AHMET16 / airbnb_classification Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

airbnb_classification / notebook.ipynb

AHMET16 add project5 History

1 contributor

2.55 MB

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 [2]:

```
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, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, mean_squared_error, mean_squared_log_error, roc_cu
from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score, classific
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 [3]:

```
#Data obtained from http://insideairbnb.com/san-diego
full_df = pd.read_csv('listings.csv')
```

In [4]:

```
full_df.info()
```

```
RangeIndex: 14188 entries, 0 to 14187
Data columns (total 75 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   id                                            14188 non-null  int64
 1   listing_url                                   14188 non-null  object
 2   scrape_id                                     14188 non-null  int64
 3   last_scraped                                  14188 non-null  object
 4   source                                        14188 non-null  object
 5   name                                          14188 non-null  object
 6   description                                   14060 non-null  object
 7   neighborhood_overview                         9306 non-null   object
```

```
 8   picture_url                                   14188 non-null  object
 9   host_id                                       14188 non-null  int64
 10  host_url                                      14188 non-null  object
 11  host_name                                     14174 non-null  object
 12  host_since                                    14174 non-null  object
 13  host_location                                 11669 non-null  object
 14  host_about                                    9132 non-null   object
 15  host_response_time                            13042 non-null  object
 16  host_response_rate                            13042 non-null  object
 17  host_acceptance_rate                          13465 non-null  object
 18  host_is_superhost                             14175 non-null  object
 19  host_thumbnail_url                            14174 non-null  object
 20  host_picture_url                              14174 non-null  object
 21  host_neighbourhood                            12131 non-null  object
 22  host_listings_count                           14174 non-null  float64
 23  host_total_listings_count                     14174 non-null  float64
 24  host_verifications                            14188 non-null  object
 25  host_has_profile_pic                          14174 non-null  object
 26  host_identity_verified                        14174 non-null  object
 27  neighbourhood                                 9306 non-null   object
 28  neighbourhood_cleansed                        14188 non-null  object
 29  neighbourhood_group_cleansed                  0 non-null      float64
 30  latitude                                      14188 non-null  float64
 31  longitude                                     14188 non-null  float64
 32  property_type                                 14188 non-null  object
 33  room_type                                     14188 non-null  object
 34  accommodates                                  14188 non-null  int64
 35  bathrooms                                     0 non-null      float64
 36  bathrooms_text                                14184 non-null  object
 37  bedrooms                                      12915 non-null  float64
 38  beds                                          14027 non-null  float64
 39  amenities                                     14188 non-null  object
 40  price                                         14188 non-null  object
 41  minimum_nights                                14188 non-null  int64
 42  maximum_nights                                14188 non-null  int64
 43  minimum_minimum_nights                        14186 non-null  float64
 44  maximum_minimum_nights                        14186 non-null  float64
 45  minimum_maximum_nights                        14186 non-null  float64
 46  maximum_maximum_nights                        14186 non-null  float64
 47  minimum_nights_avg_ntm                        14186 non-null  float64
 48  maximum_nights_avg_ntm                        14186 non-null  float64
 49  calendar_updated                              0 non-null      float64
 50  has_availability                              14188 non-null  object
 51  availability_30                               14188 non-null  int64
 52  availability_60                               14188 non-null  int64
 53  availability_90                               14188 non-null  int64
 54  availability_365                              14188 non-null  int64
 55  calendar_last_scraped                         14188 non-null  object
 56  number_of_reviews                             14188 non-null  int64
 57  number_of_reviews_ltm                         14188 non-null  int64
 58  number_of_reviews_l30d                        14188 non-null  int64
 59  first_review                                  12523 non-null  object
 60  last_review                                   12523 non-null  object
 61  review_scores_rating                          12523 non-null  float64
 62  review_scores_accuracy                        12502 non-null  float64
 63  review_scores_cleanliness                     12502 non-null  float64
 64  review_scores_checkin                         12500 non-null  float64
 65  review_scores_communication                   12502 non-null  float64
 66  review_scores_location                        12500 non-null  float64
 67  review_scores_value                           12500 non-null  float64
 68  license                                       152 non-null    object
 69  instant_bookable                              14188 non-null  object
 70  calculated_host_listings_count                14188 non-null  int64
 71  calculated_host_listings_count_entire_homes   14188 non-null  int64
 72  calculated_host_listings_count_private_rooms  14188 non-null  int64
 73  calculated_host_listings_count_shared_rooms   14188 non-null  int64
 74  reviews_per_month                             12523 non-null  float64
dtypes: float64(23), int64(17), object(35)
memory usage: 8.1+ MB
```

In [5]:

```python
base_df = full_df[['price', 'review_scores_rating', 'review_scores_accuracy',
                      'review_scores_cleanliness', 'review_scores_checkin', 'review_scores_com
                      'review_scores_location', 'review_scores_value','accommodates', 'bedroom
```

```
                        'instant_bookable', 'property_type', 'room_type', 'amenities', 'availabi
                        'availability_30','availability_90','host_id', 'calculated_host_listings
                        'host_response_time', 'host_response_rate','host_is_superhost']]
```

In [6]:

```
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 [7]:

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

Out[7]:

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

In [8]:

```
#using lambda function to strip $ and , out of each price record. replacing with blank space.
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 float
df['price'].head(2)
```

Out[8]:

```
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 [9]:

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

Out[9]:

```
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 [10]:

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

low_listings.value_counts()
```

Out[10]:

```
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 [11]:

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

Out[11]:

```
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 [12]:

```
df['bedrooms'].describe()
```

Out[12]:

```
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 [13]:

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

Out[13]:

```
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 [14]:

```
#Changing availability to a percentage named availability rate.
df['availability_30_rate'] = df['availability_30'].apply(lambda x: x/ 30)
df['availability_90_rate'] = df['availability_90'].apply(lambda x: x/ 90)
```

In [15]:

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

In [16]:

```
avaibility = df[['availability_30_rate', 'booked_rate_30','availability_90_rate','booked_rate_
avaibility.head()
```

Out[16]:

	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 [17]:

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

Out[17]:
```
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 [18]:
```
#creating a classifier that captures whether a host has a perfect response rate or not.
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[18]:
```
True     9916
False    4272
Name: host_response_100, dtype: int64
```

Fixing Host is Superhost & Instant Bookable

- Features are currently strings instead of bools.

In [19]:
```
#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 [20]:
```
#making sure that I captured both the True and False classification.
df['superhost'].value_counts()
```

Out[20]:
```
False    8782
True     5406
Name: superhost, dtype: int64
```

In [21]:
```
#making sure that I captured both the True and False Classifications.
df['instant_bookable'].value_counts()
```

Out[21]:
```
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 [22]:

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

Out[22]: 1665

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

In [23]:

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

In [24]:

```
len(nulls)
```

Out[24]: 1665

In [25]:

```
nulls.head(5)
```

Out[25]:

	price	review_scores_rating	review_scores_accuracy	review_scores_cleanliness	review_scores_checkin	revie
104	307.0	NaN	NaN	NaN	NaN	
118	900.0	NaN	NaN	NaN	NaN	
180	150.0	NaN	NaN	NaN	NaN	
183	235.0	NaN	NaN	NaN	NaN	
274	404.0	NaN	NaN	NaN	NaN	

5 rows × 32 columns

In [26]:

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

In [27]:

```
len(df)
```

Out[27]: 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 [28]:
```
#creating classifier and then checking to see how many units ae in each category.
df['elite'] = df['review_scores_rating'] >=4.9
df['elite'].value_counts()
```

Out[28]:
```
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 [29]:
```
#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[29]:
```
True     10503
False     2020
Name: entire_home, dtype: int64
```

In [30]:
```
#dropping room_typesince I now classifier inits place
df.drop(['room_type'], axis=1,inplace=True)
```

Host Response Time

In [31]:
```
#checking to see how many records I have of each response speed.
df['host_response_time'].value_counts()
```

Out[31]:
```
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 [32]:
```
#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 hour'
df['response_within_hour'].value_counts()
```

Out[32]:
```
True     9676
```

```
False    2847
Name: response_within_hour, dtype: int64
```

In [33]:

```
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 [34]:

```
#Creating a classifier for each review metric with the same critieria as my target (4.9 - 5.0

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 [35]:

```
#Printing a list with the number of units that are Elite in each category.
print("Number of Elite Accuracy Units:", len(df[df['accuracy_5']== True]))
print("Number of Elite Cleanliness Units:", len(df[df['cleanliness_5']== True]))
print("Number of Elite Checkin Units:", len(df[df['checkin_5']== True]))
print("Number of Elite Location Units:", len(df[df['location_5']== True]))
print("Number of Elite Value Units:", len(df[df['value_5']== True]))
print("Number of Elite Communication Units:", len(df[df['communication_5']== True]))
```

```
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 [36]:

```
df['value_5'].value_counts()
```

Out[36]:

```
False    9178
True     3345
Name: value_5, dtype: int64
```

New Feature: Price Above Median

In [37]:

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

Out[37]:
```
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 [38]:
```
df['price_280+'] =df['price'] >=280
```

In [39]:
```
df['price_280+'].value_counts()
```

Out[39]:
```
False    8756
True     3767
Name: price_280+, dtype: int64
```

Creating Analysis_df

In [40]:
```
#copying my dataframe as 'analysis_df' so I can easily pull back up my df with all classifiers

analysis_df = df.copy()
```

Preparing for Modeling

In [41]:
```
#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 [42]:

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

need_to_encode = df[['price_280+', 'elite', 'accuracy_5', 'cleanliness_5', 'checkin_5', 'locat
                     'communication_5', 'entire_home', 'bedrooms_2+',
                     'bookings_above_avg', 'instant_bookable', 'capacity_5+', 'calculated_host
#calling encoder and fitting it to the features that need to be encoded.
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_encode.columns))
ohe_df.head(5)
```

Out[42]:

	price_280+_False	price_280+_True	elite_False	elite_True	accuracy_5_False	accuracy_5_True	cleanliness_5_F
0	1.0	0.0	1.0	0.0	1.0	0.0	
1	1.0	0.0	1.0	0.0	1.0	0.0	
2	1.0	0.0	1.0	0.0	1.0	0.0	
3	0.0	1.0	1.0	0.0	1.0	0.0	
4	0.0	1.0	0.0	1.0	0.0	1.0	

