In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from kmodes.kmodes import KModes
import seaborn as sns
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing
```

In [2]:
```
#pip install kmodes
```

In [2]:
```
# imports the dataset
df = pd.read_csv("sf_airbnb_listings.csv", sep=",", header=None, engine='python', encoding="utf-8-sig")
```

In [3]:

```
header = df.iloc[0]
# take the rest of your data minus the header row
df = df[1:]
# set the header row as the df header
df.columns = header
pd.set_option('display.max_rows', 7500)
pd.set_option('display.max_columns', 106)
df
```

Out[3]:

	id	listing_url	scrape_id	last_scraped	name	summary	space	description
1	958	https://www.airbnb.com/rooms/958	2.02E+13	6/2/2019	Bright, Modern Garden Unit - 1BR/1B	New update: the house next door is under const...	Newly remodeled, modern, and bright garden uni...	New update: the house next door is under const...
2	5858	https://www.airbnb.com/rooms/5858	2.02E+13	6/2/2019	Creative Sanctuary	NaN	We live in a large Victorian house on a quiet ...	We live in a large Victorian house on a quiet ...
3	7918	https://www.airbnb.com/rooms/7918	2.02E+13	6/2/2019	A Friendly Room - UCSF/USF - San Francisco	Nice and good public transportation. 7 minute...	Room rental-sunny view room/sink/Wi Fi (inner ...	Nice and good public transportation. 7 minute...
4	8142	https://www.airbnb.com/rooms/8142	2.02E+13	6/2/2019	Friendly Room Apt. Style - UCSF/USF - San Franc...	Nice and good public transportation. 7 minute...	Room rental Sunny view Rm/Wi-Fi/TV/sink/large ...	Nice and good public transportation. 7 minute...
5	8339	https://www.airbnb.com/rooms/8339	2.02E+13	6/2/2019	Historic Alamo Square Victorian	Pls email before booking. Interior featured i...	Please send us a quick message before booking ...	Pls email before booking. Interior featured i...
...	...	...	...	...	...	...	...	...
7571	35284961	https://www.airbnb.com/rooms/35284961	2.02E+13	6/2/2019	Brand New Designer 2 BR SF Condo	Luxury spacious 2 bedroom condo located in SF,...	Private dedicated entrance NEST temperature c...	Luxury spacious 2 bedroom condo located in SF,...
7572	35285751	https://www.airbnb.com/rooms/35285751	2.02E+13	6/2/2019	Beautiful 1x1 in Historic Mission Tudor Building	A beautifully remodeled one bedroom in a great...	NaN	A beautifully remodeled one bedroom in a great...

	id	listing_url	scrape_id	last_scraped	name	summary	space	description
7573	35286441	https://www.airbnb.com/rooms/35286441	2.02E+13	6/2/2019	Beautiful Queen Victorian in the heart of Mission	Our place is a charming Victorian located in t...	The house is a quintessential remodeled Victor...	Our place is a charming Victorian located in t...
7574	35288483	https://www.airbnb.com/rooms/35288483	2.02E+13	6/2/2019	New comfortable, convenient place for family	This new place is comfortable, with easy commu...	NaN	This new place is comfortable, with easy commu...
7575	35291911	https://www.airbnb.com/rooms/35291911	2.02E+13	6/2/2019	Spacious 2bdrm/2bath in the heart of SF	Freshly remodeled in May 2019, 2 bedroom 2 ba...	- Centrally located - Steps away from 16th and...	Freshly remodeled in May 2019, 2 bedroom 2 ba...

7575 rows × 106 columns

In [4]:

```
df['zipcode'] = df.zipcode.astype('category')
df['latitude'] = df.latitude.astype('category')
df['longitude'] = df.longitude.astype('category')
df['maximum_nights'] = df.maximum_nights.astype('category')
df['minimum_minimum_nights'] = df.minimum_minimum_nights.astype('category')
df['amenities'] = df.amenities.astype('category')
df['bed_type'] = df.bed_type.astype('category')
df['property_type'] = df.property_type.astype('category')

# df['neighbourhood'] = df.neighbourhood.astype(int)
# df['room_type'] = df.room_type.astype(int)
```

In [5]:
```
df.dtypes
```

