AirBnb Market Analysis for Oceanside Property Management

- Prepared by: Jonathan Holt

1 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:

- 1) **A model** that will predict whether a specific rental unit should get a 5 Star Overall score based on other available information.
- 2) **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:

- 1) Is there a significant advantage to being a Superhost? (is it worth all the effort to get this status and maintain it?)
- 2)How do we determine whether a Host "should" be getting 5 Star reviews?
- 3) 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.

2 Understanding AirBnb

2.1 Who uses AirBnb?

information from: https://listwithclever.com/research/airbnb-vs-hotels-study/#sources (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/ (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

Information from: https://listwithclever.com/research/airbnb-vs-hotels-study/ (https://listwithclever.com/research/airbnb-vs-hotels-study/) accessed 7/7/22

- 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

2.2 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.

2.3 Importance of Superhost

- information from https://www.airbnb.com/d/superhost (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.

2.4 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 (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.

3 My Process

3.1 The Problem

The big concern that the Airbnb Hosts who are reaching out to OPM is **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.**

3.2 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.**

3.3 Goals for my Model:

What I will be looking for in my models:

- 1. **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.
- 2. **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).
- 3. **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).
- 4. **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.

4 Preprocessing

4.1 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_sq
from sklearn.metrics import precision_score, recall_score, accuracy_sco
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')
```

executed in 1.13s, finished 00:24:21 2022-07-14

4.2 Full_df: Dataframe Containing All Available Columns

In [2]:
```
full_df = pd.read_csv('listings.csv.gz')
```

executed in 446ms, finished 00:24:21 2022-07-14

In [3]:
```
1 full_df.info()
```

executed in 27ms, finished 00:24:21 2022-07-14

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10935 entries, 0 to 10934
Data columns (total 74 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   id                                            10935 non-null  int64
 1   listing_url                                   10935 non-null  object
 2   scrape_id                                     10935 non-null  int64
 3   last_scraped                                  10935 non-null  object
 4   name                                          10935 non-null  object
 5   description                                   10809 non-null  object
 6   neighborhood_overview                         7440 non-null   object
 7   picture_url                                   10935 non-null  object
 8   host_id                                       10935 non-null  int64
 9   host_url                                      10935 non-null  object
 10  host_name                                     10931 non-null  object
 11  host_since                                    10931 non-null  object
 12  host_location                                 10924 non-null  object
 13  host_about                                    7436 non-null   object
 14  host_response_time                            10113 non-null  object
 15  host_response_rate                            10113 non-null  object
 16  host_acceptance_rate                          10543 non-null  object
 17  host_is_superhost                             10931 non-null  object
 18  host_thumbnail_url                            10931 non-null  object
 19  host_picture_url                              10931 non-null  object
 20  host_neighbourhood                            9900 non-null   object
 21  host_listings_count                           10931 non-null  float6
4
 22  host_total_listings_count                     10931 non-null  float6
4
 23  host_verifications                            10935 non-null  object
 24  host_has_profile_pic                          10931 non-null  object
 25  host_identity_verified                        10931 non-null  object
 26  neighbourhood                                 7440 non-null   object
 27  neighbourhood_cleansed                        10935 non-null  object
 28  neighbourhood_group_cleansed                  0 non-null      float6
4
 29  latitude                                      10935 non-null  float6
4
 30  longitude                                     10935 non-null  float6
4
 31  property_type                                 10935 non-null  object
 32  room_type                                     10935 non-null  object
 33  accommodates                                  10935 non-null  int64
 34  bathrooms                                     0 non-null      float6
4
 35  bathrooms_text                                10930 non-null  object
 36  bedrooms                                      9905 non-null   float6
4
 37  beds                                          10813 non-null  float6
4
 38  amenities                                     10935 non-null  object
 39  price                                         10935 non-null  object
```

```
 40  minimum_nights                                 10935 non-null  int64
 41  maximum_nights                                 10935 non-null  int64
 42  minimum_minimum_nights                         10934 non-null  float6
4
 43  maximum_minimum_nights                         10934 non-null  float6
4
 44  minimum_maximum_nights                         10934 non-null  float6
4
 45  maximum_maximum_nights                         10934 non-null  float6
4
 46  minimum_nights_avg_ntm                         10934 non-null  float6
4
 47  maximum_nights_avg_ntm                         10934 non-null  float6
4
 48  calendar_updated                               0 non-null      float6
4
 49  has_availability                               10935 non-null  object
 50  availability_30                                10935 non-null  int64
 51  availability_60                                10935 non-null  int64
 52  availability_90                                10935 non-null  int64
 53  availability_365                               10935 non-null  int64
 54  calendar_last_scraped                          10935 non-null  object
 55  number_of_reviews                              10935 non-null  int64
 56  number_of_reviews_ltm                          10935 non-null  int64
 57  number_of_reviews_l30d                         10935 non-null  int64
 58  first_review                                   9408 non-null   object
 59  last_review                                    9408 non-null   object
 60  review_scores_rating                           9408 non-null   float6
4
 61  review_scores_accuracy                         9385 non-null   float6
4
 62  review_scores_cleanliness                      9385 non-null   float6
4
 63  review_scores_checkin                          9383 non-null   float6
4
 64  review_scores_communication                    9385 non-null   float6
4
 65  review_scores_location                         9383 non-null   float6
4
 66  review_scores_value                            9383 non-null   float6
4
 67  license                                        0 non-null      float6
4
 68  instant_bookable                               10935 non-null  object
 69  calculated_host_listings_count                 10935 non-null  int64
 70  calculated_host_listings_count_entire_homes    10935 non-null  int64
 71  calculated_host_listings_count_private_rooms   10935 non-null  int64
 72  calculated_host_listings_count_shared_rooms    10935 non-null  int64
 73  reviews_per_month                              9408 non-null   float6
4
dtypes: float64(24), int64(17), object(33)
memory usage: 6.2+ MB
```

4.3 Base DF

- I am slicing out the columns that will be useful for analyzing and modeling the data, to make the dataframe more manageable.

In [4]:
```
base_df = full_df[['price', 'review_scores_rating', 'review_scores_accu
                   'review_scores_cleanliness', 'review_scores_check
                   'review_scores_location', 'review_scores_value','
                   'instant_bookable', 'property_type', 'room_type',
                   'availability_30','availability_90','host_id', 'c
                   'host_response_time', 'host_response_rate','host_
```

executed in 4ms, finished 00:24:21 2022-07-14

In [5]:
```
df = base_df
```

executed in 2ms, finished 00:24:21 2022-07-14

5 Exploratory Data Analysis

5.1 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(2)
```

executed in 4ms, finished 00:24:21 2022-07-14

Out[6]:
```
0     $60.00
1    $282.00
Name: price, dtype: object
```

In [7]:
```
df['price'] = df['price'].map(lambda x: x.replace('$',' '))
df['price'] = df['price'].map(lambda x: x.replace(',',''))
df['price'] = df['price'].astype(float)
df['price'].head(2)
```

executed in 10ms, finished 00:24:21 2022-07-14

Out[7]:
```
0     60.0
1    282.0
Name: price, dtype: float64
```

5.2 New Feature: Host Listings_5-

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

In [8]:
```
df['calculated_host_listings_count'].describe()
```

executed in 6ms, finished 00:24:21 2022-07-14

Out[8]:
```
count    10935.000000
mean        18.456607
std         39.761546
min          1.000000
25%          1.000000
50%          3.000000
75%         14.000000
max        219.000000
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 219 listings.

In [9]:
```
low_listings = df['calculated_host_listings_count'] <= 2
low_listings.value_counts()
```

executed in 5ms, finished 00:24:21 2022-07-14

Out[9]:
```
False    5899
True     5036
Name: calculated_host_listings_count, dtype: int64
```

Since so many hosts have just 1 or 2 rental units, everything 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 OPM's service.

In [10]:
```
df['host_listings_5-'] =df['calculated_host_listings_count'] <= 5
df['host_listings_5-'].value_counts()
```

executed in 5ms, finished 00:24:21 2022-07-14

Out[10]:
```
True     6802
False    4133
Name: host_listings_5-, dtype: int64
```

5.3 New Feature: Capacity_5+

In [11]:
```
1 df['accommodates'].describe()
```
executed in 5ms, finished 00:24:21 2022-07-14

Out[11]:
```
count    10935.000000
mean         4.818930
std          3.139456
min          0.000000
25%          2.000000
50%          4.000000
75%          6.000000
max         16.000000
Name: accommodates, dtype: float64
```

In [12]:
```
1 df['capacity_5+'] = df['accommodates'] >=5
```
executed in 2ms, finished 00:24:21 2022-07-14

Analysis:

- Again, a binary classifier will be the most useful. Since a "family" unit would accommodate 4 or less, I am setting the classifier at 5 or more in order to determine whether the rental units are hotel room sized (2 or 4 people), or larger.

In [13]:
```
1 df['capacity_5+'].value_counts()
```
executed in 3ms, finished 00:24:21 2022-07-14

Out[13]:
```
False    6386
True     4549
Name: capacity_5+, dtype: int64
```

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

5.4 New Feature: Bedrooms_2+

In [14]:
```
1 df['bedrooms'].describe()
```
executed in 6ms, finished 00:24:21 2022-07-14

Out[14]:
```
count    9905.000000
mean        1.969611
std         1.217462
min         1.000000
25%         1.000000
50%         2.000000
75%         3.000000
max        12.000000
Name: bedrooms, dtype: float64
```

Most of the units in this dataset have 1-3 bedrooms. I will create a feature that groups units as

either 1 bedroom or 2 or more.

In [15]:
```
df['bedrooms_2+'] = df['bedrooms'] >= 2
df['bedrooms_2+'].value_counts()
```

executed in 4ms, finished 00:24:21 2022-07-14

Out[15]:
```
False    5651
True     5284
Name: bedrooms_2+, dtype: int64
```

5.5 New Feature: Booking Rates

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

In [16]:
```
df['availability_30_rate'] = df['availability_30'].apply(lambda x: x /
df['availability_90_rate'] = df['availability_90'].apply(lambda x: x /
```

executed in 7ms, finished 00:24:21 2022-07-14

In [17]:
```
df['booked_rate_30'] =  df['availability_30_rate'].apply(lambda x: 1 -
df['booked_rate_90'] =  df['availability_90_rate'].apply(lambda x: 1 -

```

executed in 7ms, finished 00:24:21 2022-07-14

In [18]:
```
availability = df[['availability_30_rate', 'booked_rate_30', 'availabil
availability.head()
```

executed in 8ms, finished 00:24:21 2022-07-14

Out[18]:

	availability_30_rate	booked_rate_30	availability_90_rate	booked_rate_90
0	0.000000	1.000000	0.566667	0.433333
1	0.400000	0.600000	0.700000	0.300000
2	0.166667	0.833333	0.055556	0.944444
3	0.333333	0.666667	0.588889	0.411111
4	0.833333	0.166667	0.944444	0.055556

5.6 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 [19]:
```
df['bookings_above_avg'] = df['booked_rate_90'] >=.512
df['bookings_above_avg'].value_counts()
```

executed in 5ms, finished 00:24:21 2022-07-14

Out[19]:
```
True     5476
False    5459
Name: bookings_above_avg, dtype: int64
```

5.7 New Feature: Host Response Rate 100

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

In [20]:
```
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()
```

executed in 9ms, finished 00:24:21 2022-07-14

Out[20]:
```
True     7229
False    3706
Name: host_response_100, dtype: int64
```

5.8 Fixing Host is Superhost & Instant Bookable

Features are currently strings instead of bools.

In [21]:
```
df['superhost'] = df['host_is_superhost'] == 't'
df['instant_bookable'] = df['instant_bookable'] == 't'
```

executed in 4ms, finished 00:24:21 2022-07-14

In [22]:
```
df['superhost'].value_counts()
```

executed in 4ms, finished 00:24:21 2022-07-14

Out[22]:
```
False    6095
True     4840
Name: superhost, dtype: int64
```

In [23]:
```
df['instant_bookable'].value_counts()
```

executed in 4ms, finished 00:24:21 2022-07-14

Out[23]:
```
False    5912
True     5023
Name: instant_bookable, dtype: int64
```

5.9 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.

5.9.1 Dealing with Nulls

In [24]:
```
1 df['review_scores_rating'].isna().sum()
```
executed in 3ms, finished 00:24:21 2022-07-14

Out[24]: 1527

There are 1527 Null records that need to be dealt with. If I drop them, I will lose 14% of my data.

In [25]:
```
1 nulls = df[df['review_scores_rating'].isna()]
```
executed in 4ms, finished 00:24:21 2022-07-14

In [26]:
```
1 nulls.head(3)
```
executed in 11ms, finished 00:24:21 2022-07-14

Out[26]:

	price	review_scores_rating	review_scores_accuracy	review_scores_cleanliness	review_scores_c
12	1200.0	NaN	NaN	NaN	
23	217.0	NaN	NaN	NaN	
70	129.0	NaN	NaN	NaN	

3 rows × 33 columns

nulls appear to have no ratings. Let's drop them for now.

