Linear Regression on LA Airbnb Data:

Predicting Airbnb Rental Price using Multiple Linear Regression

Alexander Vaillant

College of Information Technology, Western Governors University

D214: MSDA Capstone

Dr. Daniel Smith

August 27, 2021

Research Question

“To what extent do the independent variables of Airbnb rentals predict the rental price in the Los Angeles Market?**”**

Justification for Research Question

According to the 2021 Airbnb Statistics Article released by Steve Deane, there are over “14,000 new hosts joining Airbnb each month in 2021” (Deane, 2021). Deane references Airbnb statistics from 2019 which deemed Los Angeles, California as one of the most popular cities for Airbnb in the US (Deane, 2021). To successfully enter a competitive market like Los Angeles, new hosts must know the financial impact that different features of an Airbnb rental have on its price. In their paper “Real Estate Value Prediction Using Linear Regression”, Ghosalkar and Dhage utilize linear regression to predict the value of real estate (Ghosalkar & Dhage, 2018). Multiple Linear Regression has been a proven method used to accurately predict price based on various features while accounting for their impact on the variance of price.

Context

The contribution of this study to the MSDA program and the Data Analytics field is to create a predictive model which approximates an Airbnb’s rental price so that a new host in the Los Angeles market may gauge a potential property’s affordability and revenue against competitors. With 32,241 listings in the Los Angeles market, price and the variables with influence on price play a crucial role in the revenue of an Airbnb. In this study, a Multiple Linear Regression model will be utilized to analyze the statistical significance of independent, or predictor, variables which have the most influence on an Airbnb’s rental price (dependent variable). When these highly influential predictor variables are known, a host may cater to those areas to attract customers. “Multiple regression allows for a relationship to be modeled between multiple independent variables and a single dependent variable where the independent variables are being used to predict the dependent variable” (Laerd Statistics, 2015). In their paper “Real Estate Value Prediction Using Linear Regression”, Ghosalkar and Dhage utilize linear regression to predict the value of real estate (Ghosalkar & Dhage, 2018). Like with real estate value, linear regression can be used to predict the rental price of an Airbnb rental.

Null and Alternate Hypotheses

The null hypothesis (H_0) of this statistical analysis is that a statistically significant model cannot be created to predict the Airbnb rental price. The alternate hypothesis (H_a) is that a statistically significant model can be created to predict the Airbnb rental price. The acceptance or rejection of the null hypothesis will be decided based on the p-value of the Multiple Linear Regression model created.

Data Collection

Description of Relevant Data

The data needed to be collected for this study is publicly available through Inside Airbnb website. "Inside Airbnb is an independent, non-commercial set of tools and data that allows you to explore how Airbnb is really being used in cities around the world" (Cox, n.d.). The data was compiled from public information on the Airbnb website by Murray Cox. Before any data cleansing and removal is done, the Los Angeles dataset contains 32,241 rows with 74 columns and data sparsity of less than 10%. Host demographic and PII data will be removed before analysis.

This dataset is available through Inside Airbnb here: http://insideairbnb.com/get-the-data.html. The study will place a delimitation on the dataset by limiting the neighborhood group to only the city of Los Angeles. There are two other neighborhood groups with observations that will be removed. Another potential delimitation would be the number of features allowed in the model based on stepwise regression. The limitations of this dataset are the presence of host PII and the data sparsity of 10% in both categorical and continuous variables. These limitations can be worked around in the data cleansing process. After PII data is removed, the dataset contains the following 31 usable variables:

Variable	Type	Intention
Id	Continuous	Index
host_response_time	Categorical	Predictor/Independent

host_response_rate	Continuous	Predictor/Independent
host_acceptance_rate	Continuous	Predictor/Independent
host_is_superhost	Categorical	Predictor/Independent
host_listings_count	Continuous	Predictor/Independent
host_has_profile_pic	Categorical	Predictor/Independent
host_identity_verified	Categorical	Predictor/Independent
neighbourhood_group_cleansed	Categorical	Predictor/Independent
room_type	Categorical	Predictor/Independent
accommodates	Continuous	Predictor/Independent
bathrooms_text	Continuous (once cleansed)	Predictor/Independent
bedrooms	Continuous	Predictor/Independent
beds	Continuous	Predictor/Independent
price	Continuous	Response/Dependent
minimum_nights	Continuous	Predictor/Independent
maximum_nights	Continuous	Predictor/Independent
has_availability	Categorical	Predictor/Independent
availability_30	Continuous	Predictor/Independent
availability_60	Continuous	Predictor/Independent
availability_90	Continuous	Predictor/Independent
availability_365	Continuous	Predictor/Independent
number_of_reviews	Continuous	Predictor/Independent
review_scores_rating	Continuous	Predictor/Independent
review_scores_accuracy	Continuous	Predictor/Independent
review_scores_cleanliness	Continuous	Predictor/Independent
review_scores_checkin	Continuous	Predictor/Independent
review_scores_communication	Continuous	Predictor/Independent
review_scores_location	Continuous	Predictor/Independent
review_scores_value	Continuous	Predictor/Independent
instant_bookable	Categorical	Predictor/Independent

The data was compiled from public information on the Airbnb website by Murray Cox. “The data is available under a Creative Commons CC0 1.0 Universal (CC0 1.0) ‘Public Domain Dedication’ license” (Cox, n.d.). Based on the Creative Commons CC0 1.0 Universal license, a user can “copy modify, distribute and perform the work, even for commercial purposes, all without asking permission” (Creative Commons, n.d.). it can be used for commercial purposes. There are several other Airbnb datasets available on Kaggle with public domain licenses. For example, another dataset can be found at https://www.kaggle.com/kritikseth/us-airbnb-open-data with the CC0: Public Domain license. All host PII data will be removed at the start of the analysis to increase privacy.