```
Out[5]: 0
id                                    object
listing_url                           object
scrape_id                             object
last_scraped                          object
name                                  object
summary                               object
space                                 object
description                           object
experiences_offered                   object
neighborhood_overview                 object
notes                                 object
transit                               object
access                                object
interaction                           object
house_rules                           object
thumbnail_url                         object
medium_url                            object
picture_url                           object
xl_picture_url                        object
host_id                               object
host_url                              object
host_name                             object
host_since                            object
host_location                         object
host_about                            object
host_response_time                    object
host_response_rate                    object
host_acceptance_rate                  object
host_is_superhost                     object
host_thumbnail_url                    object
host_picture_url                      object
host_neighbourhood                    object
host_listings_count                   object
host_total_listings_count             object
host_verifications                    object
host_has_profile_pic                  object
host_identity_verified                object
street                                object
neighbourhood                         object
neighbourhood_cleansed                object
neighbourhood_group_cleansed          object
city                                  object
```

```
state                                              object
zipcode                                          category
market                                             object
smart_location                                     object
country_code                                       object
country                                            object
latitude                                         category
longitude                                        category
is_location_exact                                  object
property_type                                    category
room_type                                          object
accommodates                                       object
bathrooms                                          object
bedrooms                                           object
beds                                               object
bed_type                                         category
amenities                                        category
square_feet                                        object
price                                              object
weekly_price                                       object
monthly_price                                      object
security_deposit                                   object
cleaning_fee                                       object
guests_included                                    object
extra_people                                       object
minimum_nights                                     object
maximum_nights                                   category
minimum_minimum_nights                           category
maximum_minimum_nights                             object
minimum_maximum_nights                             object
maximum_maximum_nights                             object
minimum_nights_avg_ntm                             object
maximum_nights_avg_ntm                             object
calendar_updated                                   object
has_availability                                   object
availability_30                                    object
availability_60                                    object
availability_90                                    object
availability_365                                   object
calendar_last_scraped                              object
number_of_reviews                                  object
number_of_reviews_ltm                              object
first_review                                       object
```

```
last_review                                     object
review_scores_rating                            object
review_scores_accuracy                          object
review_scores_cleanliness                       object
review_scores_checkin                           object
review_scores_communication                     object
review_scores_location                          object
review_scores_value                             object
requires_license                                object
license                                         object
jurisdiction_names                              object
instant_bookable                                object
is_business_travel_ready                        object
cancellation_policy                             object
require_guest_profile_picture                   object
require_guest_phone_verification                object
calculated_host_listings_count                  object
calculated_host_listings_count_entire_homes     object
calculated_host_listings_count_private_rooms    object
calculated_host_listings_count_shared_rooms     object
reviews_per_month                               object
dtype: object
```

In [6]:

```
columns="host_id host_listings_count host_total_listings_count accommodates bathrooms bedrooms beds price wee
kly_price monthly_price security_deposit cleaning_fee guests_included minimum_nights maximum_minimum_nights m
inimum_maximum_nights maximum_maximum_nights minimum_nights_avg_ntm maximum_nights_avg_ntm availability_30 av
ailability_60 availability_90 availability_365 number_of_reviews number_of_reviews_ltm review_scores_rating r
eview_scores_accuracy review_scores_cleanliness review_scores_checkin review_scores_communication review_scor
es_location review_scores_value calculated_host_listings_count calculated_host_listings_count_entire_homes ca
lculated_host_listings_count_private_rooms calculated_host_listings_count_shared_rooms neighbourhood room_typ
e reviews_per_month".split()
```

In [7]:
```
columns
```

Out[7]:
```
['host_id',
 'host_listings_count',
 'host_total_listings_count',
 'accommodates',
 'bathrooms',
 'bedrooms',
 'beds',
 'price',
 'weekly_price',
 'monthly_price',
 'security_deposit',
 'cleaning_fee',
 'guests_included',
 'minimum_nights',
 'maximum_minimum_nights',
 'minimum_maximum_nights',
 'maximum_maximum_nights',
 'minimum_nights_avg_ntm',
 'maximum_nights_avg_ntm',
 'availability_30',
 'availability_60',
 'availability_90',
 'availability_365',
 'number_of_reviews',
 'number_of_reviews_ltm',
 'review_scores_rating',
 'review_scores_accuracy',
 'review_scores_cleanliness',
 'review_scores_checkin',
 'review_scores_communication',
 'review_scores_location',
 'review_scores_value',
 'calculated_host_listings_count',
 'calculated_host_listings_count_entire_homes',
 'calculated_host_listings_count_private_rooms',
 'calculated_host_listings_count_shared_rooms',
 'neighbourhood',
 'room_type',
 'reviews_per_month']
```

In [8]:
```
df2=pd.DataFrame(df, columns=columns)
df2.dtypes
```

