img

Create Airbnb ¶

Author: Stephanie Ciaccia

Overview

Crete is a popular destination in Greece, known for its stunning beaches, rich history, and cuisine. The tourism industry is a major contributor to the island's economy, accounting for a significant portion of its GDP and providing employment opportunities to many local residents.

Between January and March 2023, traffic numbers from non-European airports have increased by double digits compared to the pre-pandemic period in 2019, indicating a strong recovery in the tourism industry.

Business Problem

As the 2023 high season approaches, the Greek National Tourism Organisation is collaborating with Crete's Municipal Department of Tourism to discuss the vacation rental market with new airbnb hosts and its anticipated increase in demand.

The Greek National Tourism Organisation will be sharing the results of their predictive analysis of Airbnb listings with the local housing committee. The aim of this presentation is to identify the key features associated with high ratings (4 stars and above) on Airbnb listings, to assist new entrants to the Airbnb market. The objective is to offer insights and recommendations to hosts on how to enhance their listings and ultimately improve overall guest satisfaction.

Data Understanding

In [181]:
```
import pandas as pd
import numpy as np
import math
from datetime import datetime
import datetime

import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
%matplotlib inline
import plotly as plty
from matplotlib.ticker import StrMethodFormatter

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

from sklearn.impute import MissingIndicator, SimpleImputer

from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectFromModel

from sklearn.metrics import plot_confusion_matrix, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_roc_curve

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.metrics import precision_score, recall_score, accuracy_score
from sklearn.model_selection import train_test_split, GridSearchCV,\
cross_val_score, RandomizedSearchCV

from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

from sklearn.model_selection import train_test_split, cross_validate
from sklearn.preprocessing import normalize
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import log_loss

from collections import Counter
from nltk.corpus import stopwords

import pandas as pd
from IPython.display import display
```

Function for printing long lists

In [205]:
```
def print_full(x):
    pd.set_option('display.max_rows', len(x))
    print(x)
    pd.reset_option('display.max_rows')
```

Source #1: Airbnb Listings

I pulled the March 2023 Crete Detailed Listings data from insideairbnb.com. This dataset included over 30,000 listings from the last quarter. Inside Airbnb is a mission driven project that provides data and advocacy about Airbnb's impact on residential communities.

http://insideairbnb.com/ (http://insideairbnb.com/)

The dataset includes over 30,000 entries and 74 rows of features related to the physical airbnb properties and characteristics of the hosts and reviews. The target column, will be created from **review_scores_rating** which is the overall stars the property has.

I will need to create a new column with the target categorical variable after cleaning the data.

In [206]:
```
#importing Inside Airbnb data
df = pd.read_csv("data/listings_crete.csv")
```

In [207]:
```
df.head()
```

Out[207]:

	id	listing_url	scrape_id	last_scraped	source	name	description	neighborho
0	28970	https://www.airbnb.com/rooms/28970	20230330024735	2023-03-31	previous scrape	artists' house in the old town	This is a fully renovated stone house from 191...	its calm v ladies that a
1	27966	https://www.airbnb.com/rooms/27966	20230330024735	2023-03-30	city scrape	Heraklion-Pinelopi Apartment	For an unforgettable stay!! Just 10 minutes wa...	Ammou cute are
2	29849	https://www.airbnb.com/rooms/29849	20230330024735	2023-03-30	city scrape	Villa Kallergi - Nefeli, 6 guests	Villa Nefeli Kallergi with private pool in the...	The traditic Loutra wa
3	29130	https://www.airbnb.com/rooms/29130	20230330024735	2023-03-30	city scrape	Villa Kallergi - Athena, 12 guests	Villa Athena Kallergi is in the heart of the C...	The traditic Loutra wa
4	31789	https://www.airbnb.com/rooms/31789	20230330024735	2023-03-30	city scrape	Kissamos Windmills	The space Kissamos Windmills apart...	

5 rows × 75 columns

In [208]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23372 entries, 0 to 23371
Data columns (total 75 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   id                                            23372 non-null  int64
 1   listing_url                                   23372 non-null  object
 2   scrape_id                                     23372 non-null  int64
 3   last_scraped                                  23372 non-null  object
 4   source                                        23372 non-null  object
 5   name                                          23370 non-null  object
 6   description                                   22838 non-null  object
 7   neighborhood_overview                         13180 non-null  object
 8   picture_url                                   23372 non-null  object
 9   host_id                                       23372 non-null  int64
 10  host_url                                      23372 non-null  object
 11  host_name                                     23372 non-null  object
 12  host_since                                    23372 non-null  object
 13  host_location                                 16236 non-null  object
 14  host_about                                    11631 non-null  object
 15  host_response_time                            18171 non-null  object
 16  host_response_rate                            18171 non-null  object
 17  host_acceptance_rate                          21053 non-null  object
 18  host_is_superhost                             23339 non-null  object
 19  host_thumbnail_url                            23372 non-null  object
 20  host_picture_url                              23372 non-null  object
 21  host_neighbourhood                            2630 non-null   object
 22  host_listings_count                           23372 non-null  int64
 23  host_total_listings_count                     23372 non-null  int64
 24  host_verifications                            23372 non-null  object
 25  host_has_profile_pic                          23372 non-null  object
 26  host_identity_verified                        23372 non-null  object
 27  neighbourhood                                 13180 non-null  object
 28  neighbourhood_cleansed                        23372 non-null  object
 29  neighbourhood_group_cleansed                  0 non-null      float64
 30  latitude                                      23372 non-null  float64
 31  longitude                                     23372 non-null  float64
 32  property_type                                 23372 non-null  object
 33  room_type                                     23372 non-null  object
 34  accommodates                                  23372 non-null  int64
 35  bathrooms                                     0 non-null      float64
 36  bathrooms_text                                23330 non-null  object
 37  bedrooms                                      21907 non-null  float64
 38  beds                                          23160 non-null  float64
 39  amenities                                     23372 non-null  object
 40  price                                         23372 non-null  object
 41  minimum_nights                                23372 non-null  int64
 42  maximum_nights                                23372 non-null  int64
 43  minimum_minimum_nights                        23372 non-null  int64
 44  maximum_minimum_nights                        23372 non-null  int64
 45  minimum_maximum_nights                        23372 non-null  int64
 46  maximum_maximum_nights                        23372 non-null  int64
 47  minimum_nights_avg_ntm                        23372 non-null  float64
 48  maximum_nights_avg_ntm                        23372 non-null  float64
 49  calendar_updated                              0 non-null      float64
 50  has_availability                              23372 non-null  object
 51  availability_30                               23372 non-null  int64
 52  availability_60                               23372 non-null  int64
 53  availability_90                               23372 non-null  int64
 54  availability_365                              23372 non-null  int64
 55  calendar_last_scraped                         23372 non-null  object
 56  number_of_reviews                             23372 non-null  int64
 57  number_of_reviews_ltm                         23372 non-null  int64
 58  number_of_reviews_l30d                        23372 non-null  int64
 59  first_review                                  17250 non-null  object
 60  last_review                                   17250 non-null  object
 61  review_scores_rating                          17250 non-null  float64
```

```
 62  review_scores_accuracy                         17181 non-null  float64
 63  review_scores_cleanliness                      17181 non-null  float64
 64  review_scores_checkin                          17180 non-null  float64
 65  review_scores_communication                    17181 non-null  float64
 66  review_scores_location                         17180 non-null  float64
 67  review_scores_value                            17180 non-null  float64
 68  license                                        1293 non-null   object
 69  instant_bookable                               23372 non-null  object
 70  calculated_host_listings_count                 23372 non-null  int64
 71  calculated_host_listings_count_entire_homes    23372 non-null  int64
 72  calculated_host_listings_count_private_rooms   23372 non-null  int64
 73  calculated_host_listings_count_shared_rooms    23372 non-null  int64
 74  reviews_per_month                              17250 non-null  float64
dtypes: float64(17), int64(23), object(35)
memory usage: 13.4+ MB
```

In [209]:
```
df.describe()
```

Out[209]:

	id	scrape_id	host_id	host_listings_count	host_total_listings_count	neighbourhood_group_clea
count	2.337200e+04	2.337200e+04	2.337200e+04	23372.000000	23372.000000	
mean	1.806035e+17	2.023033e+13	1.820018e+08	41.852687	69.361287	
std	3.069823e+17	0.000000e+00	1.550618e+08	126.775145	231.480183	
min	2.796600e+04	2.023033e+13	5.127900e+04	1.000000	1.000000	
25%	2.130233e+07	2.023033e+13	4.406371e+07	1.000000	2.000000	
50%	3.971123e+07	2.023033e+13	1.379316e+08	4.000000	4.000000	
75%	5.592472e+17	2.023033e+13	2.940258e+08	14.000000	16.000000	
max	8.577857e+17	2.023033e+13	5.074666e+08	2429.000000	5180.000000	

8 rows × 40 columns

Data Preparation - cleaning

To startoff I looked at null values to begin thinking about replacement methods for null values. I then further inspected the columns with a correlation matrix which columns should be dropped, to avoid overfitting the model.

In [210]:
```
#inspecting nulls
df.isna().sum().sort_values(ascending=False).head(30)
```

Out[210]:
```
bathrooms                        23372
neighbourhood_group_cleansed     23372
calendar_updated                 23372
license                          22079
host_neighbourhood               20742
host_about                       11741
neighborhood_overview            10192
neighbourhood                    10192
host_location                     7136
review_scores_value               6192
review_scores_location            6192
review_scores_checkin             6192
review_scores_cleanliness         6191
review_scores_accuracy            6191
review_scores_communication       6191
review_scores_rating              6122
last_review                       6122
first_review                      6122
reviews_per_month                 6122
host_response_rate                5201
host_response_time                5201
host_acceptance_rate              2319
bedrooms                          1465
description                        534
beds                               212
bathrooms_text                      42
host_is_superhost                   33
name                                 2
host_listings_count                  0
number_of_reviews                    0
dtype: int64
```

In [211]:
```
#dropping columns that are mainly null

no_values = ['bathrooms', 'neighbourhood_group_cleansed','calendar_updated','license']

for x in no_values:

    df.drop(columns=[x], inplace=True)
```

In [212]:
```
#dropping columns that are not needed

not_necessary = ['host_neighbourhood','last_scraped','source','scrape_id',
                 'host_location','host_about','host_acceptance_rate', 'host_thumbnail_url',
                 'host_url','host_name','picture_url','last_review', 'first_review', "host_
                 "host_total_listings_count",'host_picture_url',
                 'host_identity_verified','host_has_profile_pic',
                 'host_verifications', 'calendar_last_scraped',"host_id","has_availability"
                 "instant_bookable","calculated_host_listings_count", "calculated_host_list
                 "calculated_host_listings_count_private_rooms","calculated_host_listings_c
                "minimum_minimum_nights", "maximum_minimum_nights", "minimum_maximum_nights
                "maximum_maximum_nights", "minimum_nights_avg_ntm", "maximum_nights_avg_ntm
                'beds']

for x in not_necessary:

    df.drop(columns=[x], inplace=True)
```

In [213]:
```
#looking at remaining nulls
df.isna().sum().sort_values(ascending=False).head(10)
```

Out[213]:
```
neighborhood_overview          10192
neighbourhood                  10192
review_scores_value             6192
review_scores_location          6192
review_scores_checkin           6192
review_scores_communication     6191
review_scores_cleanliness       6191
review_scores_accuracy          6191
reviews_per_month               6122
review_scores_rating            6122
dtype: int64
```

In [214]:
```
#replacing null values in description with empty strings with n/a as these are important to

no_value = ['neighborhood_overview', 'neighbourhood', 'name', 'description', 'host_response

for x in no_value:
    df[x] = df[x].fillna('n/a')
```

After looking at the airbnb listings on the website, it appears as though null bedrooms are studio apartments or homes that do not have any bedrooms

In [215]:
```
#looking at null values of bedroom to determine whether to drop or not
df[df['bedrooms'].isnull()]
```

Out[215]:

	id	listing_url	name	description	neighborhood
102	297742	https://www.airbnb.com/rooms/297742	seaview of Libyan Sea, comfy studio with terrace	Casa Rosa Studio has such a great view of the ...	Casa Rosa just above the
113	789777	https://www.airbnb.com/rooms/789777	Evli 4, spacious apartment, 7 min walk to center	Evli Apartments is fortunately located in the...	If you are lo peaceful sta
176	685844	https://www.airbnb.com/rooms/685844	Studio apartment by the sea.	Fully-equipped studio, fully-equipped kitchen ...	
236	876115	https://www.airbnb.com/rooms/876115	Mirabella Studio with Sea View for 2	ESL 1116395 - Our studios, on the lower floor,...	A peace where you ca
317	1055530	https://www.airbnb.com/rooms/1055530	Beautiful calm rooftop studio with stunning views	A beautiful self-contained, fully equipped hol...	The neighbou in
...	...	...	...	...	
23065	844700621948252197	https://www.airbnb.com/rooms/844700621948252197	Irene Luxury Apartments Emmanuel	Απολαύστε την χαλάρωση στα νέα διαμερίσματα IR...	
23186	852066997317337107	https://www.airbnb.com/rooms/852066997317337107	Enastron Apartment 6 *View-Pool-Parking-BBQ*	New luxury apartment for 2 to 3 people. Enast...	
23242	852676361817393851	https://www.airbnb.com/rooms/852676361817393851	BalconyStudio Panorama Meerblick	Erlebe in dieser besonderen und familienfreund...	
23298	854872123674361782	https://www.airbnb.com/rooms/854872123674361782	Ροδιά στούντιο με πισίνα και θεα	Χαλαρώστε κάνοντας μια μοναδική και ήρεμη απόδ...	
23314	856479422999011796	https://www.airbnb.com/rooms/856479422999011796	Veranda stalis	Enjoy your stay in a cozy & stylish apartment ...	

1465 rows × 35 columns

In [216]:
```
#replacing null scores with 0