The dataset will be downloaded from the Inside Airbnb website in .gz format. From there, the listings.csv.gz can be scraped with a few for loops in Python and the final listings.csv file will be exported to be cleansed of host PII data in excel. The sparsity in this dataset is less than 10%.

Advantages and Disadvantages of Data-Gathering Methodology

With the dataset publicly available on Insider Airbnb, an advantage is the cited Creative Commons Public Domain License. This allows the user to distribute and analyze the data for commercial purposes without legal consequences. Another advantage is the data compilation's monthly frequency. In the current Covid pandemic, data and situations are undergoing continuous change, and having up-to-date data is pertinent for success in competitive markets.

In this published dataset, there are some removed and calculated features. This is a disadvantage without the presence of a fully completed data dictionary. Secondly, each feature contains missing values. Another disadvantage of this dataset is the inclusion of Airbnb host Personal Identifiable Information, or PII.

Overcoming Any Challenges Encountered in Data Collection Process

To overcome removed and calculated features, the dataset comes with a data dictionary to boost understanding of each feature. The data dictionary is thorough and explains most features. The challenge of missing values in features is overcome through K-Nearest Neighbors Imputation on continuous and binary dummy features transformed from categorical features. The final challenge of host PII is dealt with through the removal of any feature containing PII.

Data Extraction and Preparation

Extraction and Preparation Process

1. Import necessary libraries

Import necessary libraries

In [1]:
```
import pandas as pd
import numpy as np
from numpy import random
import gzip
import shutil
from sklearn.impute import KNNImputer
import re
import seaborn as sns
from sklearn.preprocessing import normalize
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import SequentialFeatureSelector
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

2. Extract data from original downloaded .gz file

Data Extraction

Extract data from original .gz file

In [4]:
```
# Extract data from the .gz file
with gzip.open('listings.csv.gz', 'rb') as file_in:
    with open('listings.csv', 'wb') as file_out:
        shutil.copyfileobj(file_in, file_out)
```

3. Load raw data into pandas dataframe for EDA and cleaning

Load raw data into a pandas dataframe for EDA and Cleaning

In [5]:
```
Airbnb_raw = pd.read_csv('C:/Users/tedda/Desktop/D214 MSDA Capstone/listings.csv', header = 0)
print("Airbnb Raw Data Shape:", Airbnb_raw.shape)
```

```
Airbnb Raw Data Shape: (32240, 74)
```

Currently, there are 74 features with several containing host PII. Avoided .head() to not show PII.

4. Select desired features without any host PII

Select desired features without host PII

In [6]:
```
#Remove PII by grabbing only desired columns
AirbnbDesiredColumns = Airbnb_raw[['host_response_time', 'host_response_rate',
       'host_acceptance_rate', 'host_is_superhost', 'host_listings_count',
       'host_has_profile_pic', 'host_identity_verified',
       'neighbourhood_group_cleansed', 'room_type', 'accommodates',
       'bathrooms_text', 'bedrooms', 'beds', 'price', 'minimum_nights',
       'maximum_nights', 'has_availability', 'availability_30',
       'availability_60', 'availability_90', 'availability_365',
       'number_of_reviews', 'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'instant_bookable']]
print("Airbnb Data Shape:", AirbnbDesiredColumns.shape)
AirbnbDesiredColumns.head()
```

```
Airbnb Data Shape: (32240, 30)
```

Out[6]:

	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost	host_listings_count	host_has_profile_pic	host_identi
0	NaN	NaN	NaN	f	1.0	t	
1	within an hour	100%	100%	t	2.0	t	
2	within an hour	100%	36%	t	2.0	t	
3	NaN	NaN	NaN	f	1.0	t	
4	within a few hours	100%	25%	f	6.0	t	

5 rows × 30 columns

The "id" column can be used to identify a host, which qualifies as PII. Decided to remove the "id" column as well. There are 30 variables left.

5. Perform Exploratory data analysis

   a. Check Initial Data Sparsity

Check initial Data Sparsity

In [7]:
```
# Get the initial data sparsity of each column.
# Goal: remove NaN values in all categorical variables so KNNImputer can be used on continuous variables
data_sparsity = AirbnbDesiredColumns.isnull().sum()/ len(AirbnbDesiredColumns)
data_sparsity
```

Out[7]:
```
host_response_time              0.266811
host_response_rate              0.266811
host_acceptance_rate            0.260577
host_is_superhost               0.001272
host_listings_count             0.001272
host_has_profile_pic            0.001272
host_identity_verified          0.001272
neighbourhood_group_cleansed    0.000000
room_type                       0.000000
accommodates                    0.000000
bathrooms_text                  0.001799
bedrooms                        0.115043
beds                            0.019789
price                           0.000000
minimum_nights                  0.000000
maximum_nights                  0.000000
has_availability                0.000000
availability_30                 0.000000
availability_60                 0.000000
availability_90                 0.000000
availability_365                0.000000
number_of_reviews               0.000000
review_scores_rating            0.247177
review_scores_accuracy          0.258065
review_scores_cleanliness       0.258033
review_scores_checkin           0.258344
review_scores_communication     0.258096
review_scores_location          0.258437
review_scores_value             0.258499
instant_bookable                0.000000
dtype: float64
```

b. Descriptive Summary of the Data

Descriptive Summary of the Data

In [8]:
```
AirbnbDesiredColumns.describe()
```

Out[8]:

	host_listings_count	accommodates	bedrooms	beds	minimum_nights	maximum_nights	avai
count	32199.000000	32240.000000	28531.000000	31602.000000	32240.000000	32240.000000	322
mean	36.371626	3.601427	1.665627	1.964401	20.317246	664.483623	
std	200.998227	2.539507	1.085385	1.552321	34.120765	506.685023	
min	0.000000	0.000000	1.000000	0.000000	1.000000	1.000000	
25%	1.000000	2.000000	1.000000	1.000000	2.000000	90.000000	
50%	2.000000	3.000000	1.000000	1.000000	30.000000	1125.000000	
75%	7.000000	4.000000	2.000000	2.000000	30.000000	1125.000000	
max	2232.000000	16.000000	15.000000	26.000000	1125.000000	10000.000000	

6. Limit data to only the City of Los Angeles (Delimitation #1)

Limit data to only the City of Los Angeles

In [9]:
```
# Delimitation 1: Limit the data to only the City of Los Angeles
AirbnbLA = AirbnbDesiredColumns[(AirbnbDesiredColumns['neighbourhood_group_cleansed'] == 'City of Los Angeles')]
print('New data shape:',AirbnbLA.shape)
```

