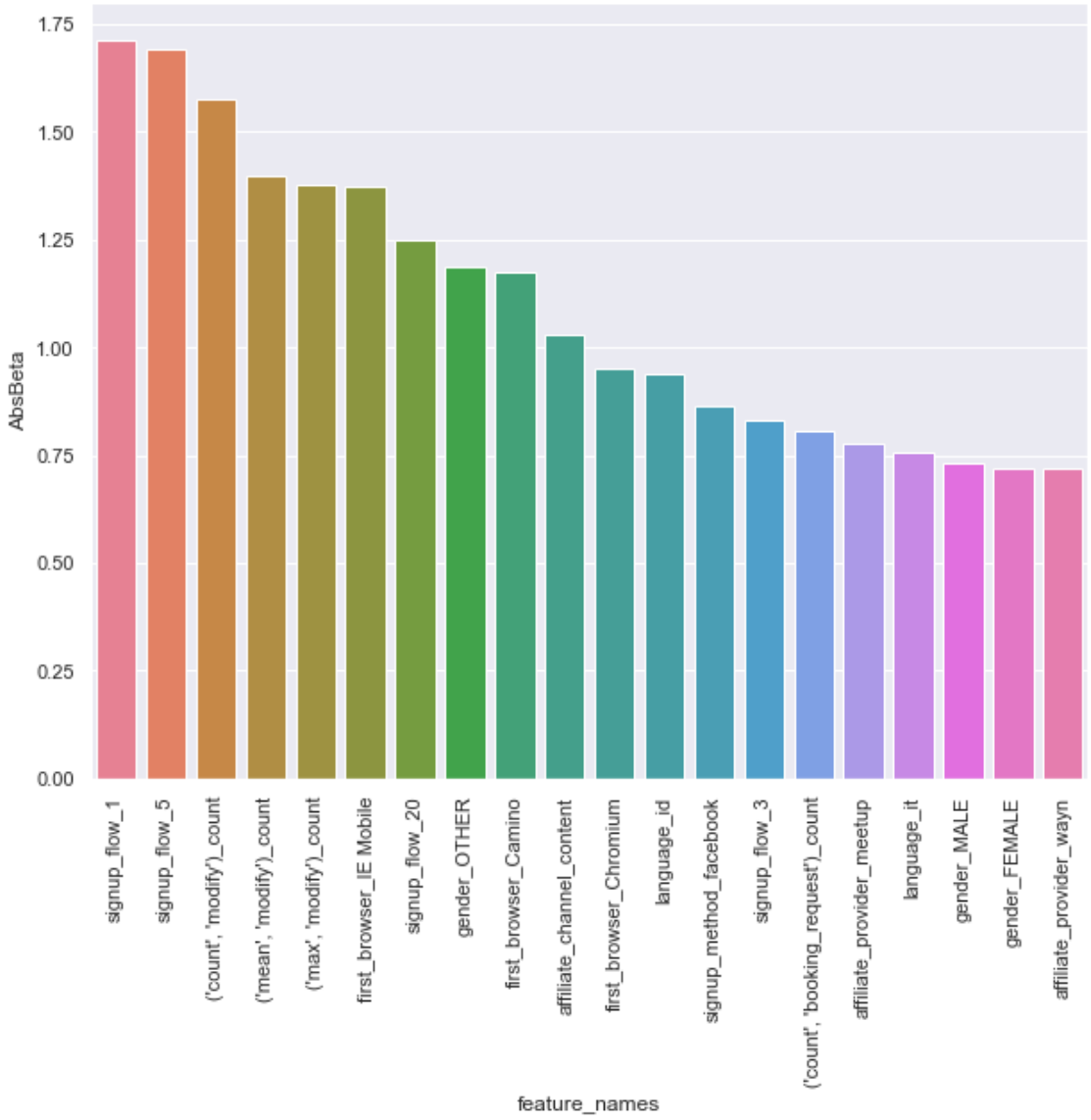
Finding out the important features

[49]:
```
#Getting best coefficiants
feature_imp=best_model.coef_
coef_table=pd.DataFrame({"feature_names":X_train_new.columns,"Beta":
 ↪feature_imp[0],"AbsBeta":abs(feature_imp[0])})
#Get the sorted values
coef_table=coef_table.sort_values(by='AbsBeta',ascending=False)
print("The most important features in the model are: \n")
coef_table=coef_table.iloc[0:20,:]

#plotting the values
sns.set()
plt.figure(figsize=(10,8))
sns.barplot(coef_table['feature_names'],coef_table['AbsBeta'],palette="husl")
plt.xticks(rotation=90)
```

```
The most important features in the model are:
```

