SparkCognition

Introduction:

In this assignment, I am going to create the script to answer the following questions. Based on the data available in .csv file provided, please try and answer the questions below:
1.Explain what you understand from the data provided, provide exploratory insights.

2.What is your approach if you have to dynamically change the pricing for In-Flight WiFi? – Technical approach will suffice.

3.What are the statistical/ predictive methodologies that could be applied with the data you have – Please implement one model & explain the output.

Source Code

Table of contents

1. Import Libraries

In [453]:
```
import pandas as pd
pd.set_option('display.max_columns', 500)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import time
from collections import Counter
import openpyxl
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_s
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import classification_report

from imblearn.over_sampling import SMOTE

import xgboost as xgb


from hyperopt import hp
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials

from scipy.stats import pearsonr
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

2. Read Data

In [12]:
```
#import the data
train_file = 'Data_Challenge.csv'
df_total = pd.read_csv(train_file)
```

In [36]:
```
#check the data sample
df_total.head()
```

Out[36]:

	Route	Flight Count	flight per week 1	flights per week 2	flights per week 3	flights per month 1	flights per month 2	flights per month 3	First Flow	Last Flown	Airline Count	First Airline
0	HND-CTS	5303	107.0	111.0	106.0	466.0	470.0	421.0	7/1/2017 0:20	7/25/2018 23:25	1	ANA
1	CTS-HND	5272	110.0	112.0	109.0	453.0	455.0	402.0	7/1/2017 0:40	7/25/2018 23:39	1	ANA
2	HND-FUK	4520	89.0	104.0	101.0	430.0	452.0	426.0	7/1/2017 2:48	7/25/2018 22:37	1	ANA
3	FUK-HND	4475	87.0	96.0	99.0	424.0	446.0	434.0	7/1/2017 1:28	7/25/2018 23:13	1	ANA
4	HND-ITM	3796	71.0	91.0	80.0	362.0	391.0	362.0	7/1/2017 0:20	7/25/2018 22:23	1	ANA

In [455]:
```
#examine the statistics of data features
df_total.describe()
```

Out[455]:

	Flight Count	flight per week 1	flights per week 2	flights per week 3	flights per month 1	flights per month 2	flights per month 3
count	5316.000000	1959.000000	1941.000000	1915.000000	2639.000000	2525.000000	2747.000000
mean	107.502822	6.262379	6.358063	6.325849	19.568776	20.123564	17.384783
std	294.557007	8.880984	9.394343	9.210832	35.369946	36.500784	32.964393
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000	1.000000	2.000000	2.000000	2.000000
50%	11.000000	4.000000	4.000000	4.000000	7.000000	7.000000	5.000000
75%	78.000000	7.000000	7.000000	7.000000	28.000000	28.000000	25.000000
max	5303.000000	110.000000	112.000000	109.000000	466.000000	470.000000	434.000000

In [456]:
```
#check data size
df_total.shape
```

Out[456]: (5316, 33)

In [457]:
```
#check data type
df_total.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5316 entries, 0 to 5315
Data columns (total 33 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Route                              5316 non-null   object 
 1   Flight Count                       5316 non-null   int64  
 2   flight per week 1                  1959 non-null   float64
 3   flights per week 2                 1941 non-null   float64
 4   flights per week 3                 1915 non-null   float64
 5   flights per month 1                2639 non-null   float64
 6   flights per month 2                2525 non-null   float64
 7   flights per month 3                2747 non-null   float64
 8   First Flow                         5316 non-null   object 
 9   Last Flown                         5316 non-null   object 
 10  Airline Count                      5316 non-null   int64  
 11  First Airline                      5316 non-null   object 
 12  Last Airline                       5316 non-null   object 
 13  Aircraft Type Count                5316 non-null   int64  
 14  First Aircraft Type                5316 non-null   object 
 15  Average of Avg - Flight Duration (MB) 5316 non-null   float64
 16  Min of Min - Flight Duration (Hrs)  5316 non-null   float64
 17  Max of Max - Flight Duration (Hrs)  5316 non-null   float64
 18  Avg - Seat Count                   5316 non-null   float64
 19  Min - Seat Count                   5316 non-null   int64  
 20  Max - Seat Count                   5316 non-null   int64  
 21  Price per User                     3944 non-null   float64
 22  avg(a.Price)                       3944 non-null   float64
 23  min(a.Price)                       3944 non-null   float64
 24  max(a.Price)                       3944 non-null   float64
 25  Average Usage (MB)                 3539 non-null   float64
 26  Min - Usage (MB)                   3539 non-null   float64
 27  Max - Usage (MB)                   3539 non-null   float64
 28  Total Usage (MB)                   3539 non-null   float64
 29  Usage per Flight (MB)              3539 non-null   float64
 30  Usage (MB)/ Min                    3539 non-null   float64
 31  Min - Total Users                  3543 non-null   float64
 32  Max - Total Users                  3543 non-null   float64
dtypes: float64(22), int64(5), object(6)
memory usage: 1.3+ MB
```

In [458]:
```
# check the data features
df_total.columns.tolist()
```

```
 'Aircraft Type Count',
 'First Aircraft Type',
 'Average of Avg - Flight Duration (MB)',
 'Min of Min - Flight Duration (Hrs)',
 'Max of Max - Flight Duration (Hrs)',
 'Avg - Seat Count',
 'Min - Seat Count',
 'Max - Seat Count',
 'Price per User',
 'avg(a.Price)',
 'min(a.Price)',
 'max(a.Price)',
 'Average Usage (MB)',
 'Min - Usage (MB)',
 'Max - Usage (MB)',
 'Total Usage (MB)',
 'Usage per Flight (MB)',
 'Usage (MB)/ Min',
 'Min - Total Users',
 'Max - Total Users']
```

In [459]:
```
# convert the price feature to numerical
df_total[['Price per User','avg(a.Price)','min(a.Price)', 'max(a.Price)']] = \
df_total[['Price per User','avg(a.Price)','min(a.Price)', 'max(a.Price)']].replac
```

3. Exploratory Data Analysis

A. Check missing data

In [46]:
```
#check missing value\
#Define a funciton to visualize the features with missing values, and the percent
def missing_value_table(df):
    #total missing value
    mis_val = df.isnull().sum()
    #percentage of the missing values
    mis_val_percent = 100*mis_val/len(df)
    #type fo the missing value
    mis_val_type = df.dtypes
    #combine the results to a table
    mis_val_table = pd.concat([mis_val, mis_val_percent, mis_val_type], axis = 1)
    #name the column
    mis_val_table_rename_col = mis_val_table.rename(columns = {0:'Missing Values'
    #sort the table by percentage of missing descending
    mis_val_table_rename_col = mis_val_table_rename_col[mis_val_table_rename_col.
    .sort_values('% of Total Values', ascending = False).round(1)

    #print
    print("Your selected dataframe has " + str(df.shape[1]) + " columns.\n" "Ther
    #return the dataframe with missing information
    return mis_val_table_rename_col
```

In [47]:
```
missing_value_table(df_total)
```