```
New data shape: (17872, 30)
```

7. Check data sparsity after delimitation in case it eliminated missing values.

Check data sparsity after removal of two neighborhood groups

In [10]:
```
#Check current data sparsity with the removal of two neighborhood groups.
AirbnbLASparsity = AirbnbLA.isnull().sum()/len(AirbnbLA)
AirbnbLASparsity
```

Out[10]:
```
host_response_time             0.282285
host_response_rate             0.282285
host_acceptance_rate           0.285307
host_is_superhost              0.000112
host_listings_count            0.000112
host_has_profile_pic           0.000112
host_identity_verified         0.000112
neighbourhood_group_cleansed   0.000000
room_type                      0.000000
accommodates                   0.000000
bathrooms_text                 0.001846
bedrooms                       0.131770
beds                           0.023500
price                          0.000000
minimum_nights                 0.000000
maximum_nights                 0.000000
has_availability               0.000000
availability_30                0.000000
availability_60                0.000000
availability_90                0.000000
availability_365               0.000000
number_of_reviews              0.000000
review_scores_rating           0.284188
review_scores_accuracy         0.295546
review_scores_cleanliness      0.295490
review_scores_checkin          0.295826
review_scores_communication    0.295490
review_scores_location         0.295882
review_scores_value            0.295938
instant_bookable               0.000000
dtype: float64
```

8. Convert percentages and currency to Floats (for MLR Algorithm)

Convert percentages and currency to Floats

In [11]:
```
AirbnbLA['host_response_rate']=AirbnbLA['host_response_rate'].replace('%','', regex = True).astype(float)/100
AirbnbLA['host_acceptance_rate']=AirbnbLA['host_acceptance_rate'].replace('%','', regex = True).astype(float)/100
AirbnbLA['price']=AirbnbLA['price'].replace('[\$,]','', regex = True).astype(float)
AirbnbLA.head()
```

Out[11]:

	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost	host_listings_count	host_has_profile_pic	host_identity_v
1	within an hour	1.0	1.00	t	2.0	t	
4	within a few hours	1.0	0.25	f	6.0	t	
5	within a few hours	1.0	0.80	t	8.0	t	
6	within a few hours	1.0	0.80	t	8.0	t	
7	within a few hours	1.0	0.80	t	8.0	t	

5 rows × 30 columns

9. Parse Bathrooms_text column and convert to Float (for MLR Algorithm)

Parse Bathrooms_text column and convert to Float

In [13]:
```
AirbnbLA['bathrooms'] = AirbnbLA['bathrooms_text'].replace(
    ("Half-bath","Shared half-bath","Private half-bath"),"0.5 baths", regex = True).str.split(' ').str[0].astype(float)
AirbnbLA.drop('bathrooms_text', axis=1,inplace=True)

# New Value Counts
AirbnbLA['bathrooms'].value_counts()
```

Out[13]:
```
1.0     11615
2.0      2842
1.5       857
2.5       713
3.0       586
3.5       276
4.0       225
4.5       169
8.0       119
5.0        99
5.5        91
6.0        40
0.0        39
6.5        36
0.5        34
11.0       25
7.0        16
8.5        15
7.5        12
10.0       10
9.0         7
11.5        3
9.5         2
12.5        2
10.5        2
13.0        2
25.0        1
12.0        1
Name: bathrooms, dtype: int64
```

10. Convert Categorical Variables into Binary Dummy Variables

Converting Categorical Variables into Binary Dummy Variables

In [14]:
```
# Transform the categorical variables into binary dummy varibles; drop_first removes 1st dummy to avoid Multicollinearity
AirbnbLA = pd.get_dummies(AirbnbLA, drop_first = True)
AirbnbLA.head()
```

Out[14]:

odates	bedrooms	beds	price	minimum_nights	maximum_nights	availability_30	...	host_response_time_within a few hours	host_response_time_within an hour	host_is_sup
1	1.0	1.0	74.0	30	366	0	...	0	1	
2	1.0	2.0	118.0	31	730	0	...	1	0	
2	1.0	1.0	50.0	30	1125	0	...	1	0	
2	1.0	1.0	65.0	30	1125	25	...	1	0	
4	2.0	2.0	130.0	30	90	0	...	1	0	

11. Perform KNN Imputation

Perform KNN Imputation

In [15]:
```python
# Instantiate the KNNImputer.
# Fill in missing values by fit_transform
AirbnbLAColumns = AirbnbLA.columns
imputer = KNNImputer(n_neighbors = 5, weights = 'uniform', metric = 'nan_euclidean')
AirbnbLA_imputed = imputer.fit_transform(AirbnbLA)
Airbnb = pd.DataFrame(AirbnbLA_imputed, columns = AirbnbLAColumns)
print(Airbnb.shape)
Airbnb.head()
```

