AHMET16 / airbnb-classification-model Public

Code Issues Pull requests Actions Projects Wiki Security

main

airbnb-classification-model / Untitled.ipynb

AHMET16 Add files via upload History

1 contributor

2240 lines (2240 sloc) | 76.2 KB

Using Predictive Modeling to Get More 5-Star Airbnb Reviews

- Prepared by: Ahmet KARAOGLAN, Data Scientist

Business Problem

Oceanside Property Management is a property management company located in San Diego California. Their main business is managing rental properties. However, they have recently noticed that a lot of Airbnb hosts have been reaching out to them for guidance. These hosts are mostly uninterested in having OPM manage their rentals, however they wany some help in increasing their success as Airbnb hosts.

There have been so many Airbnb hosts reaching out that OPM has decided that this can be a good side-business for them. So they plan to officially addmairbnb consulting as a service that they provide. In their initial research they found that the top questions that potential clients who wish to utlize this service are:

- "What can I do to get more 5 star ratings?"
- "Can you help me reach Superhost status? (or maintain Superhost status)

These questions are understandable because Airbnb puts a huge focus on getting 5 star overall ratings. They also highly publicize the benefits of getting (and maintaining) Superhost status.

Oceanside Property Management has decided that the main focus of their service will be helping clients get more 5 star reviews. Therefore they have tasked me with providing the following:

- A model that will predict whether a specific rental unit should get a 5 Star Overall score based on other available information.
- An industry analysis of AirBnb in San Diego. Specifically looking for any insight that they can give to their clients that will give them a leg up on people who don't use their consulting service.

They also want me to answer the following questions:

- Is there a significant advantage to being a Superhost? (is it worth all the effort to get this status and maintain it?)
- How do we determine whether a Host "should" be getting 5 Star reviews?
- What factors are most important in determining a 5 Star Overall Rating? (what aspects should they most focus on)

And finally, they want to know where their consulting service can make the most impact, so they know which features to market and/or which hosts to market to.

Understanding AirBnb

Who uses AirBnb?

information from: https://listwithclever.com/research/airbnb-vs-hotels-study/#sources, accessed 6/21/22

- Initially, the idea of staying in a random person's home was viewed as absurd and dangerous, but public perception of peer-to-peer (P2P) vacation rentals has shifted significantly in recent years.
- A 2016 Goldman Sachs study found that,"If people have stayed in peer-to-peer lodging in the last five years, the likelihood that they prefer traditional hotels is halved (79 percent vs. 40 percent)."
- Airbnb is becoming the preferred choice of vacationers — 60% of travelers who use both Airbnb and hotels prefer Airbnb over comparable hotels when going on vacation
- 68% of business travelers prefer staying in hotels when traveling for work, and they're more likely to have a negative experience at an Airbnb

information from: https://www.torontomu.ca/news-events/news/2016/10/why-tourists-choose-airbnb-over-hotels/ accessed 6/21/22

David Guttentag, professor at the Ted Rogers School of Hospitality and Tourism Management, identifies five types of Airbnb guests based on his 2016 study:

- Money savers: Choose Airbnb because of affordability
- Home seekers: Interested in household amenities and larger spaces
- Collaborative consumers: Motivated by the share economy philosophy and the ability to have an authentic experience
- Pragmatic novelty seekers: While not regular Airbnb users, these travelers are drawn to the novelty of Airbnb
- Interactive novelty seekers: Want to interact with their host or other locals

Importance of 5 Star Reviews and Rating

AirBnb focuses on exceeding customer expectations, which is why they strictly require that hosts maintain a near perfect rating in order to remain on the service.

Importance of Superhost

- information from https://www.airbnb.com/d/superhost. Accessed 6/16/22

Advantages:

- Superhost badge to stand out among other hosts.
- Customers can filter search results to show only superhosts.