In [27]:
```
1 df = df.dropna()
```
executed in 7ms, finished 00:24:21 2022-07-14

8385 Records are left after dropping null values

In [28]:
```
1 df.head(2)
```
executed in 12ms, finished 00:24:21 2022-07-14

Out[28]:

	price	review_scores_rating	review_scores_accuracy	review_scores_cleanliness	review_scores_ch
0	60.0	5.00	5.00	5.00	
1	282.0	4.87	4.91	4.64	

2 rows × 33 columns

5.9.2 Creating Elite Unit Classifier

- I have decided to classify "5 Star" units as ones that have a 4.9 or higher overall rating.
- 4.9 is still an incredibly high score, and is above 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 [29]:
```
1 df['elite'] = df['review_scores_rating'] >= 4.9
2 df['elite'].value_counts()
```
executed in 4ms, finished 00:24:21 2022-07-14

Out[29]:
```
False    4546
True     3264
Name: elite, dtype: int64
```

5.10 Room Type

In [30]:
```
1 df['room_type'].value_counts()
```
executed in 4ms, finished 00:24:21 2022-07-14

Out[30]:
```
Entire home/apt    6487
Private room       1261
Shared room          57
Hotel room            5
Name: room_type, dtype: int64
```

In [31]:
```
1 df['entire_home'] = df['room_type'] == 'Entire home/apt'
2 df['entire_home'].value_counts()
```
executed in 5ms, finished 00:24:21 2022-07-14

Out[31]:
```
True     6487
False    1323
Name: entire_home, dtype: int64
```

In [32]:
```
#dropping room_type since I now have a classifier in it's place.
df.drop(['room_type'], axis=1, inplace=True)
```
executed in 3ms, finished 00:24:21 2022-07-14

5.11 Host Response Time

In [33]:
```
df['host_response_time'].value_counts()
```
executed in 4ms, finished 00:24:21 2022-07-14

Out[33]:
```
within an hour        6229
within a few hours     993
within a day           446
a few days or more     142
Name: host_response_time, dtype: int64
```

In [34]:
```
df['response_within_hour'] = df['host_response_time'] == 'within an hou
df['response_within_hour'].value_counts()
```
executed in 4ms, finished 00:24:21 2022-07-14

Out[34]:
```
True     6229
False    1581
Name: response_within_hour, dtype: int64
```

In [35]:
```
#dropping host_response_time since I now have a classifier in it's plac
df.drop(['host_response_time'], axis=1, inplace=True)
```
executed in 4ms, finished 00:24:21 2022-07-14

5.12 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 classifer (5-Star), I am counting counting 4.9s in with the 5.0s.

In [36]:
```
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
```
executed in 5ms, finished 00:24:21 2022-07-14

5.13 New Feature: Price Above Median

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 below the MEDIAN price. (I rounded the median of 197 to 200)

In [37]:
```
df['price'].describe()
```
executed in 5ms, finished 00:24:21 2022-07-14

Out[37]:
```
count     7810.000000
mean       294.637900
std        330.516177
min         10.000000
25%        120.000000
50%        197.000000
75%        352.750000
max      10000.000000
Name: price, dtype: float64
```

In [38]:
```
df['price_200+'] = df['price'] >= 200
```
executed in 2ms, finished 00:24:21 2022-07-14

In [39]:
```
df['price_200+'].value_counts()
```
executed in 3ms, finished 00:24:21 2022-07-14

Out[39]:
```
False    3962
True     3848
Name: price_200+, dtype: int64
```

5.14 Creating Analysis_df

In [40]:
```
analysis_df = df.copy()
```
executed in 2ms, finished 00:24:21 2022-07-14

In [41]:
```
analysis_df['elite'].value_counts()
```
executed in 3ms, finished 00:24:21 2022-07-14

Out[41]:
```
False    4546
True     3264
Name: elite, dtype: int64
```

Analysis: Out of the 7810 Records in this dataset, 41% are 5 Star Units.

6 Preparing for Modeling

In [42]:
```
1 analysis_df.info()
```

executed in 10ms, finished 00:24:21 2022-07-14

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7810 entries, 0 to 10933
Data columns (total 41 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   price                           7810 non-null   float64
 1   review_scores_rating            7810 non-null   float64
 2   review_scores_accuracy          7810 non-null   float64
 3   review_scores_cleanliness       7810 non-null   float64
 4   review_scores_checkin           7810 non-null   float64
 5   review_scores_communication     7810 non-null   float64
 6   review_scores_location          7810 non-null   float64
 7   review_scores_value             7810 non-null   float64
 8   accommodates                    7810 non-null   int64  
 9   bedrooms                        7810 non-null   float64
 10  beds                            7810 non-null   float64
 11  instant_bookable                7810 non-null   bool   
 12  property_type                   7810 non-null   object 
 13  amenities                       7810 non-null   object 
 14  availability_365                7810 non-null   int64  
 15  availability_30                 7810 non-null   int64  
 16  availability_90                 7810 non-null   int64  
 17  host_id                         7810 non-null   int64  
 18  calculated_host_listings_count  7810 non-null   int64  
 19  host_response_rate              7810 non-null   float64
 20  host_is_superhost               7810 non-null   object 
 21  host_listings_5-                7810 non-null   bool   
 22  capacity_5+                     7810 non-null   bool   
 23  bedrooms_2+                     7810 non-null   bool   
 24  availability_30_rate            7810 non-null   float64
 25  availability_90_rate            7810 non-null   float64
 26  booked_rate_30                  7810 non-null   float64
 27  booked_rate_90                  7810 non-null   float64
 28  bookings_above_avg              7810 non-null   bool   
 29  host_response_100               7810 non-null   bool   
 30  superhost                       7810 non-null   bool   
 31  elite                           7810 non-null   bool   
 32  entire_home                     7810 non-null   bool   
 33  response_within_hour            7810 non-null   bool   
 34  accuracy_5                      7810 non-null   bool   
 35  cleanliness_5                   7810 non-null   bool   
 36  checkin_5                       7810 non-null   bool   
 37  location_5                      7810 non-null   bool   
 38  value_5                         7810 non-null   bool   
 39  communication_5                 7810 non-null   bool   
 40  price_200+                      7810 non-null   bool   
dtypes: bool(17), float64(15), int64(6), object(3)
memory usage: 1.6+ MB
```

6.1 One Hot Encoding

In [43]:
```
need_to_encode = df[['price_200+', 'elite', 'accuracy_5', 'cleanliness_
                     'communication_5', 'entire_home', 'bedrooms_2+',
                     'bookings_above_avg', 'instant_bookable', 'capacit
                     'host_listings_5-', 'superhost', 'host_response_10


ohe = OneHotEncoder()
ohe.fit(need_to_encode)

ohe_1 = ohe.transform(need_to_encode).toarray()

ohe_df = pd.DataFrame(ohe_1, columns=ohe.get_feature_names(need_to_enco
ohe_df.head(2)
```

executed in 27ms, finished 00:24:21 2022-07-14

Out[43]:

	price_200+_False	price_200+_True	elite_False	elite_True	accuracy_5_False	accuracy_5_True	cle
0	1.0	0.0	0.0	1.0	0.0	1.0	
1	0.0	1.0	1.0	0.0	0.0	1.0	

2 rows × 34 columns

In [44]:
```
cleaned_df = ohe_df
```

executed in 2ms, finished 00:24:21 2022-07-14

6.1.1 Dropping One Value for Categoricals

In [45]:
```
cleaned_df.drop(['elite_False', 'accuracy_5_False', 'cleanliness_5_Fals
                 'value_5_False', 'communication_5_False', 'bedrooms_2+
                 'bookings_above_avg_False', 'instant_bookable_False',
                 'host_listings_5-_False', 'entire_home_False', 'price_
                 'superhost_False', 'host_response_100_False', 'respons
                ], axis=1, inplace=True)
```

executed in 3ms, finished 00:24:21 2022-07-14

6.1.2 Dealing with Class Imbalance

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

In [46]:
```
1 cleaned_df['elite_True'].value_counts()
```
executed in 4ms, finished 00:24:21 2022-07-14

Out[46]:
```
0.0    4546
1.0    3264
Name: elite_True, dtype: int64
```

In [47]:
```
1 cleaned_df.isna().sum()
```
executed in 4ms, finished 00:24:21 2022-07-14

Out[47]:
```
price_200+_True              0
elite_True                   0
accuracy_5_True              0
cleanliness_5_True           0
checkin_5_True               0
location_5_True              0
value_5_True                 0
communication_5_True         0
entire_home_True             0
bedrooms_2+_True             0
bookings_above_avg_True      0
instant_bookable_True        0
capacity_5+_True             0
host_listings_5-_True        0
superhost_True               0
host_response_100_True       0
response_within_hour_True    0
dtype: int64
```

6.2 Train Test Split

In [48]:
```
1  balanced_df = cleaned_df.copy()
2
3
4  X = balanced_df.drop(['elite_True'], axis=1)
5  y = balanced_df['elite_True']
6
7
8  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25
9
10 smote = SMOTE(random_state=23)
11 X_train_resampled, y_train_resampled = smote.fit_sample(X_train, y_trai
```
executed in 93ms, finished 00:24:22 2022-07-14

6.3 Choosing Evaluation Metrics

- My goal is to predict whether a person will get a 4.9-5.0 Airbnb Overall rating.
- Which is worse?
  - Model predicts that a unit is an Elite Unit, but they actually aren't? (more false Positives)

- Model predicts that someone is NOT an Elite Unit but they actually are? (more false negatives)

Decision

- I want false Positives 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.
- **Therefore, I am most concerned with Precision, balanced out by F1 score.**

6.3.1 Metrics Function

In [49]:
```
def get_metrics(clf, y_pred):

    """Function that calculates the key metrics that I want to analyze
    unneccesary evaluation metrics that I don't need to see."""

    clf_prec = precision_score(y_test, y_pred) * 100
    print('Precision is :{0}'.format(clf_prec))
    clf_f1 = f1_score(y_test, y_pred) * 100
    print('F1 Score is :{0}'.format(clf_f1))
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_t
    clf_roc_auc = auc(false_positive_rate, true_positive_rate)
    print('ROC AUC is :{0}'.format(round(clf_roc_auc, 2)))
    clf_cv_score = np.mean(cross_val_score(clf, X_train_resampled, y_tr
    print('Cross Validation Score is :{0}'.format(round(clf_cv_score, 3
```

executed in 3ms, finished 00:24:22 2022-07-14

7 Modeling

7.1 Baseline Decision Tree

In [50]:
```
dt1 = DecisionTreeClassifier(random_state=23, class_weight="balanced")
dt1.fit(X_train_resampled, y_train_resampled)
dt1_y_pred = dt1.predict(X_test)
get_metrics(dt1, dt1_y_pred)
```

executed in 139ms, finished 00:24:22 2022-07-14

```
Precision is :76.12121212121212
F1 Score is :76.53869591712372
ROC AUC is :0.8
Cross Validation Score is :0.835
```

7.1.1 Baseline Model Analysis:

- A simple decision tree gives me a good starting point. The precision is is above 76%, which is acceptable, as is the F1 Score.
- The AUC Score is already at .8, which is great for baseline!
- Likewise, the Cross Validation score is already looking good as it above .8.

7.2 Decision Tree 2

7.2.1 Refining Decision Tree through GridSearchCV

In [51]:
```
dt_param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 2, 3, 4, 5, 6],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 3, 4, 5, 6]
}
```

executed in 2ms, finished 00:24:22 2022-07-14

In [52]:
```
# Instantiate GridSearchCV
dt2 = DecisionTreeClassifier(random_state=23)

dt_grid_search = GridSearchCV(dt2, dt_param_grid, cv=3, scoring = 'prec

# Fit to the data
dt_grid_search.fit(X_train_resampled, y_train_resampled)
dt_grid_search.best_params_
```

executed in 3.78s, finished 00:24:26 2022-07-14

Out[52]:
```
{'criterion': 'entropy',
 'max_depth': 3,
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

In [53]:
```
dt2 = DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samp
                             min_samples_leaf=1, class_weight='balanced
dt2.fit(X_train_resampled, y_train_resampled)
dt2_y_pred = dt2.predict(X_test)
get_metrics(dt2, dt2_y_pred)
```

executed in 72ms, finished 00:24:26 2022-07-14

```
Precision is :83.10038119440915
F1 Score is :81.59700561447286
ROC AUC is :0.84
Cross Validation Score is :0.849
```

7.3 Random Forests

In [54]:
```
rf1_clf = RandomForestClassifier(random_state=23, class_weight="balance
rf1_clf.fit(X_train_resampled, y_train_resampled)
rf1_y_pred = rf1_clf.predict(X_test)
get_metrics(rf1_clf, rf1_y_pred)
```

executed in 3.22s, finished 00:24:29 2022-07-14

```
Precision is :79.1566265060241
F1 Score is :79.82989064398542
ROC AUC is :0.83
Cross Validation Score is :0.858
```

7.4 Random Forests 2

7.4.1 GridSearch CV

In [55]:
```
rf_param_grid = {
    'n_estimators': [10, 30, 100],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 2, 6, 10],
    'min_samples_split': [5, 10],
    'min_samples_leaf': [3, 6]
}
```

executed in 2ms, finished 00:24:29 2022-07-14

In [56]:
```
rf2_clf = RandomForestClassifier(random_state=23)


rf1_grid_search= GridSearchCV(rf2_clf, rf_param_grid, scoring = 'precis
rf1_grid_search.fit(X_train_resampled, y_train_resampled)

print("")
print(f"Random Forest  Optimal Parameters: {rf1_grid_search.best_params
```

executed in 24.1s, finished 00:24:53 2022-07-14

```
Random Forest  Optimal Parameters: {'criterion': 'gini', 'max_depth': Non
e, 'min_samples_leaf': 3, 'min_samples_split': 5, 'n_estimators': 10}
```

In [57]:
```
rf2_clf = RandomForestClassifier(criterion= 'gini', max_depth= None, mi
                                 min_samples_split= 5, n_estimators= 10,
                                 class_weight='balanced')
rf2_clf.fit(X_train_resampled, y_train_resampled)
rf2_y_pred = rf2_clf.predict(X_test)
get_metrics(rf2_clf, rf2_y_pred)
```

executed in 312ms, finished 00:24:53 2022-07-14

```
Precision is :79.95365005793744
F1 Score is :82.19178082191782
ROC AUC is :0.85
Cross Validation Score is :0.862
```

7.5 XGBoost Model

In [58]:
```
# Instantiate XGBClassifier
clf = XGBClassifier(random_state=23)

# Fit XGBClassifier
xg1 = clf.fit(X_train_resampled, y_train_resampled)

# Predict on training and test sets
training_preds = clf.predict(X_train_resampled)
xg1_y_pred = clf.predict(X_test)
get_metrics(xg1, xg1_y_pred)
```

executed in 2.36s, finished 00:24:56 2022-07-14

```
Precision is :79.19143876337694
F1 Score is :80.3862401931201
ROC AUC is :0.83
Cross Validation Score is :0.858
```