```
(17872, 33)
```

Out[15]:

	host_response_rate	host_acceptance_rate	host_listings_count	accommodates	bedrooms	beds	price	minimum_nights	n
0	1.0	1.00	2.0	1.0	1.0	1.0	74.0	30.0	
1	1.0	0.25	6.0	2.0	1.0	2.0	118.0	31.0	
2	1.0	0.80	8.0	2.0	1.0	1.0	50.0	30.0	
3	1.0	0.80	8.0	2.0	1.0	1.0	65.0	30.0	
4	1.0	0.80	8.0	4.0	2.0	2.0	130.0	30.0	

5 rows × 33 columns

12. Check final data sparsity

Check final data sparsity after KNN Imputation

In [16]:
```python
final_data_sparsity = Airbnb.isnull().sum()/len(Airbnb)
final_data_sparsity
```

Out[16]:
```
host_response_rate                        0.0
host_acceptance_rate                      0.0
host_listings_count                       0.0
accommodates                              0.0
bedrooms                                  0.0
beds                                      0.0
price                                     0.0
minimum_nights                            0.0
maximum_nights                            0.0
availability_30                           0.0
availability_60                           0.0
availability_90                           0.0
availability_365                          0.0
number_of_reviews                         0.0
review_scores_rating                      0.0
review_scores_accuracy                    0.0
review_scores_cleanliness                 0.0
review_scores_checkin                     0.0
review_scores_communication               0.0
review_scores_location                    0.0
review_scores_value                       0.0
bathrooms                                 0.0
host_response_time_within a day           0.0
host_response_time_within a few hours     0.0
host_response_time_within an hour         0.0
host_is_superhost_t                       0.0
host_has_profile_pic_t                    0.0
host_identity_verified_t                  0.0
room_type_Hotel room                      0.0
room_type_Private room                    0.0
room_type_Shared room                     0.0
has_availability_t                        0.0
instant_bookable_t                        0.0
dtype: float64
```

13. Normalize data after KNN Imputation (Normality Assumption of MLR Algorithm)

Normalize data after KNN Imputation (Normality Assumption of Linear Regression Models)

In [17]:
```python
AirbnbLANormalized = pd.DataFrame(normalize(Airbnb), columns = AirbnbLAColumns)
AirbnbLANormalized.head()
```

Out[17]:

	host_response_rate	host_acceptance_rate	host_listings_count	accommodates	bedrooms	beds	price	minimum_nights	ma
0	0.002046	0.002046	0.004092	0.002046	0.002046	0.002046	0.151422	0.061387	
1	0.001310	0.000327	0.007858	0.002619	0.001310	0.002619	0.154532	0.040597	
2	0.000852	0.000682	0.006819	0.001705	0.000852	0.000852	0.042616	0.025570	
3	0.000867	0.000694	0.006940	0.001735	0.000867	0.000867	0.056384	0.026024	
4	0.004849	0.003879	0.038789	0.019395	0.009697	0.009697	0.630326	0.145460	

5 rows × 33 columns

14. Define calculate_VIF() function to (Minimize Multicollinearity)

Define function to calculate VIF and remove features above set threshold of 10

In [21]:
```python
# Define a function that calculates the VIF of each feature
# Remove any feature with a VIF higher than the threshold (Good thresholds are between 5-10)
# Export the dataframe of features without high VIF
def calculate_vif_(df, thresh=10):
    global X_NoMulti
    const = sm.add_constant(df)
    cols = const.columns
    variables = np.arange(const.shape[1])
    vif_df = pd.Series([variance_inflation_factor(const.values, i)
               for i in range(const.shape[1])],
              index=const.columns).to_frame()

    vif_df = vif_df.sort_values(by=0, ascending=False).rename(columns={0: 'VIF'})
    vif_df = vif_df.drop('const')
    vif_df = vif_df[vif_df['VIF'] > thresh]

    print('Features above VIF threshold:\n')
    print(vif_df[vif_df['VIF'] > thresh])

    col_to_drop = list(vif_df.index)

    for i in col_to_drop:
        print('Dropping: {}'.format(i))
        df = df.drop(columns=i)

    X_NoMulti = df
    return X_NoMulti
```

15. Split dataframe into X and y values for VIF check

Split dataframe into X and y values for VIF check

In [22]:
```python
y = pd.DataFrame(AirbnbLANormalized['price'])
X = AirbnbLANormalized.drop(['price'],axis=1)
```

16. Use VIF function to Remove Multicollinearity

Calculate VIF and remove features above threshold

In [23]:
```python
calculate_vif_(X, thresh = 10)
```

```
Features above VIF threshold:

                                   VIF
review_scores_accuracy       441.642173
review_scores_checkin        367.390774
review_scores_value          338.563331
review_scores_communication  327.573744
review_scores_location       229.937998
review_scores_cleanliness    132.343336
host_has_profile_pic_t       104.349085
availability_60               34.543010
review_scores_rating          24.662996
availability_90               22.464315
host_response_rate            18.682910
Dropping: review_scores_accuracy
Dropping: review_scores_checkin
Dropping: review_scores_value
Dropping: review_scores_communication
Dropping: review_scores_location
Dropping: review_scores_cleanliness
Dropping: host_has_profile_pic_t
Dropping: availability_60
Dropping: review_scores_rating
Dropping: availability_90
Dropping: host_response_rate
```

Out[23]:

	host_acceptance_rate	host_listings_count	accommodates	bedrooms	beds	minimum_nights	maximum_nig
0	0.002046	0.004092	0.002046	0.002046	0.002046	0.061387	0.748
1	0.000327	0.007858	0.002619	0.001310	0.002619	0.040597	0.956
2	0.000682	0.006819	0.001705	0.000852	0.000852	0.025570	0.958
3	0.000694	0.006940	0.001735	0.000867	0.000867	0.026024	0.975
4	0.003879	0.038789	0.019395	0.009697	0.009697	0.145460	0.436
...	...	...	...	...	...	...	...
17867	0.000504	0.004068	0.006509	0.003255	0.003255	0.024411	0.915
17868	0.000504	0.004068	0.006509	0.003255	0.003255	0.024411	0.915
17869	0.000727	0.009192	0.005014	0.001671	0.003343	0.002507	0.940
17870	0.000832	0.012613	0.005045	0.001682	0.001682	0.001682	0.945
17871	0.000846	0.012826	0.003420	0.000855	0.000855	0.001710	0.961

17872 rows × 21 columns

17. Split Data into Training and Testing datasets (80-20; Model Evaluation)

Split Training and Testing Datasets (For Model Evaluation on Unseen Data)

Split Data into Training (80%) and Testing (20%) datasets

In [37]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X_NoMulti, y, test_size = 0.2, random_state = 123)
```

Print shapes of Training and Testing datasets

In [38]:
```
print("x_train shape:", x_train.shape)
print("x_test shape:", x_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
x_train shape: (14297, 21)
x_test shape: (3575, 21)
y_train shape: (14297, 1)
y_test shape: (3575, 1)
```

18. Perform Stepwise Regression (Feature Selection)

Perform Stepwise Regression (For Feature Selection of most impactful features on explained variance)

In [27]:
```
# Instantiate the Linear Regression algorithm
LR = LinearRegression()

# Build step forward feature selection
sfs = SequentialFeatureSelector(
    estimator = LR
    ,n_features_to_select = None
    ,direction = 'forward'
    ,scoring = 'explained_variance'
    ,cv = 10)

# Perform SFFS
sfs = sfs.fit(x_train, y_train)
x_variables = sfs.transform(x_train)

print("Selected x_variables shape:", x_variables.shape)
```

```
Selected x_variables shape: (14297, 10)
```

19. Remove non-supported columns from training and testing dataset

Show Support Values of each Feature

In [28]:
```
SupportClassification = sfs.get_support().reshape(1,x_train.shape[1])
XColumns = X_NoMulti.columns
XSupport = pd.DataFrame(SupportClassification, columns = (XColumns)).transpose().rename(columns={0: 'Support'})
XSupport.head(x_train.shape[1])
```

Out[28]:

	Support
host_acceptance_rate	False
host_listings_count	True
accommodates	True
bedrooms	True
beds	False
minimum_nights	True
maximum_nights	True
availability_30	False
availability_365	True
number_of_reviews	True
bathrooms	False
host_response_time_within a day	False
host_response_time_within a few hours	False
host_response_time_within an hour	False
host_is_superhost_t	True
host_identity_verified_t	False
room_type_Hotel room	False
room_type_Private room	True
room_type_Shared room	True
has_availability_t	False
instant_bookable_t	False

Remove non-supported columns from training and testing datasets

In [29]:
```
# Get list of columns with Support of True
XVariables = XSupport[(XSupport['Support'] == True)].transpose().columns
x_train_stepwise = x_train[XVariables]
x_test_stepwise = x_test[XVariables]

# Print shape of new dataframes
print('Post-Stepwise Regression x_train shape:', x_train_stepwise.shape)
print('Post-Stepwise Regression x_test shape:', x_test_stepwise.shape)
```

```
Post-Stepwise Regression x_train shape: (14297, 10)
Post-Stepwise Regression x_test shape: (3575, 10)
```

20. Export Post-Stepwise Regression Training and Testing Datasets (ready for modeling).

Export Post-Stepwise Regression Training and Testing datasets

In [30]:
```
x_train_stepwise.to_csv("Post-StepwiseRegression x_train.csv")
x_test_stepwise.to_csv("Post-StepwiseRegression x_test.csv")
```

Explanation of Tools and Techniques

Pandas

The Pandas library was used to load data, create dataframes, export dataframes to csv files, describe the data, count values, check data sparsity, and create binary dummy variables.

NumPy

The NumPy library was used to transform dataframes to arrays, reshape arrays, and set the random seed for reproducible results.

Gzip

The Gzip library was used to extract the csv file from the downloaded raw .gz file.

Shutil

The Shutil library was used to extract the csv file from the downloaded raw .gz file.