```
Out[8]: host_id                                         object
        host_listings_count                             object
        host_total_listings_count                       object
        accommodates                                    object
        bathrooms                                       object
        bedrooms                                        object
        beds                                            object
        price                                           object
        weekly_price                                    object
        monthly_price                                   object
        security_deposit                                object
        cleaning_fee                                    object
        guests_included                                 object
        minimum_nights                                  object
        maximum_minimum_nights                          object
        minimum_maximum_nights                          object
        maximum_maximum_nights                          object
        minimum_nights_avg_ntm                          object
        maximum_nights_avg_ntm                          object
        availability_30                                 object
        availability_60                                 object
        availability_90                                 object
        availability_365                                object
        number_of_reviews                               object
        number_of_reviews_ltm                           object
        review_scores_rating                            object
        review_scores_accuracy                          object
        review_scores_cleanliness                       object
        review_scores_checkin                           object
        review_scores_communication                     object
        review_scores_location                          object
        review_scores_value                             object
        calculated_host_listings_count                  object
        calculated_host_listings_count_entire_homes     object
        calculated_host_listings_count_private_rooms    object
        calculated_host_listings_count_shared_rooms     object
        neighbourhood                                   object
        room_type                                       object
        reviews_per_month                               object
        dtype: object
```

In [85]:
```python
# transforms categorical valuses in column to numeric
le = preprocessing.LabelEncoder()
df2['neighbourhood'] = le.fit_transform(df2.neighbourhood.values)
df2['room_type'] = le.fit_transform(df2.room_type.values)
df2.fillna(0, inplace=True)
#df2.to_csv('df2.csv')
df2
```

Out[85]:

	host_id	host_listings_count	host_total_listings_count	accommodates	bathrooms	bedrooms	beds	price	weekly_price	mon
1	1169	1.0	1.0	3.0	1.0	1.0	2.0	170.0	1120.0	
2	8904	2.0	2.0	5.0	1.0	2.0	3.0	235.0	1600.0	
3	21994	10.0	10.0	2.0	4.0	1.0	1.0	65.0	485.0	
4	21994	10.0	10.0	2.0	4.0	1.0	1.0	65.0	490.0	
5	24215	2.0	2.0	5.0	1.5	2.0	2.0	685.0	0	
...	...	...	...	...	...	...	...	...	...	
7571	94857021	2.0	2.0	2.0	1.0	2.0	2.0	475.0	0	
7572	4430421	92.0	92.0	3.0	1.0	1.0	1.0	115.0	0	
7573	50247302	2.0	2.0	6.0	1.5	2.0	2.0	500.0	0	
7574	163879334	3.0	3.0	6.0	1.0	2.0	3.0	180.0	0	
7575	236978	3.0	3.0	4.0	2.0	2.0	2.0	300.0	0	

7575 rows × 39 columns

In [97]:
```python
columns2="host_id host_listings_count host_total_listings_count accommodates bathrooms bedrooms beds weekly_price monthly_price security_deposit cleaning_fee guests_included minimum_nights mximum_nights mimimum_mimimum_nights maximum_minimum_nights minimum_maximum_nights maximum_maximum_nights minimum_nights_avg_ntm maximum_nights_avg_ntm availability_30 availability_60 availability_90 availability_365 number_of_reviews number_of_reviews_ltm review_scores_rating review_scores_accuracy review_scores_cleanliness review_scores_checkin review_scores_communication review_scores_location review_scores_value calculated_host_listings_count calculated_host_listings_count_entire_homes calculated_host_listings_count_private_rooms calculated_host_listings_count_shared_rooms neighbourhood room_type price".split()