Requirements:

- Minimum 4.8 overall rating.
- 10 stays over the last year.
- < 1% Cancellation Rate.
- At least 90% Response Rate.
- Reassessed every 3 months.

Problems with Airbnb Data and/or Ratings System

The review data is incredibly skewed because Airbnb requires such a high rating. Even though there is a 5 point scale, Anything lower than a 4.8 is seen as "bad".

- So while this is technically a 5pt scale (as a reviewer can give 1 - 5 stars, with no partial stars allowed), getting a 4.0 average could result in being de-listed from the service!

In order to stay at a 4.8 overall rating:

- a host will need to have four 5-star reviews to offset a single 4-star review.
- a host will need to have ten 5-star reviews to offset a single 3-star review.

The major problem with this review system is that airbnb guests often assume that airbnb's review scale functions similarly to a hotel review scale, which also uses 5 stars, with 3 considered average, 4 above average, and 5 star being the best possible experience.

from https://medium.com/@campbellandia/how-to-avoid-the-dreaded-4-star-review-a-guide-for-airbnb-hosts-cdf482d083fe

- The problem stems from the fundamental difference in what most people think a 5-star rating system is, and what AirBnB's system actually is. The vast majority of people think that a 4-star review is perfectly appropriate; Their stay was good, they enjoyed themselves, but your place wasn't the Vanderbilt Suite at the Plaza. What they don't understand is that if a listing gets too many 4-star reviews the AirBnB platform begins to send warnings to hosts that their listing will be removed.

My Process

The Problem

The big concern that Airbnb Hosts haveis how to ensure 5-Star Overall reviews. While the other review categories certainly factor into a guest's review of a property, the Overall rating itself doesn't factor anything else in. It is just purely what the guest put in for Overall Rating. "How to get more 5 Star Reviews" is the problem that I'm seeking to solve.

What I'm Looking For:

I have been tasked with creating a model that will predict whether or not a unit will generate 5-star reviews. The best way to find this is create a classification for whether a unit has a perfect 5.0 overall rating, and then train a model to predict whether a unit will get that classification or not.

Target: Elite Units

- I am calling my target classification Elite Units.
- An Elite Unit is any Rental Unit that has a 4.9 - 5.0 Overall Rating.
- I am including 4.9 so there is a tiny bit of wiggle room, especially as a 4.9 overall rental unit would be seen as "successful" in Airbnb's eyes.
- These are the high-performing units that OPM clients want to emulate. Creating this classifier makes it easier to determine if they are performing on target, as well as letting us analyze any common trends, etc.

Measuring Success: Booking Rate

- At first glance, you might assume that Price would be the best performance metric. However, price is relative. A low priced 5 bedroom house will often cost more than a high priced 1 bedroom house.
- However, all Airbnb Hosts desire bookings. The more that their unit is booked, the more success that they have.
- Therefore, Booking Rate will be the feature that I use to measure how successful a unit is.
- Any features that cause a positive Trend in Booking Rate will be seen as successful.

Goals for my Model:

What I will be looking for in my models:

- High Precision Score: I want to make sure that I am identifying as many airbnb units that meet my target criteria as possible. I will keep this in balance by checking F1 Score.

- Good F1 Score: While I am ultimately not concerned with Recall , a good F1 score means that the model is performing well on both Recall and Precision. Since Recall and Precision are inverses of each other, a good F1 score ensures that the model isn't skewed too far toward one or the other. (ie, a model that predicts EVERY customer is within my target would have perfect Recall, but would be useless).

- High Cross Validation Score: This ensures that the model isn't overly trained on the test data and that it does a good job of predicted unseen and unknown data. (ie, the test set).

- Area Under the Curve (AUC): The ROC AUC Score measures the Area under the ROC curve, which means that it classifies the true positive rate against the false positive rate. The higher the score, the better performing the model is.