```
Your selected dataframe has 33 columns.
There are 18 columns that have missing values.
```

Out[47]:

	Missing Values	% of Total Values	Type
flights per week 3	3401	64.0	float64
flights per week 2	3375	63.5	float64
flight per week 1	3357	63.1	float64
flights per month 2	2791	52.5	float64
flights per month 1	2677	50.4	float64
flights per month 3	2569	48.3	float64
Total Usage (MB)	1777	33.4	float64
Usage (MB)/ Min	1777	33.4	float64
Usage per Flight (MB)	1777	33.4	float64
Average Usage (MB)	1777	33.4	float64
Min - Usage (MB)	1777	33.4	float64
Max - Usage (MB)	1777	33.4	float64
Min - Total Users	1773	33.4	float64
Max - Total Users	1773	33.4	float64
min(a.Price)	1372	25.8	float64
avg(a.Price)	1372	25.8	float64
Price per User	1372	25.8	float64
max(a.Price)	1372	25.8	float64

In [48]:
```
df_price_miss = df_total[df_total['avg(a.Price)'].isnull()]
```

In [49]:
```
df_price_miss.shape
```

Out[49]:
```
(1372, 33)
```

comment

In total, there are 1372 routes have not price data

In [50]:
```
df_price_miss.head()
```

Out[50]:

	Route	Flight Count	flight per week 1	flights per week 2	flights per week 3	flights per month 1	flights per month 2	flights per month 3	First Flow	Last Flown	Airline Count	Fi Airl
366	ABJ-BKO	258	7.0	7.0	7.0	30.0	30.0	26.0	10/8/2017 19:00	7/25/2018 18:49	1	A
378	BKO-ABJ	258	7.0	7.0	7.0	30.0	30.0	26.0	10/8/2017 15:46	7/25/2018 15:21	1	A
383	CDG-BKO	263	7.0	7.0	7.0	30.0	27.0	25.0	10/8/2017 9:26	7/25/2018 9:05	1	A
384	CDG-BOG	92	7.0	7.0	7.0	30.0	29.0	8.0	4/22/2018 16:52	7/25/2018 16:32	1	A
536	BKO-CDG	261	7.0	7.0	6.0	29.0	27.0	26.0	10/8/2017 22:25	7/25/2018 23:05	1	A

In [460]:
```
#data without missing price
df_no_price_miss = df_total[~df_total['avg(a.Price)'].isnull()]
```

In [461]:
```
# double check the missing data
missing_value_table(df_no_price_miss)
```

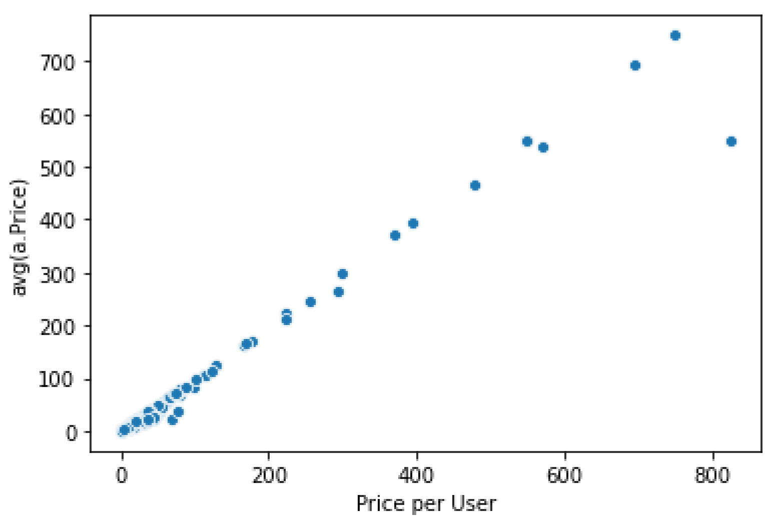
```
Your selected dataframe has 33 columns.
There are 14 columns that have missing values.
```

Out[461]:

	Missing Values	% of Total Values	Type
flights per week 3	2106	53.4	float64
flights per week 2	2096	53.1	float64
flight per week 1	2080	52.7	float64
flights per month 2	1623	41.2	float64
flights per month 1	1535	38.9	float64
flights per month 3	1409	35.7	float64
Average Usage (MB)	497	12.6	float64
Min - Usage (MB)	497	12.6	float64
Max - Usage (MB)	497	12.6	float64
Total Usage (MB)	497	12.6	float64
Usage per Flight (MB)	497	12.6	float64
Usage (MB)/ Min	497	12.6	float64
Min - Total Users	497	12.6	float64
Max - Total Users	497	12.6	float64

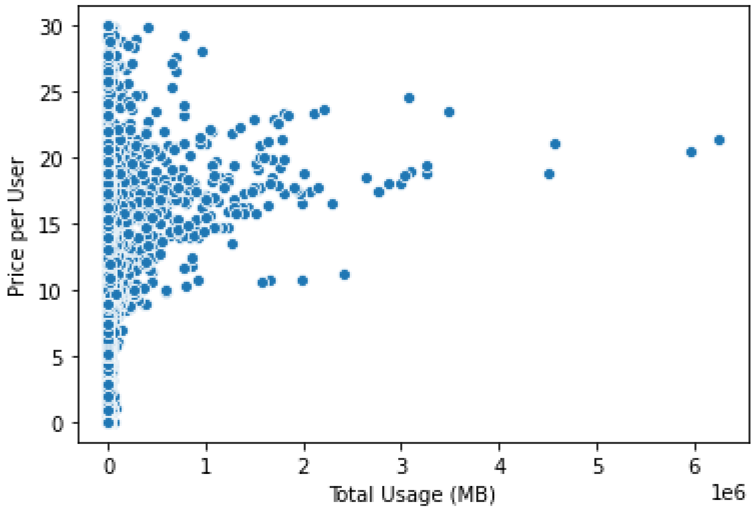
In [462]:
```
# try to understand the different price feature
sns.scatterplot(data = df_no_price_miss, x = 'Price per User', y = 'avg(a.Price)'
```

Out[462]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4fd667518>



In [463]:
```
sns.scatterplot(data = df_no_price_miss[df_no_price_miss['Price per User']<30], x
```

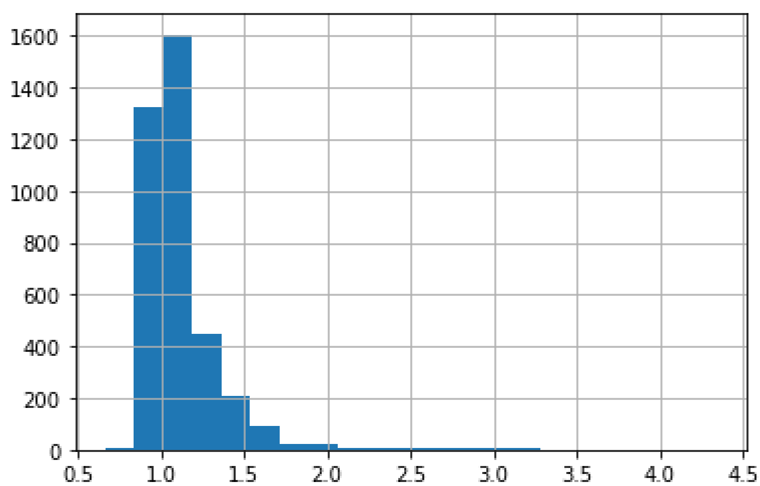
Out[463]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e493de4518>
```



In [464]:
```
df_no_price_miss['ratio'] = df_no_price_miss['Price per User']/df_no_price_miss['
```

In [465]:
```
df_no_price_miss['ratio'].hist(bins = 21)
```

Out[465]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e493b1a2e8>
```

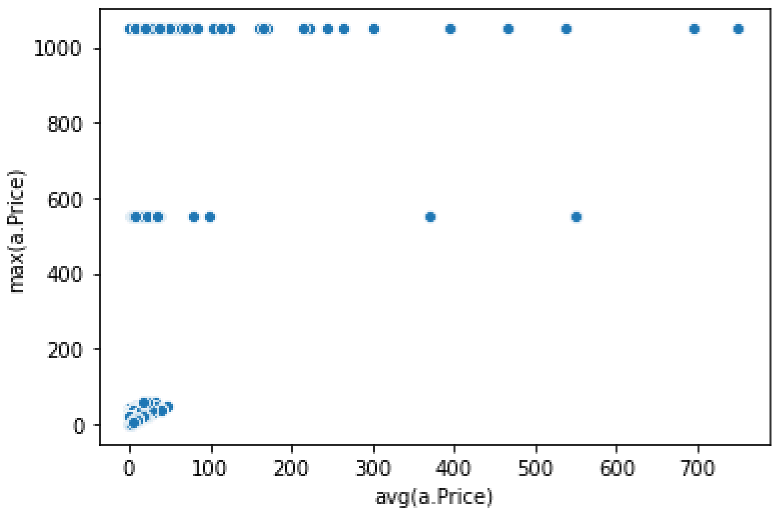


In [466]:
```
df_no_price_miss['max(a.Price)'].loc[df_no_price_miss['ratio']==1].value_counts()
```

Out[466]:
```
1.00         1
1.95         4
2.00        10
2.90        24
4.90        18
4.99        11
5.00        12
5.95         2
6.00        16
6.49        10
6.95         5
6.99         1
7.00        14
7.90         6
7.95         1
8.00       461
8.90        19
8.99        27
9.00         6
9.95         7
9.99         1
10.00       13
11.95        2
12.00       55
12.95        6
13.00        9
13.90       25
13.99       11
15.00        4
15.95        2
16.95        3
17.00       12
17.90        2
18.90        1
18.99       14
19.00      132
19.95       12
19.99        5
20.00       22
21.95       21
23.00        1
25.90        2
29.99        7
30.00       10
39.00        6
40.00        1
59.00        5
550.00      12
1050.00      8
Name: max(a.Price), dtype: int64
```

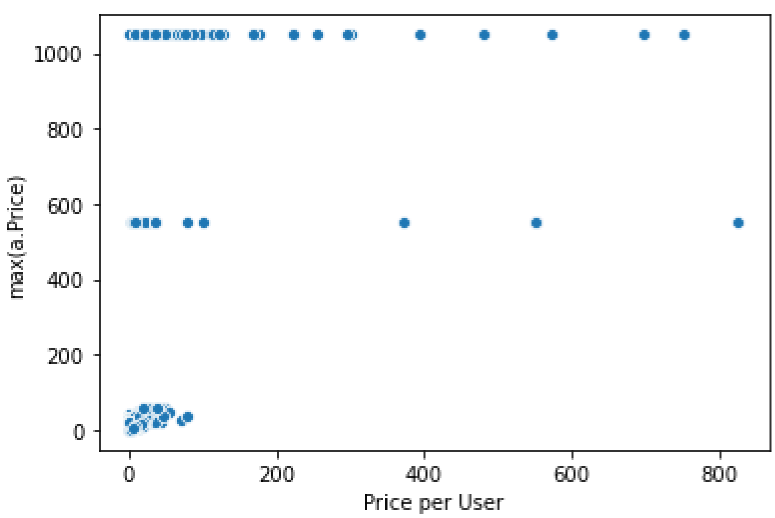
In [467]:
```
sns.scatterplot(data = df_no_price_miss, x = 'avg(a.Price)', y = 'max(a.Price)')
```

Out[467]: <matplotlib.axes._subplots.AxesSubplot at 0x1e493ea74e0>



In [468]:
```
sns.scatterplot(data = df_no_price_miss, x = 'Price per User', y = 'max(a.Price)'
```

Out[468]: <matplotlib.axes._subplots.AxesSubplot at 0x1e493f24c18>



In [469]:
```
df_no_price_miss['avg(a.Price)'].describe()
```

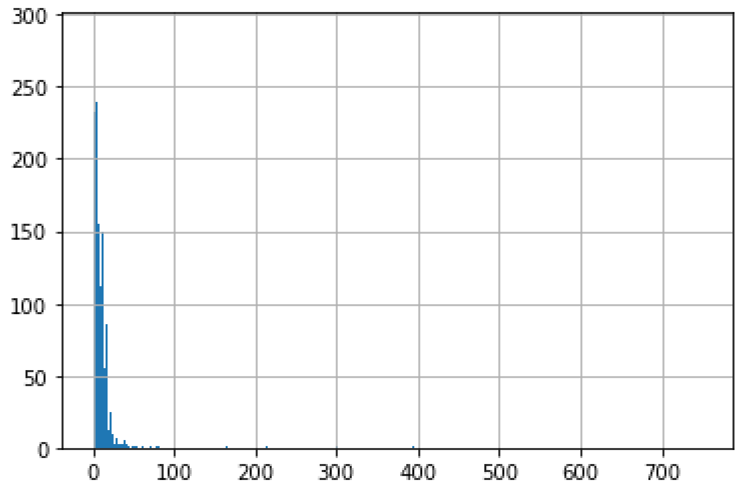
Out[469]:
```
count    3944.000000
mean       11.059019
std        33.751234
min         0.000000
25%         3.407500
50%         7.220000
75%        12.182500
max       750.000000
Name: avg(a.Price), dtype: float64
```

In [470]:
```
df_no_price_miss['avg(a.Price)'].loc[df_no_price_miss['Flight Count']>55].describ
```

Out[470]:
```
count    1499.000000
mean       14.850927
std        34.844035
min         0.000000
25%         7.470000
50%         9.990000
75%        15.805000
max       750.000000
Name: avg(a.Price), dtype: float64
```

In [471]:
```
df_no_price_miss['avg(a.Price)'].hist(bins = 1000)
```

Out[471]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e493faa8d0>
```