7.5.1 GridSearch

In [59]:
```
boost_param_grid = {
    'learning_rate': [0.1, 0.2],
    'max_depth': [6],
    'min_child_weight': [1, 2],
    'subsample': [0.5, 0.7],
    'n_estimators': [100],
}
```

executed in 3ms, finished 00:24:56 2022-07-14

7.6 XGBoost 2

In [60]:
```
xg2 = XGBClassifier(random_state=23)

grid_clf = GridSearchCV(xg2, boost_param_grid, scoring='precision', cv=
grid_clf.fit(X_train_resampled, y_train_resampled)

best_parameters = grid_clf.best_params_

print('Grid Search found the following optimal parameters: ')
for param_name in sorted(best_parameters.keys()):
    print('%s: %r' % (param_name, best_parameters[param_name]))
```

executed in 4.98s, finished 00:25:01 2022-07-14

```
Grid Search found the following optimal parameters:
learning_rate: 0.1
max_depth: 6
min_child_weight: 1
n_estimators: 100
subsample: 0.7
```

In [61]:
```
xg2 = XGBClassifier(learning_rate= 0.1, max_depth=6, min_child_weight=1
                                  n_estimators=100, subsample=0.7, random
xg2.fit(X_train_resampled, y_train_resampled)
xg2_y_pred = xg2.predict(X_test)
get_metrics(xg2, xg2_y_pred)
```

executed in 2.85s, finished 00:25:03 2022-07-14

```
Precision is :81.11638954869359
F1 Score is :82.38841978287093
ROC AUC is :0.85
Cross Validation Score is :0.861
```

8 Model Comparison and Selection

In [62]:
```
#baseline metrics
get_metrics(dt1, dt1_y_pred)
```

executed in 123ms, finished 00:25:04 2022-07-14

```
Precision is :76.12121212121212
F1 Score is :76.53869591712372
ROC AUC is :0.8
Cross Validation Score is :0.835
```

In [63]:
```
get_metrics(dt2, dt2_y_pred)
```

executed in 68ms, finished 00:25:04 2022-07-14

```
Precision is :83.10038119440915
F1 Score is :81.59700561447286
ROC AUC is :0.84
Cross Validation Score is :0.849
```

In [64]:
```
1 get_metrics(rf2_clf, rf2_y_pred)
```

executed in 339ms, finished 00:25:04 2022-07-14

```
Precision is :79.95365005793744
F1 Score is :82.19178082191782
ROC AUC is :0.85
Cross Validation Score is :0.862
```

In [65]:
```
1 get_metrics(xg2, xg2_y_pred)
```

executed in 2.29s, finished 00:25:06 2022-07-14

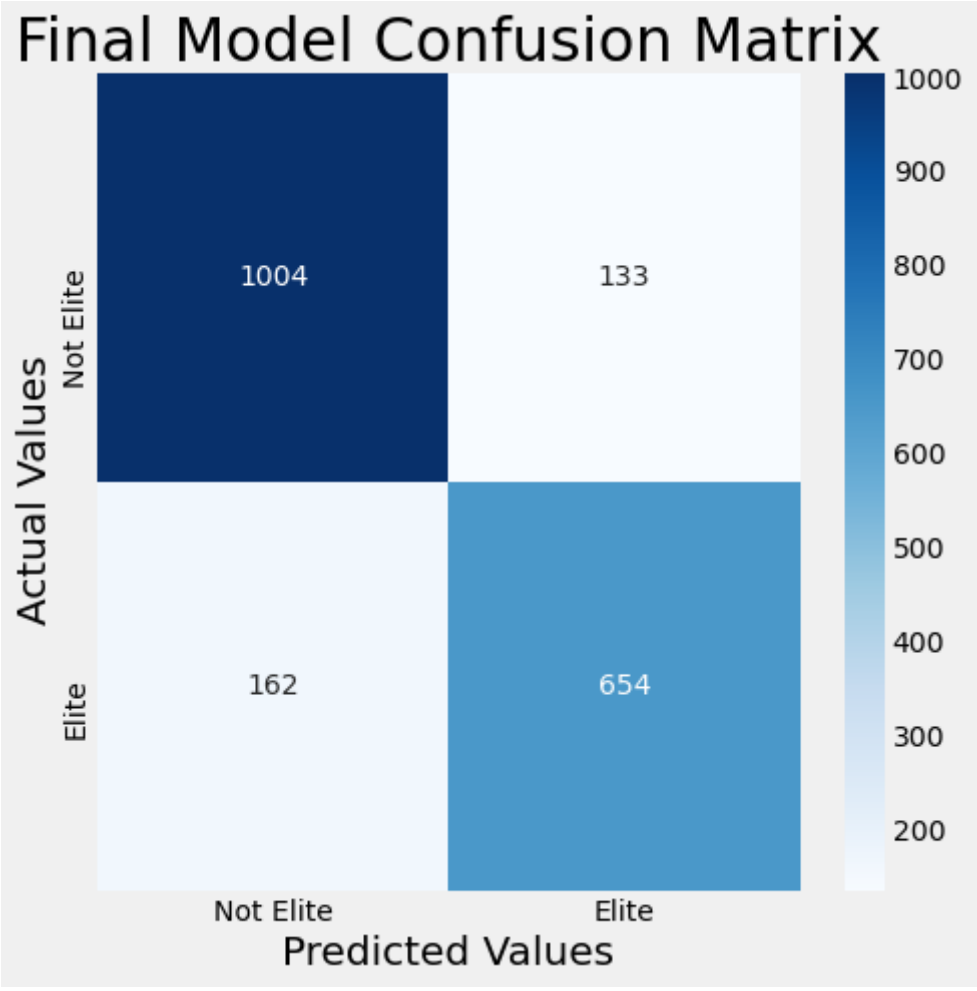
```
Precision is :81.11638954869359
F1 Score is :82.38841978287093
ROC AUC is :0.85
Cross Validation Score is :0.861
```

8.1 Model Selection: Decision Tree 2

8.2 Final Model Confusion Matrix

In [66]:
```
dt2_matrix = confusion_matrix(y_test, dt2_y_pred)

fig, ax = plt.subplots(figsize=(7,7))

ax = sns.heatmap(dt2_matrix, annot=True, cmap='Blues', fmt='d')

ax.set_title('Final Model Confusion Matrix', fontsize = 30);
ax.set_xlabel('Predicted Values',fontsize = 20)
ax.set_ylabel('Actual Values ', fontsize=20);

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['Not Elite','Elite'])
ax.yaxis.set_ticklabels(['Not Elite','Elite'])


## Display the visualization of the Confusion Matrix.
plt.show()
```

executed in 126ms, finished 00:25:06 2022-07-14



8.3 Final Model Evaluation:

- **Precision:** This Model correctly picks whether a rental will have an overall AirBnb rating between 4.9-5.0, 83% of the time.
  - This is 33% better than random guessing.

    - The Final Model is also a 7% improvement over the baseline model.
- **F1 Score:** While other models had slightly better F1 Scores, Decision Tree 2's F1 Score is only slightly worse. The F1 Score indicates that Precision is reasonably balanced with Recall, so I don't need to worry about this being an unbalanced and un-usable model. Therefore I'm fine choosing a model with a lower F1 in order to get more precsision.
- **ROC AUC Score:** Shows the True Positive Rate vs. the False Postive Rate. Some models had slightly higher scores than my Final Model, but again, it was very slight.
- **Cross Validation Score:** This model performs fairly well on data that it was not trained on and is comporable to the Cross Validation Scores of the other models.

8.4 Final Model Plot

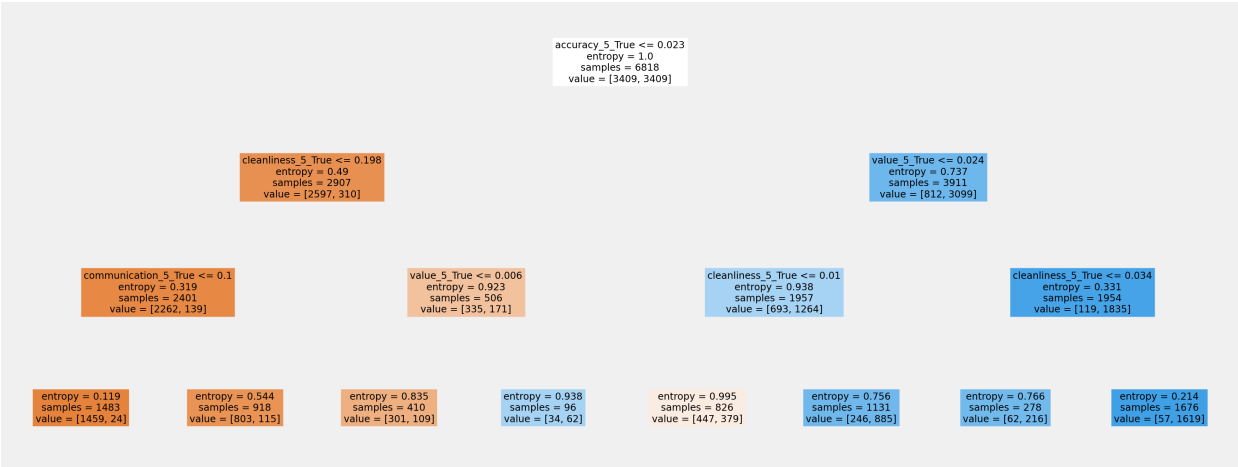
In [67]:
```
1 X.columns
```

executed in 3ms, finished 00:25:06 2022-07-14

Out[67]:
```
Index(['price_200+_True', 'accuracy_5_True', 'cleanliness_5_True',
       'checkin_5_True', 'location_5_True', 'value_5_True',
       'communication_5_True', 'entire_home_True', 'bedrooms_2+_True',
       'bookings_above_avg_True', 'instant_bookable_True', 'capacity_5+_T
rue',
       'host_listings_5-_True', 'superhost_True', 'host_response_100_Tru
e',
       'response_within_hour_True'],
      dtype='object')
```

In [68]:
```
1 fig = plt.figure(figsize=(50,20))
2 tree.plot_tree(dt2,
3                feature_names=X.columns,
4                filled=True);
```

executed in 420ms, finished 00:25:07 2022-07-14



8.5 How to use This Model going forward:

- OPM can take the data from new clients and run the model to determine whether they are performing at 5-Star level or not.

- If they are, they should be able to obtain Superhost status and OPM can focus on helping them **maintain** everything that they are doing right.
- If they are not a 5-Star rental unit, OPM can give them advice and help get them to 5-Star status.

Caveats:

- No model is perfect, and this one certainly isn't.
- This model relies on review scores from the 6 review categories. **If you don't have that data, the model does not perform reliably enough to be used.**
- That said, it can be reliably trusted as only 133 records from the test set of 1,953 were incorrectly labeled as being Elite Units when they were, in fact, not. (We aren't worried about the ones that were predicted to be not Elite incorrectly)

9 Feature Evaluation:

- Now that we have determined that the model is reasonably reliable and acceptable to use for predicting whether or not an AirBnb unit is an Elite Unit or not, we will use the model to tell us which features have the largest impact on making that classification.

9.1 Feature Importance

In [69]:
```
feature_names = list(X)
dt2_importance = dt2.feature_importances_
```

executed in 2ms, finished 00:25:07 2022-07-14

In [70]:
```
feature_importance_df = pd.DataFrame(dt2_importance, feature_names)
feature_importance_df= feature_importance_df.reset_index()
feature_importance_df.rename(columns={'index': 'Feature', 0: 'Importanc
feature_importance_df = feature_importance_df.sort_values('Importance',
feature_importance_df.head(4)
```

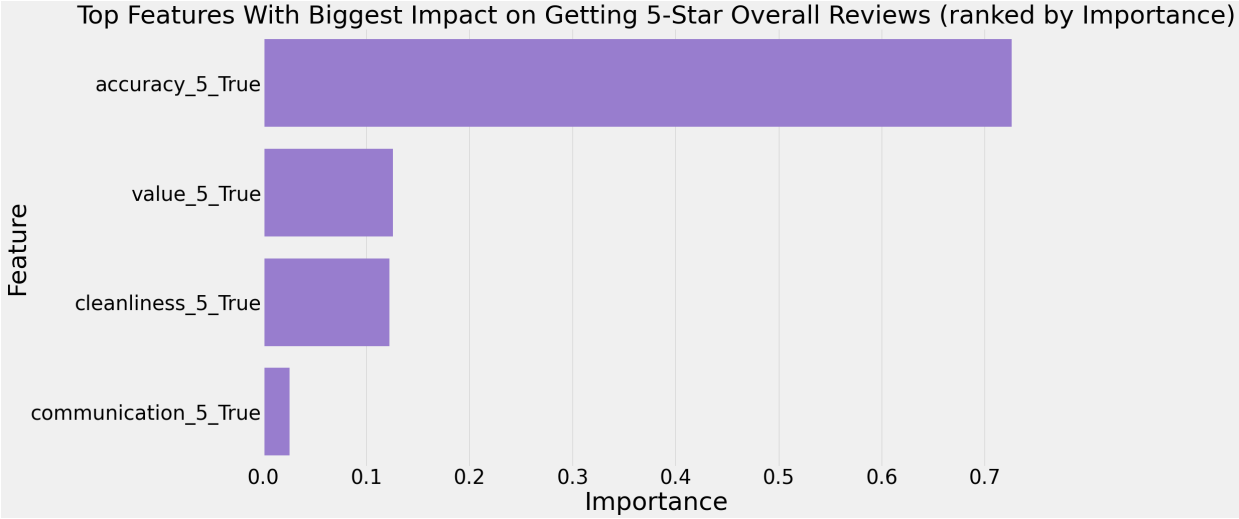
executed in 5ms, finished 00:25:07 2022-07-14

Out[70]:

	Feature	Importance
1	accuracy_5_True	0.726050
5	value_5_True	0.125592
2	cleanliness_5_True	0.122595
6	communication_5_True	0.025763

In [71]:
```
# plot feature importance
fig, ax = plt.subplots(figsize=(25,15))
p = sns.barplot(data=feature_importance_df.head(4), x='Importance', y='
p.set_xlabel("Importance", fontsize = 50)

p.set_ylabel("Feature", fontsize = 50)
plt.xticks(fontsize=40)
plt.yticks(fontsize=40)

p.set_title("Top Features With Biggest Impact on Getting 5-Star Overall
            fontsize = 50)


plt.show();
```

executed in 165ms, finished 00:25:07 2022-07-14



9.1.1 Analysis:

- **Accuracy is by far the most important feature**
- It is **5.5 Times more important** than the next features.
- **Value and Cleanliness are also important, but not nearly as much as Accuracy**
- **Communication** also has importance, but not nearly as much as the others.