5 rows × 87 columns

In [43]:

```
#creating 'cleaned_df'  as a copy of 'ohe_df' sothat I have a saved version of the df up to th
cleaned_df = ohe_df.copy()
```

Dropping One Value for Categoricals

In [44]:

```
#dropping one value from all categorical values to prevent redundancy.
cleaned_df.drop(['elite_False', 'accuracy_5_False', 'cleanliness_5_False', 'checkin_5_False',
                 'value_5_False', 'communication_5_False', 'bedrooms_2+_False',
                 'bookings_above_avg_False', 'instant_bookable_False', 'capacity_5+_False',
                 'calculated_host_listings_count_97', 'entire_home_False', 'price_280+_False',
                 'superhost_False', 'host_response_100_False', 'response_within_hour_False',
                ], 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 [45]:

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

Out[45]:

```
0.0    7385
1.0    5138
Name: elite_True, dtype: int64
```

Train Test Split

- Creating seperate Traning and Test Groups for modeling.

In [46]:

```
#creating 'balanced_df', which will end up being my df with balanced data
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 in the test set.
#Stratifying the split to minimize class imbalance.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25, stratify=y, random_st

#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_train)
```

Choosing Evaluation Metrics

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.
- Therefore, Iam most concerned with Precision, balanced out by F1 score.

Metrics Function

In [47]:

```
#creating 'get_metrics' function
def get_metrics(clf, y_pred):
    """ Function that calculates the key metrics that I want to analyze for my models. It also

    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_test,y_pred)

    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_train_resampled, cv=10))
    print('Cross Validation Score is :{0}' .format(round(clf_cv_score, 3)))
```

Modeling

Baseline Decision Tree

In [48]:

```
#Starting with a basic decision tree. Making sure that class weights are balanced.

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)
```

```
Precision is :77.18068535825545
F1 Score is :77.15064227325807
ROC AUC is :0.81
Cross Validation Score is :0.849
```

In [112…

```
y_pred_train = dt1.predict(X_train_resampled)
print(classification_report(y_train_resampled,y_pred_train))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.97      0.96      5539
         1.0       0.97      0.96      0.96      5539

    accuracy                           0.96     11078
   macro avg       0.96      0.96      0.96     11078
weighted avg       0.96      0.96      0.96     11078
```

In [113…

```
print(classification_report(y_test,dt1_y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.84      0.84      1846
         1.0       0.77      0.77      0.77      1285

    accuracy                           0.81      3131
   macro avg       0.81      0.81      0.81      3131
weighted avg       0.81      0.81      0.81      3131
```

Baseline Model Analysis:

- A simple decision tree gives me a good starting point.
- The precision is above 77%, which is acceptable , as is the F1 score.
- The AUC Score is already at 81%, which is great for baseline!
- Likewise, the Cross Validation score is alrady looking good as it is nearly %84

Decision Tree 2

Refining Decision Tree Through GridSearchCV

In [49]:

```
#Creating Grid Search to optimize Random Forest Parameters for Precision.
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]
}
```

In [104…

```
#running the decision tree again. This time using the parameters as determined from the grid s
dt2 = DecisionTreeClassifier(criterion='gini', max_depth=3, min_samples_split=2,
                             min_samples_leaf=1, class_weight='balanced', random_state=23)
dt2.fit(X_train_resampled, y_train_resampled)
dt2_y_pred = dt2.predict(X_test)
get_metrics(dt2, dt2_y_pred)
```

```
Precision is :80.24316109422493
F1 Score is :81.199538638985
ROC AUC is :0.84
Cross Validation Score is :0.862
```

In [117…

```
#Running the decision tree again. This time using the parameters as determined from the gried
```

```
dt2 = DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_split=2, min_sample
dt2.fit(X_train_resampled,y_train_resampled)
dt2_y_pred = dt2.predict(X_test)
get_metrics(dt2,dt2_y_pred)
```

```
Precision is :82.92682926829268
F1 Score is :81.11332007952285
ROC AUC is :0.84
Cross Validation Score is :0.863
```

In [121…

```
dt2_y_pred = dt2.predict(X_train_resampled)
print(classification_report(y_train_resampled,y_pred_train))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.97      0.96      5539
         1.0       0.97      0.96      0.96      5539

    accuracy                           0.96     11078
   macro avg       0.96      0.96      0.96     11078
weighted avg       0.96      0.96      0.96     11078
```

In [119…

```
print(classification_report(y_test, dt2_y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.89      0.87      1846
         1.0       0.83      0.79      0.81      1285

    accuracy                           0.85      3131
   macro avg       0.84      0.84      0.84      3131
weighted avg       0.85      0.85      0.85      3131
```

Analysis:

- All scores are improved!
- Everything is in the 80s which is excellent!
- I would be happy with this as a final model, but want to see if I can further improve my results with ensemble methods or gradient boosting.

Random Forests

In [122…

```
#Creating a Random Forests Classifier.
rf1_clf = RandomForestClassifier(random_state=23, class_weight='balanced')
rf1_clf.fit(X_train_resampled, y_train_resampled)
rf1_t_pred = rf1_clf.predict(X_test)
get_metrics(rf1_clf, rf1_t_pred)
```

```
Precision is :80.72196620583718
F1 Score is :81.25241592578276
ROC AUC is :0.84
```

```
Cross Validation Score is :0.875
```

In [129…

```
rf1_t_pred = rf1_clf.predict(X_train_resampled)
print(classification_report(y_train_resampled,y_pred_train))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.97      0.96      5539
         1.0       0.97      0.96      0.96      5539

    accuracy                           0.96     11078
   macro avg       0.96      0.96      0.96     11078
weighted avg       0.96      0.96      0.96     11078
```

In [124…

```
print(classification_report(y_test, rf1_t_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.86      0.87      1846
         1.0       0.81      0.82      0.81      1285

    accuracy                           0.85      3131
   macro avg       0.84      0.84      0.84      3131
weighted avg       0.85      0.85      0.85      3131
```

Analysis:

- This is an improvement over my baseline model. But it still isn't as good as my Optimized Decision Tree.

Random Forests 2

GridSearch CV

In [53]:

```
#Creating Grid Search to optimize Random Forest Parameters for Precision.
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]
}
```

In [54]:

```
#Running GridSearch on my Random Forests Model to get the optimal parameters.
rf2_clf = RandomForestClassifier(random_state=23)


rf1_grid_search= GridSearchCV(rf2_clf, rf_param_grid, scoring = 'precision', cv=3)
rf1_grid_search.fit(X_train_resampled, y_train_resampled)

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

```
Random Forest  Optimal Parameters: {'criterion': 'entropy', 'max_depth': None, 'min_samples_le
af': 3, 'min_samples_split': 10, 'n_estimators': 100}
```

In [131…

```
#running Model again with parameters set to the values from the grid search.
rf2_clf = RandomForestClassifier(criterion= 'entropy', max_depth= 10, min_samples_leaf= 3,
                                 min_samples_split= 10, n_estimators= 100, random_state=23,
                                 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)
```

```
Precision is :79.24393723252496
F1 Score is :82.69445478228509
ROC AUC is :0.85
Cross Validation Score is :0.883
```

In [132…

```
rf2_t_pred = rf2_clf.predict(X_train_resampled)
print(classification_report(y_train_resampled,y_pred_train))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.97      0.96      5539
         1.0       0.97      0.96      0.96      5539

    accuracy                           0.96     11078
   macro avg       0.96      0.96      0.96     11078
weighted avg       0.96      0.96      0.96     11078
```

In [130…

```
print(classification_report(y_test, rf2_y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.90      0.84      0.87      1846
         1.0       0.79      0.86      0.83      1285

    accuracy                           0.85      3131
   macro avg       0.85      0.85      0.85      3131
weighted avg       0.86      0.85      0.85      3131
```

Analysis:

- This model is better than Baseline Decision Tree and Baseline Random Forests.
- It has a slightly better F1 Score and Cross Validation score than Decision Tree 2, however Decision Tree 2 still has a higher precision score, which is my main metric for determining my final model.

Gradient Boosting (XGBoost) Model

In [133…

```
# Instantiate XGBClassifier to start a gradient boosting model
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)
print(classification_report(y_test, xg1_y_pred))
```

```
Precision is :80.54298642533936
F1 Score is :81.80773649942552
ROC AUC is :0.85
Cross Validation Score is :0.884
              precision    recall  f1-score   support

         0.0       0.88      0.86      0.87      1846
         1.0       0.81      0.83      0.82      1285

    accuracy                           0.85      3131
   macro avg       0.84      0.85      0.84      3131
weighted avg       0.85      0.85      0.85      3131
```

In [138…

```
xg1_y_pred = clf.predict(X_train_resampled)
print(classification_report(y_train_resampled,xg1_y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.93      0.91      0.92      5539
         1.0       0.91      0.93      0.92      5539

    accuracy                           0.92     11078
   macro avg       0.92      0.92      0.92     11078
weighted avg       0.92      0.92      0.92     11078
```

In [ ]:

```
```

GridSearch

In [57]:

```
# setting up grid search
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],
}
```

XGBoost 2

In [58]:

```
#running Gridsearch on Gradient Boosted model to find optimal parameters
xg2 = XGBClassifier(random_state=23)

grid_clf = GridSearchCV(xg2, boost_param_grid, scoring='precision', cv=3, n_jobs=1)
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]))
```

```
Grid Search found the following optimal parameters:
learning_rate: 0.1
max_depth: 6
min_child_weight: 1
n_estimators: 100
subsample: 0.5
```

In [110…

```
xg2 = XGBClassifier(learning_rate= 0.1, max_depth=6, min_child_weight=1,
                                n_estimators=100, subsample=0.5, random_state=23)
xg2.fit(X_train_resampled, y_train_resampled)
xg2_y_pred = xg2.predict(X_test)
get_metrics(xg2, xg2_y_pred)
print(classification_report(y_test, xg2_y_pred))
```

```
Precision is :80.16224188790561
F1 Score is :82.31730405149564
ROC AUC is :0.85
Cross Validation Score is :0.885
              precision    recall  f1-score   support

         0.0       0.89      0.85      0.87      1846
         1.0       0.80      0.85      0.82      1285

    accuracy                           0.85      3131
   macro avg       0.85      0.85      0.85      3131
weighted avg       0.85      0.85      0.85      3131
```

In [139…

```
xg2_y_pred = xg2.predict(X_train_resampled)
print(classification_report(y_train_resampled,xg2_y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.89      0.91      5539
         1.0       0.90      0.92      0.91      5539

    accuracy                           0.91     11078
   macro avg       0.91      0.91      0.91     11078
weighted avg       0.91      0.91      0.91     11078
```

Analysis:

- Improves all scores from both baseline decision tree as well as baseline boosted model.
- Scores all are comparable to the Optimized Decision Tree.