no_value = ['review_scores_value', 'review_scores_location','review_scores_checkin',
    'review_scores_accuracy','review_scores_communication', 'review_scores_cleanliness',
            'host_response_rate', 'bedrooms', 'reviews_per_month', 'review_scores_rating']

for x in no_value:
    df[x] = df[x].fillna(0)
```

Dropping the values that are not needed

In [217]:
```
#checking for nulls
df.isna().sum().sort_values(ascending=False).head(30)
```

Out[217]:
```
bathrooms_text                 42
host_is_superhost              33
reviews_per_month               0
room_type                       0
property_type                   0
longitude                       0
latitude                        0
neighbourhood_cleansed          0
neighbourhood                   0
host_response_rate              0
host_response_time              0
host_since                      0
neighborhood_overview           0
description                     0
name                            0
listing_url                     0
accommodates                    0
bedrooms                        0
review_scores_value             0
number_of_reviews_l30d          0
review_scores_location          0
review_scores_communication     0
review_scores_checkin           0
review_scores_cleanliness       0
review_scores_accuracy          0
review_scores_rating            0
number_of_reviews_ltm           0
amenities                       0
number_of_reviews               0
availability_365                0
dtype: int64
```

In [218]:
```
#dropping empty superhost rows
df['host_is_superhost'].dropna(inplace=True)
```

Inspecting values and cleaning them prior to changing datatype

In [219]:
```
df['host_response_time'].value_counts()
```

Out[219]:
```
within an hour        14045
n/a                    5201
within a few hours     2206
within a day           1634
a few days or more      286
Name: host_response_time, dtype: int64
```

In [220]:
```
df['host_is_superhost'].value_counts()
```

Out[220]:
```
f    14893
t     8446
Name: host_is_superhost, dtype: int64
```

In [221]:
```
#changing host_is_superhost to binary values

bool_list = ['host_is_superhost']

for x in bool_list:

    df[x].replace(['f', 't'], [0, 1], inplace=True)
```

In [222]:
```
#sanity check
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23372 entries, 0 to 23371
Data columns (total 35 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   id                           23372 non-null  int64  
 1   listing_url                  23372 non-null  object 
 2   name                         23372 non-null  object 
 3   description                  23372 non-null  object 
 4   neighborhood_overview        23372 non-null  object 
 5   host_since                   23372 non-null  object 
 6   host_response_time           23372 non-null  object 
 7   host_response_rate           23372 non-null  object 
 8   host_is_superhost            23339 non-null  float64
 9   neighbourhood                23372 non-null  object 
 10  neighbourhood_cleansed       23372 non-null  object 
 11  latitude                     23372 non-null  float64
 12  longitude                    23372 non-null  float64
 13  property_type                23372 non-null  object 
 14  room_type                    23372 non-null  object 
 15  accommodates                 23372 non-null  int64  
 16  bathrooms_text               23330 non-null  object 
 17  bedrooms                     23372 non-null  float64
 18  amenities                    23372 non-null  object 
 19  price                        23372 non-null  object 
 20  availability_30              23372 non-null  int64  
 21  availability_60              23372 non-null  int64  
 22  availability_90              23372 non-null  int64  
 23  availability_365             23372 non-null  int64  
 24  number_of_reviews            23372 non-null  int64  
 25  number_of_reviews_ltm        23372 non-null  int64  
 26  number_of_reviews_l30d       23372 non-null  int64  
 27  review_scores_rating         23372 non-null  float64
 28  review_scores_accuracy       23372 non-null  float64
 29  review_scores_cleanliness    23372 non-null  float64
 30  review_scores_checkin        23372 non-null  float64
 31  review_scores_communication  23372 non-null  float64
 32  review_scores_location       23372 non-null  float64
 33  review_scores_value          23372 non-null  float64
 34  reviews_per_month            23372 non-null  float64
dtypes: float64(12), int64(9), object(14)
memory usage: 6.2+ MB
```

In [223]:
```
#inspecting individual object types
df_object = df.select_dtypes(include="object")

for col in df_object.columns:

    print("Column Name: ", col)
    print("Object Type: ", df[col].dtype)
    print("String: ", isinstance(df[col], str))
```

```
Column Name:  listing_url
Object Type:  object
String:  False
Column Name:  name
Object Type:  object
String:  False
Column Name:  description
Object Type:  object
String:  False
Column Name:  neighborhood_overview
Object Type:  object
String:  False
Column Name:  host_since
Object Type:  object
String:  False
Column Name:  host_response_time
Object Type:  object
String:  False
Column Name:  host_response_rate
Object Type:  object
String:  False
Column Name:  neighbourhood
Object Type:  object
String:  False
Column Name:  neighbourhood_cleansed
Object Type:  object
String:  False
Column Name:  property_type
Object Type:  object
String:  False
Column Name:  room_type
Object Type:  object
String:  False
Column Name:  bathrooms_text
Object Type:  object
String:  False
Column Name:  amenities
Object Type:  object
String:  False
Column Name:  price
Object Type:  object
String:  False
```

In [224]:
```
#changing columns to str

remove_char_list = ['bathrooms_text', 'price','host_response_rate']

#loop to remove non numerical characters

for x in remove_char_list:

    df[x].astype(str)
    df[x] = df[x].str.replace(r'[^\d.]+', '')
```

In [225]:
```
#renaming column

df.rename(columns={"bathrooms_text":"bathrooms"}, inplace=True)
```

In [226]:
```
#converting host_response_rate to percentage
df['host_response_rate'] = df['host_response_rate'].astype(float)

df['host_response_rate'] = df['host_response_rate'].div(100).round(2)

df['host_response_rate'] = df['host_response_rate'].fillna(0)

#sanity check of null values
df['host_response_rate'].isna().sum()
```

Out[226]:
```
0
```

Converting data types

In [227]:
```
#changing to numeric
df['bathrooms'] = pd.to_numeric(df['bathrooms'])
df['price'] = pd.to_numeric(df['price'])
```

In [229]:
```
#changing datasypes
float_type = ["bathrooms", "price"]
string_type = ["listing_url", "host_response_time", "name", "description","property_type",
               "neighborhood_overview","host_response_time","neighbourhood"]

change_type = [float_type, string_type]

for x in change_type:

    df[float_type].astype(float)
    df[string_type].astype(str)
```

In [230]:
```
#resetting index
df.reset_index(drop=True, inplace=True)
```

Creating final dataframe for feature selection and modeling

In [231]:
```
#making new dataframe from final cleaned df dataframe
feature_df = df
```

In [260]:
```
#looking at target variables
feature_df['review_scores_rating']
```

Out[260]:
```
0        4.50
1        4.91
2        5.00
3        5.00
4        5.00
         ...
23367    0.00
23368    0.00
23369    0.00
23370    0.00
23371    0.00
Name: review_scores_rating, Length: 23372, dtype: float64
```

In [132]:
```
# #changing to category for sanity check purposes

# feature_df['listing_rating'] = feature_df['review_scores_rating'].astype(int)

# feature_df['listing_rating'] = feature_df['listing_rating'].astype(str)

# feature_df['listing_rating'] = feature_df['listing_rating'] + ' Stars'
```

In [133]:
```
#sanity check
#feature_df['listing_rating'].value_counts()
```

In [134]:
```
#renaming column
feature_df.rename(columns={'review_scores_rating':'airbnb_rating'}, inplace=True)
```

In [233]:
```
feature_df['airbnb_rating'] = feature_df['review_scores_rating']
```

Manually claeaning amenities column

After inspecting the description, name, and amenities columns and analyzing the most common words present in each of these columns (comprised on long string), I decided to only use the **amenities** column as an indicator of the listings key features. If I had additional time, I would have liked to do a sentiment analysis on the titles and descriptions to see how this impacted scores.

- Cleaning: Since there are over 21000 different amenities and that they play an important role in listings, I cleaned the columns by removing stop words, unnecessary words and phrases, and then used Counter to count the most common features to help analyze which ones should be kept. I tried to limit repetitive features such as soap, linens, hair dryer. I iterated through the list many times until I cleaned up the values.

In [234]:
```
#making all values lowercase
feature_df['amenities'] = feature_df['amenities'].apply(str.lower)

#removing all characters that aren't letters
feature_df['amenities'] = feature_df['amenities'].str.replace('[^a-zA-Z\s]', '', regex=True
```

In [235]:
```
itional list of stop words and unneeded strings to drop
'space</b><br','/>The','/><br','is', 'you',
'located', 'The', "και", "/><b>The", "<br", "one", "two", "room", "living", "fully", "equipp
It", "με", "double", "also", 'bed', 'beautiful', 'floor', 'enjoy', 'away', 'bathroom', 'bedı
offers', 'dining', '3', 'ideal', 'km', 'large', '/><b>Guest', 'access</b><br', 'minutes', '
από','city','σε', '/>', 'free', '4', 'Cretan', '-', '2', '3', 'old', 'near', 'home', '1',"al
'water', 'maker', "hangers","hot", 'u', "street", 'cooking', "premises", "clothing", "silve
"extinguisher", "first", "aid", "kit", "street", "shower", 'wine', 'glasses', 'storage', 'r
greets', 'host', 'iron', 'entrance', 'extra', 'outdoor', 'free', 'private', 'table', 'dryin
'high','pillows', 'blankets','baking', 'sheet', 'kettle', 'furniture', 'unit', 'pack', 'un'
'roomdarkening', 'childrenus', 'chair', 'shades','smoke', 'freezer', 'books', 'shared','bod
system', 'lockbox','soap', 'body', 'linens', 'hair_dryer', 'toaster', 'basics', 'cleaning_p
wardrobe", "upon", "request", "fenced", 'safe', 'type', 'toys', 'drip', 'single', 'level',
baby', 'mini','fridge', "stainless", "steel", "carbon", "carbon", "monoxide", "central", "b
"coutryard", "public","ethernet", "connection"]

itional words after filtering through results
menities'] = feature_df['amenities'].apply(lambda x: ' '.join([word for word in x.split() i
```

In [236]:
```
#updating feature name
feature_df.replace({"amenities": {'long term stays':'long_term_stays', "air conditioning":
                                  "beach access":"beach_access", "hair dryer":"hair_dryer",
                                  "shared pool":"shared_pool", "luggage dropoff": "luggage_
                                  "private pool":"private_pool","fast wifi":"fast_wifi","se
                                 "hot water":"hot_water","dedicated workspace":"dedicated_w
                                  "private parking":"private_parking","cleaning products":"
                                  "bbq grill": "bbq_grill","self checkin":"self_checkin", "
                                  "ac split ductless":"ac_split_ductless"}}, regex=True, in

stop_words = set(stopwords.words("english"))

# removing stopwords
feature_df['amenities'] = feature_df['amenities'].apply(lambda x: ' '.join([word for word i

#making dataframe
top_attributes_amen = pd.DataFrame(Counter(" ".join(feature_df['amenities']).split()).most_
top_attributes_amen = top_attributes_amen[top_attributes_amen[0] != "Crete"]

#changing column name and datatype
top_attributes_amen.rename(columns={0:"amenity", 1:"amenity_count"}, inplace=True)
top_attributes_amen['amenity'] = top_attributes_amen['amenity'].astype(str)

#dropping repeat rows
top_attributes_amen = top_attributes_amen.drop([18, 27, 28, 36, 37, 38, 39, 40])