In [472]:
```
df_no_price_miss['avg(a.Price)'].describe()
```

Out[472]:
```
count    3944.000000
mean       11.059019
std        33.751234
min         0.000000
25%         3.407500
50%         7.220000
75%        12.182500
max       750.000000
Name: avg(a.Price), dtype: float64
```

In [195]:
```
upper_quartile = np.percentile(df_no_price_miss['avg(a.Price)'].tolist(), 99)
upper_quartile
```

Out[195]:
```
57.57420000000066
```

In [473]:
```
upper_quartile = np.percentile(df_no_price_miss['avg(a.Price)'].tolist(), 75)
lower_quartile = np.percentile(df_no_price_miss['avg(a.Price)'].tolist(), 25)
IQR = upper_quartile - lower_quartile
df_no_price_outlier = df_no_price_miss.loc[df_no_price_miss['avg(a.Price)']<=uppe
df_price_outlier = df_no_price_miss.loc[df_no_price_miss['avg(a.Price)']>upper_qu
```

In [474]:
```
df_no_price_outlier['avg(a.Price)'].describe()
```

Out[474]:
```
count    3801.000000
mean        7.789682
std         5.564332
min         0.000000
25%         3.280000
50%         6.910000
75%        11.730000
max        25.150000
Name: avg(a.Price), dtype: float64
```

In [475]:
```
df_price_outlier['avg(a.Price)'].describe()
```

Out[475]:
```
count    143.000000
mean      97.959371
std      151.366564
min       25.360000
25%       31.965000
50%       38.510000
75%       63.170000
max      750.000000
Name: avg(a.Price), dtype: float64
```

In [476]:
```
df_no_price_outlier = df_no_price_miss.loc[df_no_price_miss['avg(a.Price)']<=100]
df_price_outlier = df_no_price_miss.loc[df_no_price_miss['avg(a.Price)']>100]
```

In [477]:
```
df_no_price_miss.shape
```

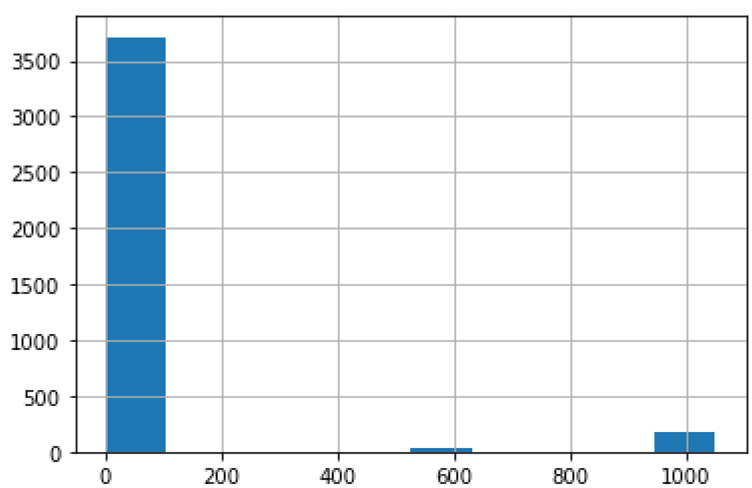
Out[477]: (3944, 34)

In [478]:
```
df_no_price_outlier.shape
```

Out[478]: (3919, 34)

In [479]:
```
df_no_price_outlier['max(a.Price)'].hist(bins = 10)
```

Out[479]: <matplotlib.axes._subplots.AxesSubplot at 0x1e493f4c668>



In [480]:
```
df_no_price_miss['avg(a.Price)'].loc[df_no_price_miss['max(a.Price)']>800].descri
```

Out[480]:
```
count    195.000000
mean      52.962923
std       96.624223
min        0.980000
25%       18.185000
50%       28.760000
75%       41.080000
max      750.000000
Name: avg(a.Price), dtype: float64
```

comment

The price per user and avarge price is hightly correlated. I am going to just use avg price as the target value. And there are some average price outliers, they'll be removed from the training dataset.

In [481]:
```
df_usage_miss = df_no_price_outlier[df_no_price_outlier['Average Usage (MB)'].isn
```

In [482]:
```
df_usage_miss.head()
```

Out[482]:

	Route	Flight Count	flight per week 1	flights per week 2	flights per week 3	flights per month 1	flights per month 2	flights per month 3	First Flow	Last Flown	Airline Count	A
1223	PHX-SAN	29	NaN	1.0	3.0	9.0	1.0	3.0	9/7/2017 20:42	7/25/2018 15:55	1	
1247	DAL-MDW	42	NaN	2.0	3.0	8.0	3.0	2.0	12/15/2017 12:38	7/24/2018 11:34	1	
1336	PHX-ATL	33	1.0	3.0	1.0	7.0	9.0	5.0	9/20/2017 13:51	7/10/2018 13:17	1	
1356	BOS-BWI	36	1.0	1.0	2.0	6.0	5.0	11.0	11/24/2017 19:05	7/9/2018 0:46	1	
1358	BOS-MDW	25	1.0	NaN	3.0	6.0	5.0	4.0	12/17/2017 18:39	7/9/2018 20:47	1	

In [483]:
```
df_usage_miss.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 497 entries, 1223 to 5161
Data columns (total 34 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   Route                                497 non-null    object 
 1   Flight Count                         497 non-null    int64  
 2   flight per week 1                    66 non-null     float64
 3   flights per week 2                   67 non-null     float64
 4   flights per week 3                   69 non-null     float64
 5   flights per month 1                  204 non-null    float64
 6   flights per month 2                  229 non-null    float64
 7   flights per month 3                  255 non-null    float64
 8   First Flow                           497 non-null    object 
 9   Last Flown                           497 non-null    object 
 10  Airline Count                        497 non-null    int64  
 11  First Airline                        497 non-null    object 
 12  Last Airline                         497 non-null    object 
 13  Aircraft Type Count                  497 non-null    int64  
 14  First Aircraft Type                  497 non-null    object 
 15  Average of Avg - Flight Duration (MB) 497 non-null    float64
 16  Min of Min - Flight Duration (Hrs)   497 non-null    float64
 17  Max of Max - Flight Duration (Hrs)   497 non-null    float64
 18  Avg - Seat Count                     497 non-null    float64
 19  Min - Seat Count                     497 non-null    int64  
 20  Max - Seat Count                     497 non-null    int64  
 21  Price per User                       497 non-null    float64
 22  avg(a.Price)                         497 non-null    float64
 23  min(a.Price)                         497 non-null    float64
 24  max(a.Price)                         497 non-null    float64
 25  Average Usage (MB)                   0 non-null      float64
 26  Min - Usage (MB)                     0 non-null      float64
 27  Max - Usage (MB)                     0 non-null      float64
 28  Total Usage (MB)                     0 non-null      float64
 29  Usage per Flight (MB)                0 non-null      float64
 30  Usage (MB)/ Min                      0 non-null      float64
 31  Min - Total Users                    0 non-null      float64
 32  Max - Total Users                    0 non-null      float64
 33  ratio                                466 non-null    float64
dtypes: float64(23), int64(5), object(6)
memory usage: 135.9+ KB
```

In [484]:
```
df_usage_miss.describe()
```

Out[484]:

	Flight Count	flight per week 1	flights per week 2	flights per week 3	flights per month 1	flights per month 2	flights per month 3	Airlin Cour
count	497.000000	66.000000	67.000000	69.000000	204.000000	229.000000	255.000000	497.00000
mean	8.074447	1.242424	1.208955	1.333333	2.068627	2.877729	2.364706	1.00402
std	9.622538	0.431834	0.508632	0.634004	1.473977	1.987435	1.673182	0.06337
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
25%	2.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
50%	5.000000	1.000000	1.000000	1.000000	2.000000	2.000000	2.000000	1.00000
75%	10.000000	1.000000	1.000000	2.000000	3.000000	4.000000	3.000000	1.00000
max	62.000000	2.000000	3.000000	4.000000	9.000000	9.000000	11.000000	2.00000

comment

There are additional 497 routes which have not wifi usage data. In those routes, the total flight count are relatively small, and the mean flight count is 8. Thus, I plan to remove those data from the training datasets to keep the model simple.

