AirBnB Seattle Project

This project is part of the Udacity's Data Science Nanodegree program where I have chosen to write a blog post using AirBnB's Seattle Open Data.

The CRISP-DM (Cross Industry Process for Data Mining) process serves as a guide for the project and is as follows:

- Business Understanding
- Data Understanding
- Data Preparation
- Modeling
- Evaluation
- Deploy

Business Understanding

By exploring the AirBnB Seattle data, I hope to gain more understanding on the Seattle rental market, specifically, questions regarding:

- What property features determine the listing price?
- What property features determine the its popularity? (with reviews per month as proxy)
- When is the most popular month to rent in Seattle?

Data Understanding

In [1]:
```
# Import libraries
```

```python
import glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import helper_functions as helper

from datetime import datetime
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import make_scorer,fbeta_score, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.feature_extraction.text import CountVectorizer
from scipy.sparse import csr_matrix

pd.options.display.max_columns = 500
pd.options.display.max_rows = 500
```

In [2]:
```python
csvs = glob.glob('./input/seattle/*.csv')
csvs
```

Out[2]:
```
['./input/seattle\\calendar.csv',
 './input/seattle\\listings.csv',
 './input/seattle\\reviews.csv']
```

In [3]:
```python
# Listings
base = pd.read_csv(csvs[1])
listings_df = base.copy()
listings_df.head(2)
```

Out[3]:

id	listing_url	scrape_id	last_scraped	name	summary

	id	listing_url	scrape_id	last_scraped	name	summar
0	241032	https://www.airbnb.com/rooms/241032	20160104002432	2016-01-04	Stylish Queen Anne Apartment	Na
1	953595	https://www.airbnb.com/rooms/953595	20160104002432	2016-01-04	Bright & Airy Queen Anne Apartment	Chemicall sensitive' We'v remove the irrita..

Features that are available for analysis: Including Non-Null count and Datatypes

In [4]:
```
listings_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3818 entries, 0 to 3817
Data columns (total 92 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   id                                3818 non-null   int64
 1   listing_url                       3818 non-null   object
 2   scrape_id                         3818 non-null   int64
 3   last_scraped                      3818 non-null   object
 4   name                              3818 non-null   object
 5   summary                           3641 non-null   object
 6   space                             3249 non-null   object
 7   description                       3818 non-null   object
 8   experiences_offered               3818 non-null   object
 9   neighborhood_overview             2786 non-null   object
 10  notes                             2212 non-null   object
 11  transit                           2884 non-null   object
```

```
12  thumbnail_url                     3498 non-null   object
13  medium_url                        3498 non-null   object
14  picture_url                       3818 non-null   object
15  xl_picture_url                    3498 non-null   object
16  host_id                           3818 non-null   int64
17  host_url                          3818 non-null   object
18  host_name                         3816 non-null   object
19  host_since                        3816 non-null   object
20  host_location                     3810 non-null   object
21  host_about                        2959 non-null   object
22  host_response_time                3295 non-null   object
23  host_response_rate                3295 non-null   object
24  host_acceptance_rate              3045 non-null   object
25  host_is_superhost                 3816 non-null   object
26  host_thumbnail_url                3816 non-null   object
27  host_picture_url                  3816 non-null   object
28  host_neighbourhood                3518 non-null   object
29  host_listings_count               3816 non-null   float64
30  host_total_listings_count         3816 non-null   float64
31  host_verifications                3818 non-null   object
32  host_has_profile_pic              3816 non-null   object
33  host_identity_verified            3816 non-null   object
34  street                            3818 non-null   object
35  neighbourhood                     3402 non-null   object
36  neighbourhood_cleansed            3818 non-null   object
37  neighbourhood_group_cleansed      3818 non-null   object
38  city                              3818 non-null   object
39  state                             3818 non-null   object
40  zipcode                           3811 non-null   object
41  market                            3818 non-null   object
42  smart_location                    3818 non-null   object
43  country_code                      3818 non-null   object
44  country                           3818 non-null   object
45  latitude                          3818 non-null   float64
46  longitude                         3818 non-null   float64
47  is_location_exact                 3818 non-null   object
48  property_type                     3817 non-null   object
49  room_type                         3818 non-null   object
50  accommodates                      3818 non-null   int64
```

```
51  bathrooms                         3802 non-null   float64
52  bedrooms                          3812 non-null   float64
53  beds                              3817 non-null   float64
54  bed_type                          3818 non-null   object
55  amenities                         3818 non-null   object
56  square_feet                       97 non-null     float64
57  price                             3818 non-null   object
58  weekly_price                      2009 non-null   object
59  monthly_price                     1517 non-null   object
60  security_deposit                  1866 non-null   object
61  cleaning_fee                      2788 non-null   object
62  guests_included                   3818 non-null   int64
63  extra_people                      3818 non-null   object
64  minimum_nights                    3818 non-null   int64
65  maximum_nights                    3818 non-null   int64
66  calendar_updated                  3818 non-null   object
67  has_availability                  3818 non-null   object
68  availability_30                   3818 non-null   int64
69  availability_60                   3818 non-null   int64
70  availability_90                   3818 non-null   int64
71  availability_365                  3818 non-null   int64
72  calendar_last_scraped             3818 non-null   object
73  number_of_reviews                 3818 non-null   int64
74  first_review                      3191 non-null   object
75  last_review                       3191 non-null   object
76  review_scores_rating              3171 non-null   float64
77  review_scores_accuracy            3160 non-null   float64
78  review_scores_cleanliness         3165 non-null   float64
79  review_scores_checkin             3160 non-null   float64
80  review_scores_communication       3167 non-null   float64
81  review_scores_location            3163 non-null   float64
82  review_scores_value               3162 non-null   float64
83  requires_license                  3818 non-null   object
84  license                           0 non-null      float64
85  jurisdiction_names                3818 non-null   object
86  instant_bookable                  3818 non-null   object
87  cancellation_policy               3818 non-null   object
88  require_guest_profile_picture     3818 non-null   object
89  require_guest_phone_verification  3818 non-null   object
```

```
 90  calculated_host_listings_count    3818 non-null   int64
 91  reviews_per_month                 3191 non-null   float64
dtypes: float64(17), int64(13), object(62)
memory usage: 2.7+ MB
```