9.1.2 Features with Little Impact on Target:

- All of the other Features show 0 importance in determining our Target status. However, I suspect that they play into the Accuracy, Value, etc, and will investigate that later.

10 Analysis of Top Features

10.1 Review Metric DF (or Feature Analysis DF)

In [72]:
```
feature_analysis_df = df.copy()
```
executed in 3ms, finished 00:25:07 2022-07-14

In [73]:
```
feature_analysis_df.drop(['review_scores_rating', 'review_scores_cleanl
                      'review_scores_accuracy', 'review_scores_communicat
                      'review_scores_checkin', 'accommodates', 'price', '
                      'availability_30', 'availability_90', 'availability
                      'host_id', 'bedrooms', 'beds', 'availability_30_rat
                       'host_is_superhost', 'amenities', 'property_type',
                      ], axis=1, inplace=True)
```
executed in 3ms, finished 00:25:07 2022-07-14

In [74]:
```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```
executed in 2ms, finished 00:25:07 2022-07-14

10.2 Function get_stats( )

In [75]:
```
def get_stats(df):

    """Takes the wide-form output of a groupby operation and transposes
    also adding a column "delta" which calculates the difference betwee
    value for each Metric."""

    df_transposed = df.transpose()
    df_transposed = df_transposed.reset_index()
    df_transposed.rename(columns={'index': 'Metric'}, inplace=True)
    stats_df = df_transposed
    delta = stats_df.apply(lambda x: x[1.0] - x[0.0], axis=1)
    stats_df['delta'] = delta

    return stats_df. sort_values('delta', ascending=False)
```
executed in 3ms, finished 00:25:07 2022-07-14

11 Top Features

In [76]:
```
#analysis_df sorted by calculated_host_listings_count, to make it easie
host_listings = analysis_df.sort_values('calculated_host_listings_count
```
executed in 4ms, finished 00:25:07 2022-07-14

In [77]:
```
superhost_df = host_listings[host_listings['superhost'] == True]
not_superhost_df = host_listings[host_listings['superhost'] == False]
```

executed in 4ms, finished 00:25:07 2022-07-14

In [78]:
```
elite_df = host_listings[host_listings['elite'] == True]
not_elite_df = host_listings[host_listings['elite'] == False]
```

executed in 4ms, finished 00:25:07 2022-07-14

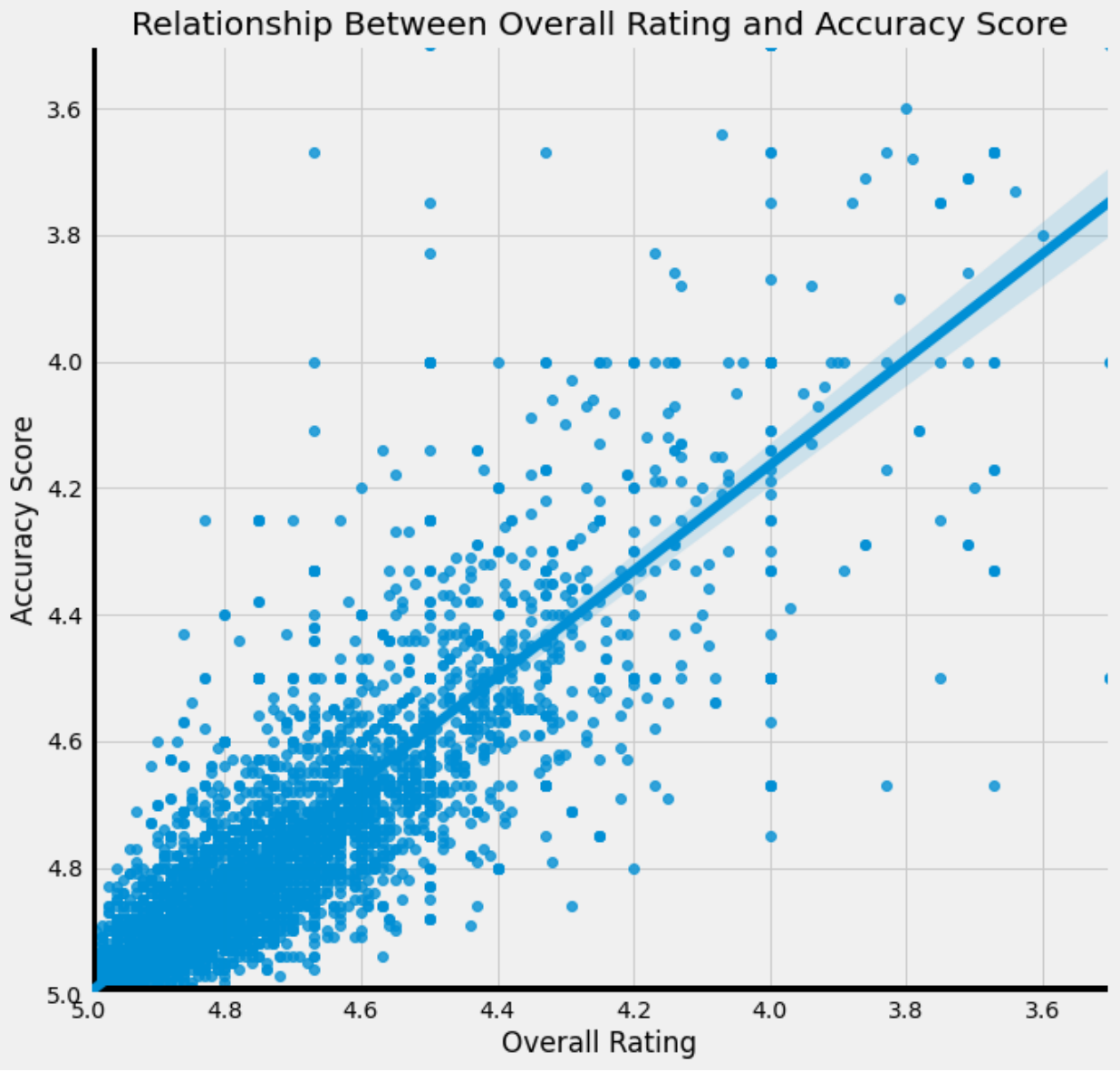
11.1 Top Feature #1: Accuracy

Accuracy is by far the most important feature in my model. Let's look at the relationship between Accuracy Score and Overall Rating.

In [79]:

```
fig, ax = plt.subplots(figsize=(10,10))
p = ax.invert_xaxis()
p = ax.invert_yaxis()

ax.axvline(5, color='black', linewidth=(10))
ax.axhline(5, color='black', linewidth=(10))


p =sns.regplot('review_scores_rating','review_scores_accuracy', data=df

ax.set_xlabel('Overall Rating')
ax.set_ylabel('Accuracy Score')

ax.set_title('Relationship Between Overall Rating and Accuracy Score')

ax.set_xlim(5.0, 3.5)
ax.set_ylim(5.0, 3.5);
```

executed in 1.78s, finished 00:25:09 2022-07-14



Analysis:

- As I suspected. There is a linear relationship between the two. Whatever the Accuracy Score is, the Overall Rating will likely be very similar as there is a nearly direct linear relationship.
- **Therefore, focusing on Accuracy is the best way to get 5 Star Reviews.**

In [80]:
```
1 df['review_scores_accuracy'].describe()
```

executed in 4ms, finished 00:25:09 2022-07-14

Out[80]:
```
count   7810.000
mean       4.799
std        0.363
min        1.000
25%        4.760
50%        4.900
75%        5.000
max        5.000
Name: review_scores_accuracy, dtype: float64
```

In [81]:
```
accuracy_metrics = feature_analysis_df.groupby('accuracy_5').mean()
accuracy_stats = get_stats(accuracy_metrics)
accuracy_stats
```

executed in 10ms, finished 00:25:09 2022-07-14

Out[81]:

accuracy_5	Metric	False	True	delta
9	elite	0.091	0.733	0.642
12	cleanliness_5	0.172	0.690	0.518
16	communication_5	0.425	0.895	0.469
15	value_5	0.057	0.468	0.411
13	checkin_5	0.487	0.891	0.404
14	location_5	0.423	0.759	0.337
8	superhost	0.379	0.640	0.261
1	host_listings_5-	0.521	0.735	0.214
7	host_response_100	0.679	0.784	0.104
6	bookings_above_avg	0.449	0.535	0.086
5	booked_rate_90	0.485	0.549	0.065
4	booked_rate_30	0.651	0.711	0.060
17	price_200+	0.502	0.484	-0.018
11	response_within_hour	0.809	0.787	-0.022
10	entire_home	0.846	0.816	-0.030
3	bedrooms_2+	0.568	0.516	-0.052
2	capacity_5+	0.518	0.430	-0.088
0	instant_bookable	0.536	0.413	-0.123

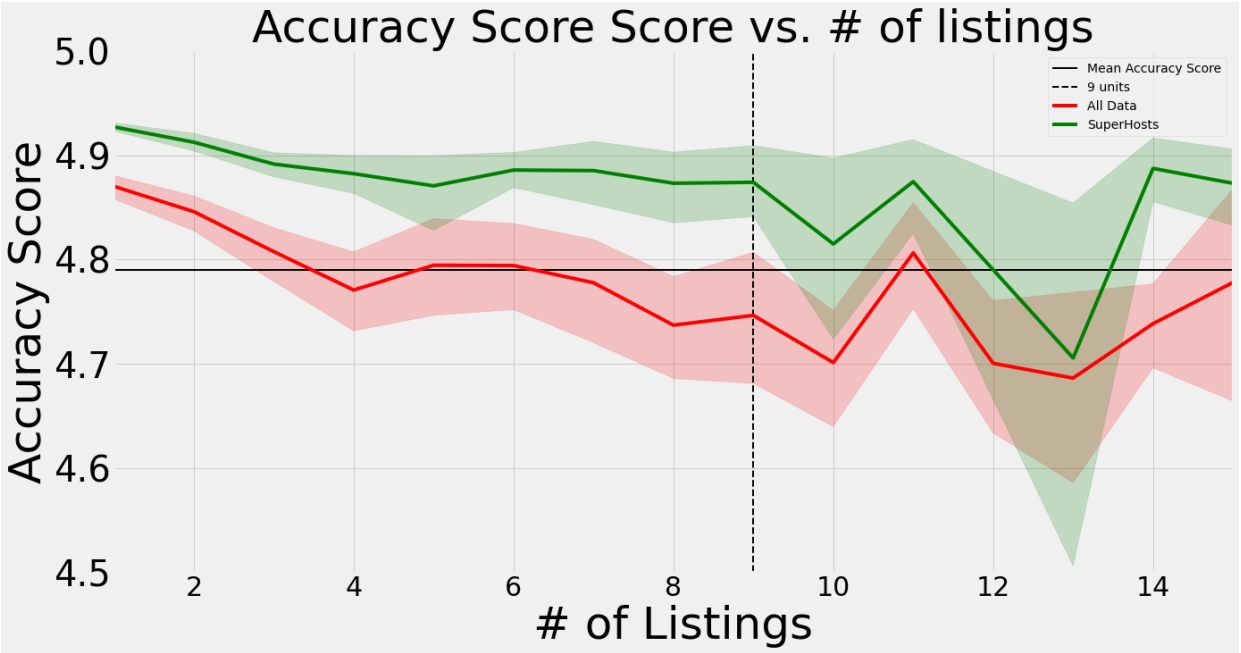
11.1.1 Analysis:

- This matches what I found in my research. **The most important aspect of renting an AirBnb is that the listing is accurate, to ensure that Guest expectations are met.**
- Nearly all units that have an accuracy score of 4.9-5.0 also scored high in the other 5 review metrics.
- Nearly all units that did not have an accuracy score of 5 did not score highly on others as well.
- **Significantly more likely to be Elite Units and/or SuperHosts**
- More likely to have a 100% Response Rate
- **73% of units that scored 4.9-5.0 on accuracy were in our target 5-star range.**
- They are less likely to use the instant book feature, although 41% of units with 5.0 accuracy do each.

In [82]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(4.79, color='black', linewidth=(2), label='Mean Accuracy Sco
ax.axvline(9, ls='--', color='black', linewidth=(2), label='9 units')


ax.set_xlim(1, 15)
ax.set_ylim(4.5, 5.0)
p = sns.lineplot(data=host_listings, x='calculated_host_listings_count'
                 color ='red' , label='All Data');

p = sns.lineplot(data=superhost_df, x='calculated_host_listings_count',
                 color ='green' , label='SuperHosts');


p.set_ylabel("Accuracy Score", fontsize = 50)

p.set_xlabel("# of Listings", fontsize = 50)
plt.xticks(fontsize=30)
plt.yticks(fontsize=40)


p.set_title("Accuracy Score Score vs. # of listings", fontsize = 50)

plt.show();
```