Model Selection

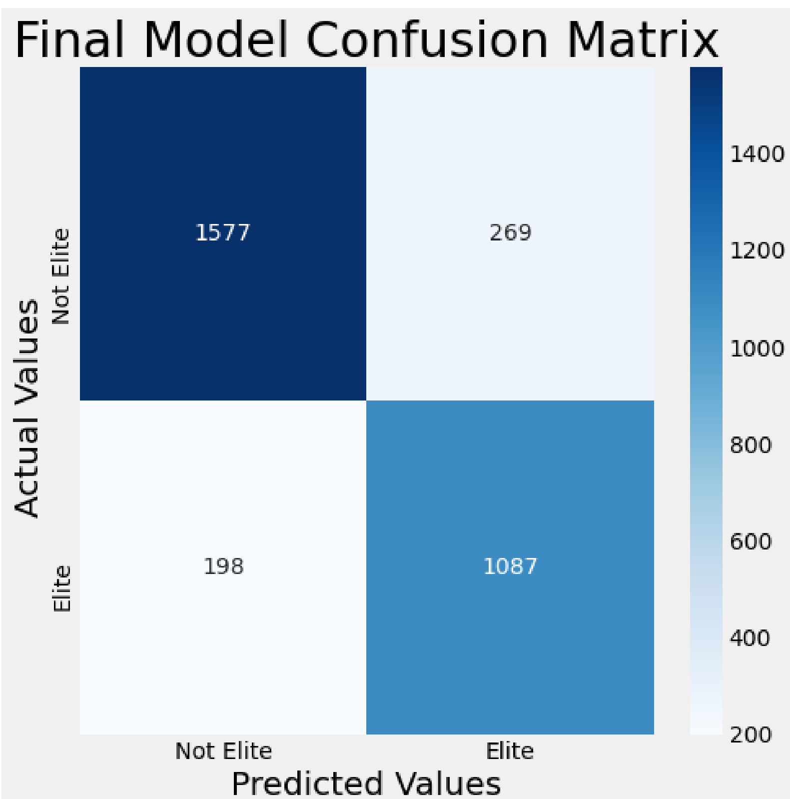
Final Model: Optimized Decision Tree (Decision Tree 2)

- The advantages of the Optimized Boosted Model (F1 Score and Cross Validation) aren't significant enough to cancel out the slight edge that the Optimized Decision Tree has in Precision.

Final Model Confusion Matrix

In [61]:

```
#checking confusino matrix to visualize my final model's predictions
dt2_matrix = confusion_matrix(y_test, xg2_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()
```



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. (50% chance of getting it correct)
  - The Final Model is also a 6% 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.

Final Model Plot

In [62]:

```
#getting the column names
X.columns
```

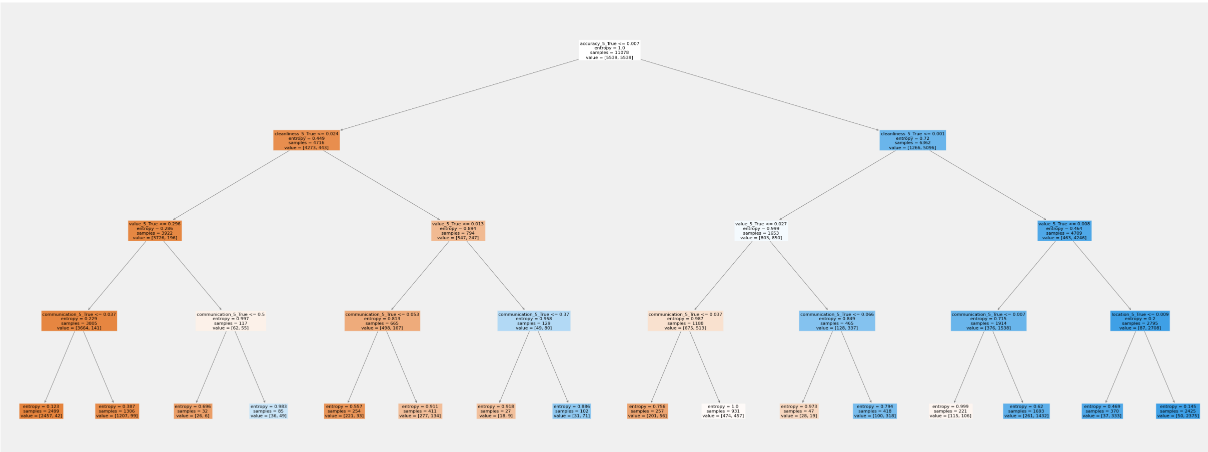
Out[62]:

```
Index(['price_280+_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+_True',
       'calculated_host_listings_count_1', 'calculated_host_listings_count_2',
       'calculated_host_listings_count_3', 'calculated_host_listings_count_4',
       'calculated_host_listings_count_5', 'calculated_host_listings_count_6',
       'calculated_host_listings_count_7', 'calculated_host_listings_count_8',
       'calculated_host_listings_count_9', 'calculated_host_listings_count_10',
       'calculated_host_listings_count_11',
       'calculated_host_listings_count_12',
       'calculated_host_listings_count_13',
       'calculated_host_listings_count_14',
       'calculated_host_listings_count_15',
       'calculated_host_listings_count_16',
       'calculated_host_listings_count_17',
       'calculated_host_listings_count_18',
       'calculated_host_listings_count_19',
       'calculated_host_listings_count_20',
       'calculated_host_listings_count_21',
       'calculated_host_listings_count_22',
       'calculated_host_listings_count_23',
       'calculated_host_listings_count_24',
       'calculated_host_listings_count_26',
       'calculated_host_listings_count_27',
       'calculated_host_listings_count_28',
       'calculated_host_listings_count_29',
       'calculated_host_listings_count_30',
       'calculated_host_listings_count_31',
       'calculated_host_listings_count_32',
       'calculated_host_listings_count_33',
       'calculated_host_listings_count_35',
       'calculated_host_listings_count_36',
       'calculated_host_listings_count_37',
       'calculated_host_listings_count_38',
       'calculated_host_listings_count_42',
       'calculated_host_listings_count_43',
       'calculated_host_listings_count_46',
       'calculated_host_listings_count_48',
       'calculated_host_listings_count_55',
       'calculated_host_listings_count_56',
       'calculated_host_listings_count_57',
       'calculated_host_listings_count_63',
       'calculated_host_listings_count_66',
       'calculated_host_listings_count_69',
       'calculated_host_listings_count_70',
       'calculated_host_listings_count_72',
       'calculated_host_listings_count_74',
       'calculated_host_listings_count_79',
```

```
       'calculated_host_listings_count_90',
       'calculated_host_listings_count_131',
       'calculated_host_listings_count_146',
       'calculated_host_listings_count_213', 'superhost_True',
       'host_response_100_True', 'response_within_hour_True'],
      dtype='object')
```

In [63]:

```python
#plotting a visualization of the decision tree classification process
fig = plt.figure(figsize=(50,20))
tree.plot_tree(dt2,
              feature_names=X.columns,
              filled=True);
```



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 162 records from the test set of 2,352 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)

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.

Feature Importance

- Finding out which features had the most impact on the classification of Elite Units.

In [65]:

```python
#getting a list of all feature names
```

```
feature_names = list(X)

#getting an array with the importance level of each feature
dt2_importance = dt2.feature_importances_
```

In [66]:

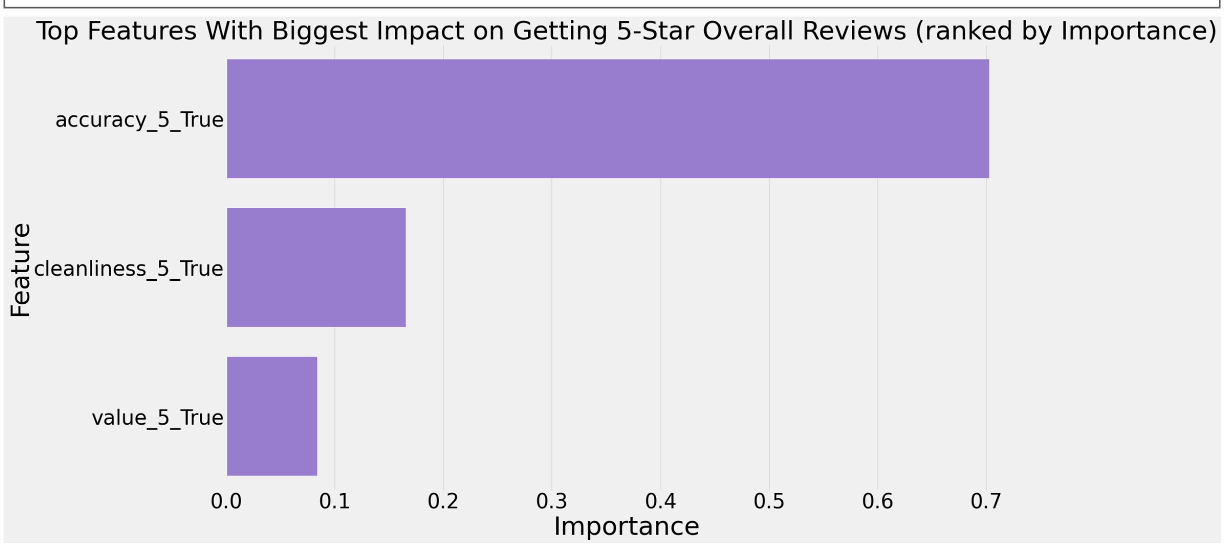
```
#Turning feature names and importances into a dataframe for analysis
feature_importance_df = pd.DataFrame(dt2_importance, feature_names)
feature_importance_df= feature_importance_df.reset_index()
feature_importance_df.rename(columns={'index': 'Feature', 0: 'Importance'}, inplace=True)
feature_importance_df = feature_importance_df.sort_values('Importance', ascending=False)
feature_importance_df.head(4)
```

Out[66]:

	Feature	Importance
1	accuracy_5_True	0.702824
2	cleanliness_5_True	0.165569
5	value_5_True	0.084057
6	communication_5_True	0.042073

In [67]:

```
# plot feature importance
fig, ax = plt.subplots(figsize=(25,15))
p = sns.barplot(data=feature_importance_df.head(3), x='Importance', y='Feature', color ='mediu
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 Reviews (ranked by Imp
            fontsize = 50)


plt.show();
```



Analysis:

- Accuracy is by far the most important feature
- It is 7 Times more important than the next features.
- Cleanliness and Value are also important, but not nearly as much as Accuracy

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.