Data Preparation

A preliminary investigation reveals that further data cleansing and preprocessing is needed before it can be used as an input for our project pipeline.

The steps that needed to be undertaken are as per follows:

- Removing Unary and URL features except for 'thumbnail_url'.
- Removing features with many missing values : >= 21% missing.
- Removing irrelevant features that lacks business justification or are too granular.
  - Text fields except for 'description'.
  - Fields with redundant information like 30,60,90 day availability, retain only availability_365.
  - Similarly with weekly, monthly price, retain only 'price' field.
  - The detailed split of review scores except the final rating.
  - More granular aspects of location.
- Remove special characters like '$' and ',' from price features.
- Remove '%' character from percentage features.
- Convert thumnail data as available or not-available.
- Convert host since from date to number of days.
- Convert amenities column to number of amenities.
- Impute Null as Zero for security deposit and cleaning fees.
- Count number of amenities from list.

Features that are Unary and URL

In [5]:
```
unary_columns, url_columns = [], []
```

```python
for i in listings_df.columns:
    if len(listings_df[i].unique())==1:
        print('Unary Column: ',i , listings_df[i].unique())
        unary_columns.append(i)
    if 'url' in i:
        url_columns = url_columns+[i]
```

```
Unary Column:  scrape_id [20160104002432]
Unary Column:  last_scraped ['2016-01-04']
Unary Column:  experiences_offered ['none']
Unary Column:  market ['Seattle']
Unary Column:  country_code ['US']
Unary Column:  country ['United States']
Unary Column:  has_availability ['t']
Unary Column:  calendar_last_scraped ['2016-01-04']
Unary Column:  requires_license ['f']
Unary Column:  license [nan]
Unary Column:  jurisdiction_names ['WASHINGTON']
```

In [6]:
```python
url_columns
```

Out[6]:
```
['listing_url',
 'thumbnail_url',
 'medium_url',
 'picture_url',
 'xl_picture_url',
 'host_url',
 'host_thumbnail_url',
 'host_picture_url']
```

In [7]:
```python
# Drop all URL variables but convert [thumbnail_url] into binary
listings_df[url_columns].isnull().sum()/len(listings_df)
```

Out[7]:
```
listing_url           0.000000
thumbnail_url         0.083814
medium_url            0.083814
picture_url           0.000000
xl_picture_url        0.083814
host_url              0.000000
```

```
host_thumbnail_url    0.000524
host_picture_url      0.000524
dtype: float64
```

In [8]:
```
url_columns = list(set(url_columns) - {'thumbnail_url'})
```

Features that are Unary and URL Features with Many Missing Values

In [9]:
```
percent_missing = listings_df.isnull().sum() * 100 / len(listings_df)
missing_value_df = pd.DataFrame({   'column_name': listings_df.columns,
                                    'percent_missing': round(percent_mi
ssing,2)})
missing_value_df.sort_values('percent_missing', inplace=True, ascending
=[False])
missing_value_df.reset_index(drop=True)
```