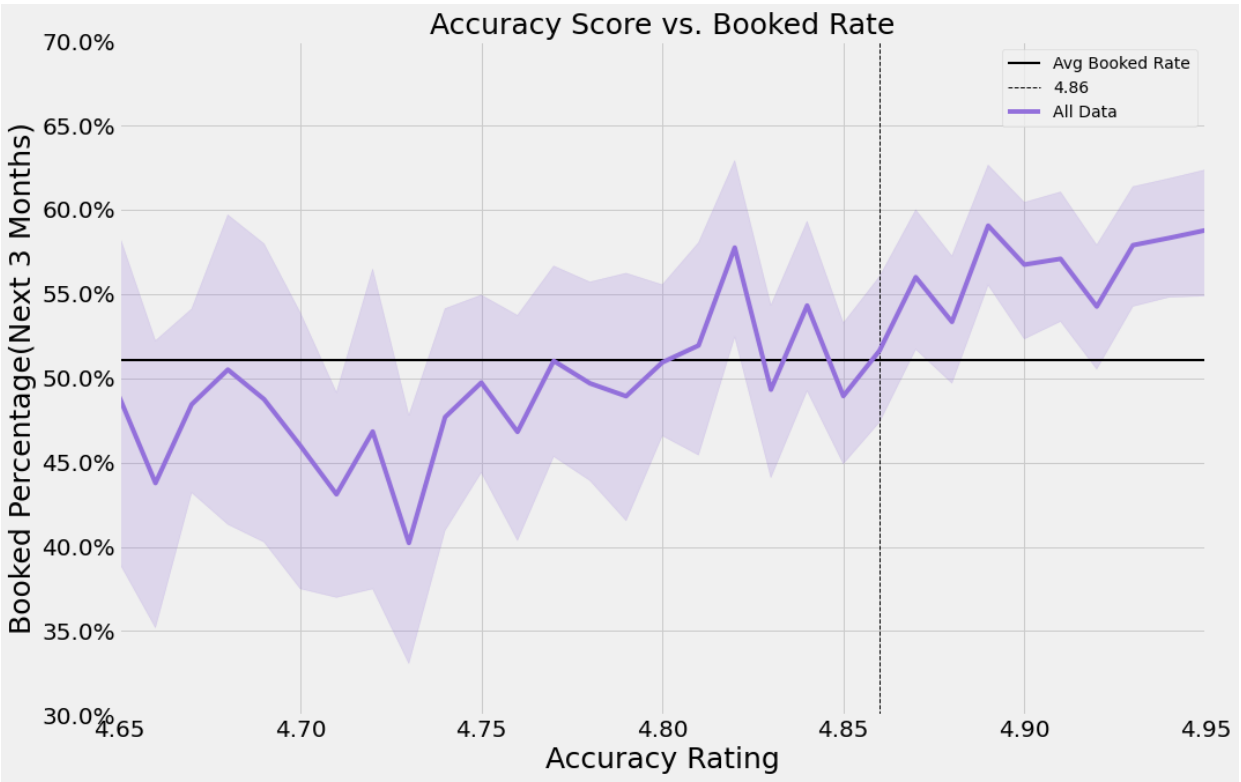
executed in 1.15s, finished 00:25:10 2022-07-14



In [83]:

```
fig, ax = plt.subplots(figsize=(15,10))

ax.axhline(.511,  color='black', linewidth=(2), label='Avg Booked Rate'
ax.axvline(4.86 , ls='--', color='black', linewidth=(1), label='4.86')


p = sns.lineplot(data=host_listings, x='review_scores_accuracy', y='boo
                 color ='mediumpurple', label='All Data' );


p.set_xlim(4.65,4.95)
p.set_ylim(.3, .7)

p.set_ylabel("Booked Percentage(Next 3 Months)", fontsize = 25)

p.set_xlabel("Accuracy Rating", fontsize = 25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

ax.yaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=No

p.set_title( "Accuracy Score vs. Booked Rate", fontsize = 25)


plt.show();
```

executed in 1.43s, finished 00:25:11 2022-07-14



Analysis:

- Mean Accuracy Score stays above average starting at **4.86** with a postive trend continuing.
- Also much more certainty in the values as they stay closer to the mean.
- The Accuracy Score for SuperHosts begins a downward trend around 8 units.

11.2 Top Feature #2: Value

In [84]:
```
1 df['review_scores_value'].describe()
```

executed in 4ms, finished 00:25:11 2022-07-14

Out[84]:
```
count   7810.000
mean       4.698
std        0.389
min        1.000
25%        4.630
50%        4.790
75%        4.900
max        5.000
Name: review_scores_value, dtype: float64
```

In [85]:
```
value_metrics = feature_analysis_df.groupby('value_5').mean()
value_stats = get_stats(value_metrics)
value_stats
```

executed in 7ms, finished 00:25:11 2022-07-14

Out[85]:

value_5	Metric	False	True	delta
9	elite	0.254	0.870	0.616
12	accuracy_5	0.369	0.896	0.526
13	cleanliness_5	0.299	0.815	0.516
16	communication_5	0.568	0.930	0.362
15	location_5	0.510	0.827	0.317
14	checkin_5	0.609	0.922	0.313
1	host_listings_5-	0.574	0.784	0.209
6	bookings_above_avg	0.473	0.549	0.077
7	host_response_100	0.713	0.786	0.073
5	booked_rate_90	0.500	0.565	0.065
4	booked_rate_30	0.668	0.717	0.049
8	superhost	0.500	0.545	0.046
11	response_within_hour	0.815	0.748	-0.067
0	instant_bookable	0.504	0.389	-0.115
10	entire_home	0.862	0.745	-0.117
3	bedrooms_2+	0.573	0.455	-0.118
17	price_200+	0.527	0.399	-0.128
2	capacity_5+	0.509	0.374	-0.135

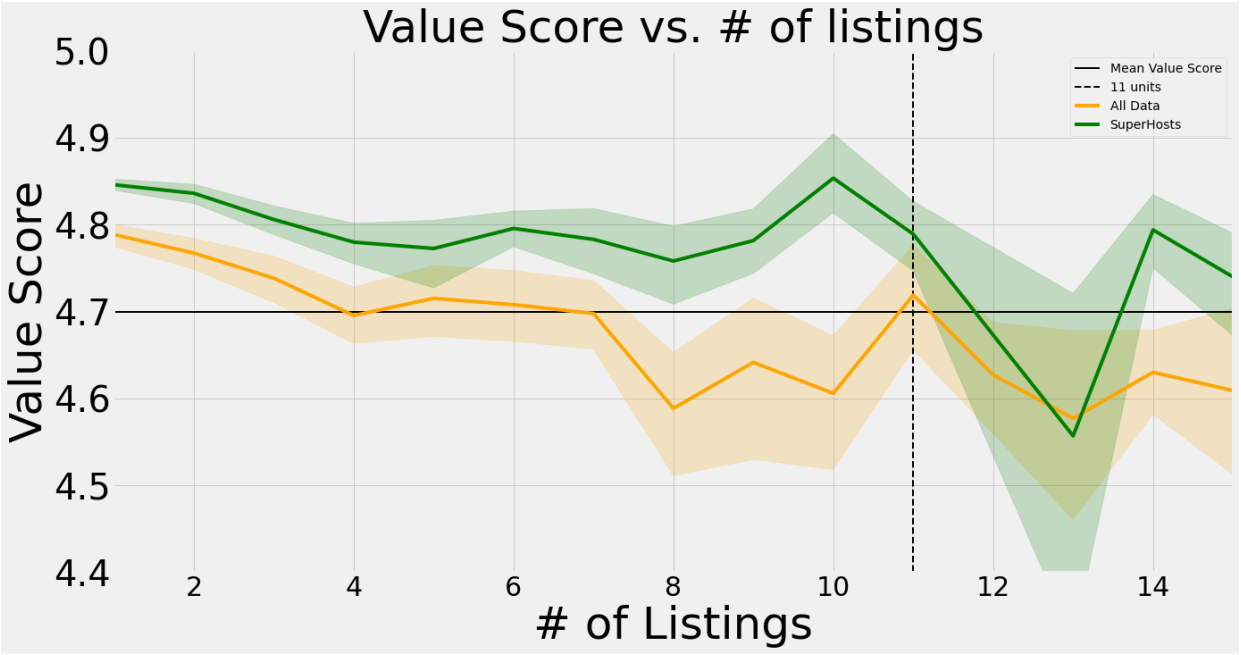
11.2.1 Analysis:

- Units with a Value Score of 4.9+ are significantly more likely to have higher scores on all review metrics.
- They are also more likely to be a 5-Star Unit, with **87% of units with high value being 5-Star Units.**

In [86]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(4.7, color='black', linewidth=(2), label='Mean Value Score')
ax.axvline(11, ls='--', color='black', linewidth=(2), label='11 units')


ax.set_xlim(1, 15)
ax.set_ylim(4.4, 5.0)
p = sns.lineplot(data=host_listings, x='calculated_host_listings_count'
                 color ='orange' , label='All Data');

p = sns.lineplot(data=superhost_df, x='calculated_host_listings_count',
                 color ='green' , label='SuperHosts');

p.set_ylabel("Value Score", fontsize = 50)

p.set_xlabel("# of Listings", fontsize = 50)
plt.xticks(fontsize=30)
plt.yticks(fontsize=40)


p.set_title( "Value Score vs. # of listings", fontsize = 50)

plt.show();
```

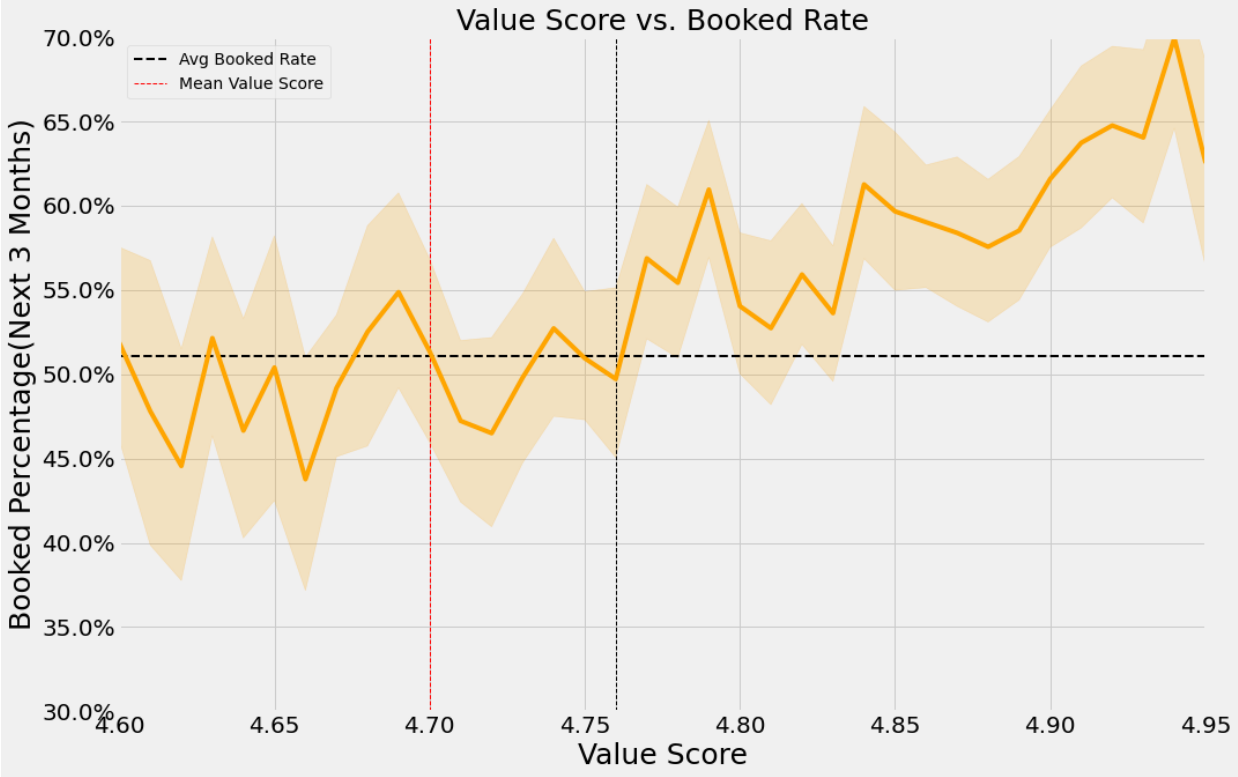
executed in 1.14s, finished 00:25:13 2022-07-14



In [87]:

```
fig, ax = plt.subplots(figsize=(15,10))

ax.axhline(.511, ls='--', color='black', linewidth=(2), label='Avg Book
ax.axvline(4.7 ,ls='--', color='red', linewidth=(1), label='Mean Value
ax.axvline(4.76 ,ls='--', color='black', linewidth=(1), label='')


p = sns.lineplot(data=host_listings, x='review_scores_value', y='booked
                 color ='orange')


p.set_xlim(4.6,4.95)
p.set_ylim(.3, .7)

p.set_ylabel("Booked Percentage(Next 3 Months)", fontsize = 25)

p.set_xlabel("Value Score", fontsize = 25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

ax.yaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=No

p.set_title( "Value Score vs. Booked Rate", fontsize = 25)


plt.show();
```

executed in 1.55s, finished 00:25:14 2022-07-14



- Higher Value Scores translate into higher booked rates.
- **There is no real differnce between being a Superhost or Five-Star Unit vs. normal here. All benefit equally from an increase in value.**

- Value stays above average booking rate with positive trend starting at 4.76.
- **Value increases booking rate more than Accuracy does.**

11.3 Top Feature #3: Cleanliness

In [88]:
```
1 df['review_scores_cleanliness'].describe()
```

executed in 4ms, finished 00:25:14 2022-07-14

Out[88]:
```
count   7810.000
mean       4.759
std        0.373
min        1.000
25%        4.690
50%        4.860
75%        4.980
max        5.000
Name: review_scores_cleanliness, dtype: float64
```

In [89]:
```
cleanliness_metrics = feature_analysis_df.groupby('cleanliness_5').mean
cleanliness_stats = get_stats(cleanliness_metrics)
cleanliness_stats
```

executed in 7ms, finished 00:25:14 2022-07-14

Out[89]:

cleanliness_5	Metric	False	True	delta
9	elite	0.165	0.746	0.581
12	accuracy_5	0.279	0.806	0.527
15	value_5	0.087	0.497	0.410
16	communication_5	0.503	0.873	0.371
13	checkin_5	0.555	0.870	0.315
14	location_5	0.488	0.732	0.244
8	superhost	0.422	0.629	0.208
1	host_listings_5-	0.562	0.718	0.156
7	host_response_100	0.698	0.778	0.080
4	booked_rate_30	0.671	0.695	0.024
5	booked_rate_90	0.509	0.528	0.019
6	bookings_above_avg	0.486	0.502	0.017
11	response_within_hour	0.798	0.796	-0.002
17	price_200+	0.502	0.480	-0.022
10	entire_home	0.841	0.817	-0.025
3	bedrooms_2+	0.570	0.506	-0.064
2	capacity_5+	0.513	0.421	-0.093
0	instant_bookable	0.514	0.420	-0.094

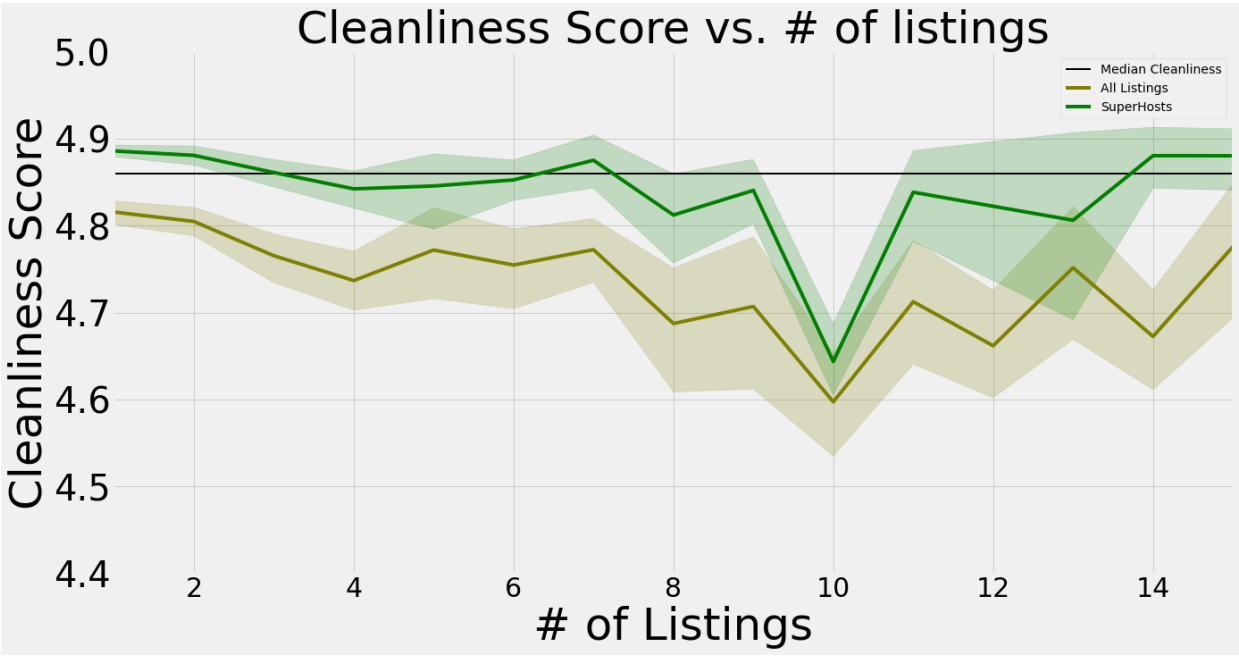
11.3.1 Analysis:

- More likely to score higher in all review metrics.
- 75% of Cleanliness 5.0 units have 5-Star Status.

In [90]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(4.86, color='black', linewidth=(2), label='Median Cleanlines

ax.set_xlim(1, 15)
ax.set_ylim(4.4, 5.0)
p = sns.lineplot(data=host_listings, x='calculated_host_listings_count'
                 color ='olive' , label='All Listings');

p = sns.lineplot(data=superhost_df, x='calculated_host_listings_count',
                  color ='green', label='SuperHosts' );


p.set_ylabel("Cleanliness Score", fontsize = 50)

p.set_xlabel("# of Listings", fontsize = 50)
plt.xticks(fontsize=30)
plt.yticks(fontsize=40)


p.set_title( "Cleanliness Score vs. # of listings", fontsize = 50)

plt.show();
```

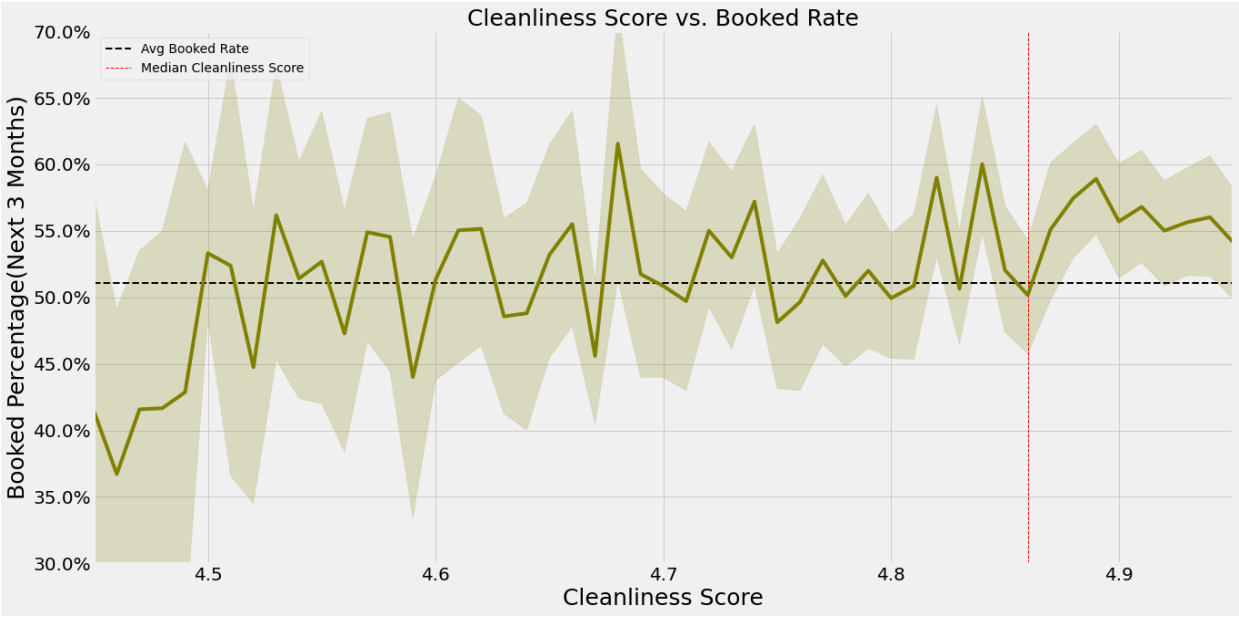
executed in 1.13s, finished 00:25:15 2022-07-14



In [91]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(.511, ls='--', color='black', linewidth=(2), label='Avg Book
ax.axvline(4.86 ,ls='--', color='red', linewidth=(1), label='Median Cle


p = sns.lineplot(data=host_listings, x='review_scores_cleanliness', y='
                 color ='olive')

p.set_xlim(4.45,4.95)
p.set_ylim(.3, .7)

p.set_ylabel("Booked Percentage(Next 3 Months)", fontsize = 25)

p.set_xlabel("Cleanliness Score", fontsize = 25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

ax.yaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=No

p.set_title( "Cleanliness Score vs. Booked Rate", fontsize = 25)


plt.show();
```

executed in 1.51s, finished 00:25:17 2022-07-14



11.4 Top Feature #4: Communication

In [92]:
```
1 df['review_scores_communication'].describe()
```

executed in 4ms, finished 00:25:17 2022-07-14

Out[92]:
```
count   7810.000
mean       4.854
std        0.344
min        1.000
25%        4.850
50%        4.950
75%        5.000
max        5.000
Name: review_scores_communication, dtype: float64
```

In [93]:
```
1 communication_metrics = feature_analysis_df.groupby('communication_5').
2 communication_stats = get_stats(communication_metrics)
3 communication_stats.sort_values(True, ascending=False)
```

executed in 8ms, finished 00:25:17 2022-07-14

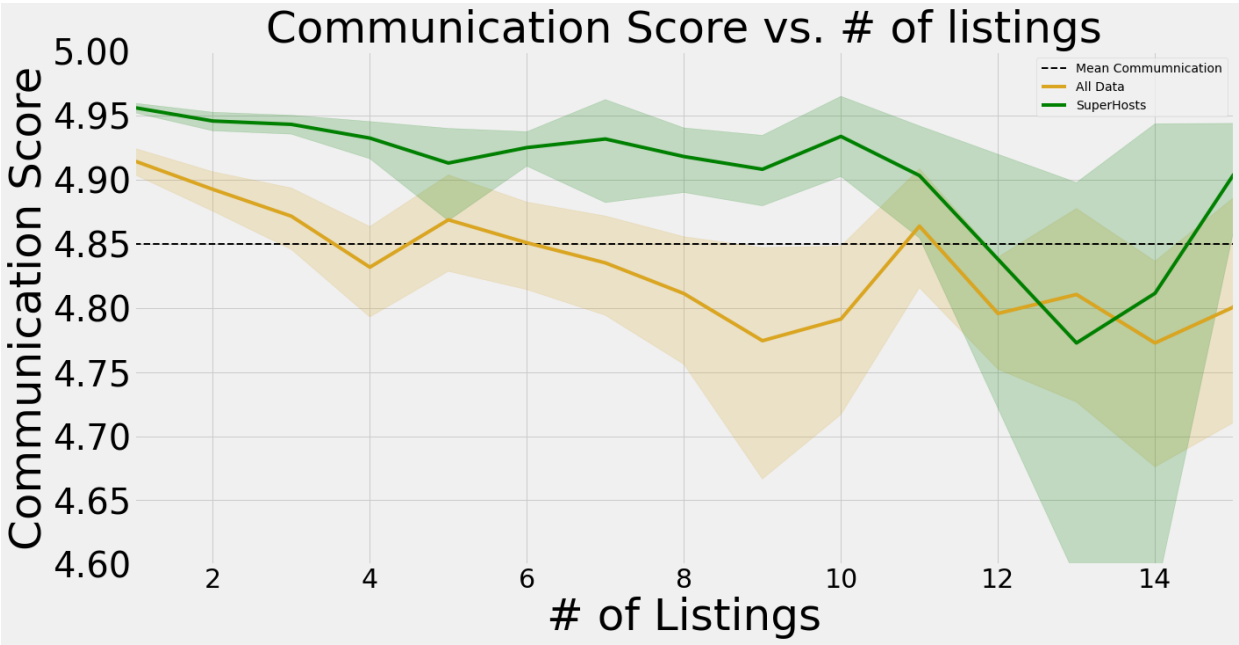
Out[93]:

communication_5	Metric	False	True	delta
14	checkin_5	0.331	0.876	0.545
10	entire_home	0.833	0.830	-0.003
11	response_within_hour	0.791	0.801	0.010
7	host_response_100	0.625	0.787	0.162
1	host_listings_5-	0.467	0.712	0.245
4	booked_rate_30	0.632	0.706	0.074
15	location_5	0.386	0.699	0.314
12	accuracy_5	0.160	0.686	0.526
8	superhost	0.286	0.626	0.340
9	elite	0.088	0.585	0.496
13	cleanliness_5	0.165	0.573	0.408
5	booked_rate_90	0.464	0.545	0.081
6	bookings_above_avg	0.424	0.528	0.104
3	bedrooms_2+	0.575	0.525	-0.051
17	price_200+	0.520	0.479	-0.041
2	capacity_5+	0.523	0.447	-0.076
0	instant_bookable	0.569	0.425	-0.144
16	value_5	0.055	0.372	0.317

In [94]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(4.85, ls='--', color='black', linewidth=(2), label='Mean Com

ax.set_xlim(1, 15)
ax.set_ylim(4.6, 5.0)
p = sns.lineplot(data=host_listings, x='calculated_host_listings_count'
                 color ='goldenrod', label='All Data' );

p = sns.lineplot(data=superhost_df, x='calculated_host_listings_count',
                color ='green', label='SuperHosts');


p.set_ylabel("Communication Score", fontsize = 50)

p.set_xlabel("# of Listings", fontsize = 50)
plt.xticks(fontsize=30)
plt.yticks(fontsize=40)


p.set_title( "Communication Score vs. # of listings", fontsize = 50)

plt.show();
```