In [485]:
```
# create a dataset without price and usage missing value
df_train = df_no_price_outlier[~df_no_price_outlier['Average Usage (MB)'].isnull(
```

In [486]:
```
df_train.shape
```

Out[486]:
```
(3422, 34)
```

In [487]:
```
missing_value_table(df_train)
```

```
Your selected dataframe has 34 columns.
There are 7 columns that have missing values.
```

Out[487]:

	Missing Values	% of Total Values	Type
flights per week 3	1657	48.4	float64
flights per week 2	1645	48.1	float64
flight per week 1	1630	47.6	float64
flights per month 2	1339	39.1	float64
flights per month 1	1225	35.8	float64
flights per month 3	1147	33.5	float64
ratio	156	4.6	float64

In [488]:
```
missing_value_table(df_train).index
```

```
Your selected dataframe has 34 columns.
There are 7 columns that have missing values.
```

Out[488]:
```
Index(['flights per week 3', 'flights per week 2', 'flight per week 1',
       'flights per month 2', 'flights per month 1', 'flights per month 3',
       'ratio'],
      dtype='object')
```

In [489]:
```
#monthly average flight
df_train['First Flow'] = pd.to_datetime(df_train['First Flow'])
df_train['Last Flown'] = pd.to_datetime(df_train['Last Flown'])
```

In [490]:
```
first_flow
```

Out[490]:
```
0       2017-07-01 00:20:00
1       2017-07-01 00:40:00
2       2017-07-01 02:48:00
3       2017-07-01 01:28:00
4       2017-07-01 00:20:00
                ...
5305    2017-07-02 18:43:00
5307    2017-07-15 08:23:00
5309    2018-04-09 11:31:00
5311    2017-07-01 17:01:00
5314    2017-07-02 15:44:00
Name: First Flow, Length: 3447, dtype: datetime64[ns]
```

In [491]:
```
df_train['diff_weeks'] =(df_train['Last Flown']-df_train['First Flow'])/np.timede
```

In [492]:
```
df_train['diff_weeks'].loc[df_train['diff_weeks']<1] = 1
```

In [493]:
```
df_train['diff_weeks'].describe()
```

Out[493]:
```
count    3422.000000
mean       33.303116
std        20.313666
min         1.000000
25%        15.746280
50%        35.435665
75%        55.268874
max        55.712302
Name: diff_weeks, dtype: float64
```

In [494]:
```
df_train['flight_per_week'] = df_train['Flight Count']/df_train['diff_weeks']
```

comment

create a flight per week feature to replace the average flight count

B. Plot the statistics of Features

Route Features:

'Route', **'Flight Count'**, 'flight per week 1', 'flights per week 2', 'flights per week 3', 'flights per month 1', 'flights per month 2', 'flights per month 3',

Flight Related Features:

'First Flow', 'Last Flown', 'Airline Count', 'First Airline', 'Last Airline', 'Aircraft Type Count', 'First Aircraft Type', **'Average of Avg - Flight Duration (MB)'**, 'Min of Min - Flight Duration (Hrs)', 'Max of Max - Flight Duration (Hrs)', **'Avg - Seat Count'**, 'Min - Seat Count', 'Max - Seat Count',

In-flight WiFi Price:

'Price per User', **'avg(a.Price)'**, 'min(a.Price)', 'max(a.Price)',

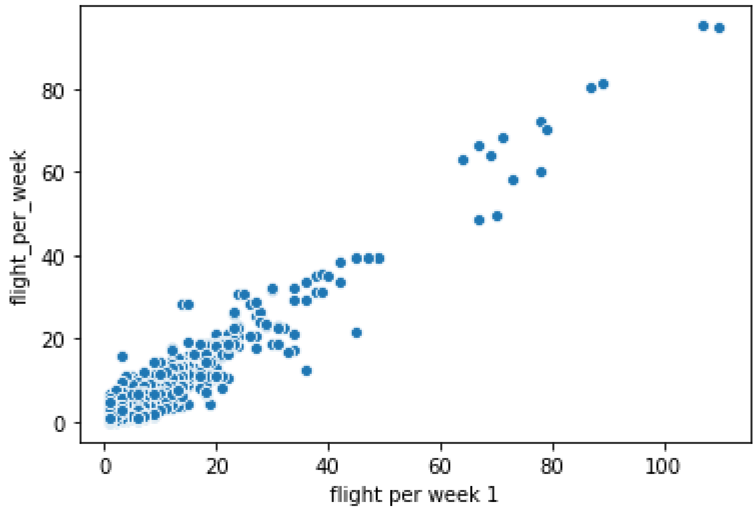
In-flight WiFi Usage:

'Average Usage (MB)', 'Min - Usage (MB)', 'Max - Usage (MB)', **'Total Usage (MB)'**, **'Usage per Flight (MB)'**, **'Usage (MB)/ Min'**, 'Min - Total Users', **'Max - Total Users'**]

2.2.1 Flight per week

In [218]:
```
sns.scatterplot(data = df_train[~df_train['flight per week 1'].isnull()], x = 'fl
```

Out[218]: <matplotlib.axes._subplots.AxesSubplot at 0x1e48792f1d0>



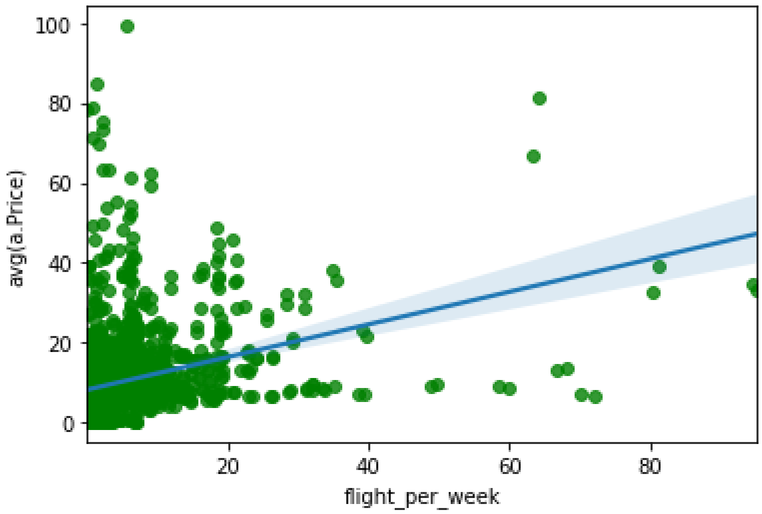
In [219]:
```
corr, _  = pearsonr(df_train['flight per week 1'].loc[~df_train['flight per week
                    df_train['flight_per_week'].loc[~df_train['flight per week 1'
print('Pearsons correlation: %.3f' % corr)
```

```
Pearsons correlation: 0.955
```

In [220]:
```
flight_count_feature_list = ['flights per week 3', 'flights per week 2', 'flight
       'flights per month 2', 'flights per month 1', 'flights per month 3']
fig, ax = plt.subplots(figsize = (10,6))
cm_df = sns.heatmap(df_train[flight_count_feature_list].corr(), annot = True, fmt
```

In [351]:
```
sns.regplot(data = df_train[df_train['avg(a.Price)']<100], x = 'flight_per_week',
```

Out[351]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e4fd4f3550>
```



comment

The calculated flight per week is highly correlated flight_per_week1, which has a log missing value. I am going to use the calculated flight per week as the new feature to represent flight count.

2.2.2 Route

In [225]:
```
df_train['route1'] = df_train['Route'].apply(lambda x: x[:3])
```

In [327]:
```
df_no_price_miss['route1'] = df_no_price_miss['Route'].apply(lambda x: x[:3])
```

In [226]:
```
df_train['route1'].head()
```

Out[226]:
```
0    HND
1    CTS
2    HND
3    FUK
4    HND
Name: route1, dtype: object
```

In [227]:
```
df_train['route2'] = df_train['Route'].apply(lambda x: x[-3:])
```

In [331]:
```
df_no_price_miss['route2'] = df_no_price_miss['Route'].apply(lambda x: x[-3:])
```

In [228]:
```
df_train['route2'].head()
```

Out[228]:
```
0    CTS
1    HND
2    FUK
3    HND
4    ITM
Name: route2, dtype: object
```

In [229]:
```
df_train['route1'].value_counts()
```

Out[229]:
```
MAD    77
YYZ    66
DOH    64
YYC    62
MIA    60
       ..
FKS     1
BOD     1
LTO     1
KIN     1
BSL     1
Name: route1, Length: 454, dtype: int64
```

In [230]:
```
df_train['route2'].value_counts()
```

Out[230]:
```
MAD    78
???    77
YYZ    69
DOH    62
HND    60
       ..
KZN     1
AXT     1
SAP     1
LTO     1
TAO     1
Name: route2, Length: 440, dtype: int64
```

In [231]:
```
df_train[df_train['route2']=='???'].head()
```

Out[231]:

	Route	Flight Count	flight per week 1	flights per week 2	flights per week 3	flights per month 1	flights per month 2	flights per month 3	First Flow	Last Flown	Airline Count	Ai
836	HND-???	148	10.0	5.0	3.0	22.0	6.0	2.0	2017-07-14 01:53:00	2018-07-24 06:47:00	1	
1781	MAD-???	18	NaN	NaN	1.0	3.0	2.0	1.0	2017-08-30 11:07:00	2018-07-25 10:15:00	2	
1967	HEL-???	10	NaN	NaN	NaN	2.0	1.0	1.0	2017-10-19 14:48:00	2018-06-07 21:11:00	1	
2106	YYZ-???	5	NaN	1.0	NaN	2.0	NaN	NaN	2018-01-10 15:10:00	2018-06-28 01:43:00	1	
2171	BKK-???	2	NaN	NaN	NaN	1.0	NaN	NaN	2018-03-14 01:17:00	2018-06-01 07:08:00	2	

In [329]:
```
df_no_price_miss.groupby('route1').agg({'avg(a.Price)':'mean'}).sort_values('avg(
```

Out[329]:

route1	avg(a.Price)
TNA	550.000000
XFW	393.750000
AKJ	289.720000
TOY	145.882500
MMY	131.084000
FKS	121.895000
HKD	108.161667
NTQ	99.250000
ISG	96.771429
MMV	96.330000

In [236]:
```
df_train[df_train['route1']=='OIT'].head()
```

Out[236]:

	Route	Flight Count	flight per week 1	flights per week 2	flights per week 3	flights per month 1	flights per month 2	flights per month 3	First Flow	Last Flown	Airline Count	Firs Airline
255	OIT-HND	1031	22.0	8.0	7.0	47.0	63.0	59.0	2017-07-01 22:50:00	2018-07-25 22:45:00	1	ANA

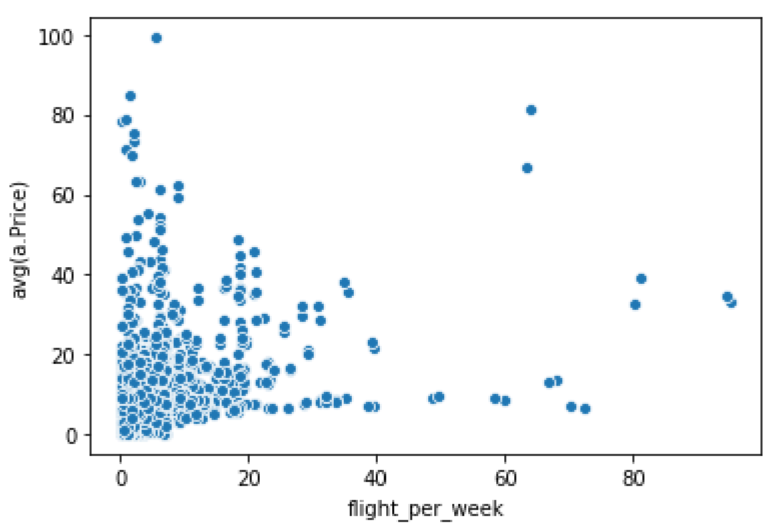
In [332]:
```
df_no_price_miss.groupby('route2').agg({'avg(a.Price)':'mean'}).sort_values('avg(
```

Out[332]:

route2	avg(a.Price)
OVB	393.750000
AKJ	382.855000
OIT	299.370000
TAK	203.240000
TOY	191.416667
NTQ	161.240000
KIJ	139.950000
MMY	122.694000
ISG	112.830000
MYJ	93.313333

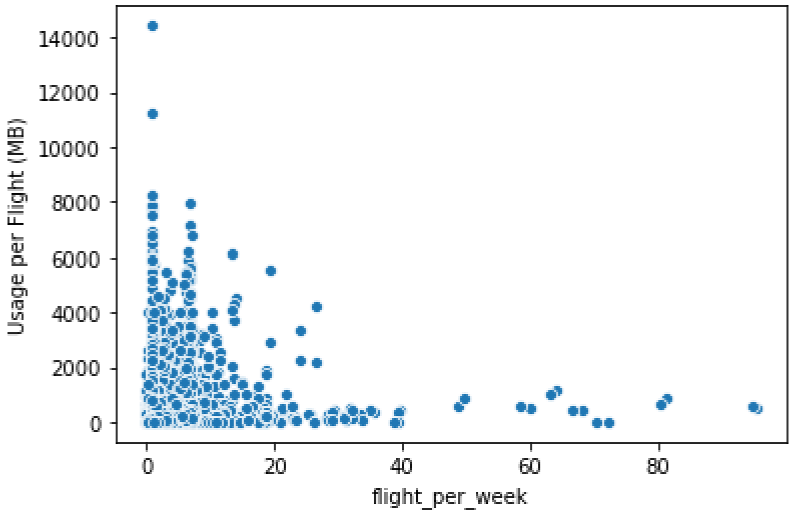
In [325]:
```
sns.scatterplot(data = df_train, x = 'flight_per_week', y = 'avg(a.Price)')
```

Out[325]: <matplotlib.axes._subplots.AxesSubplot at 0x1e48c18a668>



In [326]:
```
sns.scatterplot(data = df_train, x = 'flight_per_week', y = 'Usage per Flight (MB
```

Out[326]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e48c182fd0>
```