Out[9]:

	column_name	percent_missing
0	license	100.00
1	square_feet	97.46
2	monthly_price	60.27
3	security_deposit	51.13
4	weekly_price	47.38
5	notes	42.06
6	neighborhood_overview	27.03
7	cleaning_fee	26.98
8	transit	24.46
9	host_about	22.50
10	host_acceptance_rate	20.25
11	review_scores_accuracy	17.23
12	review_scores_checkin	17.23

	column_name	percent_missing
13	review_scores_value	17.18
14	review_scores_location	17.16
15	review_scores_cleanliness	17.10
16	review_scores_communication	17.05
17	review_scores_rating	16.95
18	last_review	16.42
19	first_review	16.42
20	reviews_per_month	16.42
21	space	14.90
22	host_response_rate	13.70
23	host_response_time	13.70
24	neighbourhood	10.90
25	thumbnail_url	8.38
26	medium_url	8.38
27	xl_picture_url	8.38
28	host_neighbourhood	7.86
29	summary	4.64
30	bathrooms	0.42
31	host_location	0.21
32	zipcode	0.18
33	bedrooms	0.16
34	host_name	0.05
35	host_listings_count	0.05
36	host_since	0.05
37	host_is_superhost	0.05

	column_name	percent_missing
38	host_identity_verified	0.05
39	host_picture_url	0.05
40	host_thumbnail_url	0.05
41	host_total_listings_count	0.05
42	host_has_profile_pic	0.05
43	property_type	0.03
44	beds	0.03
45	require_guest_profile_picture	0.00
46	calculated_host_listings_count	0.00
47	maximum_nights	0.00
48	calendar_updated	0.00
49	has_availability	0.00
50	require_guest_phone_verification	0.00
51	instant_bookable	0.00
52	availability_30	0.00
53	availability_60	0.00
54	availability_90	0.00
55	availability_365	0.00
56	calendar_last_scraped	0.00
57	number_of_reviews	0.00
58	cancellation_policy	0.00
59	jurisdiction_names	0.00
60	requires_license	0.00
61	extra_people	0.00
62	minimum_nights	0.00

63	id	0.00

	column_name	percent_missing
64	guests_included	0.00
65	price	0.00
66	scrape_id	0.00
67	last_scraped	0.00
68	name	0.00
69	description	0.00
70	experiences_offered	0.00
71	picture_url	0.00
72	host_id	0.00
73	host_url	0.00
74	host_verifications	0.00
75	street	0.00
76	neighbourhood_cleansed	0.00
77	neighbourhood_group_cleansed	0.00
78	city	0.00
79	state	0.00
80	market	0.00
81	smart_location	0.00
82	country_code	0.00
83	country	0.00
84	latitude	0.00
85	listing_url	0.00
86	is_location_exact	0.00
87	room_type	0.00

	column_name	percent_missing
88	accommodates	0.00
89	bed_type	0.00
90	amenities	0.00
91	longitude	0.00

In [10]:

```
missing_value_df = missing_value_df[missing_value_df['percent_missing']
>=21.00].reset_index(drop=True)
missing_value_df
```

Out[10]:

	column_name	percent_missing
0	license	100.00
1	square_feet	97.46
2	monthly_price	60.27
3	security_deposit	51.13
4	weekly_price	47.38
5	notes	42.06
6	neighborhood_overview	27.03
7	cleaning_fee	26.98
8	transit	24.46
9	host_about	22.50

Impute NULL to be ZERO for security deposit and cleaning fee

In [11]:

```
# Assume NULL to be ZERO in security_deposit, cleaning_fee
missing_columns = missing_value_df['column_name'].values.tolist()
missing_columns = list(set(missing_columns) - {'security_deposit','clea
```

```
ning_fee'})
missing_columns
```

Out[11]:
```
['license',
 'weekly_price',
 'monthly_price',
 'notes',
 'square_feet',
 'host_about',
 'transit',
 'neighborhood_overview']
```

Removing Features that are obviously irrelevant (as described above)

In [12]:
```
irrelevant_columns = [   'id'
                         ,'name'
                         ,'summary'
                         ,'space'
                         ,'host_id'
                         ,'host_name'
                         ,'host_location'
                         ,'host_neighbourhood'
                         ,'host_listings_count'
                         ,'host_total_listings_count'
                         ,'host_verifications'
                         ,'street'
                         ,'neighbourhood'
                         ,'neighbourhood_cleansed'
                         ,'city'
                         ,'state'
                         ,'zipcode'
                         ,'smart_location'
                         ,'latitude'
                         ,'longitude'
                         ,'is_location_exact'
                         ,'minimum_nights'
                         ,'maximum_nights'
                         ,'calendar_updated'
```

```
                          ,'availability_30'
                          ,'availability_60'
                          ,'availability_90'
                          ,'first_review'
                          ,'last_review'
                          ,'require_guest_profile_picture'
                          ,'require_guest_phone_verification'
                          ,'calculated_host_listings_count']
```

Features that are Unary and URL Consolidate List of Features that are to be removed

In [13]:
```
remove_columns = unary_columns + url_columns + missing_columns + irrele
vant_columns
remove_columns
```