[49]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19]),
 <a list of 20 Text major ticklabel objects>)
```



The most important factors that make a user book a place depends on: - Signup flow - session actions - gender - first browser - Signup method

1.0.8 1.6 Models

[50]:
```
#dividing the data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3,␣
↪random_state=42,stratify=y)
```

1.0.9 1.6.1 Fitting a Baseline Model Random Forest Model

[145]:
```
#importing libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

[146]:
```
rf_clf = RandomForestClassifier()

rf_clf.fit(X_train, y_train)

y_pred = rf_clf.predict(X_test)
print('Accuracy of Test Random Forest Classifier on test set: {:.2f}'.
 ↪format(rf_clf.score(X_test, y_test)))
```

```
Accuracy of Test Random Forest Classifier on test set: 0.62
```

[147]:
```
print("The NDCG Score is {}".format(ndcg_score(y_test,rf_clf.
 ↪predict_proba(X_test),k=5)))
```

```
The NDCG Score is 0.8123161092221133
```

[32]:
```
#Using GridSearch CV
param_grid={'n_estimators': range(100,110)}

rf_cv = GridSearchCV(rf_clf, param_grid, cv=5,␣
 ↪verbose=0,n_jobs=4,scoring='accuracy')
best_model=rf_cv.fit(X_train,y_train.values.ravel())

print("Using Grid Search CV")
print("tuned hpyerparameters :(best parameters) ",best_model.best_params_)
print("accuracy :",best_model.best_score_)
```

```
Using Grid Search CV
tuned hpyerparameters :(best parameters)  {'n_estimators': 103}
accuracy : 0.6009302948164508
```

[255]:
```
rf_clf = RandomForestClassifier(n_estimators=103)

rf_clf.fit(X_train, y_train.values)

y_pred = rf_clf.predict(X_test)
print('Accuracy of Test Random Forest Classifier on test set: {:.2f}'.
 ↪format(rf_clf.score(X_test, y_test.values.ravel())))
print("The NDCG Score is {}".format(ndcg_score(y_test.to_numpy(),rf_clf.
 ↪predict_proba(X_test),k=5)))
```

```
Accuracy of Test Random Forest Classifier on test set: 0.60
```

```
The NDCG Score is 0.7974843864018611
```

[56]:
```
#Using GridSearch CV
param_grid={'n_estimators': np.arange(100,700,100)}

rf_cv = GridSearchCV(rf_clf, param_grid, cv=5,␣
 ↪verbose=0,n_jobs=4,scoring='accuracy')
best_model=rf_cv.fit(X_train,y_train)

print("Using Grid Search CV")
print("tuned hpyerparameters :(best parameters) ",best_model.best_params_)
print("accuracy :",best_model.best_score_)
```

```
Using Grid Search CV
tuned hpyerparameters :(best parameters)  {'n_estimators': 400}
accuracy : 0.5993708797644145
```

[46]:
```
rf_clf = RandomForestClassifier(n_estimators=400)

rf_clf.fit(X_train, y_train)

y_pred = rf_clf.predict(X_test)
print('Accuracy of Test Random Forest Classifier on test set: {:.2f}'.
 ↪format(rf_clf.score(X_test, y_test)))
print("The NDCG Score is {}".format(ndcg_score(y_test,rf_clf.
 ↪predict_proba(X_test),k=5)))
```

```
Accuracy of Test Random Forest Classifier on test set: 0.60
The NDCG Score is 0.8037287097173264
```