#inspecting top values
top_attributes_amen.head(60)
```

Out[236]:

	amenity	amenity_count
0	parking	27648
1	view	27503
2	coffee	26132
3	essentials	23643
4	wifi	22706
5	kitchen	21891
6	tv	21407
7	hair	20856
8	air_conditioning	18590
9	dishes	17887
10	refrigerator	17857
11	crib	16568
12	balcony	16560
13	patio	16559
14	washer	16445
15	heating	16117
16	shampoo	15786
17	stove	13779
19	pool	13183
20	long_term_stays	12692
21	backyard	11614
22	luggage_dropoff	9682
23	bbq_grill	8453
24	microwave	8122
25	beach_access	8015
26	cleaning_products	7961
29	dishwasher	6431
30	self_checkin	6298
31	indoor	5698
32	cable	5584
33	smoking	5572
34	mountain	5340
35	pets	5227
41	barbecue	4288
42	utensils	4288
43	electric	4264
44	courtyard	3348
45	beachfront	2921
46	dinnerware	2809
47	netflix	2774
48	open	2657

	amenity	amenity_count
49	hours	2657
50	material	2591
51	guards	2590
52	reading	2589
53	stay	2558
54	cleaning	2521
55	conditioner	2509
56	babysitter	2454
57	recommendations	2454
58	property	2410
59	security	2406
60	cameras	2397
61	portable	2344
62	ac_split_ductless	2265
63	access	2238
64	resort	2207
65	bay	2184
66	woodburning	2179
67	mosquito	2158

Merging top characterisitcs into a dataframe

In [237]:
```
# making function to loop through two lists and replace the values
def replace_list_order(list_1, list_2, dataframe, column):
    for (a, b) in zip(list_1, list_2):
        dataframe = dataframe.replace({column: {a:b}})
    return pd.DataFrame(dataframe[column])
```

Translating town names to english

In [238]:
```
feature_df['neighbourhood_cleansed'].value_counts()
```

Out[238]:
```
Χανίων                    5192
Ρεθύμνης                  2806
Χερσονήσου                2069
Αποκορώνου                1662
Ηρακλείου                 1633
Αγίου Νικολάου            1352
Κισσάμου                  1334
Πλατανιά                  1225
Μαλεβιζίου                1032
Αγίου Βασιλείου            990
Ιεράπετρας                 877
Φαιστού                    834
Μυλοποτάμου                626
Σητείας                    583
Καντάνου - Σέλινου         313
Αρχανών - Αστερουσίων      191
Βιάννου                    157
Σφακίων                    134
Γόρτυνας                   115
Μινώα Πεδιάδας              93
Αμάριου                     68
Οροπεδίου Λασιθίου          41
Γαύδου                      24
Ανωγείων                    21
Name: neighbourhood_cleansed, dtype: int64
```

I researched the town names and created two lists in order to rename the values.

In [239]:
```
#renaming the greek towns

greek_town = ['Χανίων', 'Ρεθύμνης','Χερσονήσου','Αποκορώνου','Ηρακλείου','Αγίου Νικολάου','Κισσ
              'Πλατανιά','Μαλεβιζίου','Αγίου Βασιλείου','Ιεράπετρας','Φαιστού','Μυλοποτάμου', 'Ση
              'Καντάνου - Σέλινου','Αρχανών - Αστερουσίων','Βιάννου',
              'Σφακίων','Γόρτυνας','Μινώα Πεδιάδας','Αμάριου','Οροπεδίου Λασιθίου','Γαύδου','Αν


english_town = ["Chania", "Rethymno", "Hersonissos", "Apokoronas", "Heraklion", "Agios Niko
               "Platanias","Malevizi", "Agios Vasilios", "Ierapetra", "Phaistos", "Mylopota
               "Kantanos-Selino","Archanes-Asterousia", "Viannos", "Sfakia", "Gortyna", "Mi
               "Oropedio Lasithiou","Gavdos","Anogeia"]
```

In [240]:
```
feature_df['neighbourhood_cleansed'] = replace_list_order(greek_town, english_town, feature
```

In [241]:
```
#sanity check
feature_df['neighbourhood_cleansed'].value_counts()
```

Out[241]:
```
Chania                  5192
Rethymno                2806
Hersonissos             2069
Apokoronas              1662
Heraklion               1633
Agios Nikolaos          1352
Kissamos                1334
Platanias               1225
Malevizi                1032
Agios Vasilios           990
Ierapetra                877
Phaistos                 834
Mylopotamos              626
Siteia                   583
Kantanos-Selino          313
Archanes-Asterousia      191
Viannos                  157
Sfakia                   134
Gortyna                  115
Minoa Pediada             93
Amari                     68
Oropedio Lasithiou        41
Gavdos                    24
Anogeia                   21
Name: neighbourhood_cleansed, dtype: int64
```

Renaming the final cleaned dataset to final_feature_df.

In [242]:
```
#making list of top 15 most common words from the amenities column and adding back to new .
vocab_list = top_attributes_amen['amenity'].tolist()
top_vocab_list = vocab_list[0:40]
```

In [243]:
```
#renaming dataframe
final_feature_df = feature_df
```

In [244]:
```
#defining function that makes all letters lowercase and checks to see if keyword is in colu
def check_keyword(column, keyword):
    return int(keyword.lower() in column.lower())

# apply the function

for kw in top_vocab_list:
    feature_df[kw] = final_feature_df['amenities'].apply(check_keyword, keyword=kw)
```

In [245]:
```
final_feature_df.head()
```

Out[245]:

	id	listing_url	name	description	neighborhood_overview	host_since	host_response_t
0	28970	https://www.airbnb.com/rooms/28970	artists' house in the old town	This is a fully renovated stone house from 191...	its calm with many old ladies that adoooore s...	2010-05-14	within an
1	27966	https://www.airbnb.com/rooms/27966	Heraklion-Pinelopi Apartment	For an unforgettable stay!! Just 10 minutes wa...	Ammoudara is a very cute area with lots of bea...	2010-05-08	within an
2	29849	https://www.airbnb.com/rooms/29849	Villa Kallergi - Nefeli, 6 guests	Villa Nefeli Kallergi with private pool in the...	The traditional village of Loutra was chosen t...	2010-05-15	within an
3	29130	https://www.airbnb.com/rooms/29130	Villa Kallergi - Athena, 12 guests	Villa Athena Kallergi is in the heart of the C...	The traditional village of Loutra was chosen t...	2010-05-15	within an
4	31789	https://www.airbnb.com/rooms/31789	Kissamos Windmills	The space Kissamos Windmills apart...	n/a	2010-06-01	

5 rows × 76 columns

Renaming property type to six classes

In [246]:
```
#making all values lowercase
final_feature_df['property_type'] = final_feature_df['property_type'].apply(str.lower)

#removing all characters that aren't letters
final_feature_df['property_type'] = final_feature_df['property_type'].str.replace('[^a-zA-
```

In [247]:
```
original_value = ['entire rental unit', 'entire villa', 'entire home', 'entire condo','roo
              'entire cottage', 'entire serviced apartment', 'private room in rental unit',
              'room in boutique hotel', 'entire townhouse', 'private room in bed and breakfa
              'cycladic home', 'private room in serviced apartment', 'entire guest suite', '
              'entire bungalow', 'private room in condo', 'room in serviced apartment', 'pri
              'entire loft', 'private room in home', 'earthen home', 'tiny home', 'entire pl
              'private room in villa', 'farm stay', 'private room in guesthouse', 'private r
              'room in bed and breakfast', 'shared room', 'private room in vacation home',
              'private room in guest suite', 'shared room in hostel', 'boat', 'entire cabin'
              'private room in townhouse', 'private room in hostel', 'castle', 'private room
              'camperrv', 'private room in loft', 'tent', 'room in hostel', 'room in nature 
              'cave', 'private room in tiny home', 'island', 'entire chalet', 'entire bed an
              'casa particular', 'private room in tent', 'private room in earthen home', 'pr
              'private room in nature lodge', 'entire homeapt', 'private room in cycladic ho
              'shared room in vacation home', 'treehouse', 'shared room in rental unit', 'sh
              'bus', 'shared room in townhouse', 'shared room in guest suite', 'room in rent
              'hut', 'private room in tipi', 'private room in minsu', 'private room in dorm'
              'private room in casa particular', 'private room in camperrv', 'private room i
                     "private room in special"]
```

In [248]:
```
pdated_list = ['entire rental unit', 'entire villa', 'entire home', 'entire condo', 'room i
               'entire home', 'entire rental unit', 'private room', 'room in hotel', 'room
               'entire townhouse', 'private room', 'entire home', 'entire home', 'private r
               'entire rental unit', 'entire guesthouse', 'entire home', 'private room', 'p
               'private room', 'entire home', 'private room', 'entire home', 'entire home',
               'private room', 'entire home', 'private room', 'private room', 'private room
               'private room', 'private room', 'private room', 'entire home', 'boat', 'priv
               'private room', 'private room', 'special', 'private room', 'special', 'priva
               'special', 'special', 'entire villa', 'island', 'private room', 'entire bed
               'entire home', 'private room', 'private room', 'private room', 'private room
               'entire home', 'shared room', 'shared room', 'special', 'special', 'shared r
               'private room', 'private room', 'special', 'special', 'private room', 'speci
               'shared room', 'private room', 'special', 'private room', 'shared room', 'pr
```

In [249]:
```
final_feature_df['property_type'] = final_feature_df['property_type'].replace(original_valu
```

In [250]:
```
final_feature_df['property_type'].value_counts()
```

Out[250]:
```
entire rental unit          7368
entire home                 5630
entire villa                5020
entire condo                1798
room in hotel               1672
private room                1427
entire townhouse             265
entire guesthouse             91
special                       51
shared room                   28
boat                          13
entire bed and breakfast       4
island                         4
room in resort                 1
Name: property_type, dtype: int64
```

Adding final categorical target column

In [251]:
```
final_feature_df.airbnb_rating
```

Out[251]:
```
0        4.50
1        4.91
2        5.00
3        5.00
4        5.00
         ...
23367    0.00
23368    0.00
23369    0.00
23370    0.00
23371    0.00
Name: airbnb_rating, Length: 23372, dtype: float64
```

In [257]:
```
final_feature_df['airbnb_rating']
```

Out[257]:
```
0         4.5
1        4.91
2         5.0
3         5.0
4         5.0
         ...
23367     0.0
23368     0.0
23369     0.0
23370     0.0
23371     0.0
Name: airbnb_rating, Length: 23372, dtype: object
```

In [258]:
```
final_feature_df['airbnb_rating'] = final_feature_df['airbnb_rating'].astype(str)
```

In [259]:
```
#making final categorical target column which makes the ratings binary and puts them in two
# The classes are Above Four Stars and Below Four Stars

l_is_grte_4stars = []

for rating in final_feature_df.airbnb_rating:
    if int(rating.split(' ')[0]) >= 4:
        l_is_grte_4stars.append("Above Four Stars")
    else:
        l_is_grte_4stars.append("Below Four Stars")


final_feature_df['Airbnb Rating'] = l_is_grte_4stars
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-259-e6d9bd98ed9a> in <module>
      5
      6 for rating in final_feature_df.airbnb_rating:
----> 7     if int(rating.split(' ')[0]) >= 4:
      8         l_is_grte_4stars.append("Above Four Stars")
      9     else:

ValueError: invalid literal for int() with base 10: '4.5'
```

In [255]:
```
#sanity check of classification distribution
final_feature_df.airbnb_rating.value_counts(normalize=True)
```

Out[255]:
```
5.0     0.304938
0.0     0.264333
4.5     0.020837
4.67    0.018227
4.0     0.017072
          ...
1.5     0.000043
3.4     0.000043
3.95    0.000043
4.34    0.000043
4.04    0.000043
Name: airbnb_rating, Length: 133, dtype: float64
```

Analyzing Features

In [261]:
```
final_feature_df['Airbnb Rating'].hist(bins='auto');
```

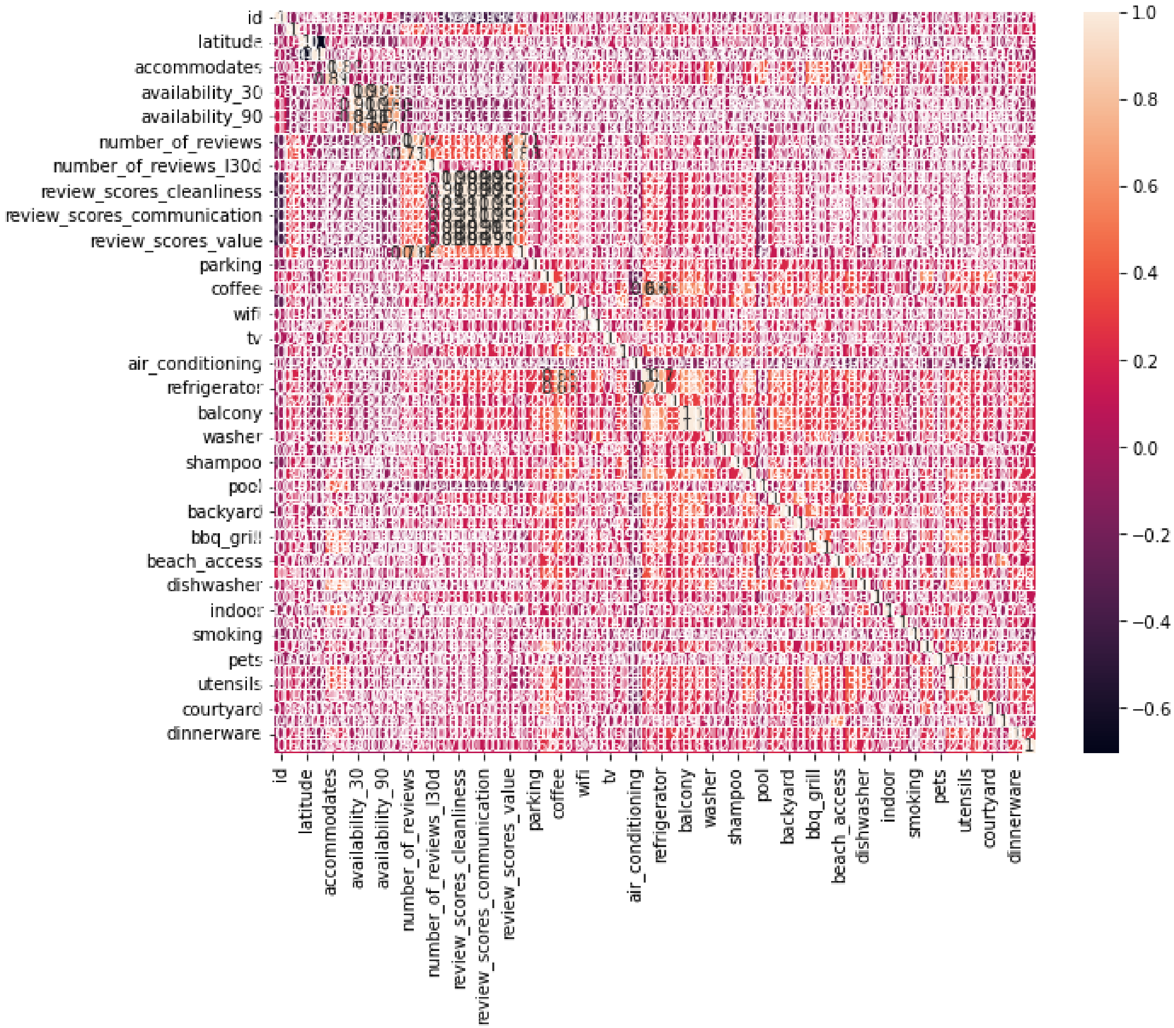
In [170]:
```
final_feature_df.head()
```

Out[170]:

	id	listing_url	name	description	neighborhood_overview	host_since	host_response_t
0	28970	https://www.airbnb.com/rooms/28970	artists' house in the old town	This is a fully renovated stone house from 191...	its calm with many old ladies that adooooore s...	2010-05-14	within an l
1	27966	https://www.airbnb.com/rooms/27966	Heraklion-Pinelopi Apartment	For an unforgettable stay!! Just 10 minutes wa...	Ammoudara is a very cute area with lots of bea...	2010-05-08	within an l
2	29849	https://www.airbnb.com/rooms/29849	Villa Kallergi - Nefeli, 6 guests	Villa Nefeli Kallergi with private pool in the...	The traditional village of Loutra was chosen t...	2010-05-15	within an l
3	29130	https://www.airbnb.com/rooms/29130	Villa Kallergi - Athena, 12 guests	Villa Athena Kallergi is in the heart of the C...	The traditional village of Loutra was chosen t...	2010-05-15	within an l
4	31789	https://www.airbnb.com/rooms/31789	Kissamos Windmills	The space Kissamos Windmills apart...	n/a	2010-06-01	

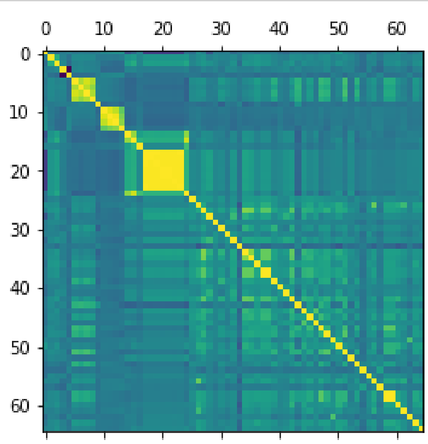
5 rows × 75 columns

In [171]:
```
# looking at corr of numeric variables which is way too big
fig, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(final_feature_df.corr(), annot=True)
plt.show()
```



In [262]:
```
corr_matrix = pd.DataFrame(df.corr(method='pearson'))
print_full(corr_matrix['review_scores_rating'].sort_values(ascending=False))
```

In [636]:
```
plt.matshow(df.corr())
plt.show()
```



In [638]:
```
final_feature_df['airbnb_rating']
```

Out[638]:
```
0        4 Stars
1        4 Stars
2        5 Stars
3        5 Stars
4        5 Stars
          ...
23081    0 Stars
23082    0 Stars
23083    0 Stars
23084    0 Stars
23085    0 Stars
Name: airbnb_rating, Length: 23086, dtype: object
```

In [639]:
```
corr_matrix = pd.DataFrame(final_feature_df.corr(method='pearson'))