comments

There are about 454 and 440 departure and arrival airports. The price may also related to the airport city, thus I created these two features as predictor.

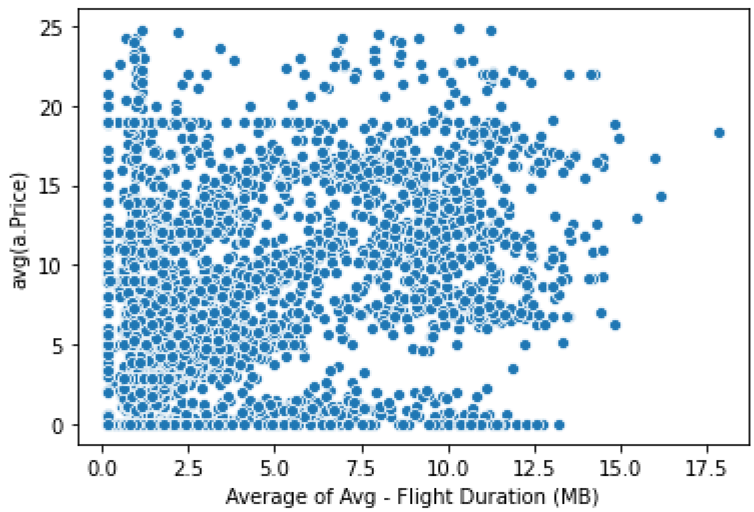
2.2.3 Average of Avg - Flight Duration (MB)

In [243]:
```
df_train['Average of Avg - Flight Duration (MB)'].describe()
```

Out[243]:
```
count    3422.000000
mean        4.407600
std         3.497652
min         0.170000
25%         1.643081
50%         3.173869
75%         6.719651
max        17.820000
Name: Average of Avg - Flight Duration (MB), dtype: float64
```

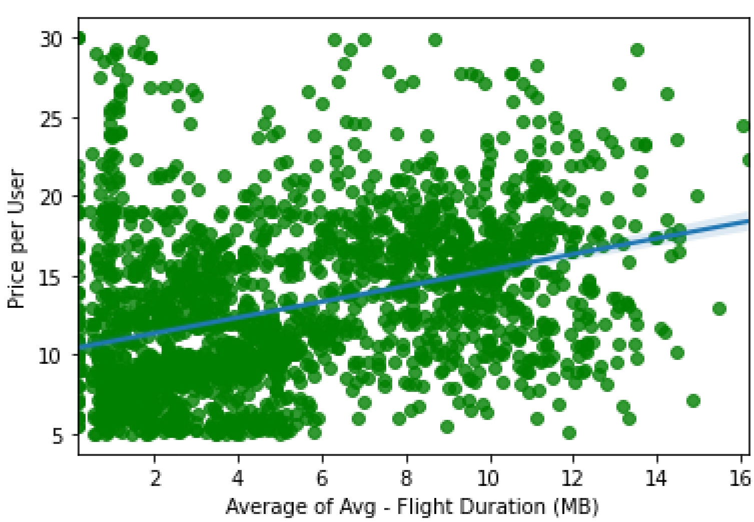
In [333]:
```
sns.scatterplot(data = df_train[df_train['avg(a.Price)']<25], x = 'Average of Avg
```

Out[333]: <matplotlib.axes._subplots.AxesSubplot at 0x1e48c325668>



In [349]:
```
sns.regplot(data = df_train[(df_train['Price per User']<30)&(df_train['Price per
```

Out[349]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4feec6128>



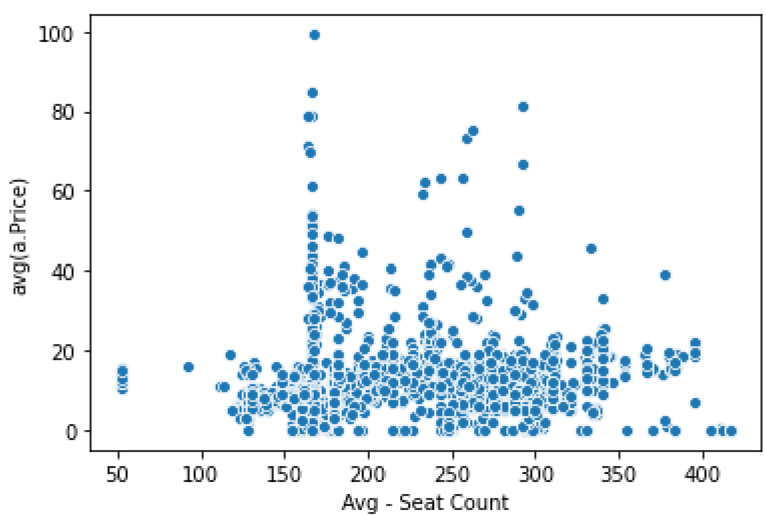
2.2.4 'Avg - Seat Count'

In [248]:
```
df_train['Avg - Seat Count'].describe()
```

Out[248]:
```
count    3422.000000
mean      218.160579
std        65.679458
min        52.000000
25%       168.327700
50%       185.739600
75%       270.623450
max       417.000000
Name: Avg - Seat Count, dtype: float64
```

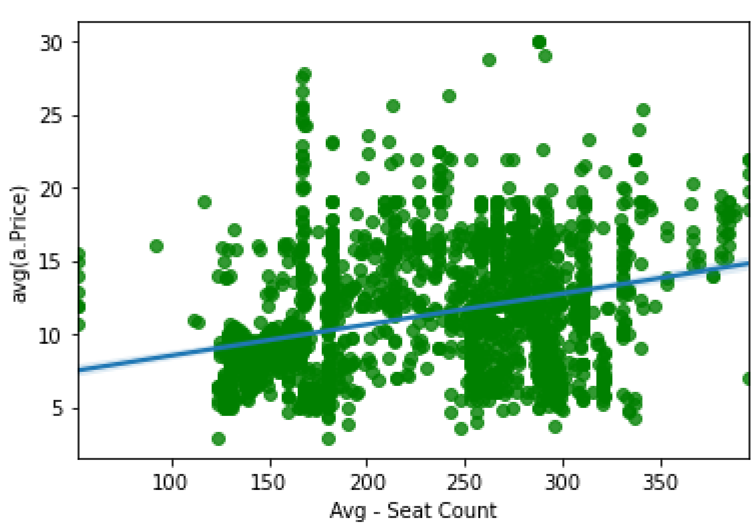
In [249]:
```
sns.scatterplot(data = df_train[df_train['avg(a.Price)']<100], x = 'Avg - Seat Co
```