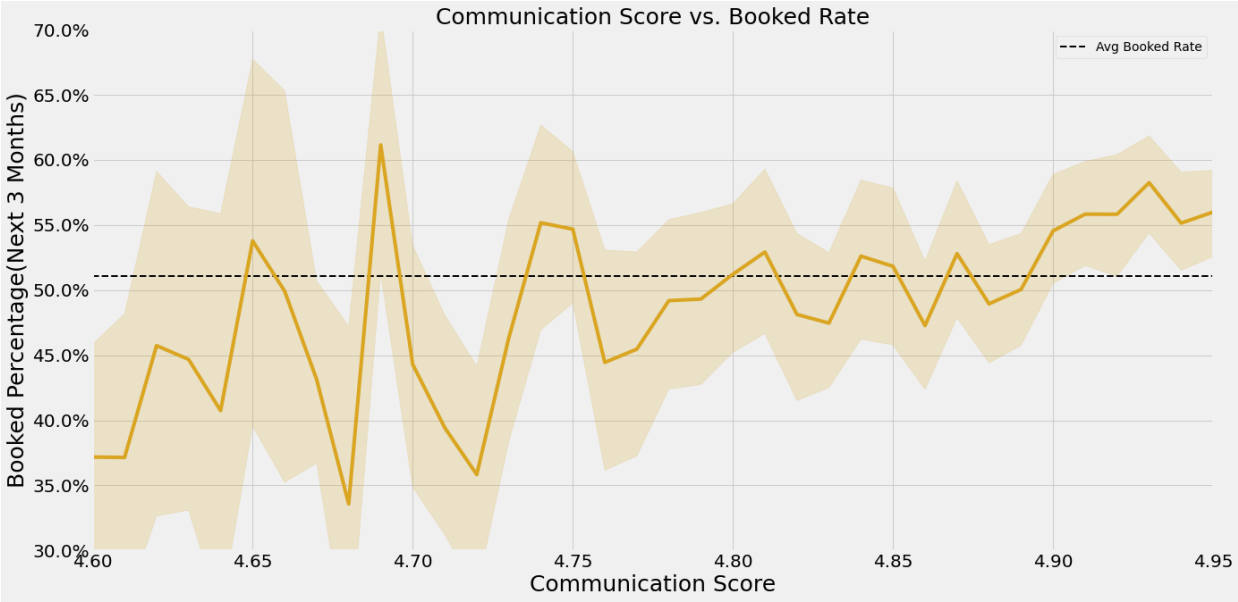
executed in 1.18s, finished 00:25:18 2022-07-14



In [95]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(.511, ls='--', color='black', linewidth=(2), label='Avg Book

p = sns.lineplot(data=host_listings, x='review_scores_communication', y
                 color ='goldenrod')

p.set_xlim(4.6,4.95)
p.set_ylim(.3, .7)

p.set_ylabel("Booked Percentage(Next 3 Months)", fontsize = 25)

p.set_xlabel("Communication Score", fontsize = 25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

ax.yaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=No

p.set_title( "Communication Score vs. Booked Rate", fontsize = 25)


plt.show();
```

executed in 1.28s, finished 00:25:19 2022-07-14



12 Questions Answered

12.1 Is there a significant advantage to being a Superhost? (is it worth all the effort to get this status and maintain it?)

In [96]:
```
superhost_metrics = feature_analysis_df.groupby('superhost').mean()
superhost_stats = get_stats(superhost_metrics)
superhost_stats
```

executed in 8ms, finished 00:25:19 2022-07-14

Out[96]:

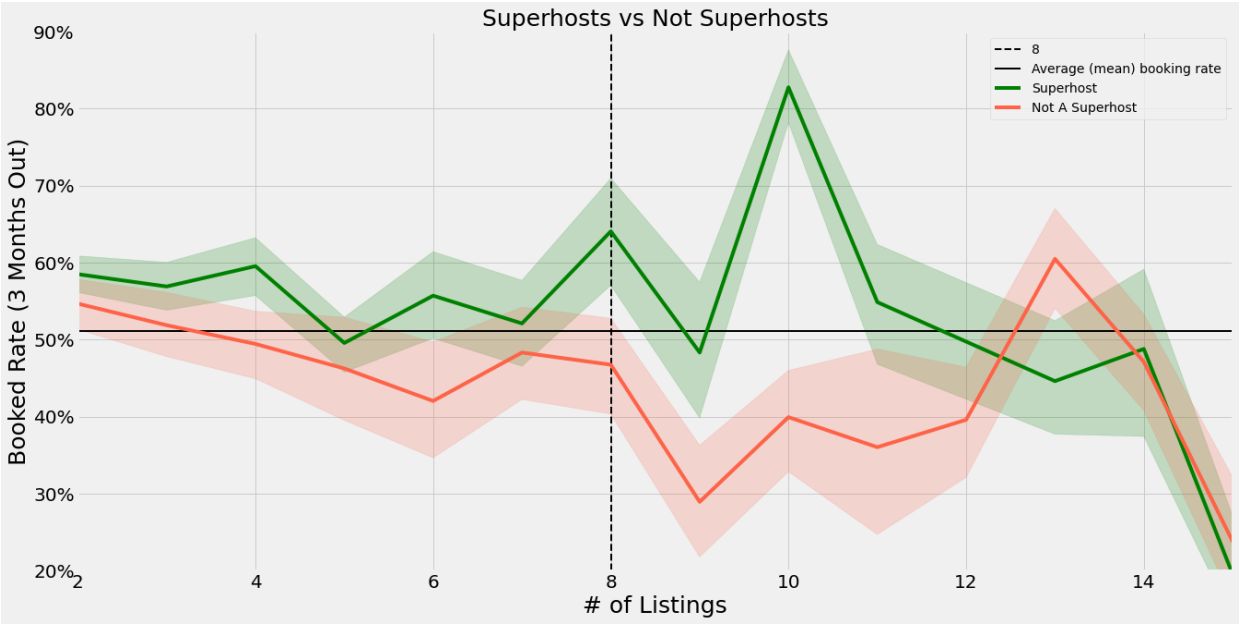
superhost	Metric	False	True	delta
16	communication_5	0.509	0.812	0.304
13	checkin_5	0.557	0.822	0.266
11	accuracy_5	0.375	0.636	0.261
8	elite	0.311	0.520	0.209
12	cleanliness_5	0.331	0.536	0.204
7	host_response_100	0.646	0.815	0.169
1	host_listings_5-	0.544	0.712	0.168
10	response_within_hour	0.730	0.862	0.131
14	location_5	0.530	0.656	0.126
6	bookings_above_avg	0.452	0.532	0.079
4	booked_rate_30	0.646	0.715	0.069
5	booked_rate_90	0.489	0.545	0.056
15	value_5	0.248	0.283	0.036
9	entire_home	0.822	0.839	0.018
17	price_200+	0.498	0.488	-0.010
2	capacity_5+	0.492	0.455	-0.037
3	bedrooms_2+	0.564	0.521	-0.043
0	instant_bookable	0.502	0.446	-0.056

- **YES!**
- **Superhosts are 21% more likely to be Elite Units than non-superhosts.**
- Superhosts and the 4 Important Features:
- 81% of Superhosts have at least 4.9 Communication Score. (30% better than non-superhosts)
- 64% of Superhosts have at least 4.9 Accuracy Score. (26% better than non-superhosts)
- Superhosts have similar Value Scores to Non-Superhosts.

In [97]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axvline(8 , ls='--', color='black', linewidth=(2), label='8')
ax.axhline(.511, color='black', linewidth=(2), label='Average (mean) bo

p = sns.lineplot(data=superhost_df,x='calculated_host_listings_count',
                 color ='green', label = 'Superhost' );

p = sns.lineplot(data=not_superhost_df,x='calculated_host_listings_coun
                  color ='tomato', label = 'Not A Superhost' );



p.set_xlim(2,15)
p.set_ylim(.2,.9)

p.set_ylabel("Booked Rate (3 Months Out)", fontsize = 25)

p.set_xlabel("# of Listings", fontsize = 25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

ax.yaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=No

p.set_title( "Superhosts vs Not Superhosts", fontsize = 25)

plt.show();
```

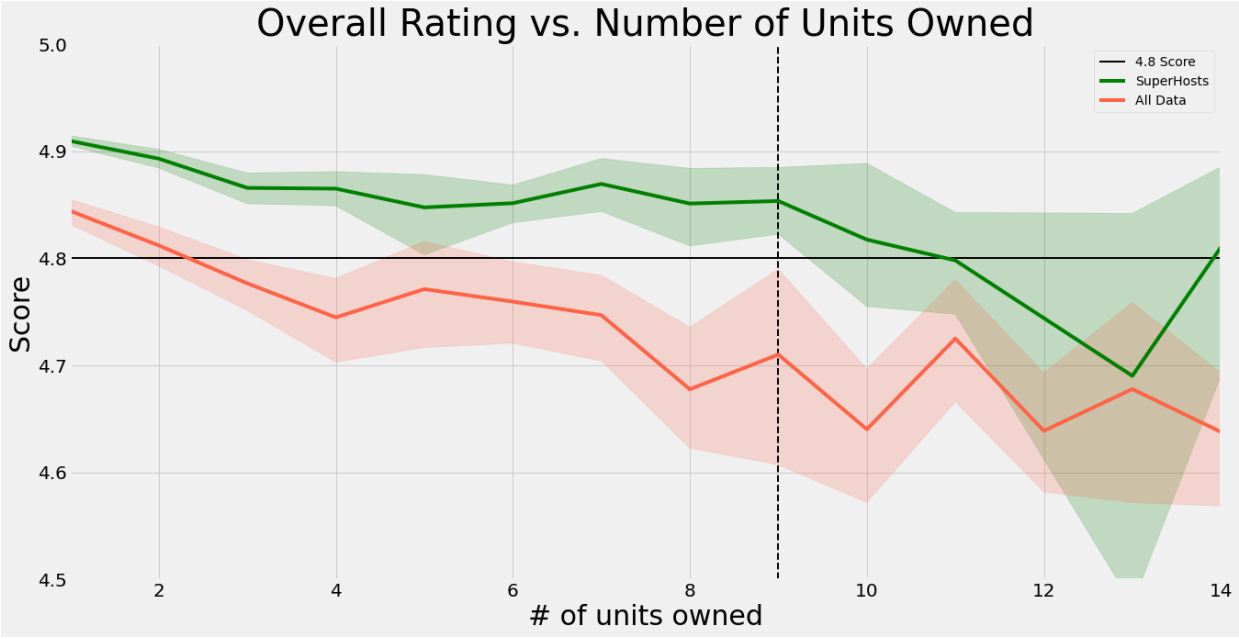
executed in 1.04s, finished 00:25:20 2022-07-14



In [98]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(4.8, color='black', linewidth=(2), label='4.8 Score')
ax.axvline(9, ls='--', color='black', linewidth=(2)),

p = sns.lineplot(data=superhost_df, x='calculated_host_listings_count',
                 color ='green', label='SuperHosts' );

p = sns.lineplot(data=host_listings, x='calculated_host_listings_count'
                  color ='tomato', label='All Data' );
p.set_xlim(1,14)
p.set_ylim(4.5, 5)


p.set_ylabel("Score", fontsize = 30)

p.set_xlabel("# of units owned", fontsize = 30)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)


p.set_title( "Overall Rating vs. Number of Units Owned", fontsize = 40)

plt.show();
```

executed in 1.13s, finished 00:25:21 2022-07-14



12.1.1 YES, Superhosts perform better

- Superhosts are better able to to handle higher numbers of listings.
- Most Superhosts can have 10 listings before it affects their 3 Month Booking Rate.
- Most Superhosts are also able to have 10 listings before their Overall Rating Drops below 4.8.

12.2 Is there a significant advantage to getting 5-Star overall Rating?

YES!

In [99]:
```
elite_metrics = feature_analysis_df.groupby('elite').mean()
elite_stats = get_stats(elite_metrics)
elite_stats
```

executed in 8ms, finished 00:25:22 2022-07-14

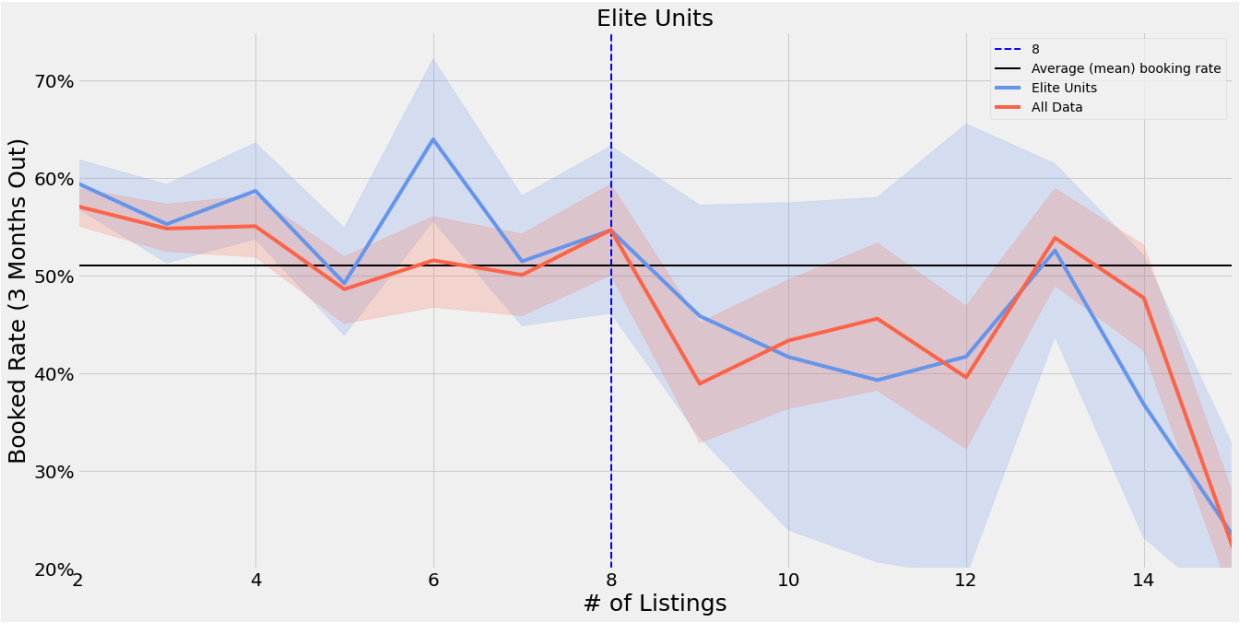
Out[99]:

elite	Metric	False	True	delta
11	accuracy_5	0.234	0.893	0.659
12	cleanliness_5	0.190	0.778	0.587
15	value_5	0.059	0.554	0.494
16	communication_5	0.474	0.929	0.455
13	checkin_5	0.534	0.913	0.379
14	location_5	0.448	0.797	0.349
8	superhost	0.422	0.637	0.215
1	host_listings_5-	0.541	0.755	0.214
7	host_response_100	0.692	0.789	0.096
6	bookings_above_avg	0.460	0.538	0.078
5	booked_rate_90	0.493	0.552	0.059
4	booked_rate_30	0.660	0.711	0.051
17	price_200+	0.484	0.505	0.020
3	bedrooms_2+	0.552	0.527	-0.025
9	entire_home	0.843	0.813	-0.031
10	response_within_hour	0.812	0.778	-0.034
2	capacity_5+	0.499	0.437	-0.062
0	instant_bookable	0.526	0.400	-0.126

In [100]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axvline(8 , ls='--', color='blue', linewidth=(2), label='8')
ax.axhline(.511, color='black', linewidth=(2), label='Average (mean) bo

p = sns.lineplot(data=elite_df,x='calculated_host_listings_count', y='b
                 color ='cornflowerblue', label = 'Elite Units' );

p = sns.lineplot(data=host_listings,x='calculated_host_listings_count',
                color ='tomato', label = 'All Data' );

p.set_xlim(2,15)
p.set_ylim(.2,.75)

p.set_ylabel("Booked Rate (3 Months Out)", fontsize = 25)

p.set_xlabel("# of Listings", fontsize = 25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

ax.yaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=No

p.set_title( "Elite Units", fontsize = 25)

plt.show();
```