print_full(corr_matrix['review_scores_rating'].sort_values(ascending=False))
```

```
review_scores_rating            1.000000
review_scores_value             0.994399
review_scores_accuracy          0.993442
review_scores_cleanliness       0.993388
review_scores_communication     0.993290
review_scores_checkin           0.992504
review_scores_location          0.989616
reviews_per_month               0.396261
number_of_reviews_ltm           0.356035
number_of_reviews               0.318674
dishes                          0.283261
essentials                      0.268578
host_is_superhost               0.252232
refrigerator                    0.245573
stove                           0.217594
coffee                          0.215107
patio                           0.205086
balcony                         0.205086
hair                            0.191936
long_term_stays                 0.186245
host_response_rate              0.185122
shampoo                         0.174105
luggage_dropoff                 0.139915
self_checkin                    0.128583
crib                            0.126889
kitchen                         0.122257
parking                         0.116028
backyard                        0.108410
netflix                         0.107255
heating                         0.103369
number_of_reviews_l30d          0.099009
dinnerware                      0.091630
washer                          0.088474
beachfront                      0.087904
wifi                            0.087488
view                            0.085987
cleaning_products               0.085984
cable                           0.082880
beach_access                    0.081893
electric                        0.066902
microwave                       0.062741
courtyard                       0.058630
mountain                        0.044292
utensils                        0.033638
barbecue                        0.033638
latitude                        0.029017
smoking                         0.027643
pets                            0.025862
dishwasher                      0.017625
bbq_grill                       0.012255
tv                              0.008294
indoor                          0.004297
beds                           -0.054039
accommodates                   -0.055533
air_conditioning               -0.057595
bedrooms                       -0.060314
longitude                      -0.064162
availability_365               -0.076722
availability_30                -0.087792
bathrooms                      -0.093087
price                          -0.103794
availability_60                -0.111976
pool                           -0.126696
availability_90                -0.133123
id                             -0.388921
Name: review_scores_rating, dtype: float64
```

In [1358]:
```
l_is_grte_4stars = []

for rating in final_feature_df.airbnb_rating:
    if int(rating.split(' ')[0]) >= 4:
        l_is_grte_4stars.append("Above Four Stars")
    else:
        l_is_grte_4stars.append("Below Four Stars")


final_feature_df['Airbnb Rating'] = l_is_grte_4stars

#sanity check of classification distribution
final_feature_df.is_grte_4stars.value_counts()
```

In [649]:
```
final_feature_df.is_grte_4stars.value_counts()
```

Out[649]:
```
1    16771
0     6315
Name: is_grte_4stars, dtype: int64
```

In [650]:
```
final_feature_df.isna().sum().sum()
```

Out[650]:
```
28
```

In [651]:
```
final_feature_df.dropna(inplace=True)
```

In [652]:
```
final_feature_df.isna().sum().sum()
```

Out[652]:
```
0
```

In [916]:
```
corr_matrix = pd.DataFrame(final_feature_df.corr(method='pearson'))

print_full(corr_matrix['is_grte_4stars'].sort_values(ascending=False))
```

```
is_grte_4stars          1.000000
dishes                  0.280013
essentials              0.266258
host_is_superhost       0.245897
refrigerator            0.243106
stove                   0.212662
coffee                  0.210239
balcony                 0.203115
patio                   0.203115
hair                    0.188089
host_response_rate      0.184905
long_term_stays         0.183684
shampoo                 0.171467
luggage_dropoff         0.139186
self_checkin            0.126356
crib                    0.123703
kitchen                 0.119160
parking                 0.114733
backyard                0.106311
netflix                 0.102052
heating                 0.101338
beachfront              0.089274
wifi                    0.086311
dinnerware              0.085772
washer                  0.084360
view                    0.082843
cleaning_products       0.081976
beach_access            0.081503
cable                   0.079186
electric                0.060891
microwave               0.057871
courtyard               0.055175
mountain                0.040831
smoking                 0.031012
utensils                0.030600
barbecue                0.030600
pets                    0.028910
latitude                0.026782
dishwasher              0.012139
tv                      0.009479
bbq_grill               0.008864
indoor                  0.000072
air_conditioning       -0.054205
beds                   -0.055396
accommodates           -0.058598
longitude              -0.063413
bedrooms               -0.063415
bathrooms              -0.095143
price                  -0.102225
pool                   -0.127897
Name: is_grte_4stars, dtype: float64
```

One hot encoding categorical features & train test split

In [263]:
```
#dropping final columns before one hot encoding
#final_feature_df.drop(columns=["name","neighbourhood","id","listing_url", "description","r
                                 'review_scores_communication','review_scores_location','re
                                "availability_30","availability_60","availability_90",
                                "availability_365","host_response_time","review_scores_accu
                               "property_type","amenities","number_of_reviews","review_scor
                               "number_of_reviews_ltm","number_of_reviews_l30d",
                              "reviews_per_month"], inplace=True)
```

In [684]:
```
categorical_features_names = ['neighbourhood_cleansed', "room_type"]

enc = OneHotEncoder()


enc_data = pd.DataFrame(enc.fit_transform(
                final_feature_df[categorical_features_names]).toarray())

enc_cols = enc.get_feature_names()

mapped_cols = []

for col in enc_cols:
    if 'x0' in col:
        mapped_col = col.replace('x0', categorical_features_names[0])
    if 'x1' in col:
        mapped_col = col.replace('x1',categorical_features_names[1])

    mapped_cols.append(mapped_col)

enc_data.columns = mapped_cols

# Merging OHC with final dataframe`
model_dt_df = pd.merge(final_feature_df, enc_data, left_index=True, right_index=True)
```

Dummy Classifier Model

In [685]:
```
# defining features
X = model_dt_df.drop(categorical_features_names + ['airbnb_rating'] + ['is_grte_4stars'], a
y = model_dt_df.is_grte_4stars

# Split for test & training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=27)
```

In [686]:
```
#sanity check
(X_test).shape
```

Out[686]:
```
(6909, 77)
```

In [687]:
```
dummy_model = DummyClassifier(strategy="most_frequent")

dummy_model.fit(X_train, y_train)
```

Out[687]:
```
DummyClassifier(strategy='most_frequent')
```

In [688]:
```
#checking distribution of target variables
y_test.value_counts(normalize=True)
```

Out[688]:
```
1    0.726299
0    0.273701
Name: is_grte_4stars, dtype: float64
```

The dummy model score gives us a cross_val_score of 67%. This means that the accuracy of the model is 67% if we always guess the majority class

In [689]:
```
# Checking cross_val_score
cv_results = cross_val_score(dummy_model, X_train, y_train, cv=3)
cv_results.mean()
```

Out[689]:
```
0.7277464156207207
```

In [1006]:
```
class ModelWithCV():
    '''Structure to save the model and more easily see its crossvalidation'''

    def __init__(self, model, model_name, X, y, cv_now=True):
        self.model = model
        self.name = model_name
        self.X = X
        self.y = y
        # For CV results
        self.cv_results = None
        self.cv_mean = None
        self.cv_median = None
        self.cv_std = None
        #
        if cv_now:
            self.cross_validate()

    def cross_validate(self, X=None, y=None, kfolds=15):
        '''
        Perform cross-validation and return results.

        Args:
          X:
            Optional; Training data to perform CV on. Otherwise use X from object
          y:
            Optional; Training data to perform CV on. Otherwise use y from object
          kfolds:
            Optional; Number of folds for CV (default is 10)
        '''

        cv_X = X if X else self.X
        cv_y = y if y else self.y

        self.cv_results = cross_val_score(self.model, cv_X, cv_y, cv=kfolds, scoring="prec
        self.cv_mean = np.mean(self.cv_results)
        self.cv_median = np.median(self.cv_results)
        self.cv_std = np.std(self.cv_results)


    def print_cv_summary(self):
        cv_summary = (
        f'''CV Results for `{self.name}` model:
            {self.cv_mean:.5f} ± {self.cv_std:.5f} precision
        ''')
        print(cv_summary)


    def print_df_results(self):
        df_cv_score = pd.DataFrame(zip(self.cv_results, self.cv_mean),
                                   columns=['cv_precision_score', 'mean_cv_precision_score'], i
        return df

    def plot_cv(self, ax):
        '''
        Plot the cross-validation values using the array of results and given
        Axis for plotting.
        '''
        ax.set_title(f'CV Precision Results for `{self.name}` Model')
        # Thinner violinplot with higher bw
        sns.violinplot(y=self.cv_results, ax=ax, bw=.4)
        sns.swarmplot(
                y=self.cv_results,
                color='orange',
                size=10,
                alpha= 0.8,
                ax=ax
        )
```

```
        return ax
```

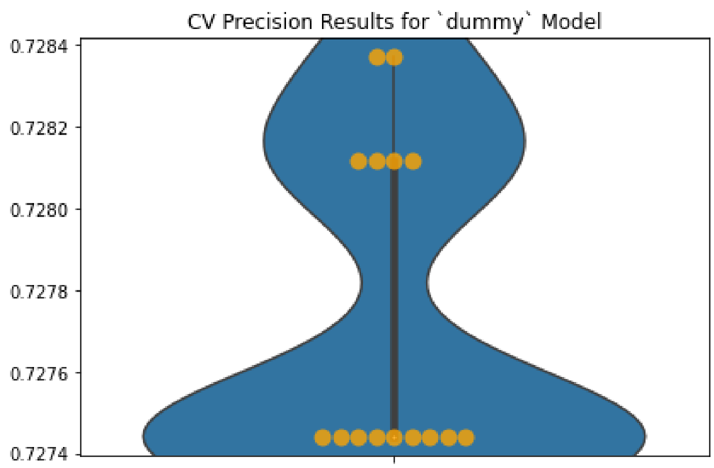
In [899]:
```
#cross validation
dummy_model_results = ModelWithCV(model=dummy_model, model_name='dummy',X=X_train, y=y_tra:
```

In [900]:
```
type(dummy_model_results)
```

Out[900]:
```
__main__.ModelWithCV
```

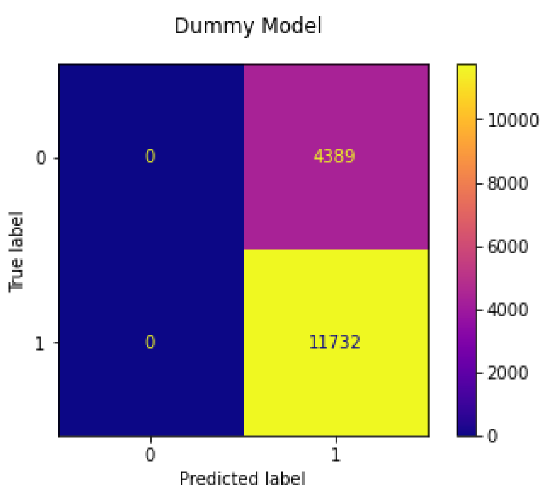
In [901]:
```
fig, ax = plt.subplots()