That said, here is the scale that I will use to evaluate my models:

- .69 or less: Model performs only slightly better than guessing and is worthless for my analysis.
- .70 - .79: Model still isn't performing very well, but is at minimum acceptable levels.
- .80 - .89: Model is performing fairly well. My goal is to be in this range or better.
- .90 - .99: Model is performing very well. I would be very happy to have a final model in this range.

In [ ]:

```

```

Preprocessing

Loading Data

In [1]:

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.ticker as mtick
from matplotlib.pylab import rcParams
import matplotlib.ticker as mtick
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, mean_squared_error, mean_
from sklearn.metrics import precision_score, recall_score, accuracy_s
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
from sklearn.metrics import plot_confusion_matrix
from xgboost import XGBClassifier
import numpy as np

pd.set_option('display.max_rows', 1000)
plt.style.use('fivethirtyeight')
```

Full_df: Dataframe Containing All Available Columns

In [2]:

```
#Data obtained from http://insideairbnb.com/san-diego
```

```
full_df = pd.read_csv('listings.csv')
```

In [3]:

```
full_df.info()
```

```
RangeIndex: 14188 entries, 0 to 14187
Data columns (total 75 columns):
 #   Column                                         Non-Null Count  Dt
ype
---  ------                                         --------------  --
---
 0   id                                             14188 non-null  in
t64
 1   listing_url                                    14188 non-null  ob
ject
 2   scrape_id                                      14188 non-null  in
t64
 3   last_scraped                                   14188 non-null  ob
ject
 4   source                                         14188 non-null  ob
ject
 5   name                                           14188 non-null  ob
ject
 6   description                                    14060 non-null  ob
ject
 7   neighborhood_overview                          9306 non-null   ob
ject
 8   picture_url                                    14188 non-null  ob
ject
 9   host_id                                        14188 non-null  in
t64
 10  host_url                                       14188 non-null  ob
ject
 11  host_name                                      14174 non-null  ob
ject
 12  host_since                                     14174 non-null  ob
ject
 13  host_location                                  11669 non-null  ob
ject
 14  host_about                                     9132 non-null   ob
ject
 15  host_response_time                             13042 non-null  ob
ject
 16  host_response_rate                             13042 non-null  ob
ject
 17  host_acceptance_rate                           13465 non-null  ob
ject
 18  host_is_superhost                              14175 non-null  ob
ject
 19  host_thumbnail_url                             14174 non-null  ob
ject
 20  host_picture_url                               14174 non-null  ob
ject
 21  host_neighbourhood                             12131 non-null  ob
ject
 22  host_listings_count                            14174 non-null  fl
oat64
 23  host_total_listings_count                      14174 non-null  fl
oat64
 24  host_verifications                             14188 non-null  ob
```

```
ject
 25  host_has_profile_pic                          14174 non-null  ob
ject
 26  host_identity_verified                        14174 non-null  ob
ject
 27  neighbourhood                                 9306 non-null   ob
ject
 28  neighbourhood_cleansed                        14188 non-null  ob
ject
 29  neighbourhood_group_cleansed                  0 non-null      fl
oat64
 30  latitude                                      14188 non-null  fl
oat64
 31  longitude                                     14188 non-null  fl
oat64
 32  property_type                                 14188 non-null  ob
ject
 33  room_type                                     14188 non-null  ob
ject
 34  accommodates                                  14188 non-null  in
t64
 35  bathrooms                                     0 non-null      fl
oat64
 36  bathrooms_text                                14184 non-null  ob
ject
 37  bedrooms                                      12915 non-null  fl
oat64
 38  beds                                          14027 non-null  fl
oat64
 39  amenities                                     14188 non-null  ob
ject
 40  price                                         14188 non-null  ob
ject
 41  minimum_nights                                14188 non-null  in
t64
 42  maximum_nights                                14188 non-null  in
t64
 43  minimum_minimum_nights                        14186 non-null  fl
oat64
 44  maximum_minimum_nights                        14186 non-null  fl
oat64
 45  minimum_maximum_nights                        14186 non-null  fl
oat64
 46  maximum_maximum_nights                        14186 non-null  fl
oat64
 47  minimum_nights_avg_ntm                        14186 non-null  fl
oat64
 48  maximum_nights_avg_ntm                        14186 non-null  fl
oat64
 49  calendar_updated                              0 non-null      fl
oat64
 50  has_availability                              14188 non-null  ob
ject
 51  availability_30                               14188 non-null  in
t64
 52  availability_60                               14188 non-null  in
t64
 53  availability_90                               14188 non-null  in
t64
 54  availability_365                              14188 non-null  in
t64
```