Out[249]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e488d83cf8>
```



In [352]:
```
sns.regplot(data = df_train[(df_train['Price per User']<30)&(df_train['Price per
```

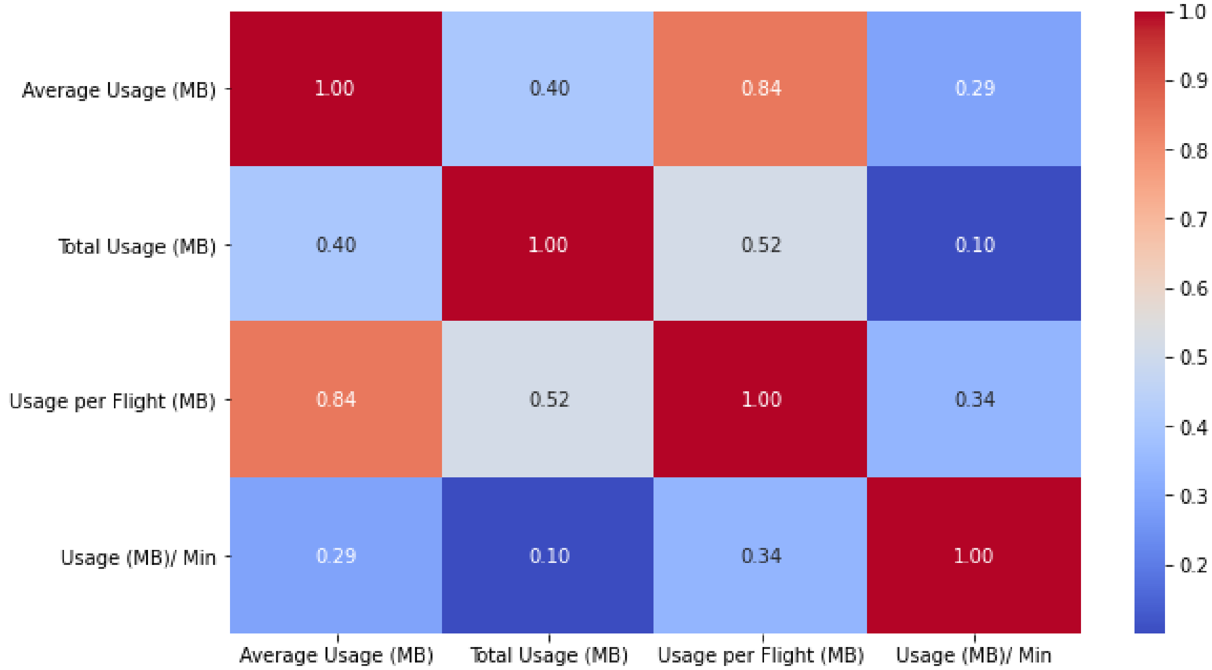
Out[352]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e48493e438>
```



2.2.5 Data Usage

'Average Usage (MB)', 'Min - Usage (MB)', 'Max - Usage (MB)', 'Total Usage (MB)', 'Usage per Flight (MB)', 'Usage (MB)/ Min', 'Min - Total Users', 'Max - Total Users'

In [251]:
```
data_usage_feature_list = ['Average Usage (MB)', 'Total Usage (MB)', 'Usage per F
fig, ax = plt.subplots(figsize = (10,6))
cm_df = sns.heatmap(df_train[data_usage_feature_list].corr(), annot = True, fmt =
```

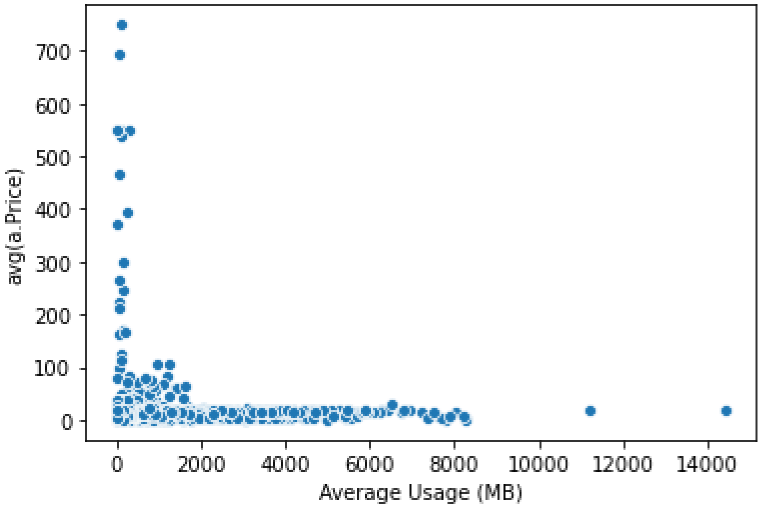


In [252]:
```
df_train['Average Usage (MB)'].describe()
```

Out[252]:
```
count     3422.000000
mean       786.213022
std       1189.176117
min          0.020000
25%         97.039999
50%        318.381678
75%        917.992848
max      14419.870120
Name: Average Usage (MB), dtype: float64
```

In [289]:
```
sns.scatterplot(data = df_no_price_miss, x = 'Average Usage (MB)', y = 'avg(a.Pri
```

Out[289]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e488cab0b8>
```

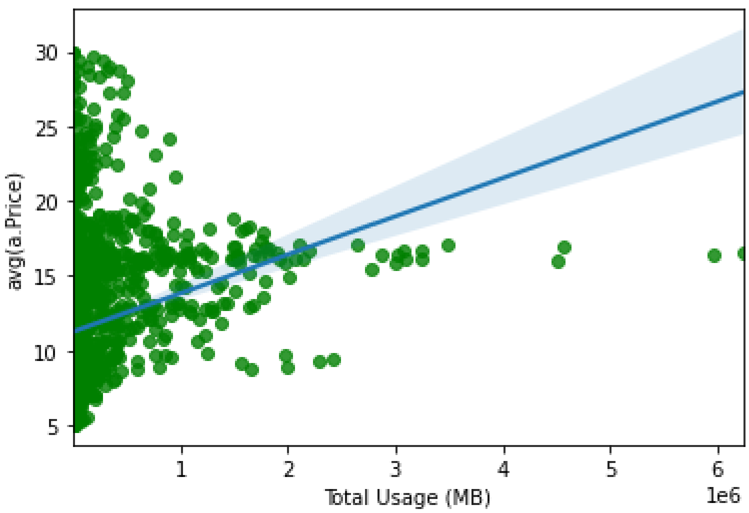


In [254]:
```
df_train['Total Usage (MB)'].describe()
```

Out[254]:
```
count    3.422000e+03
mean     1.094774e+05
std      3.799420e+05
min      2.000000e-02
25%      5.143600e+02
50%      4.207555e+03
75%      4.041058e+04
max      6.240478e+06
Name: Total Usage (MB), dtype: float64
```

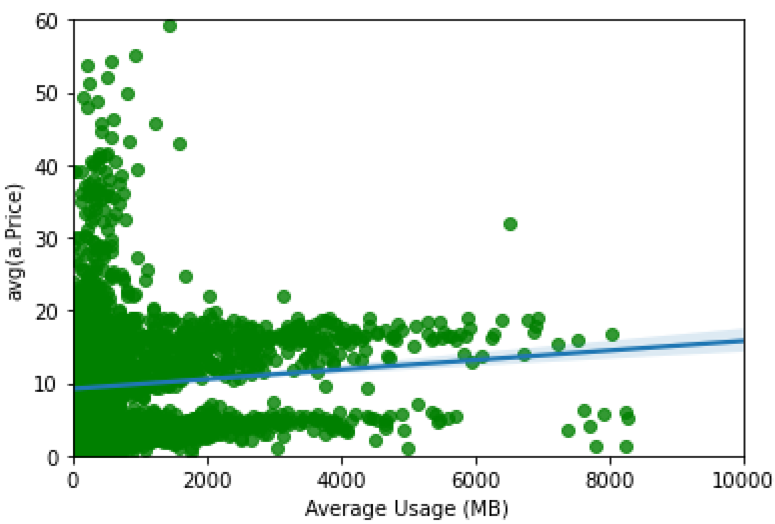
In [358]:
```
sns.regplot(data = df_train[(df_train['avg(a.Price)']<30)&(df_train['avg(a.Price)
                x = 'Total Usage (MB)', y = 'avg(a.Price)', scatter_kws={"color": "gr
```

Out[358]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4860e2c50>



In [374]:
```
sns.regplot(data = df_train[(df_train['avg(a.Price)']<60)&(df_train['avg(a.Price)
                x = 'Average Usage (MB)', y = 'avg(a.Price)', scatter_kws={"color": '
plt.ylim(0, 60)
plt.xlim(0, 10000)
```

Out[374]: (0.0, 10000.0)

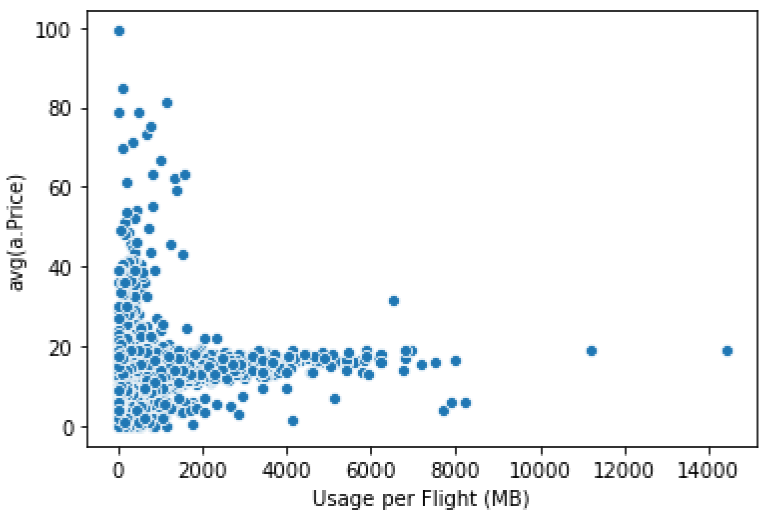


In [257]:
```
df_train['Usage per Flight (MB)'].describe()
```

Out[257]:
```
count     3422.000000
mean       515.336536
std       1007.743546
min          0.002778
25%         39.994282
50%        166.260204
75%        444.704813
max      14419.870120
Name: Usage per Flight (MB), dtype: float64
```

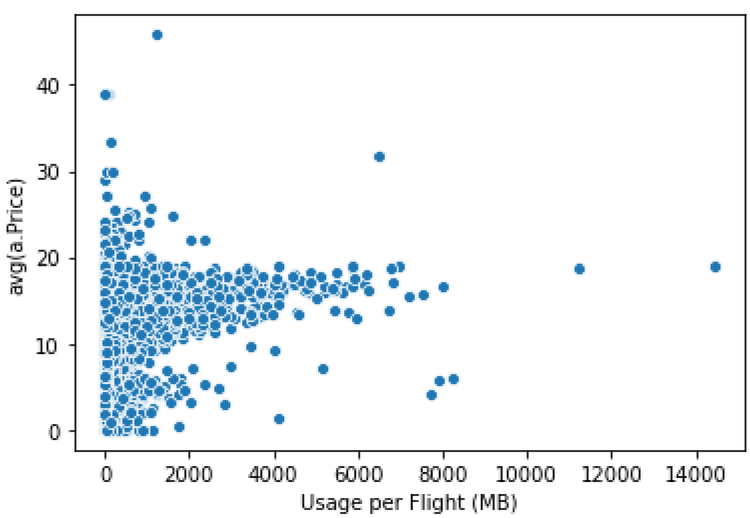
In [285]:
```
sns.scatterplot(data = df_train[df_train['avg(a.Price)']<100], x = 'Usage per Fli
```

Out[285]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e48111fe80>
```