ax = dummy_model_results.plot_cv(ax)
plt.tight_layout();

dummy_model_results.print_cv_summary()
```

```
CV Results for `dummy` model:
            0.72775 ± 0.00038 precision
```



In [ ]:
```
cv_df = get_cross_val_dataframe(dt_model_1, X_train, y_train, kfolds=5, scoring="precision'
model_cv_mean = pd.DataFrame(np.mean(cv_df), columns=['Base Model'])
```

In [902]:
```
fig, ax = plt.subplots()

fig.suptitle("Dummy Model")

plot_confusion_matrix(dummy_model, X_train, y_train, ax=ax, cmap="plasma");
```



In [903]:
```
# just the numbers (this should work even with older scikit-learn)
confusion_matrix(y_train, dummy_model.predict(X_train))
```

Out[903]:
```
array([[    0,  4389],
       [    0, 11732]])
```

Final Model Performance Metric Overview

To look at the overall performance of the model, I will first be looking at the cross validation score and **precision score**. Precision will be my main metric to determine the final model's performance.

- The cross validation score will determine the general performance of the decision tree model and help detect overfitting.
- The precision score is the main metric I will be using, as it is important to reduce false positives while predicting ratings.

When I intiially ran the base model, I included features that were too highly correlated with the target variable, so I dropped those in the data cleaning section. Those features included:

- number_of_reviews
- review_scores_cleanliness
- review_scores_rating

I originally wanted to include these in the model, but after further looking into the business problem I determined that it was best to remove these values as it did not provide insight into what to do in order to obtain reviews.

Train Test Split

In [1380]:
```
model_dt_df.rename(columns={"host_is_superhost": "superhost", "host_response_rate":"host re
```

In [1381]:
```
model_dt_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23030 entries, 0 to 23057
Data columns (total 81 columns):
 #   Column                                     Non-Null Count  Dtype  
---  ------                                     --------------  -----  
 0   host response rate                         23030 non-null  float64
 1   superhost                                  23030 non-null  int64  
 2   neighbourhood_cleansed                     23030 non-null  object 
 3   latitude                                   23030 non-null  float64
 4   longitude                                  23030 non-null  float64
 5   room_type                                  23030 non-null  object 
 6   accommodates                               23030 non-null  int64  
 7   bathrooms                                  23030 non-null  float64
 8   bedrooms                                   23030 non-null  float64
 9   beds                                       23030 non-null  float64
 10  price                                      23030 non-null  float64
 11  airbnb_rating                              23030 non-null  object 
 12  parking                                    23030 non-null  int64  
 13  view                                       23030 non-null  int64  
 14  coffee                                     23030 non-null  int64  
 15  essentials                                 23030 non-null  int64  
 16  wifi                                       23030 non-null  int64  
 17  kitchen                                    23030 non-null  int64  
 18  tv                                         23030 non-null  int64  
 19  hair                                       23030 non-null  int64  
 20  air_conditioning                           23030 non-null  int64  
 21  dishes                                     23030 non-null  int64  
 22  refrigerator                               23030 non-null  int64  
 23  crib                                       23030 non-null  int64  
 24  balcony                                    23030 non-null  int64  
 25  patio                                      23030 non-null  int64  
 26  washer                                     23030 non-null  int64  
 27  heating                                    23030 non-null  int64  
 28  shampoo                                    23030 non-null  int64  
 29  stove                                      23030 non-null  int64  
 30  pool                                       23030 non-null  int64  
 31  long_term_stays                            23030 non-null  int64  
 32  backyard                                   23030 non-null  int64  
 33  luggage_dropoff                            23030 non-null  int64  
 34  bbq_grill                                  23030 non-null  int64  
 35  microwave                                  23030 non-null  int64  
 36  beach_access                               23030 non-null  int64  
 37  cleaning_products                          23030 non-null  int64  
 38  dishwasher                                 23030 non-null  int64  
 39  self_checkin                               23030 non-null  int64  
 40  indoor                                     23030 non-null  int64  
 41  smoking                                    23030 non-null  int64  
 42  cable                                      23030 non-null  int64  
 43  mountain                                   23030 non-null  int64  
 44  pets                                       23030 non-null  int64  
 45  barbecue                                   23030 non-null  int64  
 46  utensils                                   23030 non-null  int64  
 47  electric                                   23030 non-null  int64  
 48  courtyard                                  23030 non-null  int64  
 49  beachfront                                 23030 non-null  int64  
 50  dinnerware                                 23030 non-null  int64  
 51  netflix                                    23030 non-null  int64  
 52  is_grte_4stars                             23030 non-null  object 
 53  neighbourhood_cleansed_Agios Nikolaos      23030 non-null  float64
 54  neighbourhood_cleansed_Agios Vasilios      23030 non-null  float64
 55  neighbourhood_cleansed_Amari               23030 non-null  float64
 56  neighbourhood_cleansed_Anogeia             23030 non-null  float64
 57  neighbourhood_cleansed_Apokoronas          23030 non-null  float64
 58  neighbourhood_cleansed_Archanes-Asterousia  23030 non-null  float64
 59  neighbourhood_cleansed_Chania              23030 non-null  float64
 60  neighbourhood_cleansed_Gavdos              23030 non-null  float64
 61  neighbourhood_cleansed_Gortyna             23030 non-null  float64
```

```
 62  neighbourhood_cleansed_Heraklion          23030 non-null  float64
 63  neighbourhood_cleansed_Hersonissos        23030 non-null  float64
 64  neighbourhood_cleansed_Ierapetra          23030 non-null  float64
 65  neighbourhood_cleansed_Kantanos-Selino    23030 non-null  float64
 66  neighbourhood_cleansed_Kissamos           23030 non-null  float64
 67  neighbourhood_cleansed_Malevizi           23030 non-null  float64
 68  neighbourhood_cleansed_Minoa Pediada      23030 non-null  float64
 69  neighbourhood_cleansed_Mylopotamos        23030 non-null  float64
 70  neighbourhood_cleansed_Oropedio Lasithiou  23030 non-null  float64
 71  neighbourhood_cleansed_Phaistos           23030 non-null  float64
 72  neighbourhood_cleansed_Platanias          23030 non-null  float64
 73  neighbourhood_cleansed_Rethymno           23030 non-null  float64
 74  neighbourhood_cleansed_Sfakia             23030 non-null  float64
 75  neighbourhood_cleansed_Siteia             23030 non-null  float64
 76  neighbourhood_cleansed_Viannos            23030 non-null  float64
 77  room_type_Entire home/apt                 23030 non-null  float64
 78  room_type_Hotel room                      23030 non-null  float64
 79  room_type_Private room                    23030 non-null  float64
 80  room_type_Shared room                     23030 non-null  float64
dtypes: float64(35), int64(42), object(4)
memory usage: 14.4+ MB
```

In [1382]:

```
# defining features
X = model_dt_df.drop(categorical_features_names + ['airbnb_rating'] +
                            ['is_grte_4stars'], axis=1)
y = model_dt_df.is_grte_4stars

# Split for test & training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=27)
```

Model Performance Metric Overview

To look at the overall performance of all models, I will first be looking at the cross validation scores and precision score. **Precision** will be my main metric to determine the final model's performance.

- The cross validation score will determine the general performance of the decision tree model and help detect overfitting.
- The precision score is the main metric I will be using, as it is important to reduce false positives while predicting ratings. This is essential as we do not want to misclassify low star properties as having high starts, as it will give airbnb hosts false expectations of an increase in bookings and reviews.

When I intiially ran the base model, I included features that were too highly correlated with the target variable, so I dropped those in the data cleaning section. Those features included:

- number_of_reviews
- review_scores_cleanliness
- review_scores_rating

I originally wanted to include these in the model, but after further looking into the business problem I determined that it was best to remove these values as it did not provide insight into what to do in order to obtain reviews. So though number_of_reviews is the highest correlated feature, it was removed from the model.

#1 Decision Tree

In [905]:
```
#fitting model
dt_model_1 = DecisionTreeClassifier(random_state=42)

#fitting model
dt_model_1.fit(X_train, y_train)
```

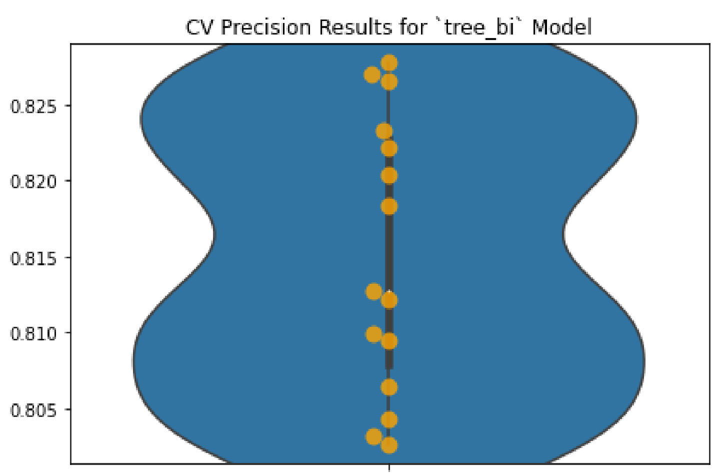
Out[905]:
```
DecisionTreeClassifier(random_state=42)
```

Base Model Analysis

The base model has a cross validation score of 81%, which is an improvement from the dummy model. The model does not appear to be overfit.

In [989]:
```
#cross validation of scores
model_results = ModelWithCV(model = dt_model_1, model_name='tree_bi',X=X_train, y=y_train)

# Plot CV results
fig, ax = plt.subplots()
ax = model_results.plot_cv(ax)
plt.tight_layout();

# Print CV results
model_results.print_cv_summary()
```

```
CV Results for `tree_bi` model:
            0.81510 ± 0.00875 precision

None
```



In [1047]:
```
#making a dataframe out of the column
cv_df = get_cross_val_dataframe(dt_model_1, X_train, y_train, kfolds=5, scoring="precision'
model_cv_mean = pd.DataFrame(np.mean(cv_df), columns=['Base Model'])
```

In [1048]:
```
model_cv_mean
```

Out[1048]:

	Base Model
precision	0.809689

The base model has precision score of 72%. This appears to be good, however due to the class imbalance of the target variable, the model does not perform well on predicting ratings below 4 starts. This means I will need to adjust the weight of the model to fix this distribution.

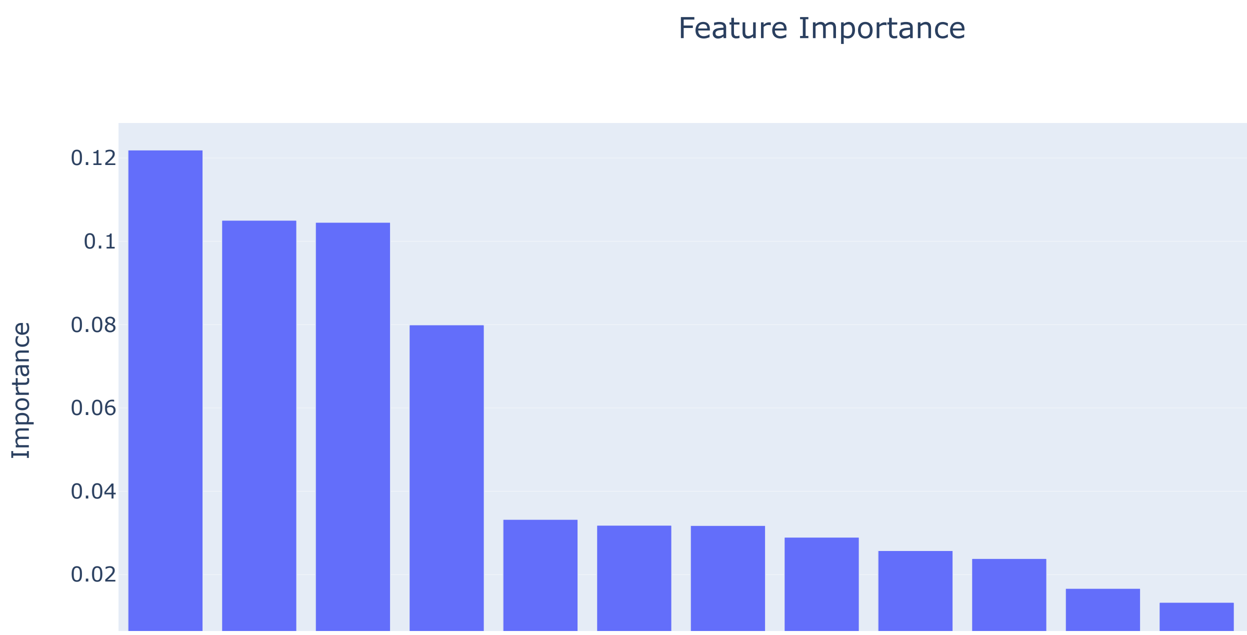
Feature Importance Function

To analyze the feature importance of the variables, I have defined a function to plot the top 15 features and display the top 20 features in a DataFrame to further analyze additional features.

In [1147]:
```
#deifning function to plot feature importance
def plot_top_15_features(model, x_train_var, feature_vals):

    tree_features = (model.feature_importances_)
    tree_features = pd.DataFrame(tree_features)

    #renaming column
    tree_features["Feature"] = x_train_var.columns.values

    #sorting the top 15 most important features
    top_15_features = tree_features.sort_values(0,ascending=False).head(feature_vals)

    #renaming feature column
    top_15_features.rename(columns={0:"Importance"}, inplace=True)