KNNImputer

KNNImputer is a function from sklearn.impute. It was used to impute missing values in categorical and continuous features.

Re

The Re, or Regular Expression, library was used to remove symbols in the process of converting text and currency columns to floats.

Seaborn

The Seaborn library was used to plot the correlation heatmap of the data.

Normalize

Normalize is a function from sklearn.preprocessing. It was used to normalize the dataset at the dataset grain level rather than at the column level.

LinearRegression

LinearRegression is a function from sklearn.linear_model. It was used in the Feature Selection process to instantiate the LinearRegression estimator.

SequentialFeatureSelector

SequentialFeatureSelector is a function from sklearn.feature_selection. It was used to perform Forward Stepwise Regression for feature selection.

Variance_Inflation_Factor

Variance_Inflation_Factor is a function from statsmodels.stats.outliers_influence. It was used to detect, remove, and minimize multicollinearity among the features.

Justification of Tools and Techniques

According to Brownlee, “Datasets may have missing values, and this can cause problems for many machine learning algorithms” (Brownlee, 2020). In preliminary exploratory data analysis, over half of the 32,241 observations have at least one feature with a missing value. Rather than remove half of the observations, this study utilizes the KNN algorithm to impute missing values. In his 2020 paper “kNN Imputation for Missing Values in Machine Learning”, Brownlee states that “the k-nearest neighbor (KNN) algorithm has proven to be generally effective” at predicting and imputing missing values (Brownlee, 2020). This imputation will account for both categorical and continuous variables.

Removal of multicollinearity is an assumption of Multiple Linear Regression models. According to Jong Hae Kim, “Diagnostic tools of multicollinearity include the variance inflation factor (VIF)” (Kim 2019). This study utilizes VIF with a threshold of 10 to reduce multicollinearity among the predictor variables.

Advantages and Disadvantages of Tools and Techniques

Advantages of using KNN Imputer would be the ease of the default hyper-parameters, how it uses an algorithm to impute missing values based on the nearest neighbors, and there are no assumptions about the data needed. The resulting missing values would be a more accurate imputation over a generic mean value imputation. One disadvantage would be that with the large amount of data processed, it is computationally expensive. The KNN Imputation stage takes a few minutes to run.

Analysis

Description of Analysis Technique and Process

Python was used for the creation, exploration, and evaluation of the Multiple Linear Regression model. After data cleansing, the dataset was randomly split into training and testing sets of 80% and 20% size, respectively. The training set is used in the model fitting phase while the testing set is used in the model evaluation phase. Forward Stepwise regression was used to identify the most impactful variables of the dataset on an Airbnb's rental price. There were 29 initial independent variables, excluding the ID field. Forward Stepwise regression built a Linear Regression model by adding one variable at a time based on the explained variance. Without any restrictions on the number of variables to include, the Forward Stepwise Regression selected 10 independent variables. The Multiple Linear Regression model uses the 10 independent variables to predict the outcome of the Airbnb Rental's price. The resulting model summary indicates that the model and each of the 10 independent variables are statistically significant with a p-value extremely close to zero.

Calculations and Outputs

Create and Fit the Linear Regression Model

```
In [31]:
# Add constant to x_train values
x_const = sm.add_constant(x_train_stepwise)

# Build and fit model to training data
MLR = sm.OLS(y_train, x_const)
MLR_Fitted = MLR.fit()

# Print Model Summary
print(MLR_Fitted.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.819
Model:                            OLS   Adj. R-squared:                  0.819
Method:                 Least Squares   F-statistic:                     6448.
Date:                Sat, 28 Aug 2021   Prob (F-statistic):               0.00
Time:                        17:02:18   Log-Likelihood:                 9594.5
No. Observations:               14297   AIC:                        -1.917e+04
Df Residuals:                   14286   BIC:                        -1.908e+04
Df Model:                          10
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
const                       0.9648      0.005    182.956      0.000       0.955       0.975
host_listings_count        -0.4453      0.008    -55.972      0.000      -0.461      -0.430
accommodates                4.2460      0.307     13.852      0.000       3.645       4.847
bedrooms                   19.5615      0.779     25.106      0.000      18.034      21.089
minimum_nights             -0.1545      0.020     -7.865      0.000      -0.193      -0.116
maximum_nights             -0.7789      0.005   -164.089      0.000      -0.788      -0.770
availability_365           -0.4642      0.005    -95.914      0.000      -0.474      -0.455
number_of_reviews          -0.4610      0.008    -58.484      0.000      -0.476      -0.446
host_is_superhost_t         8.5394      0.802     10.647      0.000       6.967      10.112
room_type_Private room    -15.2066      0.640    -23.744      0.000     -16.462     -13.951
room_type_Shared room     -29.9452      1.331    -22.498      0.000     -32.554     -27.336
==============================================================================
Omnibus:                      189.975   Durbin-Watson:                   2.038
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              294.020
Skew:                           0.134   Prob(JB):                     1.43e-64
Kurtosis:                       3.649   Cond. No.                     1.63e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.63e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Please note that the p-value, which is shown as Prob (F-statistic), is 0, statistically significant, in favor of rejecting the null hypothesis. Also, each independent variable has a p-value under 0.05, which indicates that each independent variable and the constant are statistically significant. The R-Squared and Adjusted R-Squared values are approximately 0.82 or 82% explained variance.

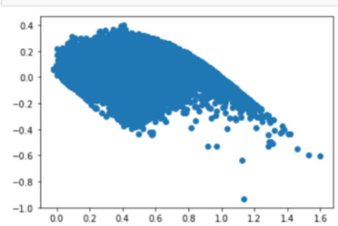
Residual Plot of Training Set