```
 55  calendar_last_scraped                          14188 non-null  ob
ject
 56  number_of_reviews                              14188 non-null  in
t64
 57  number_of_reviews_ltm                          14188 non-null  in
t64
 58  number_of_reviews_l30d                         14188 non-null  in
t64
 59  first_review                                   12523 non-null  ob
ject
 60  last_review                                    12523 non-null  ob
ject
 61  review_scores_rating                           12523 non-null  fl
oat64
 62  review_scores_accuracy                         12502 non-null  fl
oat64
 63  review_scores_cleanliness                      12502 non-null  fl
oat64
 64  review_scores_checkin                          12500 non-null  fl
oat64
 65  review_scores_communication                    12502 non-null  fl
oat64
 66  review_scores_location                         12500 non-null  fl
oat64
 67  review_scores_value                            12500 non-null  fl
oat64
 68  license                                        152 non-null    ob
ject
 69  instant_bookable                               14188 non-null  ob
ject
 70  calculated_host_listings_count                 14188 non-null  in
t64
 71  calculated_host_listings_count_entire_homes    14188 non-null  in
t64
 72  calculated_host_listings_count_private_rooms   14188 non-null  in
t64
 73  calculated_host_listings_count_shared_rooms    14188 non-null  in
t64
 74  reviews_per_month                              12523 non-null  fl
oat64
dtypes: float64(23), int64(17), object(35)
memory usage: 8.1+ MB
```

In [4]:

```
base_df = full_df[['price', 'review_scores_rating', 'review_scores_ac
                      'review_scores_cleanliness', 'review_scores_che
                      'review_scores_location', 'review_scores_value'
                      'instant_bookable', 'property_type', 'room_type
                      'availability_30','availability_90','host_id', 
                      'host_response_time', 'host_response_rate','hos
```

In [5]:

```
df = base_df
```

Exploratory Data Analysis

- Investingating the various features of my dataset to determine which features to use in my model and anaylsis, and to what extent.

Fixing Price

- Price is currently a string. I need to strip out the extra characters and convert the datatype to Float so that I can better utilize the data

In [6]:

```
df['price'].head(5)
```

Out[6]:

```
0    $225.00
1    $113.00
2    $258.00
3    $336.00
4    $333.00
Name: price, dtype: object
```

In [7]:

```
#using lambda function to strip $ and , out of each price record. rep
df['price'] = df['price'].map(lambda x: x.replace('$',''))
df['price'] = df['price'].map(lambda x: x.replace(',',''))
df['price'] = df['price'].astype(float) #changing cleaned column to f
df['price'].head(2)
```

Out[7]:

```
0    225.0
1    113.0
Name: price, dtype: float64
```

New Feature: Host Listings_5-

- Creating a new feature that classifies whether a "many" listings or not

In [8]:

```
#getting key metrics for this feature.
df['calculated_host_listings_count'].describe()
```

Out[8]:

```
count    14188.000000
mean        17.266422
std         35.920780
min          1.000000
25%          1.000000
50%          3.000000
75%         13.000000
max        213.000000
Name: calculated_host_listings_count, dtype: float64
```

Analysis:

- The majority of hosts in this dataset have between 1-14 listings. (25%-75%).
- The median is 3.
- One has 213 listings

In [9]:

```
#checking to see how many records have omly 1 or 2 listings vs the re
low_listings = df['calculated_host_listings_count'] <=2

low_listings.value_counts()
```

Out[9]:
```
False    7608
True     6580
Name: calculated_host_listings_count, dtype: int64
```

Since so many hosts have just 1 or 2 rental units, everythink is skewed toward the lower end. However, I am setting this classifier at 5 and under as people with multiple listings will be more likely to use OPC's service.

In [10]:
```
#creating classifier and checking to see how the data is split.
df['capacity_5+'] = df['accommodates'] >=5
df['capacity_5+'].value_counts()
```

Out[10]:
```
False    8221
True     5967
Name: capacity_5+, dtype: int64
```

This seems to be a good classifier as the split ends up being close to 50%.

New Feature Bedrooms_2+

In [11]:
```
df['bedrooms'].describe()
```

Out[11]:
```
count    12915.000000
mean         1.994580
std          1.235842
min          1.000000
25%          1.000000
50%          2.000000
75%          3.000000
max         23.000000
Name: bedrooms, dtype: float64
```

Analysis:

- Mean and Median are both roughly 2 Bedrooms, so I will set the classifer at 2 and above.

In [12]:
```
df['bedrooms_2+'] = df['bedrooms'] >=2
df['bedrooms_2+'].value_counts()
```

Out[12]:
```
False    7121
True     7067
Name: bedrooms_2+, dtype: int64
```

New Feature: Booking Rates

- seeing the rumber of available days is good, but in some cases it may be more helpful to see this at a percentage.

In [13]:
```
#Changing availability to a percentage named availability rate.
df['availability_30_rate'] = df['availability_30'].apply(lambda x: x/
```