    #using plotly to plot the top features
    fig = px.bar(top_15_features, x="Feature", y="Importance", title = "Top Features")
    fig.update_layout(title_text='Feature Importance', title_x=0.5)
    fig.update_traces(marker_color = "#193d99")
    fig.show()

    #displaying the top 20 features for further analysis
    display(pd.DataFrame(tree_features.sort_values(0,ascending=False)).head(20))
```

#1 Base Model - Decision Tree Feature Importance

In [907]:
```
plot_top_15_features(dt_model_1, X_train, 15)
```



	0	Feature
8	0.121926	price
3	0.105058	longitude
2	0.104561	latitude
18	0.079943	dishes
0	0.033244	host_response_rate
4	0.031836	accommodates
1	0.031775	host_is_superhost
7	0.028966	beds
12	0.025750	essentials
5	0.023850	bathrooms
6	0.016676	bedrooms
41	0.013331	pets
24	0.011495	heating
9	0.011105	parking
27	0.010993	pool
39	0.010657	cable
69	0.010210	neighbourhood_cleansed_Rethymno
38	0.009966	smoking
25	0.009403	shampoo
33	0.009305	beach_access

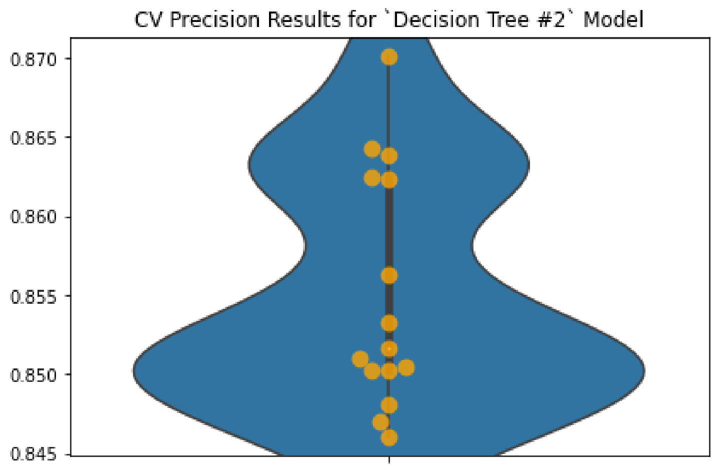
Model #2 - Decision Tree w/ Hyperparameter Adjustments

In [896]:
```
#instantiating model
dt_model_2 = DecisionTreeClassifier(criterion='entropy', random_state=42, min_samples_leaf=
                                    class_weight="balanced")

#fitting model
dt_model_2.fit(X_train, y_train)
```

Out[896]:
```
DecisionTreeClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=10, min_samples_leaf=4, random_state=42)
```

In [909]:
```
#cross validation of scores
model_results_2 = ModelWithCV(model= dt_model_2, model_name='Decision Tree #2',X=X_train, y

# Plot CV results
fig, ax = plt.subplots()
ax = model_results_2.plot_cv(ax)
plt.tight_layout();

# Print CV results
model_results_2.print_cv_summary()
```

```
CV Results for `Decision Tree #2` model:
            0.85517 ± 0.00727 precision
```



In [1049]:
```
#making a dataframe out of the column
cv_df = get_cross_val_dataframe(dt_model_2, X_train, y_train, kfolds=5, scoring="precision'
model_cv_mean['Decision Tree #2'] = np.mean(cv_df)
```

In [1050]:
```
model_cv_mean
```

Out[1050]:

	Base Model	Decision Tree #2
precision	0.809689	0.849551

In [ ]:
```
plot_top_15_features(dt_model_2, X_train, 15)
```

Model #3 - Random Forest

In [938]:
```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(max_features='sqrt', max_samples=0.3, random_state=1,class_
forest.fit(X_train, y_train)
```

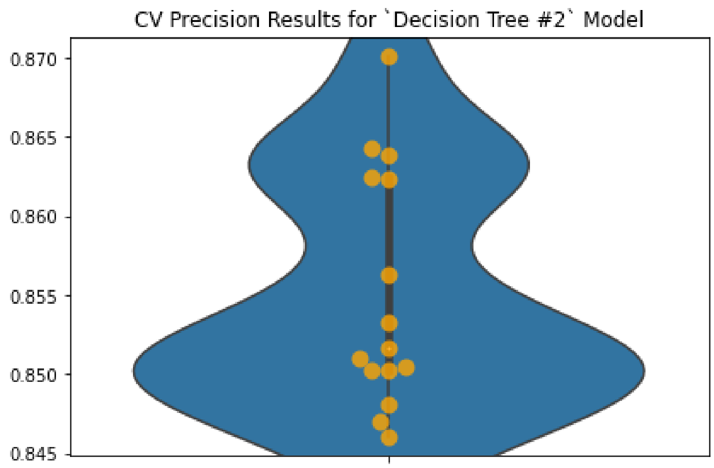
Out[938]:
```
RandomForestClassifier(class_weight='balanced', max_features='sqrt',
                       max_samples=0.3, random_state=1)
```

In [913]:
```
forest.score(X_train, y_train)
```

Out[913]:
```
0.9015569753737361
```

In [915]:
```
#cross validation of scores
forest_model_results = ModelWithCV(model= forest, model_name='Random Forest #2',X=X_train,

# Plot CV results
fig, ax = plt.subplots()
ax = model_results_2.plot_cv(ax)
plt.tight_layout();

# Print CV results
forest_model_results.print_cv_summary()
```

```
CV Results for `Random Forest #2` model:
            0.80529 ± 0.00543 precision
```



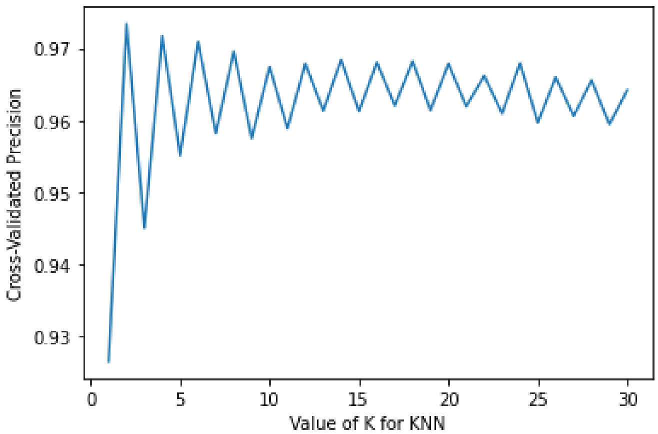
In [1051]:
```
cv_df = get_cross_val_dataframe(forest, X_train, y_train, kfolds=5, scoring="precision")
model_cv_mean['Random Forest'] = np.mean(cv_df)
model_cv_mean
```

Out[1051]:

	Base Model	Decision Tree #2	Random Forest
precision	0.809689	0.849551	0.804428

In [ ]:
```
plot_top_15_features(forest, X_train, 15)
```

In [918]:
```
import matplotlib.pyplot as plt
%matplotlib inline

# choose k between 1 to 31
k_range = range(1, 31)
k_scores = []

# use iteration to caclulator different k in models, then return the average accuracy based
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_t, y_t, cv=5, scoring='precision')
    k_scores.append(scores.mean())

    # plot to see clearly
plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Precision')
plt.show()
```



Decision Tree - Grid Search

In this grid search, I will be adjusting hyperparameters to fix the class imbalance and determine the best depth and sample leafs for the decision tree.

Due to the large class imbalance, I have parced through a variety of weights.

Currently, their is a large class imbalance.

In [935]:
```
grid = {
    'criterion':['entropy', 'gini'],
    'max_depth': [5, 6, 7, 10, 12],
    'min_samples_leaf': [1, 3, 5],
    'class_weight': ['balanced']
}

gs = GridSearchCV(estimator=DecisionTreeClassifier(), param_grid=grid, cv=5, verbose=2, sco

gs.fit(X_train, y_train)
```

```
[CV] class_weight=balanced, criterion=gini, max_depth=10, min_samples_leaf=5
[CV]  class_weight=balanced, criterion=gini, max_depth=10, min_samples_leaf=5, total=
0.1s
[CV] class_weight=balanced, criterion=gini, max_depth=10, min_samples_leaf=5
[CV]  class_weight=balanced, criterion=gini, max_depth=10, min_samples_leaf=5, total=
0.1s
[CV] class_weight=balanced, criterion=gini, max_depth=10, min_samples_leaf=5
[CV]  class_weight=balanced, criterion=gini, max_depth=10, min_samples_leaf=5, total=
0.1s
[CV] class_weight=balanced, criterion=gini, max_depth=12, min_samples_leaf=1
[CV]  class_weight=balanced, criterion=gini, max_depth=12, min_samples_leaf=1, total=
0.1s
[CV] class_weight=balanced, criterion=gini, max_depth=12, min_samples_leaf=1
[CV]  class_weight=balanced, criterion=gini, max_depth=12, min_samples_leaf=1, total=
0.1s
[CV] class_weight=balanced, criterion=gini, max_depth=12, min_samples_leaf=1
[CV]  class_weight=balanced, criterion=gini, max_depth=12, min_samples_leaf=1, total=
0.1s
[CV] class_weight=balanced, criterion=gini, max_depth=12, min_samples_leaf=1
[CV]  class_weight=balanced, criterion=gini, max_depth=12, min_samples_leaf=1, total=
```

In [939]:
```
# Best Hyperparameters
dt_gs = gs.best_params_
dt_gs
```

Out[939]:
```
{'class_weight': 'balanced',
 'criterion': 'entropy',
 'max_depth': 7,
 'min_samples_leaf': 3}
```

In [940]:
```
# Best CV score mean
dt_gs_best_cv = gs.best_score_
dt_gs_best_cv
```

Out[940]:
```
0.861690134463528
```

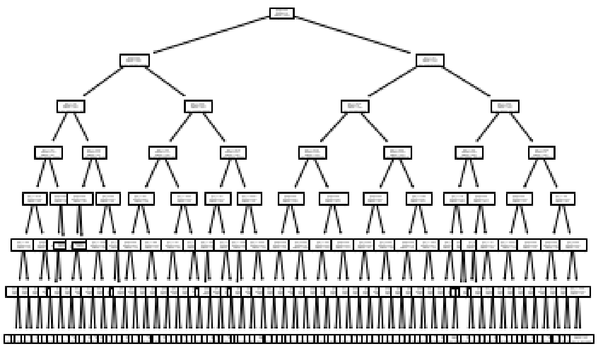
In [941]:
```
# We can find the best estimator
dt_best_model = gs.best_estimator_
dt_best_model
```

Out[941]:
```
DecisionTreeClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=7, min_samples_leaf=3)
```

In [952]:
```
#saving final decision tree best model params
dt_best_model = DecisionTreeClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=7, min_samples_leaf=3)

#fitting model
dt_best_model.fit(X_train, y_train)

dt_best_model
```

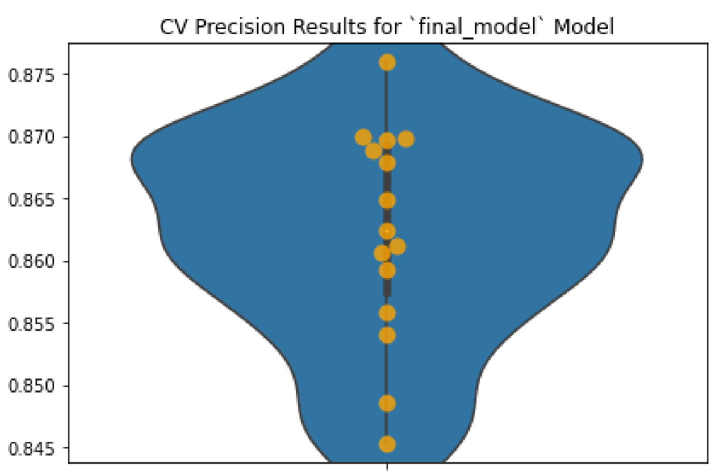
Out[952]:
```
DecisionTreeClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=7, min_samples_leaf=3)
```

In [953]:
```
#plotting tree
plot_tree(dt_best_model);
```



In [954]:
```
final_model_results = ModelWithCV(
                                model=best_model,
                                model_name='final_model',
                                X=X_train,
                                y=y_train)

# Plot CV results
fig, ax = plt.subplots()
ax = final_model_results.plot_cv(ax)
plt.tight_layout();


# Print CV results
model_results.print_cv_summary()
```

```
CV Results for `tree_bi` model:
            0.81510 ± 0.00875 precision
```



In [1052]:
```
cv_df = get_cross_val_dataframe(dt_best_model, X_train, y_train, kfolds=5, scoring="precis
model_cv_mean['Decision Tree - Grid Search'] = np.mean(cv_df)
model_cv_mean
```

Out[1052]:

	Base Model	Decision Tree #2	Random Forest	Decision Tree - Grid Search
precision	0.809689	0.849551	0.804428	0.862239

Random Forest - Grid Search

In [947]:
```
grid = {
    'max_features':['sqrt', 'gini', "entropy"],
    'max_samples': [.1 , .5, .7],
    'class_weight': ['balanced','balanced_subsample']
}

gs = GridSearchCV(estimator=RandomForestClassifier(), param_grid=grid, cv=5, verbose=2)

gs.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
[CV] class_weight=balanced, max_features=sqrt, max_samples=0.1 .......

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[CV]  class_weight=balanced, max_features=sqrt, max_samples=0.1, total=   0.5s
[CV] class_weight=balanced, max_features=sqrt, max_samples=0.1 .......