Out[13]:
```
['scrape_id',
 'last_scraped',
 'experiences_offered',
 'market',
 'country_code',
 'country',
 'has_availability',
 'calendar_last_scraped',
 'requires_license',
 'license',
 'jurisdiction_names',
 'xl_picture_url',
 'listing_url',
 'host_url',
 'picture_url',
 'medium_url',
 'host_thumbnail_url',
 'host_picture_url',
 'license',
 'weekly_price',
 'monthly_price',
 'notes',
 'square_feet',
```

```
 'square_feet',
 'host_about',
 'transit',
 'neighborhood_overview',
 'id',
 'name',
 'summary',
 'space',
 'host_id',
 'host_name',
 'host_location',
 'host_neighbourhood',
 'host_listings_count',
 'host_total_listings_count',
 'host_verifications',
 'street',
 'neighbourhood',
 'neighbourhood_cleansed',
 'city',
 'state',
 'zipcode',
 'smart_location',
 'latitude',
 'longitude',
 'is_location_exact',
 'minimum_nights',
 'maximum_nights',
 'calendar_updated',
 'availability_30',
 'availability_60',
 'availability_90',
 'first_review',
 'last_review',
 'require_guest_profile_picture',
 'require_guest_phone_verification',
 'calculated_host_listings_count']
```

In [14]:
```
listings_df.drop(listings_df[remove_columns], axis=1, inplace=True)
```

In [15]:
```
listings_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3818 entries, 0 to 3817
Data columns (total 35 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   description                   3818 non-null   object 
 1   thumbnail_url                 3498 non-null   object 
 2   host_since                    3816 non-null   object 
 3   host_response_time            3295 non-null   object 
 4   host_response_rate            3295 non-null   object 
 5   host_acceptance_rate          3045 non-null   object 
 6   host_is_superhost             3816 non-null   object 
 7   host_has_profile_pic          3816 non-null   object 
 8   host_identity_verified        3816 non-null   object 
 9   neighbourhood_group_cleansed  3818 non-null   object 
 10  property_type                 3817 non-null   object 
 11  room_type                     3818 non-null   object 
 12  accommodates                  3818 non-null   int64  
 13  bathrooms                     3802 non-null   float64
 14  bedrooms                      3812 non-null   float64
 15  beds                          3817 non-null   float64
 16  bed_type                      3818 non-null   object 
 17  amenities                     3818 non-null   object 
 18  price                         3818 non-null   object 
 19  security_deposit              1866 non-null   object 
 20  cleaning_fee                  2788 non-null   object 
 21  guests_included               3818 non-null   int64  
 22  extra_people                  3818 non-null   object 
 23  availability_365              3818 non-null   int64  
 24  number_of_reviews             3818 non-null   int64  
 25  review_scores_rating          3171 non-null   float64
 26  review_scores_accuracy        3160 non-null   float64
 27  review_scores_cleanliness     3165 non-null   float64
 28  review_scores_checkin         3160 non-null   float64
 29  review_scores_communication   3167 non-null   float64
 30  review_scores_location        3163 non-null   float64
 31  review_scores_value           3162 non-null   float64
 32  instant_bookable              3818 non-null   object 
```

```
 33  cancellation_policy            3818 non-null   object
 34  reviews_per_month              3191 non-null   float64
dtypes: float64(11), int64(4), object(20)
memory usage: 1.0+ MB
```

Invoke Helper functions to further clean the dataframe

In [16]:
```
seattle_df = helper.clean_data(listings_df)
```

Data Preprocessing: Price

Based on the questions above two different set of features were required to carry out the analysis.

- Keep remaining features except property text description for predicting price.
- Use property text description alone for price prediction.

In [17]:
```
seattle_df.drop('description', axis=1, inplace=True)
seattle_df.head()
```

Out[17]:

	host_since	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost
0	1607.0	within a few hours	96.0	100.0	0
1	1047.0	within an hour	98.0	100.0	1
2	571.0	within a few hours	67.0	100.0	0
3	789.0	NaN	NaN	NaN	0
4	1497.0	within an hour	100.0	NaN	0

In [18]:
```
seattle_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3818 entries, 0 to 3817
Data columns (total 34 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   host_since                    3816 non-null   float64
 1   host_response_time            3295 non-null   object 
 2   host_response_rate            3295 non-null   float64
 3   host_acceptance_rate          3045 non-null   float64
 4   host_is_superhost             3818 non-null   int64  
 5   host_has_profile_pic          3818 non-null   int64  
 6   host_identity_verified        3818 non-null   int64  
 7   neighbourhood_group_cleansed  3818 non-null   object 
 8   property_type                 3817 non-null   object 
 9   room_type                     3818 non-null   object 
 10  accommodates                  3818 non-null   int64  
 11  bathrooms                     3802 non-null   float64
 12  bedrooms                      3812 non-null   float64
 13  beds                          3817 non-null   float64
 14  bed_type                      3818 non-null   object 
 15  price                         3818 non-null   float64
 16  security_deposit              3818 non-null   float64
 17  cleaning_fee                  3818 non-null   float64
 18  guests_included               3818 non-null   int64  
 19  extra_people                  3818 non-null   float64
 20  availability_365              3818 non-null   int64  
 21  number_of_reviews             3818 non-null   int64  
 22  review_scores_rating          3171 non-null   float64
 23  review_scores_accuracy        3160 non-null   float64
 24  review_scores_cleanliness     3165 non-null   float64
 25  review_scores_checkin         3160 non-null   float64
 26  review_scores_communication   3167 non-null   float64
 27  review_scores_location        3163 non-null   float64
 28  review_scores_value           3162 non-null   float64
 29  instant_bookable              3818 non-null   int64  
 30  cancellation_policy           3818 non-null   int64  
 31  reviews_per_month             3191 non-null   float64
```

```
 32  has_thumbnail_url              3818 non-null   int32
 33  total_amenities                3818 non-null   int64
dtypes: float64(18), int32(1), int64(10), object(5)
memory usage: 999.4+ KB
```