Analysis of Top Features

Review Metric DF (or Feature Analysis DF)

In [68]:

```
#creating a copy of the dataframe for feature analysis
feature_analysis_df = df.copy()
```

In [69]:

```
#dropping redundant and/or unneccesary features
feature_analysis_df.drop(['review_scores_rating', 'review_scores_cleanliness', 'review_scores_
                    'review_scores_accuracy', 'review_scores_communication', 'review_scores_lo
                    'review_scores_checkin', 'accommodates', 'price', 'calculated_host_listing
                    'availability_30', 'availability_90', 'availability_365', 'host_response_r
                    'host_id', 'bedrooms', 'beds', 'availability_30_rate', 'availability_90_ra
                     'host_is_superhost', 'amenities', 'property_type',
                    ], axis=1, inplace=True)
```

In [70]:

```
#changing display option so that it displays floats to 2 decimal places.
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

Function get_stats( )

In [71]:

```
def get_stats(df):

    """Takes the wide-form output of a groupby operation and transposes it as a long-form tabl
    also adding a column "delta" which calculates the difference between the True value and th
    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)
```

Top Features

In [72]:

```
#creating a variation of analysis_df sorted by calculated_host_listings_count, to make it easi
host_listings = analysis_df.sort_values('calculated_host_listings_count', ascending=True)
```

In [73]:

```
#creating another variation that splits into two datasets. Superhosts and non-Superhosts.
superhost_df = host_listings[host_listings['superhost'] == True]
not_superhost_df = host_listings[host_listings['superhost'] == False]
```

In [74]:

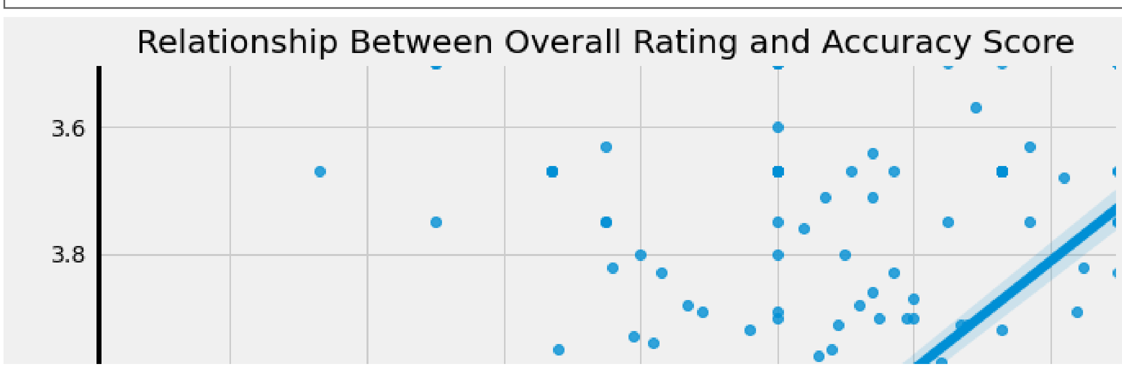
```
#creating another variation that splits into two datasets. Elite units and non-Elite units.
elite_df = host_listings[host_listings['elite'] == True]
not_elite_df = host_listings[host_listings['elite'] == False]
```

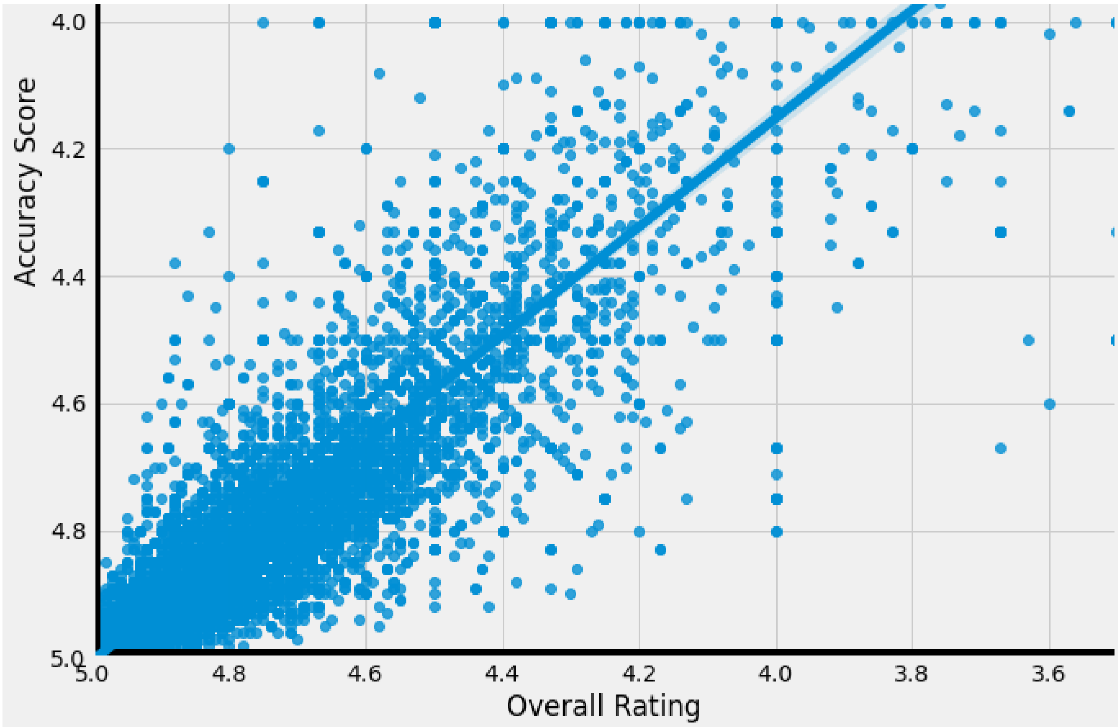
Top Feature #1: Accuracy

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

In [75]:

```
#creating a scatterplot to analyze the relationship between Accuracy Score and Overall Rating
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);
```





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 [76]:

```
#calling stats on accuracy scores.
df['review_scores_accuracy'].describe()
```

Out[76]:

```
count    12502.000
mean         4.797
std          0.359
min          1.000
25%          4.750
50%          4.900
75%          5.000
max          5.000
Name: review_scores_accuracy, dtype: float64
```

Analysis:

- While the mean is 4.79, the median is 4.9, which means that the split between elite accuracy and non-elite accuracy should be near 50%

In [77]:

```
#checking to verify my analysis
df['accuracy_5'].value_counts()
```

Out[77]:

```
True     6324
False    6199
Name: accuracy_5, dtype: int64
```

In [78]:

```
accuracy_metrics = feature_analysis_df.groupby('accuracy_5').mean()
accuracy_stats = get_stats(accuracy_metrics)
accuracy_stats
```

Out[78]:

accuracy_5	Metric	False	True	delta
8	elite	0.082	0.732	0.651
11	cleanliness_5	0.161	0.700	0.539
15	communication_5	0.399	0.883	0.483
14	value_5	0.050	0.480	0.431
12	checkin_5	0.460	0.891	0.430
13	location_5	0.384	0.749	0.365
7	superhost	0.305	0.525	0.220
6	host_response_100	0.663	0.775	0.111
5	bookings_above_avg	0.326	0.433	0.107
3	booked_rate_30	0.547	0.629	0.082
4	booked_rate_90	0.411	0.493	0.082
16	price_280+	0.282	0.319	0.037
2	bedrooms_2+	0.510	0.480	-0.030
9	entire_home	0.856	0.822	-0.034
10	response_within_hour	0.797	0.749	-0.048
1	capacity_5+	0.460	0.389	-0.071
0	instant_bookable	0.572	0.420	-0.152

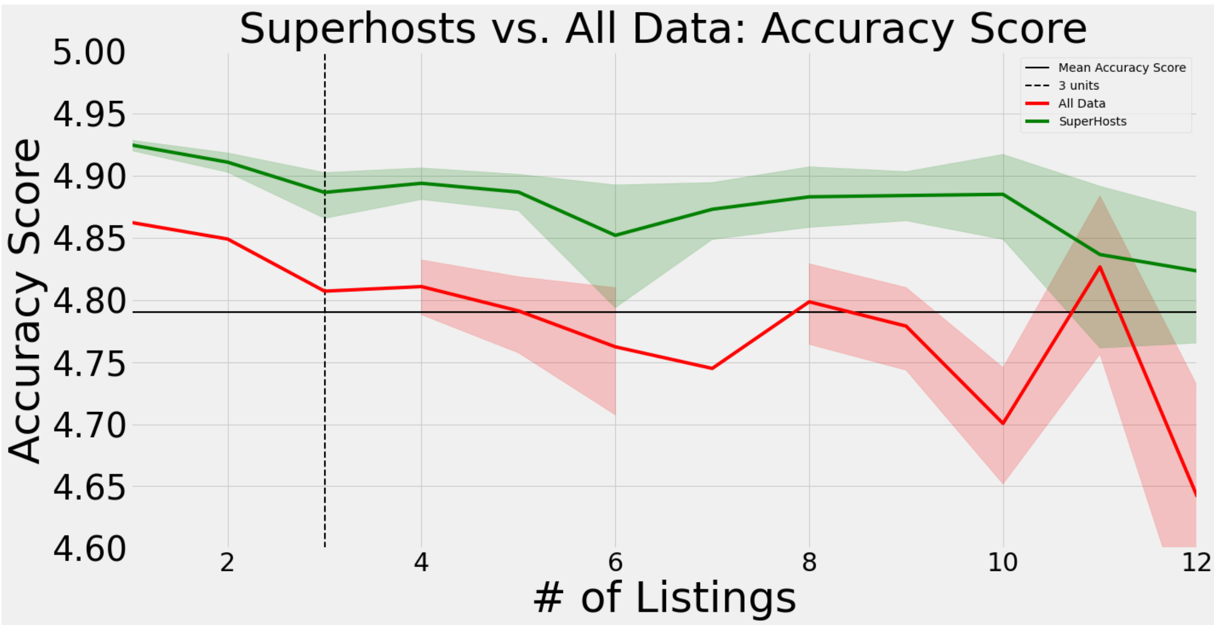
Analysis:

- 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 40% of units with 5.0 accuracy do each.

In [79]:

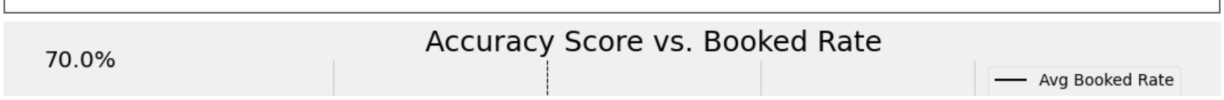
```
fig, ax = plt.subplots(figsize=(20,10))