```
In [33]:
# Get predictions of y_train values
TrainPredictions = MLR_Fitted.predict(x_const)

# Reshape y_test and predictions
y_train_array = np.array(y_train).reshape(x_const.shape[0],1)
TrainPredictions_array = np.array(TrainPredictions).reshape(x_const.shape[0],1)

# Calculate residuals
TrainResiduals = y_train_array - TrainPredictions_array

# Create a scatter plot of the residuals vs. predictions
plt.scatter(TrainPredictions_array, TrainResiduals)
plt.show()
```



Please note the above residual plot of the training set and that the residuals do not appear to be normally distributed.

Model Evaluation

Evaluation Metrics: MSE and R-Squared on Unseen Data

In [34]:
```
# Add constant to x_test values
x_test_const = sm.add_constant(x_test_stepwise)

# Get predictions of y_test values
predictions = MLR_Fitted.predict(x_test_const)

# The mean squared error
print('Unseen Data Mean squared error:', mean_squared_error(y_test, predictions), '\n')

# The R-squared/Explained Variance Score (Coefficient of Determination): 1 is perfect prediction
print('Unseen Data R-Squared/Explained Variance Score:',  r2_score(y_test, predictions))
```

```
Unseen Data Mean squared error: 0.01521526606581391

Unseen Data R-Squared/Explained Variance Score: 0.8200367463286045
```

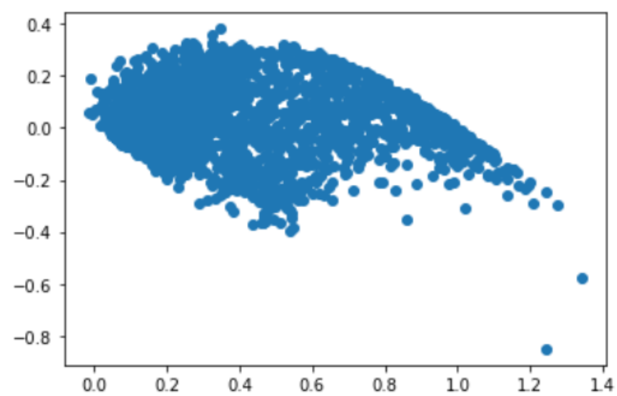
Please note that the model was evaluated on unseen, unknown data using the metrics of Mean Squared Error and R-Squared. The final model explained about 82% of the variance in Price and had a low Mean Squared Error value of ~0.015.

Residual Plot of Testing Set

In [35]:
```
# Reshape y_test and predictions
y_test_array = np.array(y_test).reshape(3575,1)
predictions_array = np.array(predictions).reshape(3575,1)

# Calculate residuals
residuals = y_test_array - predictions_array

# Create a scatter plot of the residuals vs. predictions
plt.scatter(predictions_array, residuals)
plt.show()
```



Please note the above residual plot of the testing set and that the residuals do not appear to be normally distributed.

Justification of Analysis Technique

According to a Udacity India Article by Prince Patel, “the main reason for using Python would be readability, versatility and easiness” (Patel, 2018). Since this study is meant to be accessible to new hosts and Multiple Linear Regression is “usually the first machine learning algorithm that every data scientist comes across”, it affords the newer data enthusiasts an easier route to utilize this study in commercial practice (Agarwal, 2018). According to Laerd Statistics, Multiple Linear Regression would be a viable method for this study as it is “used to predict a continuous dependent variable based on multiple independent variables” (Laerd Statistics, 2015).

Advantages and Disadvantages of Analysis Technique

According to Mohammad Waseem, the advantages of Multiple Linear Regression include that it is “easier to implement and interpret, performs well on linear data, and handles overfitting well when used with dimensionally reduction techniques” (Waseem 2021). Disadvantages of Multiple Linear Regression are that there are “assumptions on linearity” and the distributions of the data and it is “prone to multicollinearity” (Waseem 2021).

Data Summary and Implication

Summarize implications by discussing results with 1 limitation

Coefficients of Model

```
In [32]:  # Print coefficients of each feature and constant
          print(MLR_Fitted.params)

const                       0.964846
host_listings_count        -0.445316
accommodates                4.245991
bedrooms                   19.561517
minimum_nights             -0.154465
maximum_nights             -0.778914
availability_365           -0.464200
number_of_reviews          -0.460954
host_is_superhost_t         8.539390
room_type_Private room    -15.206620
room_type_Shared room     -29.945156
dtype: float64
```

Please note the above image shows the coefficients of the final Multiple Linear Regression Model on normalized data. Each coefficient represents a variable's one-normalized unit effect on price when all other variables remain the same. For example, for each one-normalized unit increase in the number of bedrooms, the rental price is increased by ~19.56 given all other variables remain the same. Variables with negative coefficients have a negative impact on price when all other variables remain the same. The five most impactful variables on price given the size of their coefficients are accommodates, bedrooms, host_is_superhost_true, room_type_Private room, and room_type_Shared room.