Splitting Data into Features/Label and Dividing Label/Price feature into (High,Low)=(1,0)

In [19]:
```
label_df   = np.where(seattle_df['price'] > seattle_df['price'].median
(), 0, 1)
feature_df = seattle_df.drop(['price'], axis=1)
```

Impute Missing Categorical Values with 'Most Frequent' and Apply Min-Max Feature Scaling

In [20]:
```
scaled_df = helper.process_features(feature_df)
```

In [21]:
```
scaled_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3818 entries, 0 to 3817
Data columns (total 73 columns):
 #   Column                                                Non-Null Count
Dtype
---  ------                                                --------------
-----
 0   host_since                                            3818 non-null
float64
 1   host_response_rate                                    3818 non-null
float64
 2   host_acceptance_rate                                  3818 non-null
float64
 3   host_is_superhost                                     3818 non-null
float64
 4   host_has_profile_pic                                  3818 non-null
float64
```

```
 5   host_identity_verified                                         3818 non-null
float64
 6   accommodates                                                   3818 non-null
float64
 7   bathrooms                                                      3818 non-null
float64
 8   bedrooms                                                       3818 non-null
float64
 9   beds                                                           3818 non-null
float64
 10  security_deposit                                               3818 non-null
float64
 11  cleaning_fee                                                   3818 non-null
float64
 12  guests_included                                                3818 non-null
float64
 13  extra_people                                                   3818 non-null
float64
 14  availability_365                                               3818 non-null
float64
 15  number_of_reviews                                              3818 non-null
float64
 16  review_scores_rating                                           3818 non-null
float64
 17  review_scores_accuracy                                         3818 non-null
float64
 18  review_scores_cleanliness                                      3818 non-null
float64
 19  review_scores_checkin                                          3818 non-null
float64
 20  review_scores_communication                                    3818 non-null
float64
 21  review_scores_location                                         3818 non-null
float64
 22  review_scores_value                                            3818 non-null
float64
 23  instant_bookable                                               3818 non-null
float64
 24  cancellation_policy                                            3818 non-null
```

```
float64
 25  reviews_per_month                              3818 non-null
float64
 26  has_thumbnail_url                              3818 non-null
float64
 27  total_amenities                                3818 non-null
float64
 28  host_response_time_a_few_days_or_more          3818 non-null
float64
 29  host_response_time_within_a_day                3818 non-null
float64
 30  host_response_time_within_a_few_hours          3818 non-null
float64
 31  host_response_time_within_an_hour              3818 non-null
float64
 32  neighbourhood_group_cleansed_Ballard           3818 non-null
float64
 33  neighbourhood_group_cleansed_Beacon_Hill       3818 non-null
float64
 34  neighbourhood_group_cleansed_Capitol_Hill      3818 non-null
float64
 35  neighbourhood_group_cleansed_Cascade           3818 non-null
float64
 36  neighbourhood_group_cleansed_Central_Area      3818 non-null
float64
 37  neighbourhood_group_cleansed_Delridge          3818 non-null
float64
 38  neighbourhood_group_cleansed_Downtown          3818 non-null
float64
 39  neighbourhood_group_cleansed_Interbay          3818 non-null
float64
 40  neighbourhood_group_cleansed_Lake_City         3818 non-null
float64
 41  neighbourhood_group_cleansed_Magnolia          3818 non-null
float64
 42  neighbourhood_group_cleansed_Northgate         3818 non-null
float64
 43  neighbourhood_group_cleansed_Other_neighborhoods  3818 non-null
float64
```

```
 44  neighbourhood_group_cleansed_Queen_Anne          3818 non-null
float64
 45  neighbourhood_group_cleansed_Rainier_Valley      3818 non-null
float64
 46  neighbourhood_group_cleansed_Seward_Park         3818 non-null
float64
 47  neighbourhood_group_cleansed_University_District  3818 non-null
float64
 48  neighbourhood_group_cleansed_West_Seattle        3818 non-null
float64
 49  property_type_Apartment                          3818 non-null
float64
 50  property_type_Bed_and_Breakfast                  3818 non-null
float64
 51  property_type_Boat                               3818 non-null
float64
 52  property_type_Bungalow                           3818 non-null
float64
 53  property_type_Cabin                              3818 non-null
float64
 54  property_type_Camper_RV                          3818 non-null
float64
 55  property_type_Chalet                             3818 non-null
float64
 56  property_type_Condominium                        3818 non-null
float64
 57  property_type_Dorm                               3818 non-null
float64
 58  property_type_House                              3818 non-null
float64
 59  property_type_Loft                               3818 non-null
float64
 60  property_type_Other                              3818 non-null
float64
 61  property_type_Tent                               3818 non-null
float64
 62  property_type_Townhouse                          3818 non-null
float64
 63  property_type_Treehouse                          3818 non-null
```

```
float64
 64  property_type_Yurt                                   3818 non-null
float64
 65  room_type_Entire_home_apt                            3818 non-null
float64
 66  room_type_Private_room                               3818 non-null
float64
 67  room_type_Shared_room                                3818 non-null
float64
 68  bed_type_Airbed                                      3818 non-null
float64
 69  bed_type_Couch                                       3818 non-null
float64
 70  bed_type_Futon                                       3818 non-null
float64
 71  bed_type_Pull_out_Sofa                               3818 non-null
float64
 72  bed_type_Real_Bed                                    3818 non-null
float64
dtypes: float64(73)
memory usage: 2.1 MB
```