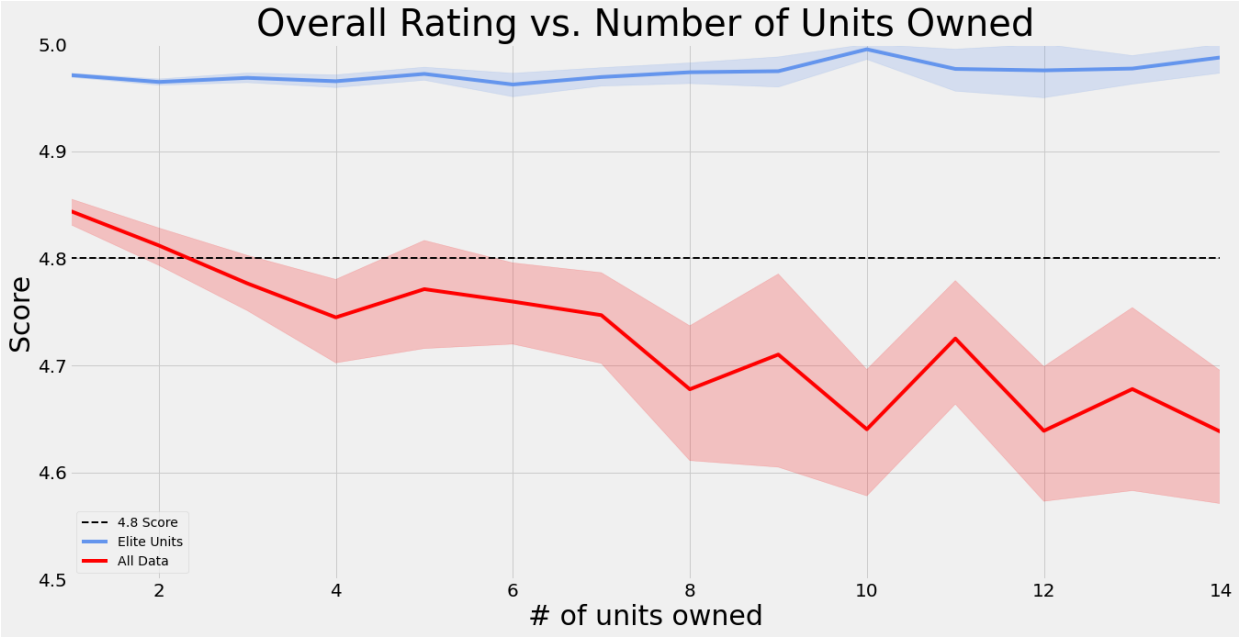
executed in 1.32s, finished 00:25:23 2022-07-14



In [101]:

```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(4.8, ls='--', color='black', linewidth=(2), label='4.8 Score

p = sns.lineplot(data=elite_df, x='calculated_host_listings_count', y='
                 color ='cornflowerblue', label='Elite Units' );

p = sns.lineplot(data=host_listings, x='calculated_host_listings_count'
                color ='red', label='All Data' );



p.set_xlim(1,14)
p.set_ylim(4.5, 5)


p.set_ylabel("Score", fontsize = 30)

p.set_xlabel("# of units owned", fontsize = 30)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)


p.set_title( "Overall Rating vs. Number of Units Owned", fontsize = 40)


plt.show();
```

executed in 1.31s, finished 00:25:24 2022-07-14



In [ ]:

```
1
```

12.3 Analysis:

- While Elite Units perform slightly better in booking rate, **being an Elite unit is the best solution to the negative trend between Overall Rating and Number of Units.**
- **As long as you can keep your units performing at the highest levels, there is no limit on how many units you list.**
- The catch is of course, learning how many that you can manage and keep at that level. This is an area where OPMs service will be invaluable!
- Offer resources to help Hosts. (Preferrred cleaners, stagers, contractors for emergencies. Maybe even a dedicated customer service phone number)

- Most Notably, **Elite Units have the biggest increase in the Top Features: accuracy, cleanliness, value, and communication.**
- **This shows that our Target does a good job of capturing the features that lead to more 5 Star Overall Reviews!**
- Elite Overall units score much higher in review metrics. This makes sense because they should have to score high in all of them to get a high overall score (even though it is a seperate metric in terms of AirBnb).
- they are also more likely to be a superhost, and more likely to have less than 5 listings.
- They are less likely to have high Capacity, or use Instant Book feature, but the differences aren't major.

Elite Units stats:

- **90% have 4.9-5.0 Average Scores in Communication, Check-in, and Accuracy.**
- 79% have perfect response rate.
- 76% have less than 5 listings
- 64% are Superhosts
- **55% of have a 4.9-5.0 Average Value Score**

13 Recommendations

13.1 The Focus of your AirBnb Consulting Service should be Improving and Maintaining Accuracy in everything that Hosts do.

- Accuracy Score has a nearly direct linear relationship with Overall Score. **Accuracy Score is by far the most important feature with effect on Overall Rating.**
- Leverage your experience in the rental market to ensure that host listings are accurate and not overly embellished.
- Be an "outside party" that understands what Airbnb guests need and want to see in listings.

This will lead to more Elite Units

- Elite Units should be eligible for becoming SuperHosts, and maintaining that status. (preferred listings, badges, “stamp of approval” from AirBnb.)

- **OPM should study the listings of units of units which consistently get 5.0 accuracy ratings to learn how to properly assess rental units and list them accurately.**
- This is the key value add that they can provide to clients.
- It's fairly easy to see that you need to have an accurate listing to perform well (many blogs and websites cite this). However, it's hard to say what practical steps a client can do to list their particular unit(s) properly. **OPM should market themselves as Accuracy Experts.**

Accuracy has linear relationships with Overall Rating, Value, Communication, and Cleanliness scores. **Performing well in Accuracy will have a positive result in ALL important features that increase the number of 5 star overall reviews.**

13.2 Provide Resources to Help Hosts Set Guest Expectations, and Then Exceed Them!

- accurate listing
- explanation of airbnb's skewed review system.
- do this without being deceptive or cooercive.

- It doesn't matter if Hosts have all the metrics and analysis to **know** that their unit deserves 5-star reviews. Their fate is in the hands of the reviewers. If they really care about getting 5 star reviews (and they should since they are critical to success on AirBnb), they need to explain this to their guests.
- It is also important to do this without begging, or deceptively cooercing your guests.

- There are many great blog posts and websites dedicated to this. The best solution that I found was this one from https://medium.com/@campbellandia/how-to-avoid-the-dreaded-4-star-review-a-guide-for-airbnb-hosts-cdf482d083fe (https://medium.com/@campbellandia/how-to-avoid-the-dreaded-4-star-review-a-guide-for-airbnb-hosts-cdf482d083fe) (accessed 6/21/22)

STAR RATINGS

When rating us after your stay, please consider that AirBnB's rating is totally different than the classic hotel star rating system: You should know that AirBnB can close down any listing that drops below an average of 4.3 stars at once. So the least we can say is that the system is rather confusing. To give you a better understanding, we designed the following (cheeky) guide to the rating system:

Please be aware of the judgement the stars may bear:

★★★★★ Perfection does not exist, but we were happy here.

★★★★ Several issues need to be improved.

★★★ Major problems that should be fixed immediately.

★★ Close it down at once.

★ Put these hosts into jail.

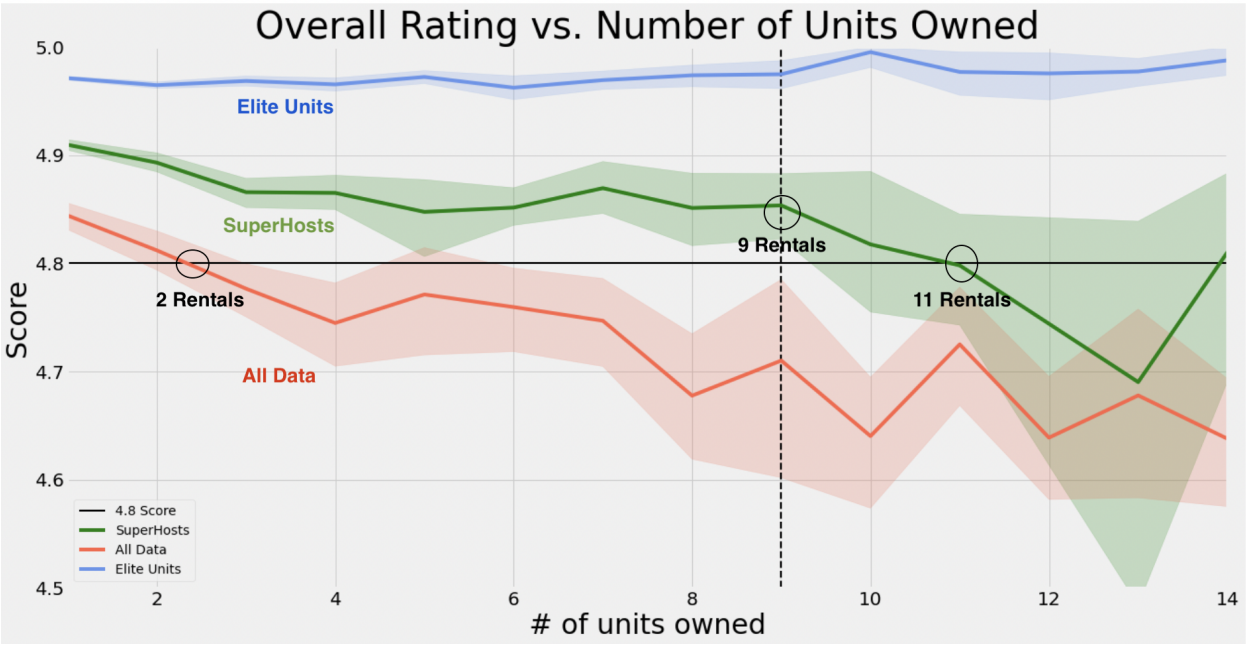
We always strive to give our guests a 5 star experience. Please let us know any problem you may have during your stay and we will do everything within our power to solve it.

13.3 Bridge the Gap Between Hosts' Self-Managing their Rentals, and OPM Fully Managing Rentals

- There is a general downward trend in overall rating as the number of units owned increases.
- Hosts with just 1 unit can likely keep everything at a very high level, and shouldn't need much help.
- Starting with 2 units, there is a negative trend in regards to most review categories, to the extent that most Hosts need some type of assistance
- If Hosts can obtain and maintain SuperHost status, they are able to handle more units on their own, usually up to 8.
- Also, I recommend that OPM offer services that help hosts to manage units once they get close to that threshold.
- - ie, prefered cleaning services, help with accurate listings, etc.

Target your consulting services at hosts with 2-8 rentals.

- If they have more than 8, try to transition them into your core business of property management.



- Non Superhosts will struggle with 2 or more properties.
- Superhosts can handle closer to 10.
- Offer Services to help these Clients that bridge the gap between Host-managed and OPM managed.
- **Give them a taste while putting them on a path toward being fully OPM managed.**
- You could also market yourself to people who haven't become Airbnb hosts yet, but want to learn how.

14 Conclusion

In my analysis of Airbnb rentals in San Diego California, I found that having a high overall rating (4.9-5.0), as well as having SuperHost status, were both beneficial to success on the platform.

- I also found that Accuracy was the biggest factor in getting a high overall rating, with a nearly 1 to 1 linear relationship.
- Other important features were Value, Cleanliness, & Communication.

Areas for OPM to Capitalize on:

- **Accuracy:** By providing a listing service which assesses client's rental units and lists their units in such a way as to maximize the accuracy.
- **Bridging the Gap between Owner-Managed and OPM Managed:** OPM can provide a la carte services which help owners who wish to keep managing their own properties, but can't handle doing so at the highest quality levels. **This is also beneficial to OPM in creating a pipeline of potential fully managed units as hosts take on more properties that they can manage.**
    - This can be structured in such a way to incentivize clients transitioning to OPMs full management service at certain thresholds (ie, 10 properties, etc).
- **Communication:** OPM can train hosts on what they can do to set expectations properly, and then exceed them with service (AirBnb's goal). This is done through how they communicate and how often they do it.

15 Further Work

15.1 Use Natural Language Processing to analyze Amenities.

- This DataSet includes **amenities, which would be very benefical to both the model and industry analysis.**
- However, they are all in string format and getting them into a useful format will be time intensive.
- Get them into a format where they can be one-hot encoded and fed into the model.

15.2 Increase the scope of this model.

- Incorporate data from the rest of California, and then the rest of the US.