```
In [31]:  # Add constant to x_train values
          x_const = sm.add_constant(x_train_stepwise)

          # Build and fit model to training data
          MLR = sm.OLS(y_train, x_const)
          MLR_Fitted = MLR.fit()

          # Print Model Summary
          print(MLR_Fitted.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.819
Model:                            OLS   Adj. R-squared:                  0.819
Method:                 Least Squares   F-statistic:                     6448.
Date:                Sat, 28 Aug 2021   Prob (F-statistic):               0.00
Time:                        17:02:18   Log-Likelihood:                 9594.5
No. Observations:               14297   AIC:                        -1.917e+04
Df Residuals:                   14286   BIC:                        -1.908e+04
Df Model:                          10
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
const                       0.9648      0.005    182.956      0.000       0.955       0.975
host_listings_count        -0.4453      0.008    -55.972      0.000      -0.461      -0.430
accommodates                4.2460      0.307     13.852      0.000       3.645       4.847
bedrooms                   19.5615      0.779     25.106      0.000      18.034      21.089
minimum_nights             -0.1545      0.020     -7.865      0.000      -0.193      -0.116
maximum_nights             -0.7789      0.005   -164.089      0.000      -0.788      -0.770
availability_365           -0.4642      0.005    -95.914      0.000      -0.474      -0.455
number_of_reviews          -0.4610      0.008    -58.484      0.000      -0.476      -0.446
host_is_superhost_t         8.5394      0.802     10.647      0.000       6.967      10.112
room_type_Private room    -15.2066      0.640    -23.744      0.000     -16.462     -13.951
room_type_Shared room     -29.9452      1.331    -22.498      0.000     -32.554     -27.336
==============================================================================
Omnibus:                      189.975   Durbin-Watson:                   2.038
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              294.020
Skew:                           0.134   Prob(JB):                     1.43e-64
Kurtosis:                       3.649   Cond. No.                     1.63e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.63e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Please note the above image shows the MLR Model’s output summary. Each independent variable and constant have a p-value of approximately 0 and are statistically significant. The final model has a p-value of nearly 0 and is statistically significant. The p-value supports a rejection of the Null Hypothesis and acceptance of the Alternate Hypothesis that a statistically significant model can be created to predict the Airbnb rental price.

The R-Squared and Adjusted R-Squared values are within a close range of each other, and this indicates that the model is not overfitting. The Condition Number is large and in the notes section, there is an indication of either strong multicollinearity or other numerical problems. VIF was used to minimize multicollinearity. However, there is still multicollinearity present in the dataset. Multiple Linear Regression requires assumptions about parameters and distributions. With the presence of multicollinearity, any further analysis may be best explored using non-Parametric analysis methods.

Recommendations

Based on the results of this study, it is recommended that a new Airbnb host in the Los Angeles market should focus on locating properties that can be rented entirely, accommodate more people, and have a larger number of bedrooms while striving to become a superhost on Airbnb. In a 12-month period, Airbnb superhosts “must maintain a 90% response rate, 1% cancellation rate, and 4.8 overall rating after completing at least 10 trips” (Airbnb n.d).

Proposals for Future Study of Dataset

Proposal One

Given the lingering presence of Multicollinearity after the VIF check and the large condition number, this study proposes using a non-Parametric method such as KNN to further analyze the dataset. The KNN algorithm does not require strict assumptions of the underlying data and would avoid the issue of multicollinearity.

Proposal Two

If Parametric analysis is necessary, this study also proposes using another feature selection method such as Principal Component Analysis for dimensionality reduction. Principal Component Analysis would reduce the number of features while further accounting for multicollinearity.

Proposal Three

Further analysis should be done on the various neighborhoods in the Los Angeles market available in this dataset. The dataset was delimited to only include the neighborhoods within the City of Los Angeles. A variety of neighborhoods and locations may present other findings of impactful variables.

References

Agarwal, A. (2018, October 5). *Linear Regression using Python*. Towards Data Science. https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2#.

Airbnb (n.d). *How do I become a Superhost*. Airbnb. (n.d.). https://www.airbnb.com/help/article/829/how-do-i-become-a-superhost.

Brownlee, J. (2020, June 24). *kNN Imputation for Missing Values in Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/knn-imputation-for-missing-values-in-machine-learning/.

Cox, M. (n.d.). *Get the Data*. Inside Airbnb. http://insideairbnb.com/behind.html.

Creative Commons. (n.d.). *CC0 1.0 Universal (CC0 1.0) Public Domain Dedication*. Creative Commons. https://creativecommons.org/publicdomain/zero/1.0/.

Deane, S. (2021, January 26). *2021 Airbnb Statistics: Usage, Demographics, and Revenue Growth*. Stratos Jet Charters Inc. https://www.stratosjets.com/blog/airbnb-statistics/.

Ghosalkar, N., & Dhage, S. N. (2018). *Real Estate Value Prediction Using Linear Regression*. Semantic Scholar. https://www.semanticscholar.org/paper/Real-Estate-Value-Prediction-Using-Linear-Ghosalkar-Dhage/f2308f0a4f0981801b518b9ca2152bcb4c797ad7.

Kim, J. H. (2019, July 15). *Multicollinearity and misleading statistical results*. National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6900425/.

Laerd Statistics (2015). *Multiple regression using SPSS Statistics*. Laerd Statistics. https://statistics.laerd.com/premium/spss/mr/multiple-regression-in-spss-20.php.

Patel, P. (2018, March 8). *Why Python is the most popular language used for Machine Learning*. Medium. https://medium.com/@UdacityINDIA/why-use-python-for-machine-learning-e4b0b4457a77.

Waseem, M. (2021, July 15). *How to Implement Linear Regression for Machine Learning?* Edureka. https://www.edureka.co/blog/linear-regression-for-machine-learning/.