```
df['availability_30_rate'] = df['availability_30'].apply(lambda x: x/
df['availability_90_rate'] = df['availability_90'].apply(lambda x: x/
```

In [14]:
```
#Changing the availability rate to the percentage of the time period
df['booked_rate_30'] = df['availability_30_rate'].apply(lambda x: 1 -
df['booked_rate_90'] = df['availability_90_rate'].apply(lambda x: 1 -
```

In [15]:
```
avaibility = df[['availability_30_rate', 'booked_rate_30','availabili
avaibility.head()
```

Out[15]:

	availability_30_rate	booked_rate_30	availability_90_rate	booked_rate_90
0	0.000000	1.000000	0.066667	0.933333
1	0.666667	0.333333	0.600000	0.400000
2	0.000000	1.000000	0.000000	1.000000
3	0.533333	0.466667	0.488889	0.511111
4	0.200000	0.800000	0.466667	0.533333

New Feature: Bookings Above Average

- I have determined that price is not a great metric for measuring rentals because the prices are relative, and no two units are exactly the same.
- However, the main thing that hosts want is to maximimze their bookings. So I want to capture and analyze how much availabilty they have so I that I have a metric to compare across the board.

In [16]:
```
df["bookings_above_avg"] = df['booked_rate_90'] >= .512
df['bookings_above_avg'].value_counts()
```

Out[16]:
```
False    8670
True     5518
Name: bookings_above_avg, dtype: int64
```

New Feature: Host Respnse Rate 100

- Feature that determines weather a host has a perfect response rate.
- SuperHost status requires a minimum of %90 response rate.

In [17]:
```
#creating a classifier that captures whether a host has a perfect res
df['host_response_rate'] = df['host_response_rate'].str.replace('%',
df['host_response_rate']  = df['host_response_rate'].astype('float')
df['host_response_100'] = df['host_response_rate'] == 100.0
df['host_response_100'].value_counts()
```

Out[17]:
```
True     9916
False    4272
Name: host_response_100, dtype: int64
```

Fixing Host is Superhost & Instant Bookable

- Features are currently strings instead of bools.

In [18]:
```
#setting up up a bool based on the old string data.
df['superhost'] = df['host_is_superhost'] == 't'
df['instant_bookable'] = df['instant_bookable'] == 't'
```

In [19]:
```
#making sure that I captured both the True and False classification.
df['superhost'].value_counts()
```

Out[19]:
```
False    8782
True     5406
Name: superhost, dtype: int64
```

In [20]:
```
#making sure that I captured both the True and False Classifications.
df['instant_bookable'].value_counts()
```

Out[20]:
```
False    7306
True     6882
Name: instant_bookable, dtype: int64
```

Target Feature: Elite Units

- This is my target feature. It classifies whether a unit is in our target 4.9 - 5.0 overall rating range or not.

Dealing with Nulls

In [21]:
```
#seeing how many records dont have a review score overall rating
df['review_scores_rating'].isna().sum()
```

Out[21]: 1665

There are 1655 Null records that need to be dealt with. If I drop them, I will lose 15% my data.

In [22]:
```
nulls = df[df['review_scores_rating'].isna()]
```

In [23]:
```
len(nulls)
```

Out[23]: 1665

In [24]:
```
nulls.head(5)
```

Out[24]:

	price	review_scores_rating	review_scores_accuracy	review_scores_cleanlines
104	307.0	NaN	NaN	NaN

118	900.0	NaN	NaN	NaN
180	150.0	NaN	NaN	NaN
183	235.0	NaN	NaN	NaN
274	404.0	NaN	NaN	NaN

5 rows × 32 columns

In [25]:
```
df.dropna(subset=['review_scores_rating'], how='all', inplace = True)
```

In [26]:
```
len(df)
```

Out[26]:
```
12523
```

12523 Records are left after dropping null values

Creating Elite Unite Classifier

- I have decided to classify "5 star" units s ones that have a 4.9 or higher oerall rating.
- 4.9 is still an incredibly high score, and is obove thresholds for success (4.8 rating, etc), so it is well worth capturing units with a 4.9 rating as high performers as well

In [27]:
```
#creating classifier and then checking to see how many units ae in ea
df['elite'] = df['review_scores_rating'] >=4.9
df['elite'].value_counts()
```

Out[27]:
```
False    7385
True     5138
Name: elite, dtype: int64
```

41% of my dataset are Elite Units

Out of 12523 Airbnb rental units, 41%(5138) are elite units, while 59(7385) are not

Room Type

In [28]:
```
#creating a new feature which turn room type into a binary classifier
df['entire_home'] = df['room_type'] == 'Entire home/apt'
df['entire_home'].value_counts()
```

Out[28]:
```
True     10503
False     2020
Name: entire_home, dtype: int64
```

In [29]:
```
#dropping room_typesince I now classifier inits place
df.drop(['room_type'], axis=1,inplace=True)
```

Host Response Time

In [30]:
```
#checking to see how many records I have of each response speed.
df['host_response_time'].value_counts()
```

Out[30]:
```
within an hour        9676
within a few hours    1237
within a day           637
a few days or more     149
Name: host_response_time, dtype: int64
```

In [31]:
```
#The majority of responses were "with an hour".
# I will change this into a binary classifier
df['response_within_hour'] = df['host_response_time'] == 'within an h
df['response_within_hour'].value_counts()
```

Out[31]:
```
True     9676
False    2847
Name: response_within_hour, dtype: int64
```

In [32]:
```
df.drop(['host_response_time'], axis=1, inplace=True)
```

Creating Review Metric Classifier Columns

- These columns will capture the number of 5 star reviews left for each review metric.
- Just like with my target classifier (5-Star), I am counting 4.9s in with the 5.0s.

In [33]:
```
#Creating a classifier for each review metric with the same critieria