Modeling and Evaluation: Price

What property features determine the listing price?

AdaBoostClassifier with decision tree base estimator and optimized with GridSearchCV.

In [22]:

```
# Initialize the classifier
clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), rando
m_state=123)

# Create the parameters list to tune, using a dictionary.
parameters = {"n_estimators": [10, 50, 100],
              "learning_rate": [0.005, .01, 0.05, 0.1],
```

```
                'base_estimator__min_samples_split' : [2, 4, 6, 8],
                'base_estimator__max_depth' : [2, 4, 6, 8]}
```

In [23]:
```
best_clf, X_train, X_test, y_train, y_test = helper.boost_classifier(cl
f, parameters, scaled_df, label_df)
```

```
C:\Users\tai_j\Anaconda3\lib\site-packages\sklearn\utils\validation.py:
71: FutureWarning: Pass scoring=make_scorer(fbeta_score, beta=0.5) as k
eyword args. From version 0.25 passing these as positional arguments wi
ll result in an error
  FutureWarning)
```

In [24]:
```
print(best_clf)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=2,
                                                         min_samples_sp
lit=6),
                   learning_rate=0.1, n_estimators=100, random_state=12
3)
```

In [25]:
```
# Unoptimized model
test_accuracy, train_accuracy = helper.prediction_scores(clf, X_train,
X_test, y_train, y_test)
helper.print_scores(test_accuracy, train_accuracy)
```

```
Accuracy score on testing data: 0.7304
Accuracy score on training data: 1.0000
```

In [26]:
```
# Optimized model
test_accuracy, train_accuracy = helper.prediction_scores(best_clf, X_tr
ain, X_test, y_train, y_test)
helper.print_scores(test_accuracy, train_accuracy)
```

```
Accuracy score on testing data: 0.8272
Accuracy score on training data: 0.8333
```

Top 10 important features

In [27]:
```
# Extract the feature importances using .feature_importances_
importances = best_clf.feature_importances_
indices = np.argsort(importances)[::-1]
print('Top 10 Important Features')
display(X_train.columns.values[indices[:10]])
```

```
Top 10 Important Features

array(['room_type_Entire_home_apt', 'accommodates', 'availability_365',
       'reviews_per_month', 'neighbourhood_group_cleansed_Downtown',
       'cleaning_fee', 'bedrooms', 'property_type_Bed_and_Breakfast',
       'neighbourhood_group_cleansed_Capitol_Hill', 'host_since'],
      dtype=object)
```

In [ ]:

Modeling and Evaluation: Popularity

What property features determine its popularity?

- Using number of reviews per month as a proxy for popularity.
- Similar data preparation as for previous price modeling.

In [28]:
```
# Remove records with Null values in 'reviews_per_month'
revpm_df         = seattle_df[seattle_df['reviews_per_month'].isnull()
!= True]

# Split Data into features and labels
label_revpm_df   = np.where(revpm_df['reviews_per_month'] > revpm_df['r
eviews_per_month'].median(), 0, 1)
feature_revpm_df = revpm_df.drop(['reviews_per_month'], axis=1)

# Feature Scaling
scaled_revpm_df  = helper.process_features(feature_revpm_df)
scaled_revpm_df.head()
```

Out[28]:

	host_since	host_response_rate	host_acceptance_rate	host_is_superhost	host_has_profile_pic
0	0.611305	0.951807	1.0	0.0	1.0
1	0.394503	0.975904	1.0	1.0	1.0
2	0.210221	0.602410	1.0	0.0	1.0
4	0.568719	1.000000	1.0	0.0	1.0
5	0.699961	1.000000	1.0	0.0	1.0

AdaBoostClassifier with decision tree base estimator. The model was optimized using GridSearchCV.