1.0.10 1.6.2 KNN

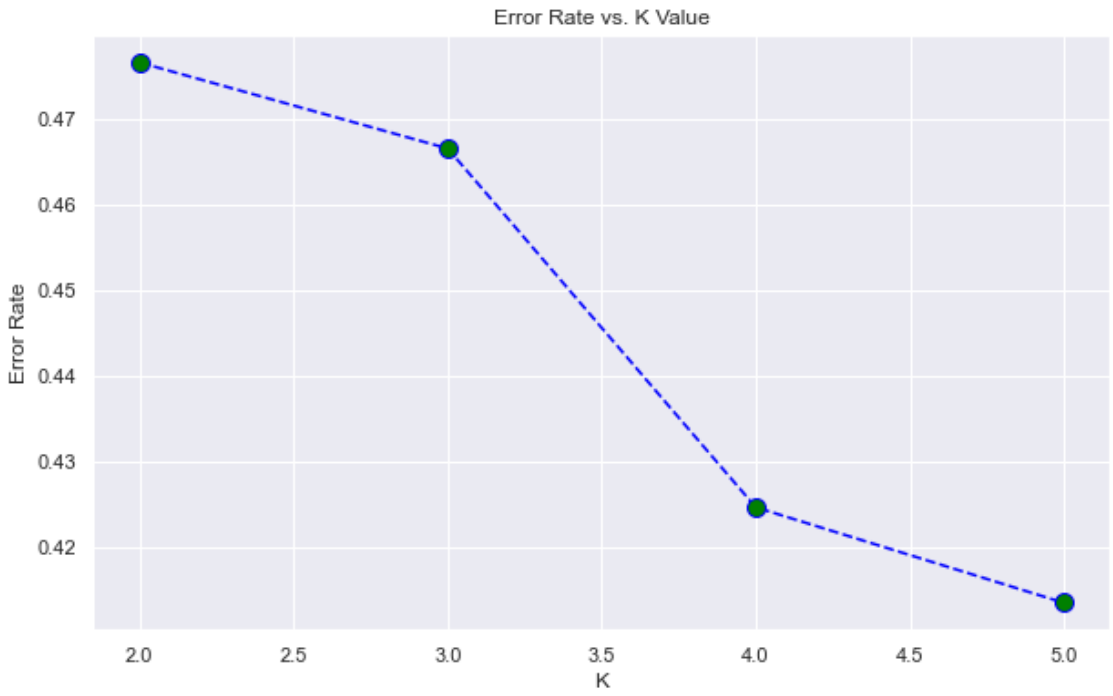
[155]:
```
from sklearn.neighbors import KNeighborsClassifier
error=[]
for i in range(2,6):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    error.append(np.mean(y_pred != np.array(y_test)))
    print("The iteration is for cluster {}".format(i))
```

```
The iteration is for cluster 2
The iteration is for cluster 3
The iteration is for cluster 4
The iteration is for cluster 5
```

[157]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
```

```
plt.plot(range(2,6),error,color='blue', linestyle='dashed',␣
 ↪marker='o',markerfacecolor='green', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

[157]: Text(0, 0.5, 'Error Rate')



[158]:
```
#trying a random k
knn = KNeighborsClassifier(n_neighbors=200)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("The error for 200 clusters is: {}".format(np.mean(y_pred != np.
 ↪array(y_test))))
```

```
The error for 200 clusters is: 0.3894684240114935
```

1.0.11 1.6.3 XGBoost

[52]:
```
from xgboost import XGBClassifier

xgbclass = XGBClassifier()
xgbclass.fit(X_train, y_train)
print('Accuracy of Test XGB Classifier on test set: {:.2f}'.format(xgbclass.
 ↪score(X_test, y_test)))
```

```
Accuracy of Test XGB Classifier on test set: 0.65
```