df['accuracy_5'] = df['review_scores_accuracy'] >= 4.9
df['cleanliness_5'] =  df['review_scores_cleanliness'] >= 4.9
df['checkin_5'] = df['review_scores_checkin'] >= 4.9
df['location_5'] = df['review_scores_location'] >=4.9
df['value_5'] = df['review_scores_value'] >= 4.9
df['communication_5'] = df['review_scores_communication'] >= 4.9
```

In [34]:
```
#Printing a list with the number of units that are Elite in each cate
print("Number of Elite Accuracy Units:", len(df[df['accuracy_5']== Tr
print("Number of Elite Cleanliness Units:", len(df[df['cleanliness_5'
print("Number of Elite Checkin Units:", len(df[df['checkin_5']== True
print("Number of Elite Location Units:", len(df[df['location_5']== Tr
print("Number of Elite Value Units:", len(df[df['value_5']== True]))
print("Number of Elite Communication Units:", len(df[df['communicatic
```

```
Number of Elite Accuracy Units: 6324
```

```
Number of Elite Cleanliness Units: 5425
Number of Elite Checkin Units: 8486
Number of Elite Location Units: 7117
Number of Elite Value Units: 3345
Number of Elite Communication Units: 8058
```

There are significantly less units that have Elite Value. I am goin to do a value count of that classifier to take a closer look.

In [35]:
```
df['value_5'].value_counts()
```

Out[35]:
```
False    9178
True     3345
Name: value_5, dtype: int64
```

New Feature: Price Above Median

In [36]:
```
#checking the mean, standard deviation, median and quatiles of price
df['price'].describe()
```

Out[36]:
```
count     12523.000000
mean        334.963347
std        1192.884956
min          10.000000
25%         115.000000
50%         181.000000
75%         318.000000
max      100000.000000
Name: price, dtype: float64
```

It is difficult to analyze price because it is relative. That said, I will create a classifier to determine whether a unit is above or belov the average(mean) price. (I rounded the mean of 279 to 280)

In [37]:
```
df['price_280+'] =df['price'] >=280
```

In [38]:
```
df['price_280+'].value_counts()
```

Out[38]:
```
False    8756
True     3767
Name: price_280+, dtype: int64
```

Creating Analysis_df

In [39]:
```
#copying my dataframe as 'analysis_df' so I can easily pull back up m

analysis_df = df.copy()
```

Preparing for Modeling

In [40]:

```
#checking to see what my dataframe currently looks like

analysis_df.info()
```

```
Int64Index: 12523 entries, 0 to 14187
Data columns (total 40 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   price                           12523 non-null  float64
 1   review_scores_rating            12523 non-null  float64
 2   review_scores_accuracy          12502 non-null  float64
 3   review_scores_cleanliness       12502 non-null  float64
 4   review_scores_checkin           12500 non-null  float64
 5   review_scores_communication     12502 non-null  float64
 6   review_scores_location          12500 non-null  float64
 7   review_scores_value             12500 non-null  float64
 8   accommodates                    12523 non-null  int64  
 9   bedrooms                        11366 non-null  float64
 10  beds                            12392 non-null  float64
 11  instant_bookable                12523 non-null  bool   
 12  property_type                   12523 non-null  object 
 13  amenities                       12523 non-null  object 
 14  availability_365                12523 non-null  int64  
 15  availability_30                 12523 non-null  int64  
 16  availability_90                 12523 non-null  int64  
 17  host_id                         12523 non-null  int64  
 18  calculated_host_listings_count  12523 non-null  int64  
 19  host_response_rate              11699 non-null  float64
 20  host_is_superhost               12513 non-null  object 
 21  capacity_5+                     12523 non-null  bool   
 22  bedrooms_2+                     12523 non-null  bool   
 23  availability_30_rate            12523 non-null  float64
 24  availability_90_rate            12523 non-null  float64
 25  booked_rate_30                  12523 non-null  float64
 26  booked_rate_90                  12523 non-null  float64
 27  bookings_above_avg              12523 non-null  bool   
 28  host_response_100               12523 non-null  bool   
 29  superhost                       12523 non-null  bool   
 30  elite                           12523 non-null  bool   
 31  entire_home                     12523 non-null  bool   
 32  response_within_hour            12523 non-null  bool   
 33  accuracy_5                      12523 non-null  bool   
 34  cleanliness_5                   12523 non-null  bool   
 35  checkin_5                       12523 non-null  bool   
 36  location_5                      12523 non-null  bool   
 37  value_5                         12523 non-null  bool   
 38  communication_5                 12523 non-null  bool   
 39  price_280+                      12523 non-null  bool   
dtypes: bool(16), float64(15), int64(6), object(3)
memory usage: 2.6+ MB
```

One Hot Encoding

In [41]:

```
#Creating a variable which captures the features that need to be one

need_to_encode = df[['price_280+', 'elite', 'accuracy_5', 'cleanlines
                    'communication_5', 'entire_home', 'bedrooms_2+',
```

```
                          'bookings_above_avg', 'instant_bookable', 'capac
#calling encoder and fitting it to the features that need to be encod
ohe = OneHotEncoder()
ohe.fit(need_to_encode)

#transforming the encoder output so that it can be modeled.
ohe_1 = ohe.transform(need_to_encode).toarray()

#adding labels
ohe_df = pd.DataFrame(ohe_1, columns=ohe.get_feature_names(need_to_en
ohe_df.head(5)
```

Out[41]:

	price_280+_False	price_280+_True	elite_False	elite_True	accuracy_5_False	ac
0	1.0	0.0	1.0	0.0	1.0	
1	1.0	0.0	1.0	0.0	1.0	
2	1.0	0.0	1.0	0.0	1.0	
3	0.0	1.0	1.0	0.0	1.0	
4	0.0	1.0	0.0	1.0	0.0	

5 rows × 87 columns

In [42]:

```
#creating 'cleaned_df'  as a copy of 'ohe_df' sothat I have a saved v
cleaned_df = ohe_df.copy()
```

Dropping One Value for Categoricals

In [43]:

```
#dropping one value from all categorical values to prevent redundancy
cleaned_df.drop(['elite_False', 'accuracy_5_False', 'cleanliness_5_Fa
                 'value_5_False', 'communication_5_False', 'bedrooms_
                 'bookings_above_avg_False', 'instant_bookable_False'
                 'calculated_host_listings_count_97', 'entire_home_Fa
                 'superhost_False', 'host_response_100_False', 'respo
                ], axis=1, inplace=True)
```

Dealing With Class imbalance

- Solution
  - Always use class weight parameter in Decision TreeClassifier
  - Always stratify Train Test Split
  - Add SMOTE to Training Sets.

In [44]:

```
#Checking to make sure that my target was properly encoded.
cleaned_df['elite_True'].value_counts()
```

Out[44]:

```
0.0    7385
1.0    5138
```

```
Name: elite_True, dtype: int64
```

Train Test Split

- Creating seperate Traning and Test Groups for modeling.

In [47]:

```
#creating 'balanced_df', which will end up being my df with balanced
balanced_df = cleaned_df.copy()

#islolating my target(y), and all other data(X)
X = balanced_df.drop(['elite_True'], axis=1)
y = balanced_df['elite_True']

#Splitting X and y into training and test sets, with 25% of the data
#Stratifying the split to minimize class imbalance.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.

#Using SMOTE to further minimize any class imbalance.
smote = SMOTE(random_state=23)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_
```

Choosing Evaluation Metrics

- My goal is to predict wheather a person will get a 4.9-5.0 Airbnb Overall rating.
- Which is worse?
    - Model predicts that a unit is an Elie Unit, but they actually are not ? (more false negative)

Decision

- I want to false Positive to be as low as possible
- if my model says that a property is an Elite Unit, I want it to be true.
- if it misses some of the Elite units in the process, that is fine.