In [29]:

```
# Initialize the classifier
clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), rando
m_state=123)

# Create the parameters list to tune, using a dictionary.
parameters = {"n_estimators": [10, 50, 100],
              "learning_rate": [0.005, .01, 0.05, 0.1],
              'base_estimator__min_samples_split' : [2, 4, 6, 8],
              'base_estimator__max_depth' : [2, 4, 6, 8]}
```

In [30]:

```
best_clf, X_train, X_test, y_train, y_test = helper.boost_classifier(cl
f, parameters, scaled_revpm_df, label_revpm_df)
```

```
C:\Users\tai_j\Anaconda3\lib\site-packages\sklearn\utils\validation.py:
71: FutureWarning: Pass scoring=make_scorer(fbeta_score, beta=0.5) as k
eyword args. From version 0.25 passing these as positional arguments wi
ll result in an error
  FutureWarning)
```

In [31]:

```
print(best_clf)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=2,
                                                         min_samples_sp
```

```
lit=8),
                              learning_rate=0.1, n_estimators=100, random_state=12
3)
```

In [32]:
```
# Unoptimized model
test_accuracy, train_accuracy = helper.prediction_scores(clf, X_train, X_test, y_train, y_test)
helper.print_scores(test_accuracy, train_accuracy)
```

```
Accuracy score on testing data: 0.7825
Accuracy score on training data: 1.0000
```

In [33]:
```
# Optimized model
test_accuracy, train_accuracy = helper.prediction_scores(best_clf, X_train, X_test, y_train, y_test)
helper.print_scores(test_accuracy, train_accuracy)
```

```
Accuracy score on testing data: 0.8607
Accuracy score on training data: 0.8687
```

In [34]:
```
# Extract the feature importances using .feature_importances_
importances = best_clf.feature_importances_
indices = np.argsort(importances)[::-1]
print('Top 10 Important Features')
display(X_train.columns.values[indices[:10]])
```

```
Top 10 Important Features

array(['number_of_reviews', 'host_since', 'availability_365',
       'cleaning_fee', 'price', 'total_amenities',
       'host_response_time_within_an_hour', 'bedrooms',
       'instant_bookable', 'host_is_superhost'], dtype=object)
```

In [ ]:
```

```

In [ ]:
```

```

When is the most popular month to rent in Seattle?

- Jul, Aug and Sep are the best period to maximise revenue. Before May are the best time for maintenance work Oct to Dec is a good time to take a break and enjoy the holidays.

In [35]:

```
# Listings Dataframe
base = pd.read_csv(csvs[1])
listings_df = base.copy()
listings_df.rename(columns={'id':'listing_id'}, inplace=True)

# Reviews Dataframe
base = pd.read_csv(csvs[2])
reviews_df = base.copy()
reviews_df = reviews_df[['id','listing_id','date']]

# Datetime Conversion
reviews_df['date'] = pd.to_datetime(reviews_df['date'])
reviews_df.head()
```

Out[35]:

	id	listing_id	date
0	38917982	7202016	2015-07-19
1	39087409	7202016	2015-07-20
2	39820030	7202016	2015-07-26
3	40813543	7202016	2015-08-02
4	41986501	7202016	2015-08-10

In [45]:

```
# Bookings Dataframe
bookings_df = pd.merge(reviews_df, listings_df, on='listing_id')
bookings_df['estimated_revenue'] = bookings_df['price'] * bookings_df[
'minimum_nights']
bookings_df.head(20)
```

Out[45]:

	id	listing_id	date	minimum_nights	price	estimated_revenue
0	38917982	7202016	2015-07-19	2	75.0	150.0
1	39087409	7202016	2015-07-20	2	75.0	150.0
2	39820030	7202016	2015-07-26	2	75.0	150.0
3	40813543	7202016	2015-08-02	2	75.0	150.0
4	41986501	7202016	2015-08-10	2	75.0	150.0
5	43979139	7202016	2015-08-23	2	75.0	150.0
6	45265631	7202016	2015-09-01	2	75.0	150.0
7	46749120	7202016	2015-09-13	2	75.0	150.0
8	47783346	7202016	2015-09-21	2	75.0	150.0
9	48388999	7202016	2015-09-26	2	75.0	150.0
10	49441269	7202016	2015-10-04	2	75.0	150.0
11	50490194	7202016	2015-10-12	2	75.0	150.0
12	53862449	7202016	2015-11-13	2	75.0	150.0
13	54562283	7202016	2015-11-21	2	75.0	150.0
14	55212826	7202016	2015-11-29	2	75.0	150.0
15	58268184	7202016	2016-01-02	2	75.0	150.0
16	20798623	3946674	2014-10-05	1	90.0	90.0
17	21224862	3946674	2014-10-13	1	90.0	90.0
18	22877803	3946674	2014-11-16	1	90.0	90.0
19	22938377	3946674	2014-11-17	1	90.0	90.0