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.5s remaining:    0.0s

[CV]  class_weight=balanced, max_features=sqrt, max_samples=0.1, total=   0.5s
[CV] class_weight=balanced, max_features=sqrt, max_samples=0.1 .......
[CV]  class_weight=balanced, max_features=sqrt, max_samples=0.1, total=   0.5s
[CV] class_weight=balanced, max_features=sqrt, max_samples=0.1 .......
[CV]  class_weight=balanced, max_features=sqrt, max_samples=0.1, total=   0.5s
[CV] class_weight=balanced, max_features=sqrt, max_samples=0.1 .......
[CV]  class_weight=balanced, max_features=sqrt, max_samples=0.1, total=   0.5s
[CV] class_weight=balanced, max_features=sqrt, max_samples=0.5 .......
[CV]  class_weight=balanced, max_features=sqrt, max_samples=0.5, total=   1.2s
[CV] class_weight=balanced, max_features=sqrt, max_samples=0.5 .......
```

In [948]:
```
# Best Hyperparameters
rf_gs = gs.best_params_
rf_gs
```

Out[948]:
```
{'class_weight': 'balanced_subsample',
 'max_features': 'sqrt',
 'max_samples': 0.7}
```

In [949]:
```
# Best CV score mean
rf_gs_best_cv = gs.best_score_
rf_gs_best_cv
```

Out[949]:
```
0.797655663915979
```

In [950]:
```
# We can find the best estimator
rf_best_model = gs.best_estimator_
rf_best_model
```

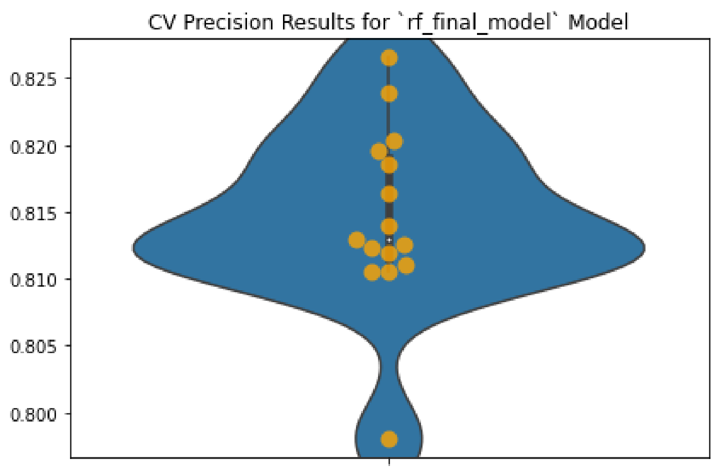
Out[950]:
```
RandomForestClassifier(class_weight='balanced_subsample', max_features='sqrt',
                       max_samples=0.7)
```

In [951]:
```
#saving final random forest best model params
rf_best_model = RandomForestClassifier(class_weight='balanced_subsample', max_features='sq
                       max_samples=0.7)

#fitting
rf_best_model.fit(X_train, y_train)

rf_best_model
```

Out[951]:
```
RandomForestClassifier(class_weight='balanced_subsample', max_features='sqrt',
                       max_samples=0.7)
```

In [955]:
```
final_model_results = ModelWithCV(
                        model=rf_best_model,
                        model_name='rf_final_model',
                        X=X_train_2,
                        y=y_train_2)

# Plot CV results
fig, ax = plt.subplots()
ax = final_model_results.plot_cv(ax)
plt.tight_layout();

# Print CV results
final_model_results.print_cv_summary()
```

```
CV Results for `rf_final_model` model:
            0.81462 ± 0.00656 precision
```



In [1053]:
```
cv_df = get_cross_val_dataframe(rf_best_model, X_train, y_train, kfolds=5, scoring="precis
model_cv_mean['Random Forest - Grid Search'] = np.mean(cv_df)
model_cv_mean
```

Out[1053]:

	Base Model	Decision Tree #2	Random Forest	Decision Tree - Grid Search	Random Forest - Grid Search
precision	0.809689	0.849551	0.804428	0.862239	0.81129

Final Model Selection - Decision Tree from Grid Search

After analyzing the cross vaildation score, **dt_best_model** is the best performing model on the training data.

After looking at the testing data, the overall performance dropped from **86% to 81%**. This is only a slight drop, and shows that the model is performing well.

The precision for the target class, "Above 4 Stars" is slightly higher, which is not surprising given the large class imbalance, which was addressed by adjusting the weights.

In [1054]:
```
model_cv_mean
```

Out[1054]:

	Base Model	Decision Tree #2	Random Forest	Decision Tree - Grid Search	Random Forest - Grid Search
precision	0.809689	0.849551	0.804428	0.862239	0.81129

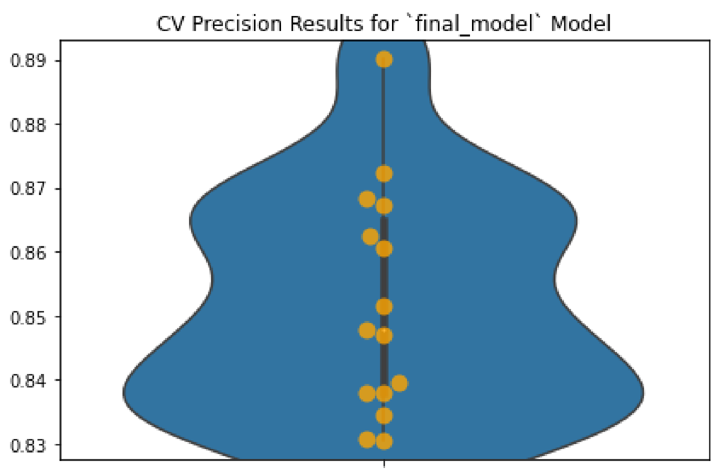
In [1168]:
```
y_pred_test = rf_best_model.predict(X_test)

print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.73      0.41      0.52      1891
           1       0.81      0.94      0.87      5018

    accuracy                           0.80      6909
   macro avg       0.77      0.67      0.70      6909
weighted avg       0.79      0.80      0.77      6909
```

Validating on test data

In [963]:
```
final_model_results = ModelWithCV(
                        model=best_model,
                        model_name='final_model',
                        X=X_test,
                        y=y_test)

# Plot CV results
fig, ax = plt.subplots()
ax = final_model_results.plot_cv(ax)
plt.tight_layout();


# Print CV results
model_results.print_cv_summary()
```

```
CV Results for `tree_bi` model:
            0.81510 ± 0.00875 precision
```



In [1059]:
```
cv_df = get_cross_val_dataframe(rf_best_model, X_test, y_test, kfolds=5, scoring="precisio
model_cv_mean['Final Decision Tree'] = np.mean(cv_df)
```

In [1231]:
```
transformed_model = model_cv_mean.transpose()
transformed_model.reset_index(inplace=True)
transformed_model.rename(columns={"index":"Model", "precision":"CV Precision Mean"}, inpla
```

In [1232]:
```
keep = transformed_model
keep
```

Out[1232]:

	Model	CV Precision Mean
0	Base Model	0.809689
1	Decision Tree #2	0.849551
2	Random Forest	0.804428
3	Decision Tree - Grid Search	0.862239
4	Random Forest - Grid Search	0.811290
5	Final Decision Tree	0.793948

In [1250]:
```
transformed_model
```

Out[1250]:

	Model	CV Precision Mean
0	Base Model	0.81
3	Final Model	0.86

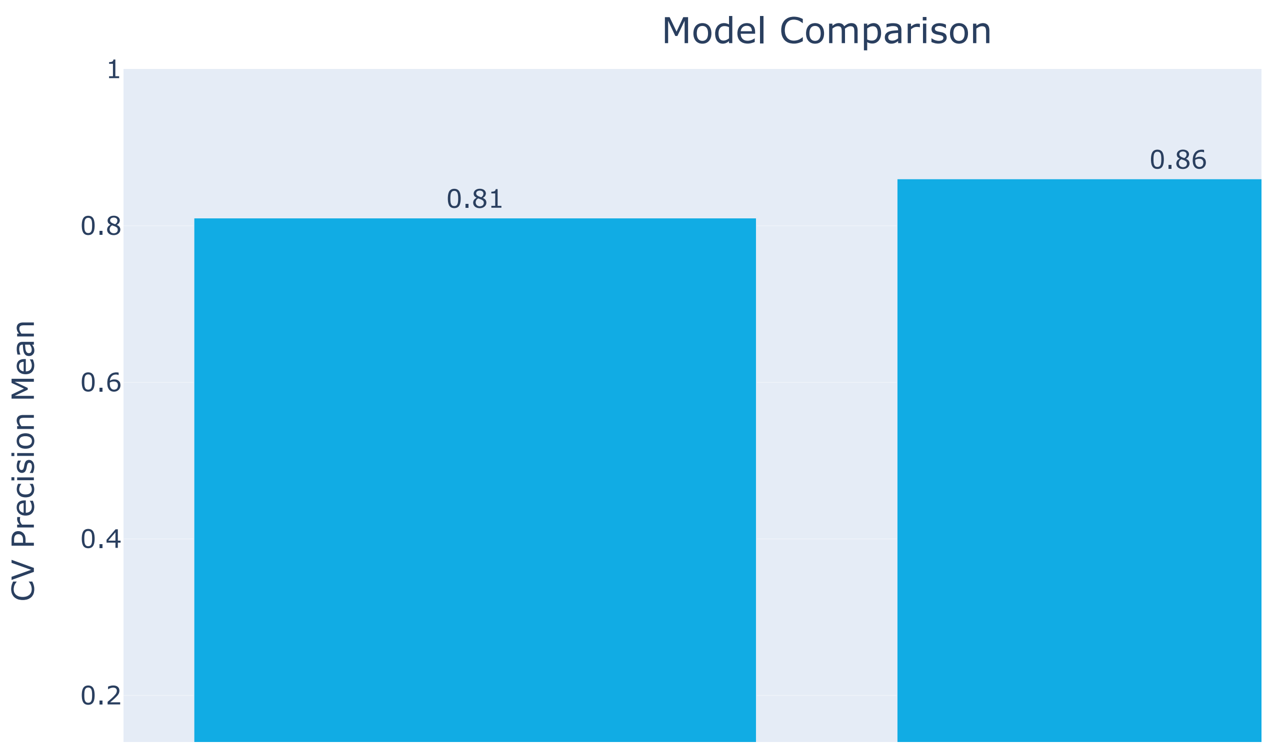
In [1233]:
```
transformed_model['CV Precision Mean'] = transformed_model['CV Precision Mean'].round(2)
```

In [1235]:
```
transformed_model = transformed_model.loc[(transformed_model['Model'] == "Base Model") | (t
```

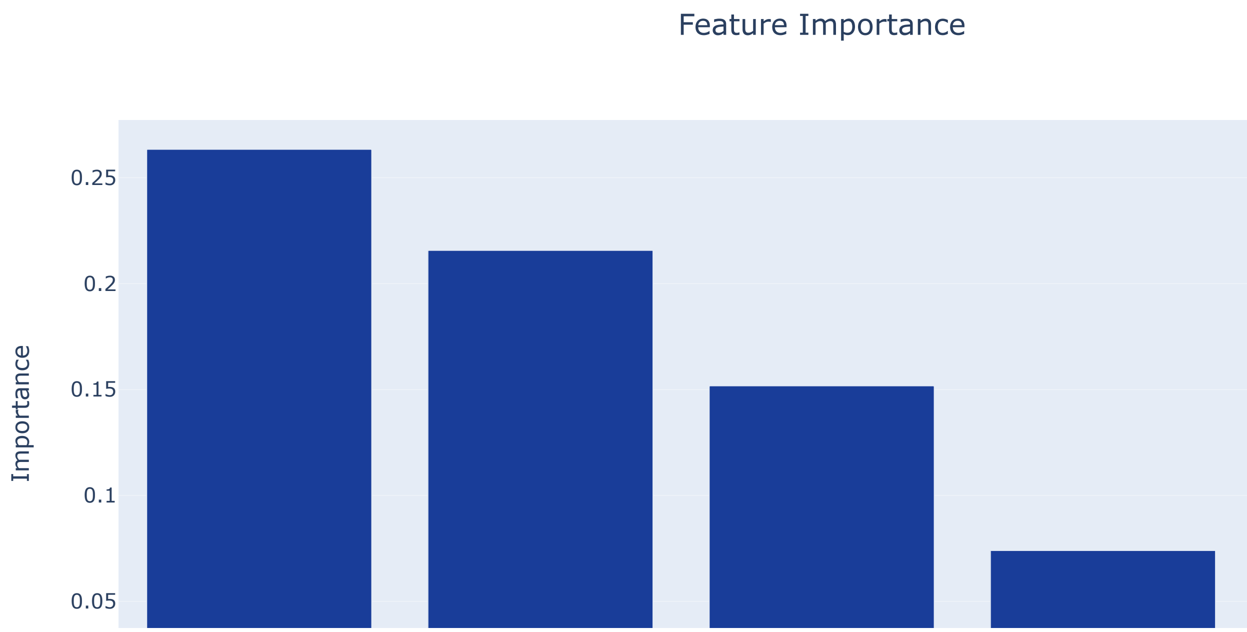
In [1249]:
```
transformed_model['Model'] = transformed_model['Model'].str.replace('Decision Tree - Grid S
```

```
<ipython-input-1249-cb077641adad>:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [1251]:
```
import plotly.graph_objects as go

#plotting bar chart
fig = px.bar(transformed_model, x="Model", y="CV Precision Mean",height=600, text="CV Prec:

#centering title
fig.update_layout(title_text='Model Comparison', title_x=0.5, title_y=0.95, xaxis_type="cat
                  font = dict(size=15),paper_bgcolor="#ffffff")

fig.update_traces(marker_color = "#11ace4", textposition='outside')

fig.update_yaxes(range=[0, 1])

newnames = {'Base Model':'Base', 'Decision Tree - Grid Search': 'Final'}

fig.show()
```