ax.axhline(4.79, color='black', linewidth=(2), label='Mean Accuracy Score')
#ax.axvline(12, ls='--', color='black', linewidth=(2), label='12 units')
ax.axvline(3, ls='--', color='black', linewidth=(2), label='3 units')


ax.set_xlim(1, 12)
ax.set_ylim(4.6, 5.0)

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

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

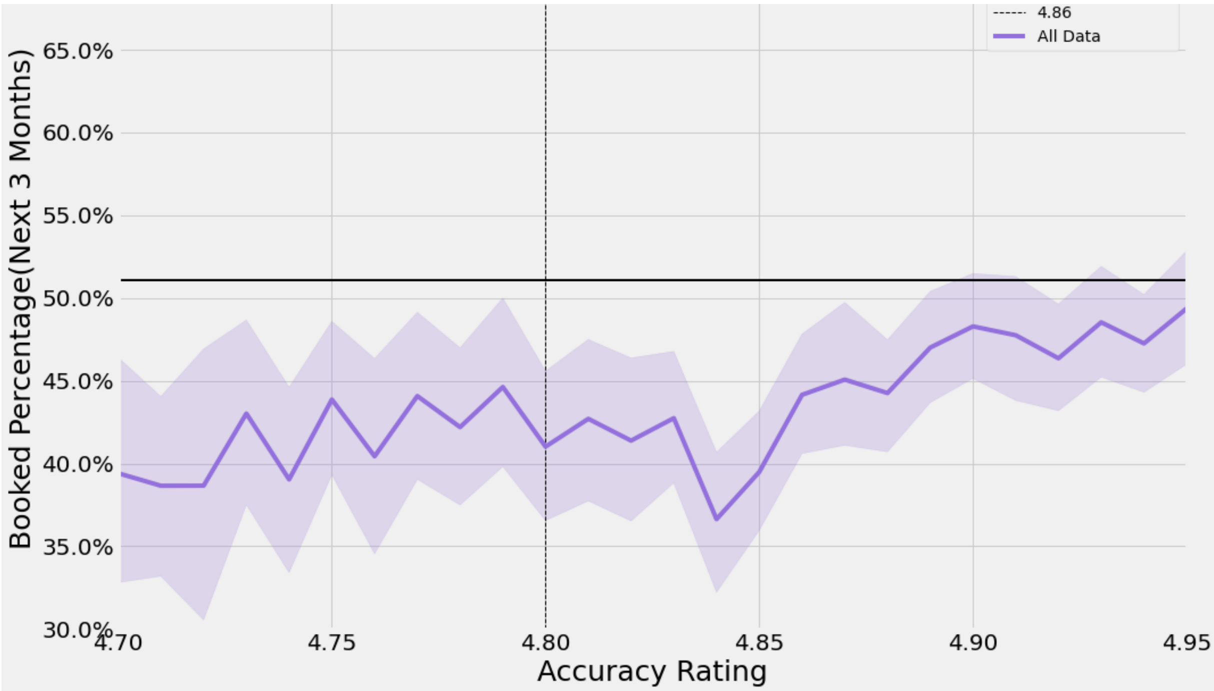
```
p.set_ylabel("Accuracy Score", fontsize = 50)

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


p.set_title("Superhosts vs. All Data: Accuracy Score", fontsize = 50)

plt.show();
```



In [80]:

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

ax.axhline(.511,  color='black', linewidth=(2), label='Avg Booked Rate')
ax.axvline(4.8 , ls='--', color='black', linewidth=(1), label='4.86')


p = sns.lineplot(data=host_listings, x='review_scores_accuracy', y='booked_rate_90',
                color ='mediumpurple', label='All Data' );


p.set_xlim(4.7,4.95)
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=None, symbol='%', is_late

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


plt.show();
```





Analysis:

- There is a positive correlation between Accuracy Rating and Booked Rate.
- Starting at 4.8 Average Accuracy, units consistantly book at a higher rate that all other units.
- Also much more certainty in the values as they stay closer to the mean.

Top Feature #2: Cleanliness

In [81]:

```
df['review_scores_cleanliness'].describe()
```

Out[81]:

```
count    12502.000
mean         4.756
std          0.380
min          1.000
25%          4.690
50%          4.860
75%          4.980
max          5.000
Name: review_scores_cleanliness, dtype: float64
```

In [82]:

```
cleanliness_metrics = feature_analysis_df.groupby('cleanliness_5').mean()
cleanliness_stats = get_stats(cleanliness_metrics)
cleanliness_stats
```

Out[82]:

cleanliness_5	Metric	False	True	delta
8	elite	0.153	0.747	0.595
11	accuracy_5	0.267	0.816	0.548
14	value_5	0.094	0.494	0.401
15	communication_5	0.472	0.868	0.397
12	checkin_5	0.532	0.869	0.337

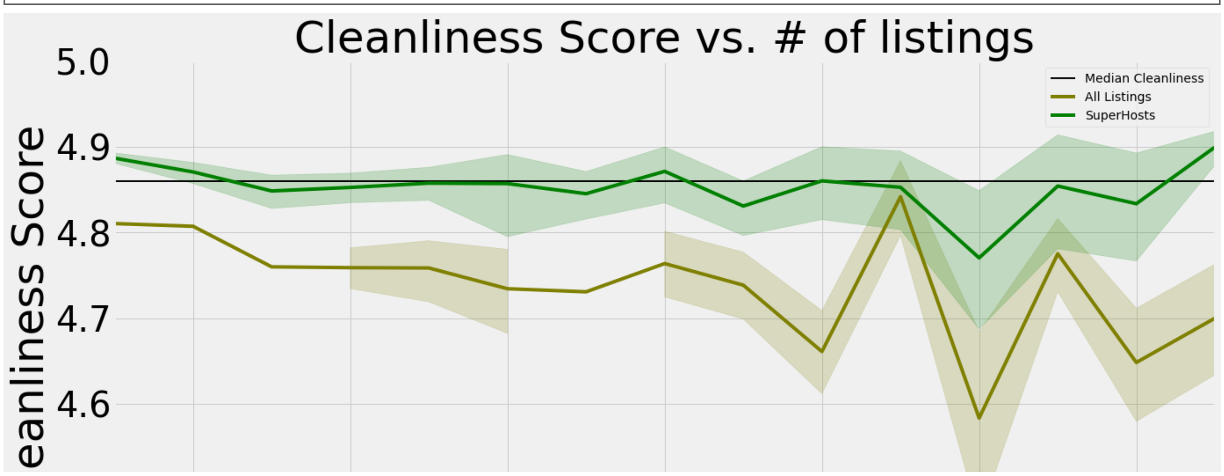
13	location_5	0.451	0.722	0.271
7	superhost	0.340	0.514	0.174
6	host_response_100	0.675	0.778	0.103
5	bookings_above_avg	0.354	0.413	0.059
4	booked_rate_90	0.434	0.477	0.043
3	booked_rate_30	0.570	0.613	0.043
16	price_280+	0.294	0.310	0.016
9	entire_home	0.851	0.823	-0.027
10	response_within_hour	0.785	0.756	-0.029
2	bedrooms_2+	0.512	0.471	-0.041
1	capacity_5+	0.456	0.383	-0.074
0	instant_bookable	0.553	0.420	-0.133

Analysis:

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

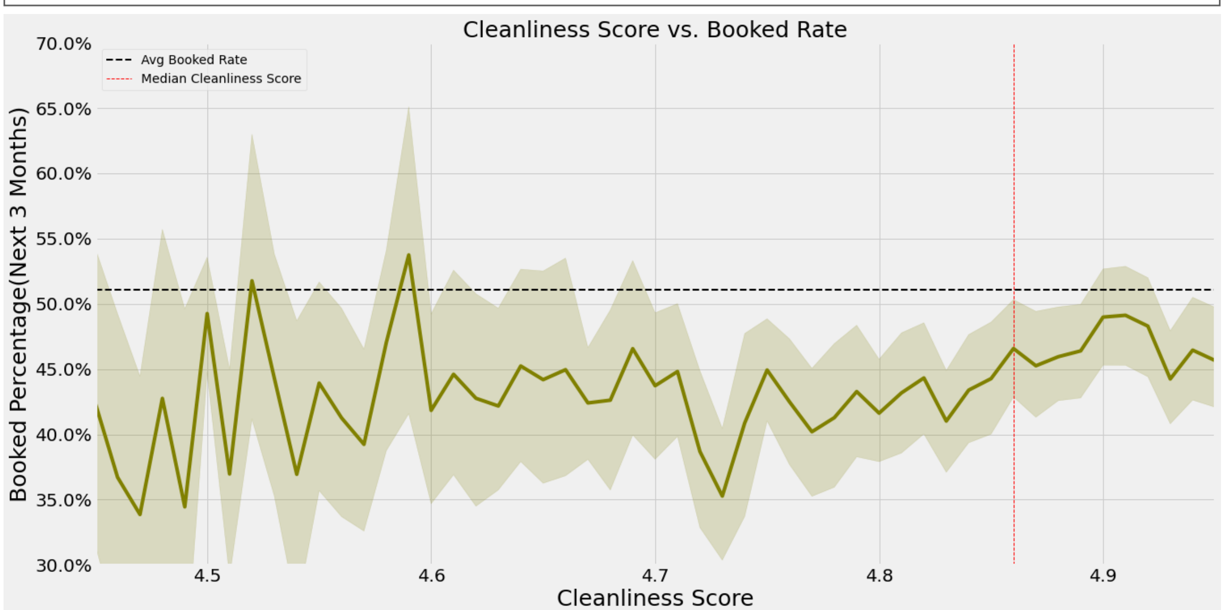
In [83]:

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

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

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

p = sns.lineplot(data=superhost_df, x='calculated_host_listings_count', y='review_scores_clean
                 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();
```