In [49]:

```
# get revenue by listings
listings_df=listings_df[['listing_id','minimum_nights','price']]
listings_df['price'] = listings_df['price'].map(lambda x: helper.conver
t_to_price(x))
```

```
listings_df_revenue = bookings_df[['listing_id','estimated_revenue']].g
roupby(['listing_id']).sum()
listings_df = pd.merge(listings_df, listings_df_revenue, on='listing_i
d', how='left')
listings_df.at[listings_df['estimated_revenue'].isnull(), 'estimated_re
venue'] = 0
listings_df
```

```
C:\Users\tai_j\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imp
orts until
```

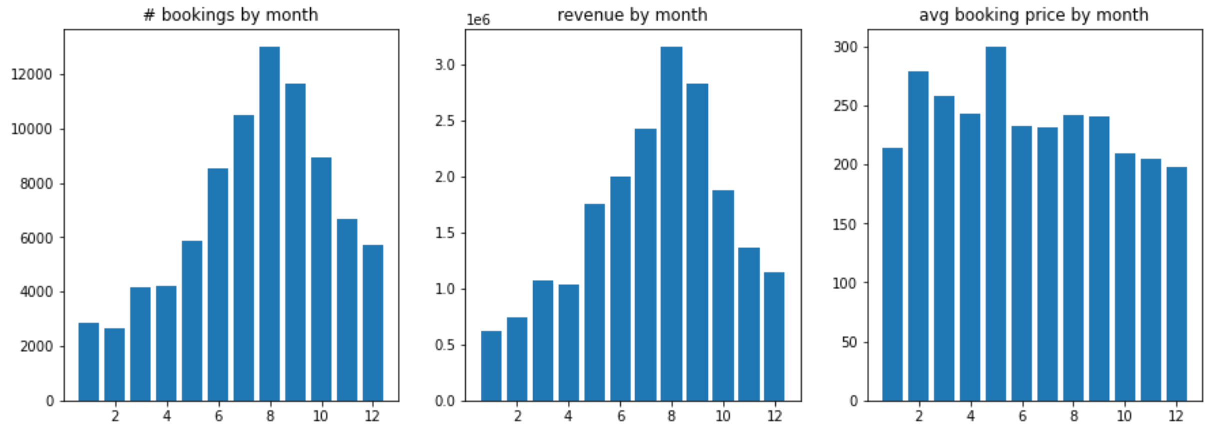
Out[49]:

	listing_id	minimum_nights	price	estimated_revenue
0	241032	1	85.0	17595.0
1	953595	2	150.0	12900.0
2	3308979	4	975.0	78000.0
3	7421966	1	100.0	0.0
4	278830	1	450.0	17100.0
...	...	...	...	...
3813	8101950	3	359.0	1077.0
3814	8902327	2	79.0	316.0
3815	10267360	1	93.0	0.0
3816	9604740	3	99.0	0.0
3817	10208623	1	87.0	0.0

3818 rows × 4 columns

In [50]:

```
plt.figure(figsize=(15, 5))

# # bookings by month
plotdata1 = reviews_df[['date']].groupby(reviews_df["date"].dt.month).c
ount()
plotdata1.rename(columns={'date':'# of bookings'}, inplace=True)

ax = plt.subplot(1, 3, 1)
ax.set_title("# bookings by month")
plt.bar(plotdata1.index, plotdata1['# of bookings'])

# revenue by month
plotdata2 = bookings_df[['date','estimated_revenue']].groupby(bookings_
df["date"].dt.month).sum()
plotdata2.rename(columns={'estimated_revenue':'revenue'}, inplace=True)

ax = plt.subplot(1, 3, 2)
ax.set_title("revenue by month")
plt.bar(plotdata2.index, plotdata2['revenue'])

# avg booking price by month
plotdata3 = pd.concat([plotdata1, plotdata2], axis=1)
plotdata3['avg booking price'] = plotdata3['revenue'] / plotdata3['# of
 bookings']
plotdata3.head()

ax = plt.subplot(1, 3, 3)
ax.set_title("avg booking price by month")
plt.bar(plotdata3.index, plotdata3['avg booking price'])

_ = plt.plot()
```



In [ ]:

```

```

Conclusion

This simple and straightforward data-driven investigation of the Seattle Airbnb market has provided us with insights may be helpful to anything keen to be involved in the property rental business.

A data-driven strategy would be important for any potential Airbnb host to get the important things right, and to fine-tune their existing listings to obtain higher premiums for their listing prices.

As per our investigation, there are factors that Airbnb hosts can tune: like (1) setting optimal prices/cleaning fees, (2) quick to reply, and (3) making the lodging available for more days of the year, and to some extent: (4) more amenities (perhaps adding on wifi, free breakfast, etc).

And there are factors that will be more relevant to people who are investing in a property or who have the resources to modify their lodgings, like: (1) making sure the property is situated in Downtown/Capitol Hill, (2) more rooms, (3) making whole apartments available instead of single-rooms, (4) enlarging the unit to accommodate more.

Understanding all these factors in depth will definitely provide an edge to anyone competing in the property rental market.