In [286]:
```
sns.scatterplot(data = df_train[df_train['max(a.Price)']<200], x = 'Usage per Fli
```

Out[286]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e481174f60>
```

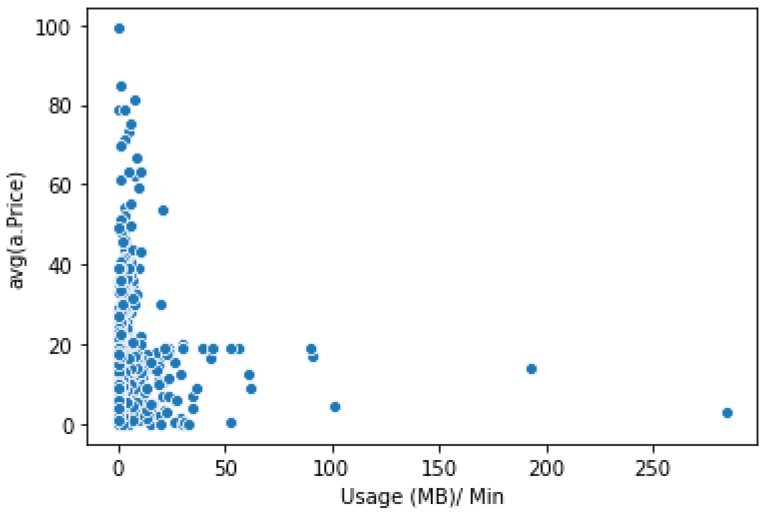


In [259]:
```
df_train['Usage (MB)/ Min'].describe()
```

Out[259]:
```
count    3422.000000
mean        2.284971
std         7.634427
min         0.000067
25%         0.242239
50%         0.866365
75%         2.311940
max       284.316992
Name: Usage (MB)/ Min, dtype: float64
```

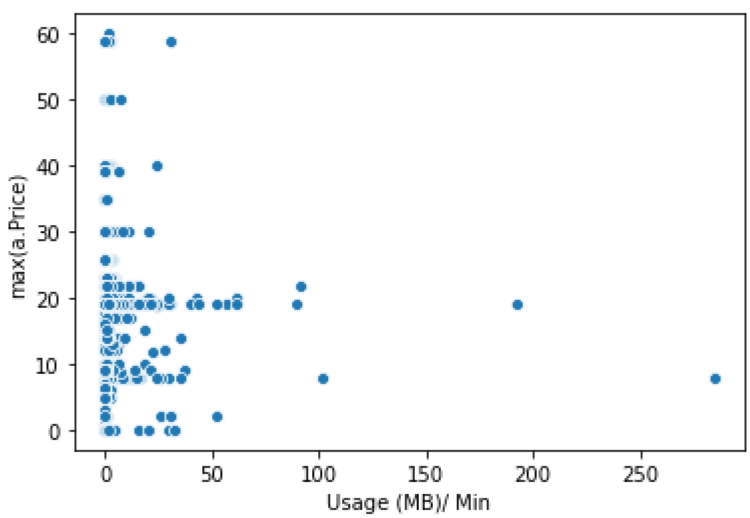
In [284]:
```
sns.scatterplot(data = df_train[df_train['avg(a.Price)']<100], x = 'Usage (MB)/ M
```

Out[284]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e489082748>
```



In [282]:
```
sns.scatterplot(data = df_train[df_train['max(a.Price)']<200], x = 'Usage (MB)/ M
```

Out[282]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e485f9c860>
```



comments

3)Generally, the average wifi price increases with the average usage volume of wifi. However, in certain routes, customers paid higher bills for very small data usage, while some customer paid cheap price for excessive wifi use. In the long term, this mismatch between usage and price may cause customer dissatisfaction and reduced revenue for the service provider.

4. Implement Model

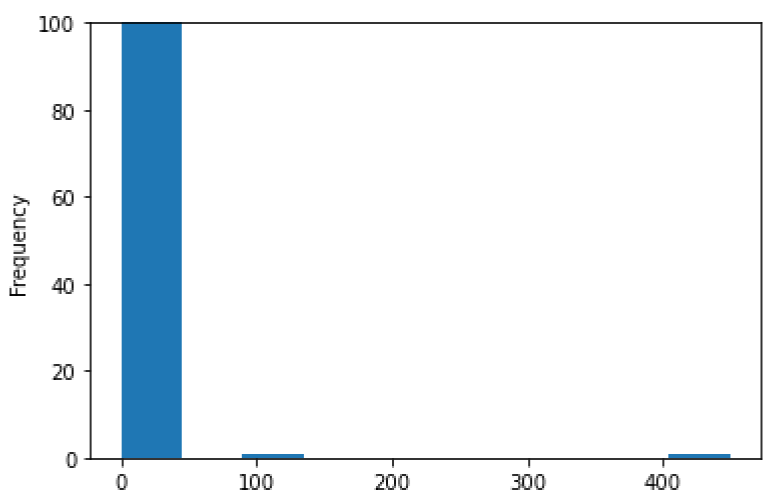
In [495]:
```
#create a feature price_per_mb to identify the problematic prices
df_no_price_miss['price_per_mb'] = df_no_price_miss['avg(a.Price)']/ df_no_price_
```

In [496]:
```
df_no_price_miss['price_per_mb'].describe()
```

Out[496]:
```
count    3447.000000
mean        0.321570
std         7.985295
min         0.000000
25%         0.005231
50%         0.022237
75%         0.078829
max       449.500000
Name: price_per_mb, dtype: float64
```

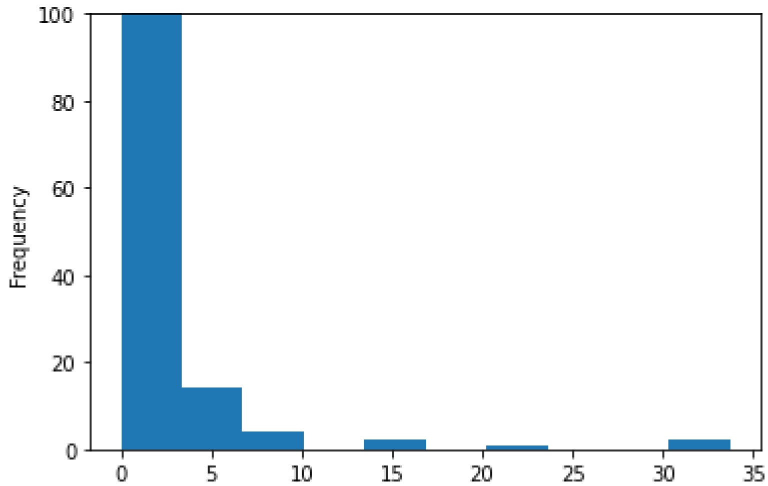
In [497]:
```
df_no_price_miss['price_per_mb'].plot.hist(ylim=(0,100))
```

Out[497]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e495b27a90>
```



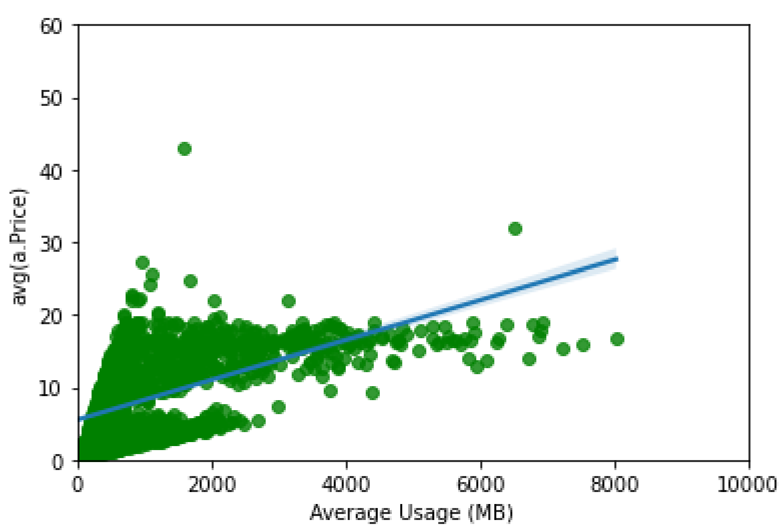
In [498]:
```
df_no_price_miss['price_per_mb'].loc[df_no_price_miss['price_per_mb']<100].plot.h
```

Out[498]: <matplotlib.axes._subplots.AxesSubplot at 0x1e495b4ad30>



In [499]:
```
sns.regplot(data = df_train[(df_no_price_miss['price_per_mb']<0.03)&((df_no_price
            x = 'Average Usage (MB)', y = 'avg(a.Price)', scatter_kws={"color": '
plt.ylim(0, 60)
plt.xlim(0, 10000)
```

Out[499]: (0.0, 10000.0)



In [500]:
```
df_train = df_no_price_miss[~df_no_price_miss['Average Usage (MB)'].isnull()]
```

In [501]:
```
df_train['First Flow'] = pd.to_datetime(df_train['First Flow'])
df_train['Last Flown'] = pd.to_datetime(df_train['Last Flown'])
```

In [502]:
```
df_train1 = df_train[(df_no_price_miss['price_per_mb']<0.03)&((df_no_price_miss['
df_test1 = df_train[(df_no_price_miss['price_per_mb']>=0.03)|((df_no_price_miss['
```