In [84]:

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

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


p = sns.lineplot(data=host_listings, x='review_scores_cleanliness', y='booked_rate_90',
                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=None, symbol='%', is_late

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


plt.show();
```



Top Feature #3: Value

In [85]:

```
df['review_scores_value'].describe()
```

Out[85]:

```
count    12500.000
mean         4.691
```

```
std          0.390
min          1.000
25%          4.600
50%          4.780
75%          4.900
max          5.000
Name: review_scores_value, dtype: float64
```

In [86]:

```
value_metrics = feature_analysis_df.groupby('value_5').mean()
value_stats = get_stats(value_metrics)
value_stats
```

Out[86]:

value_5	Metric	False	True	delta
8	elite	0.244	0.868	0.624
11	accuracy_5	0.358	0.908	0.550
12	cleanliness_5	0.299	0.801	0.503
15	communication_5	0.538	0.932	0.393
14	location_5	0.473	0.830	0.357
13	checkin_5	0.590	0.918	0.328
5	bookings_above_avg	0.343	0.480	0.137
4	booked_rate_90	0.426	0.525	0.098
3	booked_rate_30	0.567	0.648	0.081
6	host_response_100	0.705	0.760	0.055
7	superhost	0.404	0.448	0.043
16	price_280+	0.309	0.278	-0.031
2	bedrooms_2+	0.517	0.434	-0.083
10	response_within_hour	0.797	0.705	-0.092
9	entire_home	0.867	0.762	-0.105
1	capacity_5+	0.452	0.348	-0.105
0	instant_bookable	0.531	0.398	-0.133

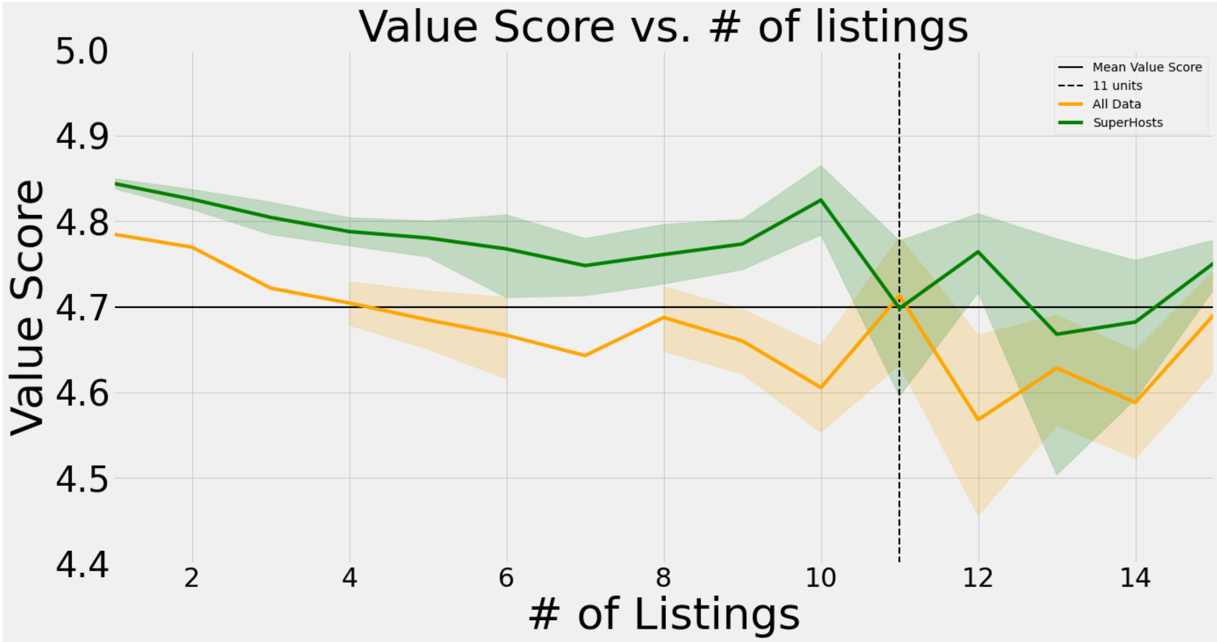
Analysis:

- 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 [87]:

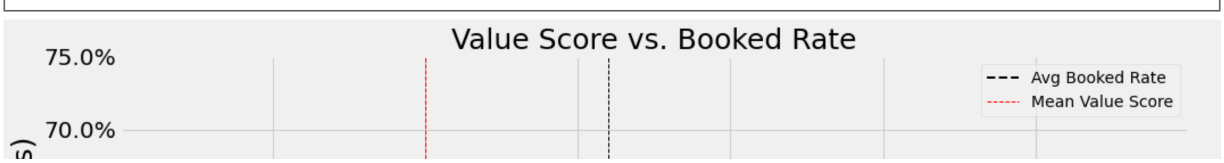
```
fig, ax = plt.subplots(figsize=(20,10))

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', y='review_scores_valu
                  color ='orange' , label='All Data');

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

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

p.set_xlabel("# of Listings", fontsize = 50)
```

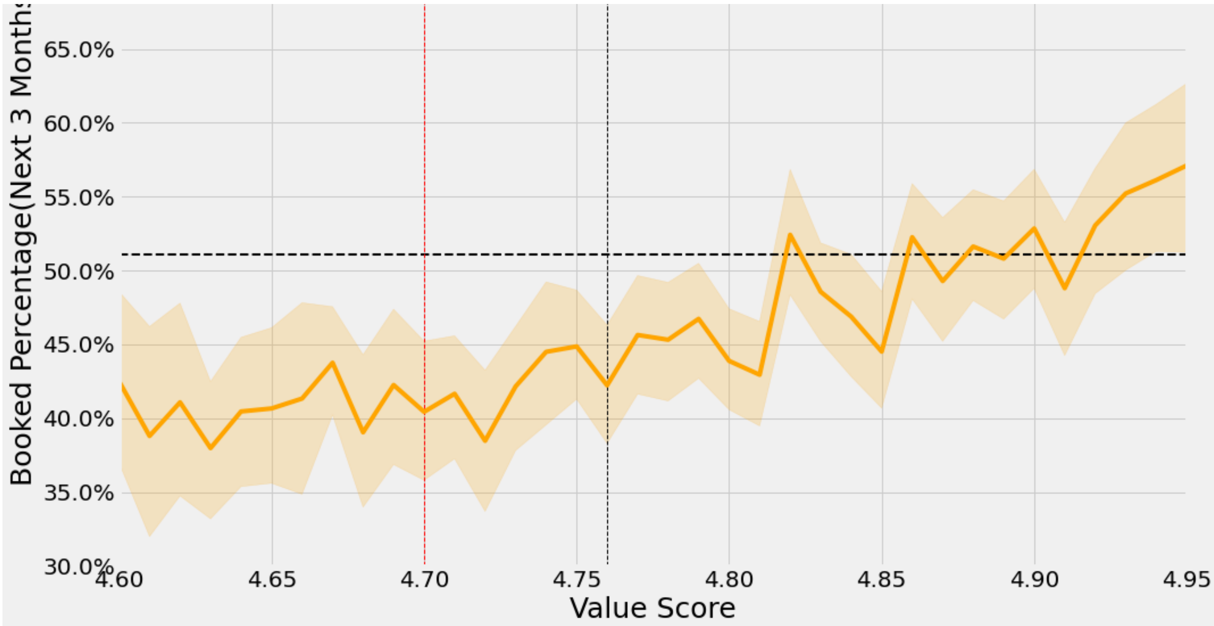
```
plt.xticks(fontsize=30)
plt.yticks(fontsize=40)


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

plt.show();
```



In [88]:

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

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


p = sns.lineplot(data=host_listings, x='review_scores_value', y='booked_rate_90',
                 color ='orange')


p.set_xlim(4.6,4.95)
p.set_ylim(.3, .75)

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=None, symbol='%', is_late

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


plt.show();
```





- Higher Value Scores translate into higher booked rates.
- There is no real difference 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 slightly after 4.7.
- Value increases booking rate more than Accuracy does.