In [1384]:
```
plot_top_15_features(dt_best_model, X_test, 5)
```



	0	Feature
18	0.263396	dishes
8	0.215680	price
1	0.151749	superhost
0	0.074012	host response rate
12	0.056199	essentials
3	0.035790	longitude
5	0.022886	bathrooms
2	0.017281	latitude
19	0.017228	refrigerator
24	0.013655	heating
44	0.013467	electric
26	0.012690	stove
47	0.009056	dinnerware
25	0.008872	shampoo
46	0.008474	beachfront
9	0.007239	parking
41	0.006615	pets
7	0.005821	beds
33	0.005444	beach_access
43	0.004453	utensils

In [1132]:

```
tree_features = (dt_best_model.feature_importances_)
tree_features = pd.DataFrame(tree_features)

#renaming column
tree_features["Feature"] = X_test.columns.values

#sorting the top 15 most important features
top_15_features = tree_features.sort_values(0,ascending=False).head(20)

#renaming feature column
top_15_features.rename(columns={0:"Importance"}, inplace=True)
```

In [1133]:
```
top_15_features
```

Out[1133]:

	Importance	Feature
18	0.263396	dishes
8	0.215680	price
1	0.151749	host_is_superhost
0	0.074012	host_response_rate
12	0.056199	essentials
3	0.035790	longitude
5	0.022886	bathrooms
2	0.017281	latitude
19	0.017228	refrigerator
24	0.013655	heating
44	0.013467	electric
26	0.012690	stove
47	0.009056	dinnerware
25	0.008872	shampoo
46	0.008474	beachfront
9	0.007239	parking
41	0.006615	pets
7	0.005821	beds
33	0.005444	beach_access
43	0.004453	utensils

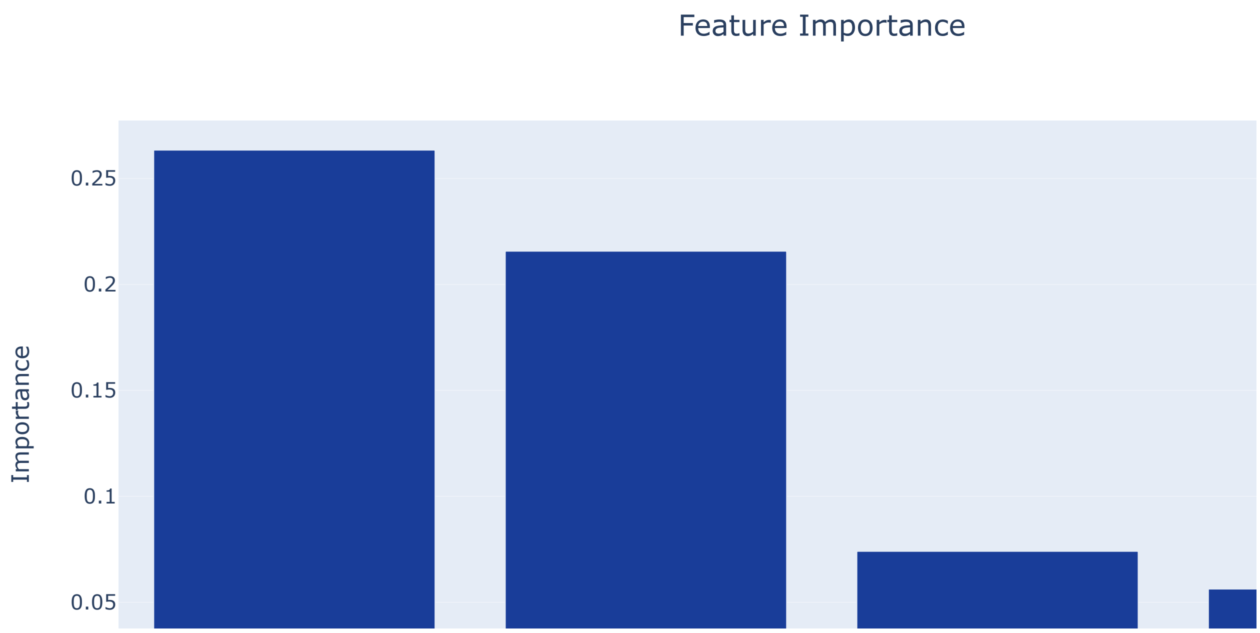
In [1134]:
```
top_15_features['Feature'] = top_15_features['Feature'].astype(str)
```

In [264]:
```
top_15_features_no = top_15_features[top_15_features['Feature'].str.contains("host_is_super
top_15_features_no.head()
```

In [1279]:
```
top_15_features = top_15_features.rename(columns={"host_response_rate":"host response rate'
```

In [1376]:
```
top_15_features_no = top_15_features_no.rename(columns={"host_response_rate":"host response
```

In [1146]:
```
#using plotly to plot the top featurescolor=
fig = px.bar(top_15_features_no.head(4), x="Feature", y="Importance", title = "Top Features
fig.update_layout(title_text='Feature Importance', title_x=0.5)
fig.update_traces(marker_color = "#193d99")
fig.show()
```



In [ ]:
```
#using plotly to plot the top features
fig = px.bar(top_15_features.head(5), x="Feature", y="Importance", title = "Top Features")
fig.update_layout(title_text='Feature Importance', title_x=0.5)
fig.show()
```

In [1252]:
```
above_4_stars = model_dt_df.loc[model_dt_df]
```

Out[1252]:

	host_response_rate	host_is_superhost	neighbourhood_cleansed	latitude	longitude	room_type	accommodates
0	1.0	0	Heraklion	35.340050	25.128090	Entire home/apt	6
1	1.0	1	Malevizi	35.331980	25.081820	Entire home/apt	3
2	1.0	1	Rethymno	35.354410	24.587130	Entire home/apt	6
3	1.0	1	Rethymno	35.355960	24.584100	Entire home/apt	12
4	0.0	0	Kissamos	35.497620	23.697680	Entire home/apt	6
...	...	...	...	...	...	...	...
23053	0.0	0	Agios Vasilios	35.169409	24.452855	Entire home/apt	7
23054	1.0	1	Amari	35.247463	24.567706	Entire home/apt	15
23055	0.9	0	Heraklion	35.339489	25.121428	Entire home/apt	2
23056	1.0	0	Kissamos	35.499830	23.647820	Entire home/apt	3
23057	1.0	0	Rethymno	35.379069	24.584246	Private room	4

23030 rows × 81 columns

In [ ]:
```
#plotting bar chart
fig = px.bar(model_dt_df, x="is_grte_4stars", y="host_is_superhost",height=600, text="host_
            labels={1:'Above 4 Stars', 0: 'Below 4 Stars'})

fig.update_layout(title_text='Feature Importance', title_x=0.5)

fig.show()
```

In [1309]:
```
model_dt_df['price'].sort_values(ascending=False)
```

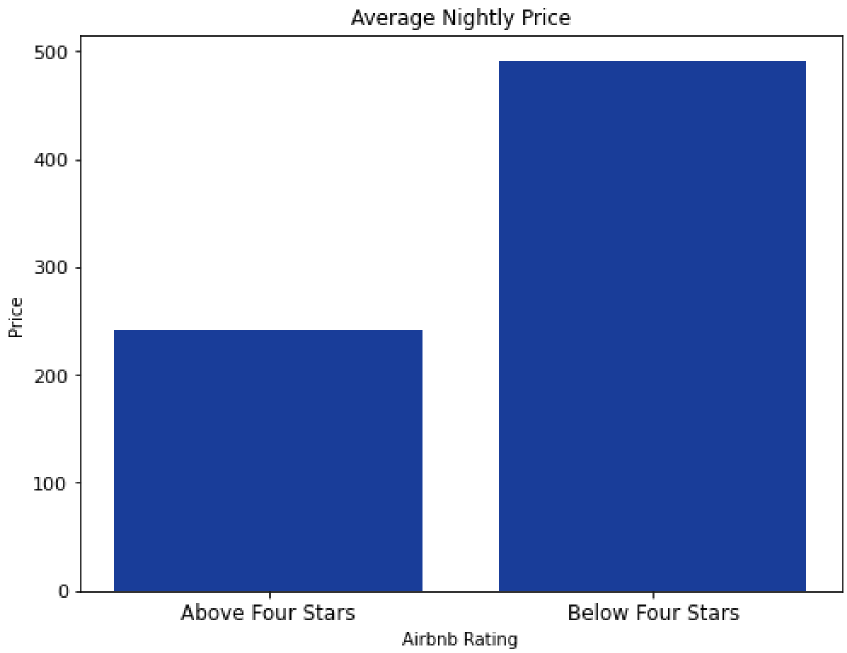
Out[1309]:
```
12385    20079.0
10332    17221.0
8423     15960.0
17963    11600.0
17968    11600.0
          ...
15293       10.0
17863       10.0
4171        10.0
9546        10.0
21002       10.0
Name: price, Length: 23030, dtype: float64
```

In [1293]:
```
four_star = model_dt_df[["is_grte_4stars", "host_is_superhost"]].copy()
```

In [1296]:
```
four_star.rename(columns={"is_grte_4stars":"above four stars", "host_is_superhost":"host is
```

In [ ]:
```
model_dt_df.rename()
```

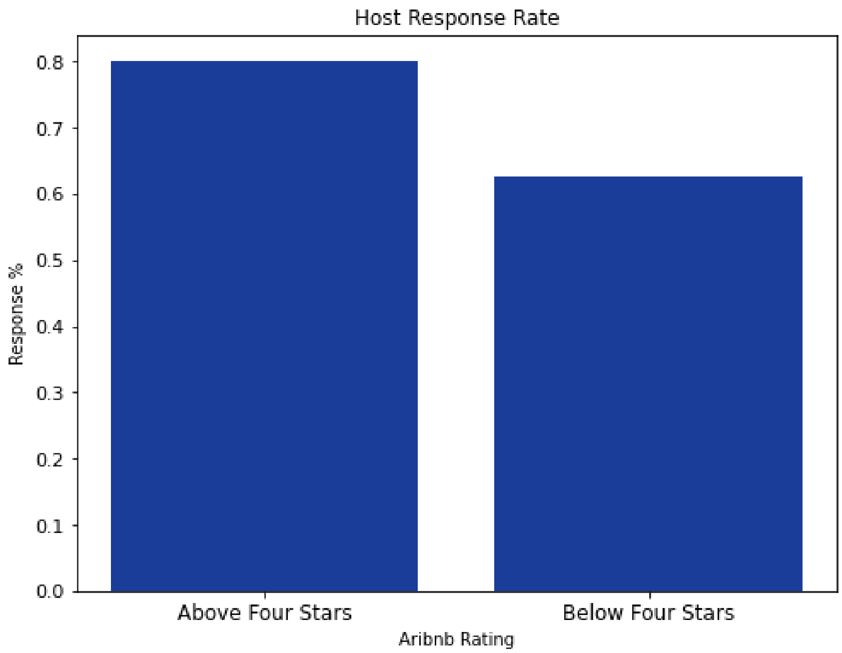
In [1371]:
```
# group the data by the "is_grte_4stars" column and calculate the mean price for each grou
grouped = final_feature_df.groupby("is_grte_4stars")["price"].mean()

fig = plt.figure(figsize = (8, 6))

# create a bar plot
plt.bar(grouped.index, grouped.values, color="#193d99")

# set the x-axis label
plt.xlabel("Airbnb Rating")

# set the y-axis label
plt.ylabel("Price")

#adding title
plt.title("Average Nightly Price")

# show the plot
plt.show()
```



In [1372]:
```
final_feature_df.groupby("is_grte_4stars")["price"].mean()
```

Out[1372]:
```
is_grte_4stars
Above Four Stars    241.648737
Below Four Stars    490.650706
Name: price, dtype: float64
```

In [1386]:
```
# group the data by the "is_grte_4stars" column and calculate the mean price for each grou
grouped = final_feature_df.groupby("is_grte_4stars")["host_response_rate"].mean()

fig = plt.figure(figsize = (8, 6))

# create a bar plot
plt.bar(grouped.index, grouped.values, color="#193d99")

# set the x-axis label
plt.xlabel("Aribnb Rating")

# set the y-axis label
plt.ylabel("Response %")

#adding title
plt.title("Host Response Rate")

# show the plot
plt.show()
```



In [1387]:
```
final_feature_df.groupby("is_grte_4stars")["host_response_rate"].mean()
```

Out[1387]:
```
is_grte_4stars
Above Four Stars    0.800374
Below Four Stars    0.626025
Name: host_response_rate, dtype: float64
```

In [1332]:
```
four_stars = feature_df.loc[feature_df["airbnb_rating"] == "4 Stars" | feature_df["airbnb_
```

Conclusions

As a result of this analysis, three features with the highest feature importance have been identified as the most important in classifying airbnb ratings in Crete:

Price

Increasing footage of home by approximately 964sq.ft increases the price by a factor of 1.227 or 22.7%.

Home Essentials

Kitchen and home essentials are among the strongest features that impact rating. It is important that hosts stock on on these.

Host Response Rate

Responsive hosts lead to higher ratings

Next Steps

From the initial modeling research, it is clear that the number of ratings and overall ratings play a large role in predicting whether an airbnb listing will be highly rated. To gain a more comprehensive understanding of why it would be beneficial to do further reserach into this to determine what other features are important in prediticting prices.

In addition, it would be beneficial to examine more airbnb data to improve on the class imbalance that was present.