In [504]:
```
# select the key features for model building
df_train2 = df_train1[[
 'Flight Count',
 'Average of Avg - Flight Duration (MB)',
 'Avg - Seat Count',
 'Average Usage (MB)',
 'Total Usage (MB)',
 'Usage per Flight (MB)',
'avg(a.Price)']]
```

In [505]:
```
df_test2 = df_test1[[
 'Flight Count',
 'Average of Avg - Flight Duration (MB)',
 'Avg - Seat Count',
 'Average Usage (MB)',
 'Total Usage (MB)',
 'Usage per Flight (MB)',
'avg(a.Price)']]
```

In [506]:
```
X = df_train2[[
 'Flight Count',
 'Average of Avg - Flight Duration (MB)',
 'Avg - Seat Count',
 'Average Usage (MB)',
 'Total Usage (MB)',
 'Usage per Flight (MB)']].values
y = df_train2['avg(a.Price)'].values.reshape(-1,1)
```

In [508]:
```
# use the random forest algorithm
from sklearn.ensemble import RandomForestRegressor
```

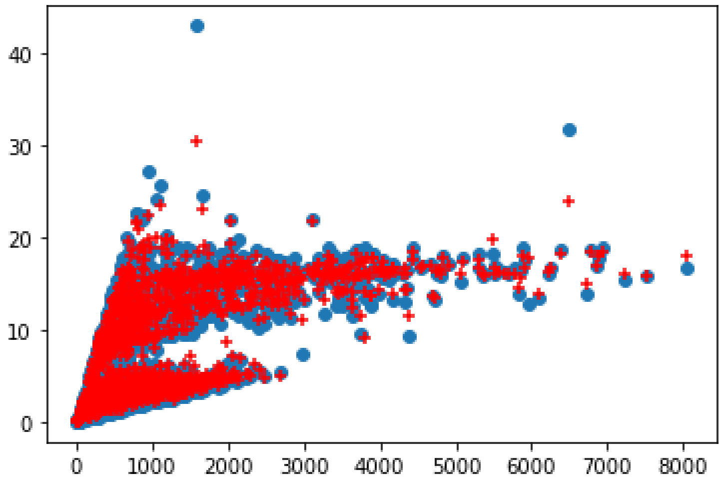
In [509]:
```
RF = RandomForestRegressor(n_estimators=100, random_state=100)
```

In [510]:
```
RF.fit(X,y)
```

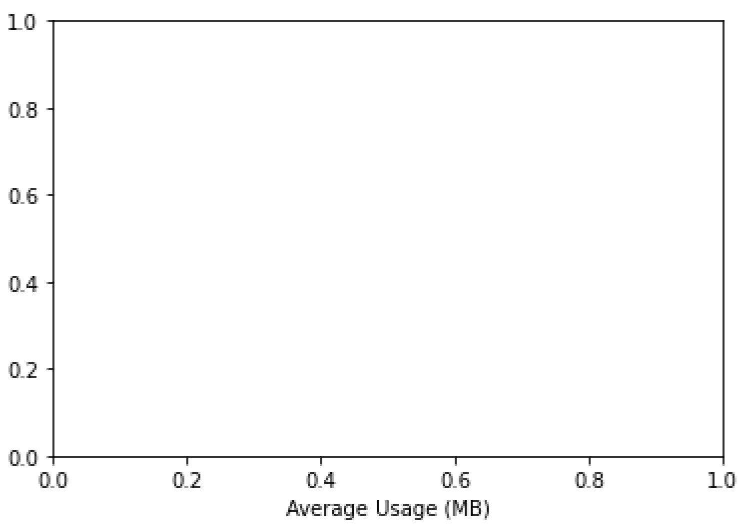
Out[510]:
```
RandomForestRegressor(random_state=100)
```

In [511]:
```
y_pred = RF.predict(X)
```

In [426]:
```
plt.scatter(df_train2['Average Usage (MB)'], y)
plt.scatter(df_train2['Average Usage (MB)'], y_pred, color='red', marker = '+')
plt.show()
plt.xlabel('Average Usage (MB)')
```



Out[426]: Text(0.5, 0, 'Average Usage (MB)')



In [512]:
```
X_test = df_test2[[
 'Flight Count',
 'Average of Avg - Flight Duration (MB)',
 'Avg - Seat Count',
 'Average Usage (MB)',
 'Total Usage (MB)',
 'Usage per Flight (MB)']].values
```

In [513]:
```
y_pred_test = RF.predict(X_test)
```

In [514]:
```
y_pred_test
```

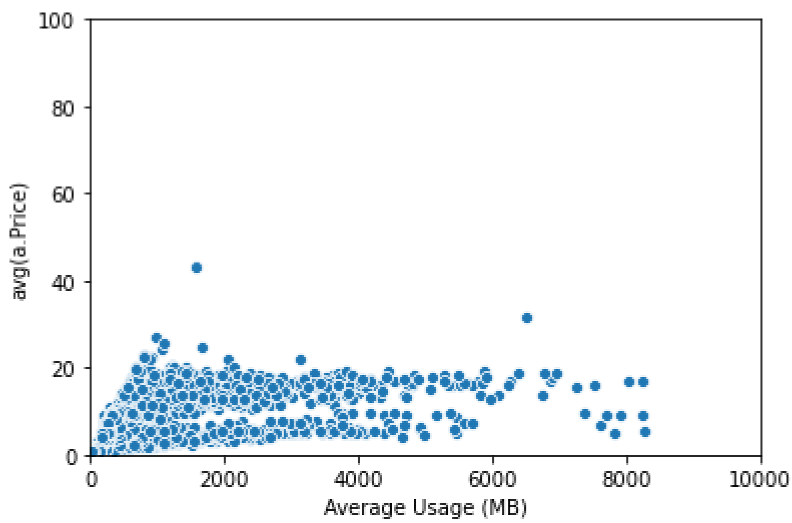
Out[514]:
```
array([11.9316, 12.2075, 14.1616, ...,  0.3702,  0.3899,  0.6443])
```

In [515]:
```
df_train3 = df_train
```

In [516]:
```
df_train3['avg(a.Price)'].loc[(df_no_price_miss['price_per_mb']>=0.03)|((df_no_pr
```

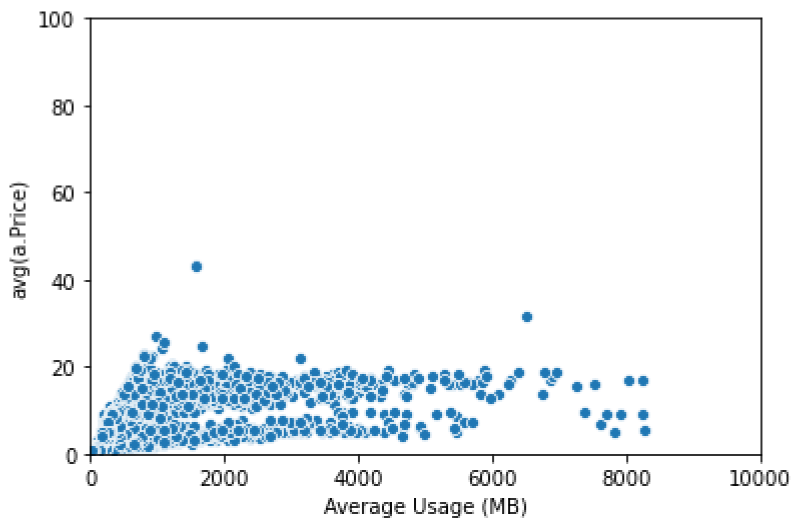
In [518]:
```
sns.scatterplot(data = df_train[(df_train['avg(a.Price)']<100)&(df_train['avg(a.P
                x = 'Average Usage (MB)', y = 'avg(a.Price)')
plt.ylim(0, 100)
plt.xlim(0, 10000)
```

Out[518]:
```
(0.0, 10000.0)
```



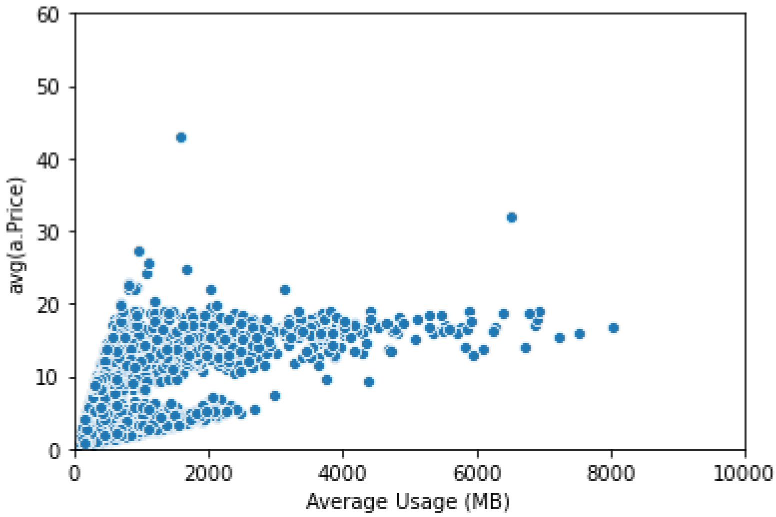
In [519]:
```
sns.scatterplot(data = df_train3[(df_train['avg(a.Price)']<100)&(df_train['avg(a.
                x = 'Average Usage (MB)', y = 'avg(a.Price)')
plt.ylim(0, 100)
plt.xlim(0, 10000)
```

Out[519]: (0.0, 10000.0)



In [445]:
```
sns.scatterplot(data = df_train1[(df_train1['avg(a.Price)']<60)&(df_train1['avg(a
                x = 'Average Usage (MB)', y = 'avg(a.Price)')
plt.ylim(0, 60)
plt.xlim(0, 10000)
```

Out[445]: (0.0, 10000.0)



Comments

The average price predictive model was able to provide a reasonable In-Flight WiFi price reference for routes the price is either too high or too low. The result is pretty preliminary, not much models and parameters were tested. In the future, we can include more data if available or improve the result by using testing different models and fine-tuning the parameters.