Top Feature #4: Communication

In [89]:

```
df['review_scores_communication'].describe()
```

Out[89]:

```
count    12502.000
mean         4.844
std          0.342
min          1.000
25%          4.830
50%          4.950
75%          5.000
max          5.000
Name: review_scores_communication, dtype: float64
```

In [90]:

```
communication_metrics = feature_analysis_df.groupby('communication_5').mean()
communication_stats = get_stats(communication_metrics)
#communication_stats.sort_values(True, ascending=False)
communication_stats
```

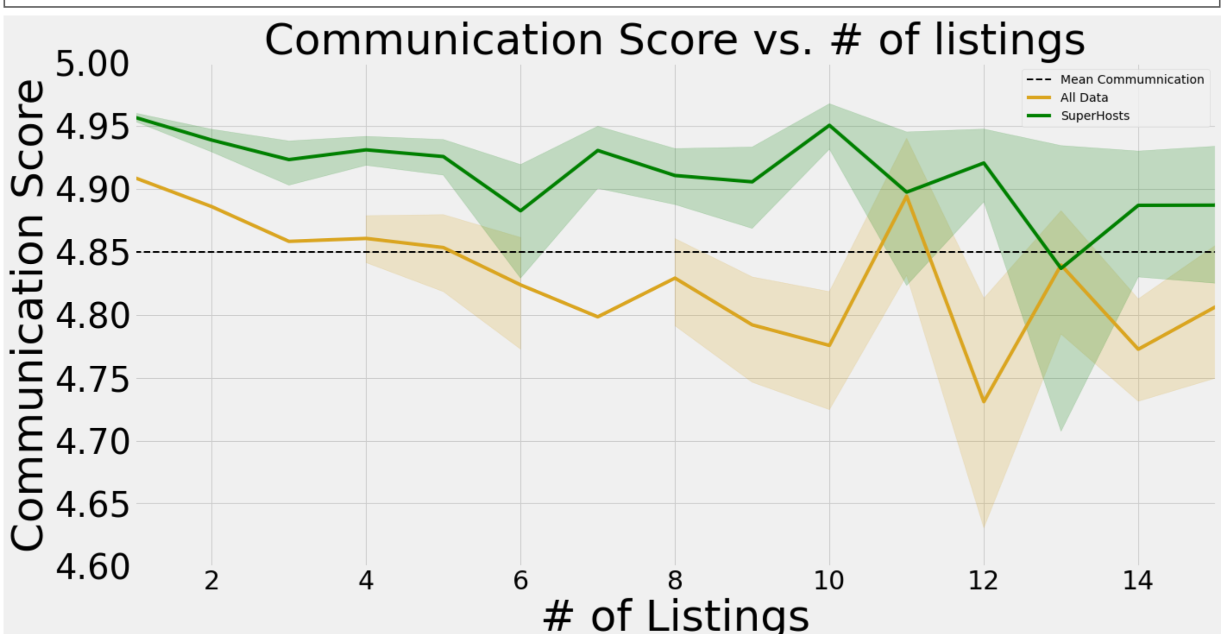
Out[90]:

communication_5	Metric	False	True	delta
13	checkin_5	0.318	0.877	0.560
11	accuracy_5	0.166	0.693	0.527
8	elite	0.083	0.592	0.509
12	cleanliness_5	0.160	0.585	0.425
15	value_5	0.051	0.387	0.336

14	location_5	0.364	0.682	0.318
7	superhost	0.223	0.522	0.299
6	host_response_100	0.595	0.789	0.194
5	bookings_above_avg	0.313	0.417	0.104
3	booked_rate_30	0.531	0.620	0.089
4	booked_rate_90	0.398	0.483	0.084
16	price_280+	0.296	0.303	0.007
9	entire_home	0.845	0.835	-0.010
10	response_within_hour	0.782	0.768	-0.014
2	bedrooms_2+	0.514	0.484	-0.030
1	capacity_5+	0.463	0.403	-0.060
0	instant_bookable	0.612	0.431	-0.181

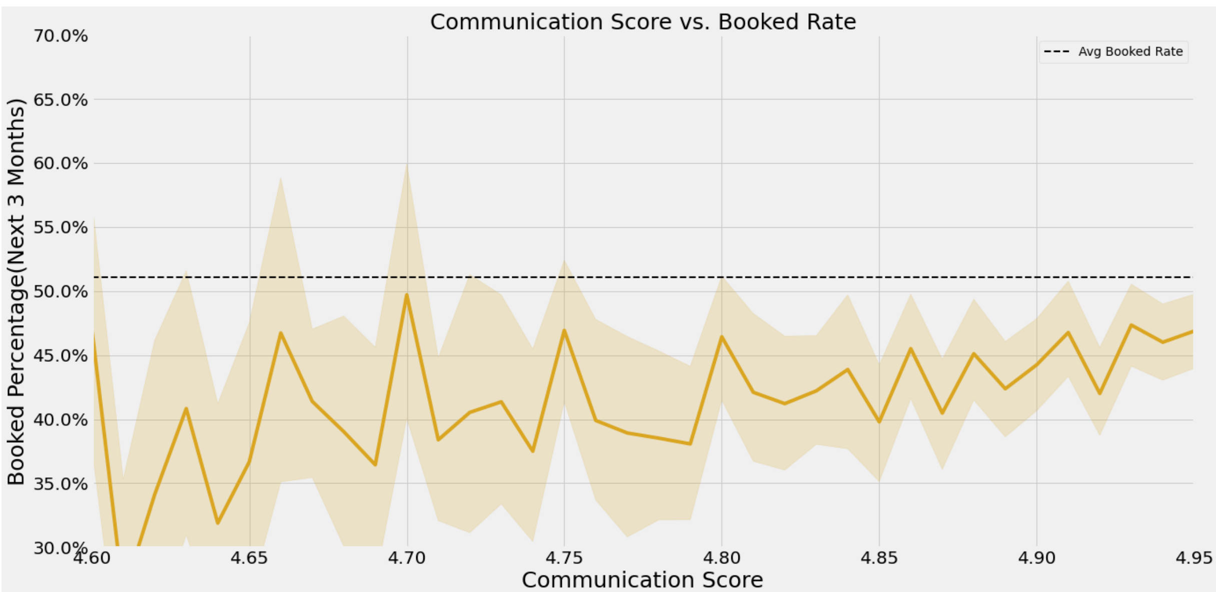
In [91]:

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

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

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

p = sns.lineplot(data=superhost_df, x='calculated_host_listings_count', y='review_scores_commu
                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();
```



In [92]:

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

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

p = sns.lineplot(data=host_listings, x='review_scores_communication', y='booked_rate_90',
                 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=None, symbol='%', is_late

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


plt.show();
```



Questions Answered

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

In [93]:

```
superhost_metrics = feature_analysis_df.groupby('superhost').mean()
superhost_stats = get_stats(superhost_metrics)
superhost_stats
```

Out[93]:

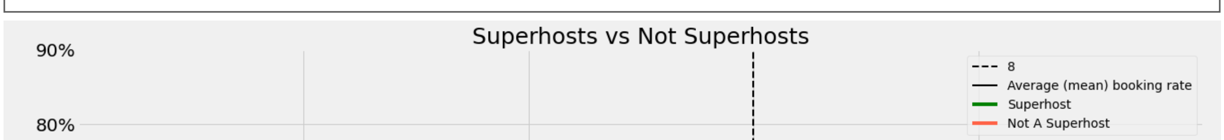
superhost	Metric	False	True	delta
15	communication_5	0.526	0.808	0.282
12	checkin_5	0.572	0.827	0.255

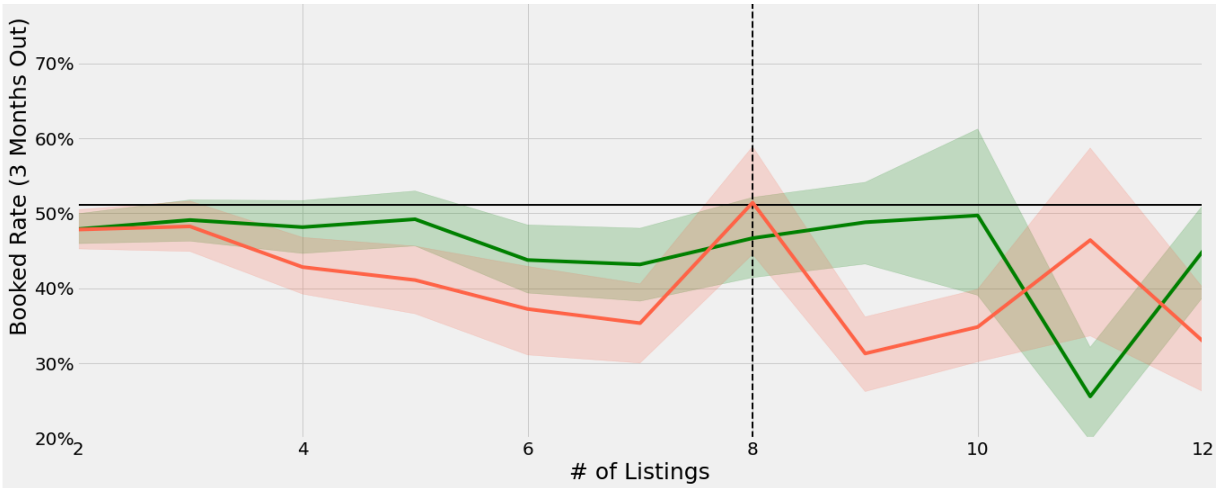
6	host_response_100	0.616	0.865	0.249
10	accuracy_5	0.411	0.637	0.226
7	elite	0.332	0.521	0.189
11	cleanliness_5	0.360	0.536	0.176
9	response_within_hour	0.710	0.861	0.151
13	location_5	0.524	0.630	0.106
3	booked_rate_30	0.560	0.628	0.068
14	value_5	0.253	0.288	0.035
4	booked_rate_90	0.438	0.472	0.034
5	bookings_above_avg	0.370	0.394	0.024
8	entire_home	0.829	0.853	0.024
16	price_280+	0.311	0.286	-0.026
2	bedrooms_2+	0.507	0.476	-0.031
1	capacity_5+	0.441	0.401	-0.040
0	instant_bookable	0.538	0.435	-0.103

- 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 [94]:

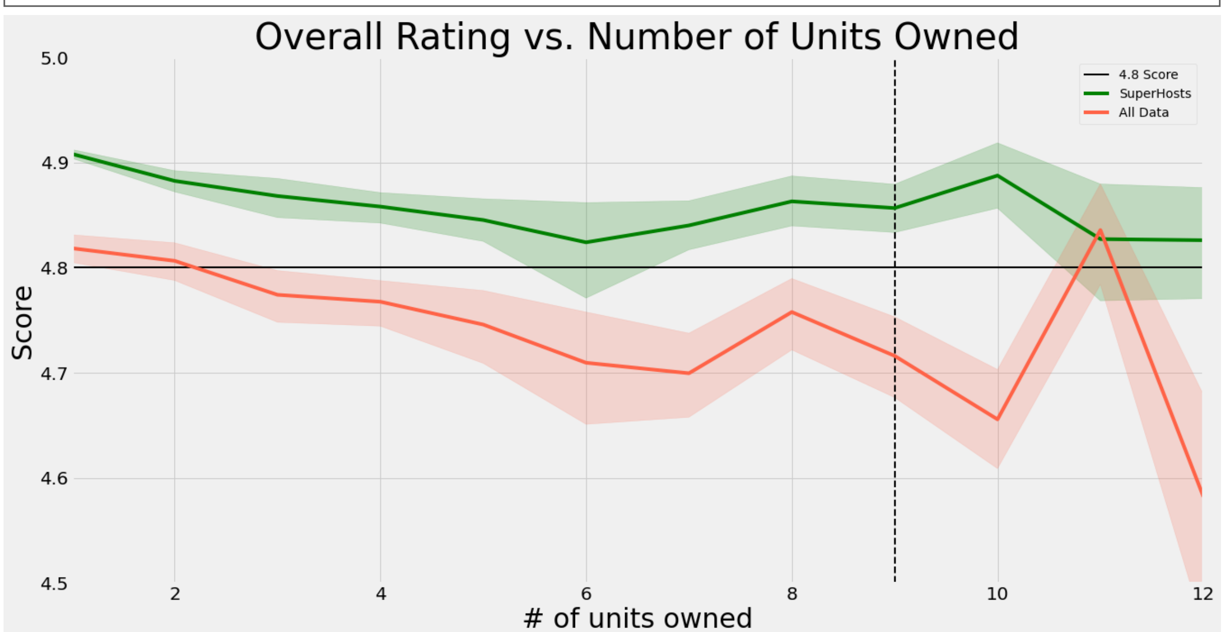
```
fig, ax = plt.subplots(figsize=(20,10))

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

p = sns.lineplot(data=superhost_df,x='calculated_host_listings_count', y='booked_rate_90',
                 color ='green', label = 'Superhost' );

p = sns.lineplot(data=not_superhost_df,x='calculated_host_listings_count', y='booked_rate_90',
                  color ='tomato', label = 'Not A Superhost' );



p.set_xlim(2,12)
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=None, symbol='%', is_late

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

plt.show();
```





In [95]:

```
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', y='review_scores_ratin
                color ='green', label='SuperHosts' );

p = sns.lineplot(data=host_listings, x='calculated_host_listings_count', y='review_scores_rati
                 color ='tomato', label='All Data' );
p.set_xlim(1,12)
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();
```



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.

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

YES!

In [96]:

```
elite_metrics = feature_analysis_df.groupby('elite').mean()
elite_stats = get_stats(elite_metrics)
elite_stats
```

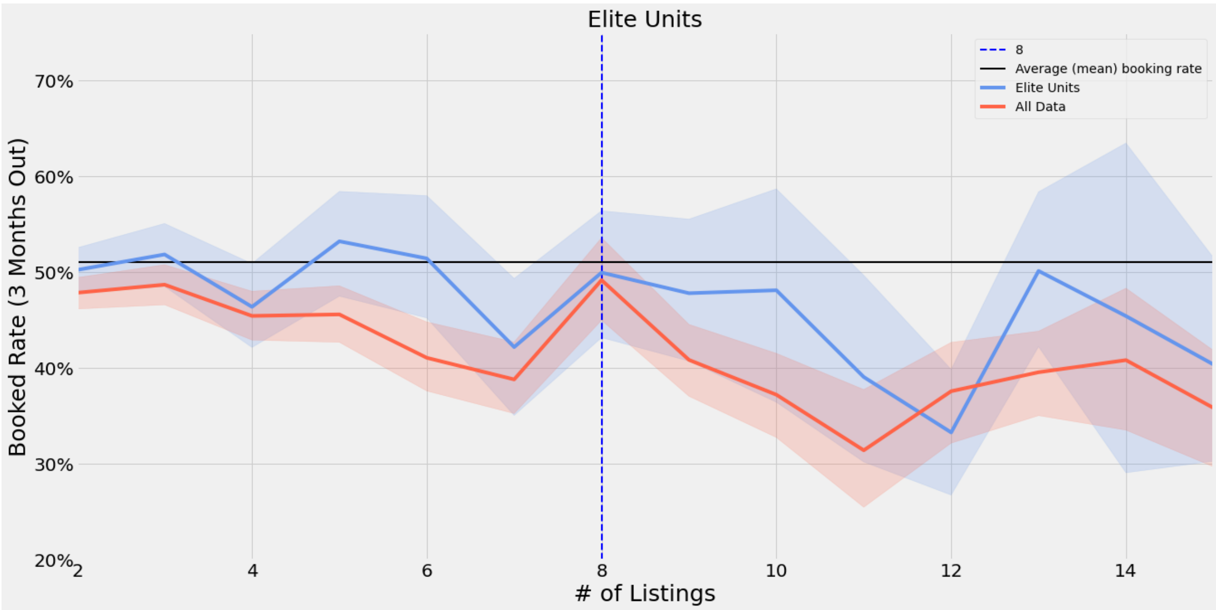
Out[96]:

elite	Metric	False	True	delta
10	accuracy_5	0.229	0.902	0.672
11	cleanliness_5	0.186	0.789	0.603
14	value_5	0.060	0.565	0.505
15	communication_5	0.445	0.928	0.483
12	checkin_5	0.512	0.916	0.403
13	location_5	0.411	0.794	0.383
7	superhost	0.338	0.528	0.190
5	bookings_above_avg	0.336	0.443	0.107
6	host_response_100	0.677	0.781	0.104
4	booked_rate_90	0.420	0.500	0.080
3	booked_rate_30	0.558	0.633	0.076
16	price_280+	0.270	0.345	0.074
2	bedrooms_2+	0.492	0.498	0.006
8	entire_home	0.851	0.822	-0.029
1	capacity_5+	0.440	0.402	-0.037
9	response_within_hour	0.798	0.736	-0.062
0	instant_bookable	0.560	0.403	-0.157

In [97]:

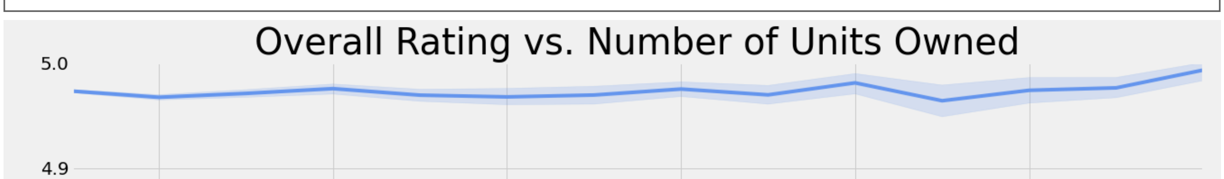
```
fig, ax = plt.subplots(figsize=(20,10))

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

p = sns.lineplot(data=elite_df,x='calculated_host_listings_count', y='booked_rate_90',
                  color ='cornflowerblue', label = 'Elite Units' );

p = sns.lineplot(data=host_listings,x='calculated_host_listings_count', y='booked_rate_90',
                 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)
```

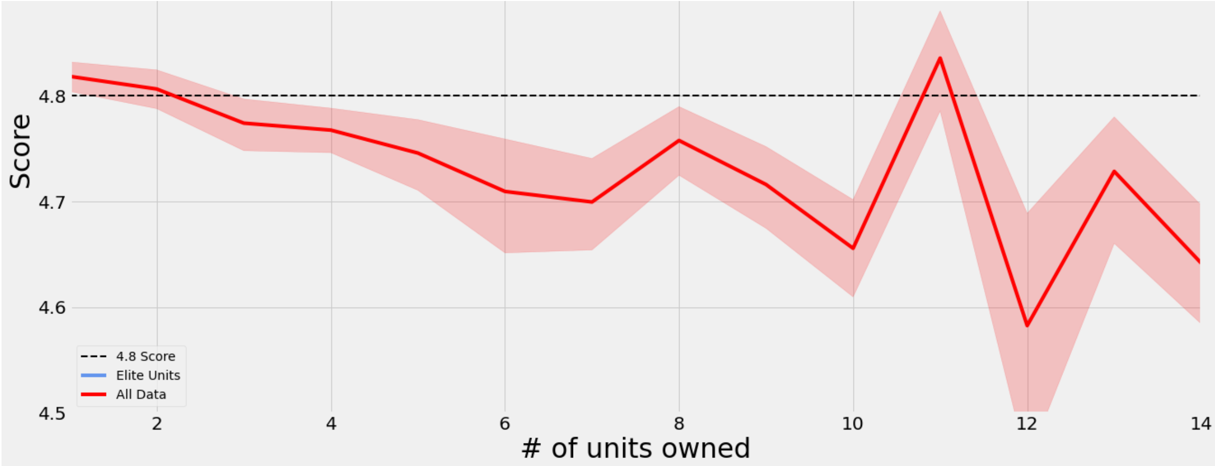
```
plt.yticks(fontsize=20)

ax.yaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=None, symbol='%', is_late

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

plt.show();
```



In [98]:

```
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='review_scores_rating',
                  color ='cornflowerblue', label='Elite Units' );

p = sns.lineplot(data=host_listings, x='calculated_host_listings_count', y='review_scores_rati
                 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();
```





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

Recommendations

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.

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 (accessed 6/21/22)

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.

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.

Further Work

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.

Increase the scope of this model.

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

In [ ]:

```

```

In [ ]:

```

```