Understanding the hyper paramaters

[53]:
```
xgbclass
```

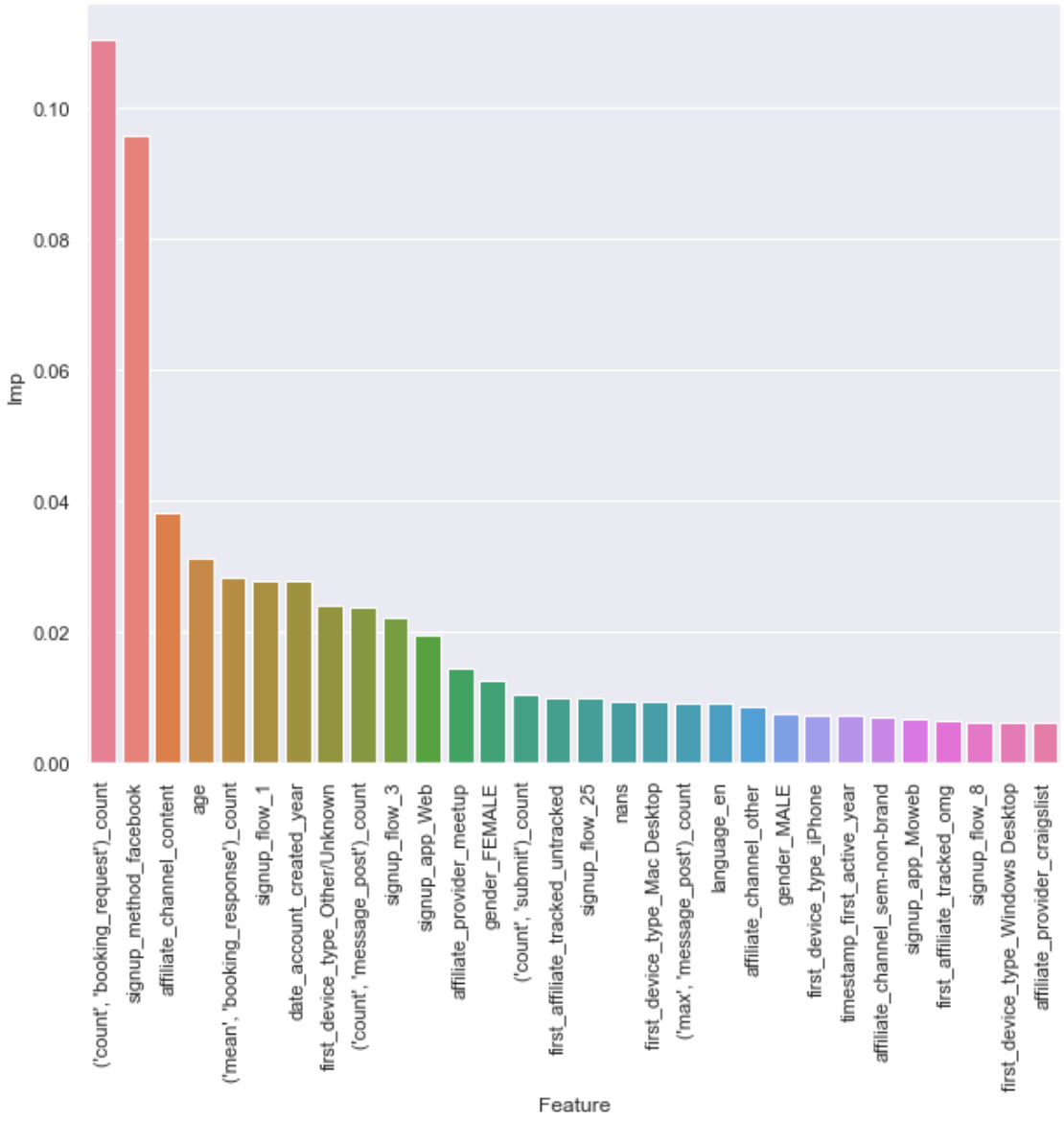
[53]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

Finding the important features

[54]:
```
important_df=pd.DataFrame(np.column_stack([X_train.columns,xgbclass.
 ↪feature_importances_]),columns=['Feature','Imp'])
important_df=important_df.sort_values(by=['Imp'],ascending=False)[0:30]

#plotting the values
sns.set()
plt.figure(figsize=(10,8))
sns.barplot(important_df['Feature'],important_df['Imp'],palette="husl")
plt.xticks(rotation=90)
```

[54]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]),
 <a list of 30 Text major ticklabel objects>)
```



When compared Random Fores, KNN, XGBOOST : XGB gives the best performance
The most important features for users to select a country are - number of times a user makes booking request - When they signup by Facebook - affiliate channel content - Age - Mean time a user spends for booking response - Signup flow 1 - The year the account was created - First device type is a part of "others" - count of message posts - Signup flow 3

1.0.12 Preparing the Dataset for testing and getting the submission results

Replace the modelname here

[55]:
```
ytest_pred=xgbclass.predict_proba(test_df)
```

[56]:
```
idx=test_users['id']
ids=[]
cts=[]
for i in range(len(idx)):
    ids += [idx[i]] * 5
    cts += le.inverse_transform(np.argsort(ytest_pred[i])[::-1])[:5].tolist()
submission=pd.DataFrame(np.column_stack((ids, cts)), columns=['id', 'country'])
```

[57]:
```
submission=submission.set_index('id')
submission1=submission.to_csv('output_test.csv')
```

1.0.13 1.8 Challenges Faced

- New Evaluation Metric: Understand why its used
- Creating dummy variables one-hot encoding vs integer mapping
- Understanding hyperparamaters of all the models used and how to vary them
- Sometimes getting data and feature engineering is more important than the model itself

1.0.14 1.9 Future Work

- Hyperparamater Tuning
- Feature Selection
- Imbalanced Dataset