```

In [10]:
```
df3=pd.DataFrame(df, columns=columns)
df3
```

Out[10]:

	host_id	host_listings_count	host_total_listings_count	accommodates	bathrooms	bedrooms	beds	price	weekly_price	mon
1	1169	1.0	1.0	3.0	1.0	1.0	2.0	170.0	1120.0	
2	8904	2.0	2.0	5.0	1.0	2.0	3.0	235.0	1600.0	
3	21994	10.0	10.0	2.0	4.0	1.0	1.0	65.0	485.0	
4	21994	10.0	10.0	2.0	4.0	1.0	1.0	65.0	490.0	
5	24215	2.0	2.0	5.0	1.5	2.0	2.0	685.0	NaN	
...	...	...	...	...	...	...	...	...	...	
7571	94857021	2.0	2.0	2.0	1.0	2.0	2.0	475.0	NaN	
7572	4430421	92.0	92.0	3.0	1.0	1.0	1.0	115.0	NaN	
7573	50247302	2.0	2.0	6.0	1.5	2.0	2.0	500.0	NaN	
7574	163879334	3.0	3.0	6.0	1.0	2.0	3.0	180.0	NaN	
7575	236978	3.0	3.0	4.0	2.0	2.0	2.0	300.0	NaN	

7575 rows × 39 columns

In [11]:
```
df3 = df3.dropna(axis=1)
df3
```

Out[11]:

	host_id	host_listings_count	host_total_listings_count	accommodates	price	guests_included	minimum_nights	maximum_min
1	1169	1.0	1.0	3.0	170.0	2.0	1.0	
2	8904	2.0	2.0	5.0	235.0	2.0	30.0	
3	21994	10.0	10.0	2.0	65.0	1.0	32.0	
4	21994	10.0	10.0	2.0	65.0	1.0	32.0	
5	24215	2.0	2.0	5.0	685.0	2.0	4.0	
...	...	...	...	...	...	...	...	
7571	94857021	2.0	2.0	2.0	475.0	1.0	3.0	
7572	4430421	92.0	92.0	3.0	115.0	1.0	30.0	
7573	50247302	2.0	2.0	6.0	500.0	4.0	1.0	
7574	163879334	3.0	3.0	6.0	180.0	4.0	30.0	
7575	236978	3.0	3.0	4.0	300.0	4.0	1.0	

7575 rows × 24 columns

In [12]:
```python
pd.set_option('display.max_rows', 7500)
pd.set_option('display.max_columns', 106)
df3
```

Out[12]:

	host_id	host_listings_count	host_total_listings_count	accommodates	price	guests_included	minimum_nights	maximum_min
1	1169	1.0	1.0	3.0	170.0	2.0	1.0	
2	8904	2.0	2.0	5.0	235.0	2.0	30.0	
3	21994	10.0	10.0	2.0	65.0	1.0	32.0	
4	21994	10.0	10.0	2.0	65.0	1.0	32.0	
5	24215	2.0	2.0	5.0	685.0	2.0	4.0	
...	...	...	...	...	...	...	...	
7571	94857021	2.0	2.0	2.0	475.0	1.0	3.0	
7572	4430421	92.0	92.0	3.0	115.0	1.0	30.0	
7573	50247302	2.0	2.0	6.0	500.0	4.0	1.0	
7574	163879334	3.0	3.0	6.0	180.0	4.0	30.0	
7575	236978	3.0	3.0	4.0	300.0	4.0	1.0	

7575 rows × 24 columns

In [13]:
```
df3.dtypes
```

Out[13]:
```
host_id                                         object
host_listings_count                             object
host_total_listings_count                       object
accommodates                                    object
price                                           object
guests_included                                 object
minimum_nights                                  object
maximum_minimum_nights                          object
minimum_maximum_nights                          object
maximum_maximum_nights                          object
minimum_nights_avg_ntm                          object
maximum_nights_avg_ntm                          object
availability_30                                 object
availability_60                                 object
availability_90                                 object
availability_365                                object
number_of_reviews                               object
number_of_reviews_ltm                           object
calculated_host_listings_count                  object
calculated_host_listings_count_entire_homes     object
calculated_host_listings_count_private_rooms    object
calculated_host_listings_count_shared_rooms     object
neighbourhood                                   object
room_type                                       object
dtype: object
```

In [57]:
```
# categorical data
```

In [730]:

```
data = df3

# model
km = KModes(n_clusters=5, init='Huang', n_init=5, verbose=1)
clusters = km.fit_predict(data)

# Print the results of clustering centroids
print(km.cluster_centroids_)
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 2418, cost: 129298.0
Run 1, iteration: 2/100, moves: 111, cost: 129289.0
Run 1, iteration: 3/100, moves: 1, cost: 129289.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 2060, cost: 129841.0
Run 2, iteration: 2/100, moves: 512, cost: 129553.0
Run 2, iteration: 3/100, moves: 110, cost: 129553.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 2735, cost: 129828.0
Run 3, iteration: 2/100, moves: 477, cost: 129770.0
Run 3, iteration: 3/100, moves: 23, cost: 129770.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 4, iteration: 1/100, moves: 2863, cost: 129566.0
Run 4, iteration: 2/100, moves: 403, cost: 129408.0
Run 4, iteration: 3/100, moves: 134, cost: 129404.0
Run 4, iteration: 4/100, moves: 1, cost: 129404.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 5, iteration: 1/100, moves: 1958, cost: 130072.0
Run 5, iteration: 2/100, moves: 74, cost: 130072.0
Best run was number 1
[['173206762' '2.0' '2.0' '2.0'
  '{TV,Elevator,Heating,Smoke detector,Essentials}' 'Real Bed' 'House'
  '37.75369' '-122.40637' '100.0' '1.0' '30.0' '30.0' '1125.0' '1125.0'
  '30.0' '1125.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '2.0' '0.0' '1.0'
  '0.0' 'Mission District' 'Private room']
 ['55782784' '1.0' '1.0' '2.0'
  '{TV,Cable TV,Air conditioning,Paid parking off premises,Gym,Elevator,Heating,Family/kid friendly,Washer,Dr
yer,Smoke detector,Carbon monoxide detector,Essentials,Shampoo,24-hour check-in,Hangers,Hair dryer,Iron,Self
check-in,Building staff,Hot water,Luggage dropoff allowed,Paid parking on premises}'
  'Real Bed' 'House' '37.78995' '-122.40956' '250.0' '1.0' '2.0' '2.0'
  '14.0' '14.0' '2.0' '14.0' '0.0' '0.0' '0.0' '0.0' '1.0' '1.0' '1.0'
  '1.0' '0.0' '0.0' 'Bernal Heights' 'Entire home/apt']
```

```
 ['117141107' '2.0' '2.0' '2.0'
  '{TV,Cable TV,Internet,Wifi,Air conditioning,Gym,Elevator,Hot tub,Heating,Smoke detector,Carbon monoxide de
tector,First aid kit,Fire extinguisher,Essentials,Shampoo,Lock on bedroom door,Hangers,Hair dryer,Iron,Laptop
friendly workspace,Self check-in,Building staff,Private entrance,Pack ,Äôn Play/travel crib,Hot water,Bed lin
ens,Extra pillows and blankets,Microwave,Coffee maker,Refrigerator,Dishes and silverware,Garden or backyard,L
uggage dropoff allowed,Wide hallways,Wide entrance for guests,Flat path to guest entrance,Wide entrance,Extra
space around bed,Accessible-height toilet,Wide clearance to shower, toilet,Paid parking on premises,Fixed gra
b bars for shower,Fixed grab bars for toilet,Roll-in shower}'
  'Real Bed' 'Apartment' '37.77981' '-122.40826' '100.0' '1.0' '1.0'
  '1.0' '1125.0' '1125.0' '1.0' '1125.0' '30.0' '60.0' '90.0' '365.0'
  '0.0' '0.0' '2.0' '0.0' '1.0' '0.0' 'Downtown' 'Private room']
 ['48005494' '1235.0' '1235.0' '2.0'
  '{TV,Internet,Wifi,Kitchen,Pets allowed,Heating,Washer,Dryer,Smoke detector,Carbon monoxide detector,Essent
ials,Shampoo,Hangers,Hair dryer,Iron,Laptop friendly workspace,Private entrance}'
  'Real Bed' 'Apartment' '37.77296' '-122.40798' '140.0' '1.0' '30.0'
  '30.0' '1125.0' '1125.0' '30.0' '1125.0' '0.0' '0.0' '0.0' '365.0'
  '0.0' '0.0' '241.0' '241.0' '0.0' '0.0' 'SoMa' 'Entire home/apt']
 ['219930816' '1.0' '1.0' '4.0'
  '{TV,Cable TV,Wifi,Kitchen,Pets allowed,Elevator,Free street parking,Heating,Washer,Dryer,Smoke detector,Ca
rbon monoxide detector,Fire extinguisher,Essentials,Shampoo,Hangers,Hair dryer,Iron,Laptop friendly workspac
e,Private living room,Hot water,Bed linens,Coffee maker,Refrigerator,Dishwasher,Dishes and silverware,Cooking
basics,Oven,Stove,BBQ grill,Patio or balcony,Long term stays allowed,Host greets you}'
  'Real Bed' 'Apartment' '37.75909' '-122.41243' '150.0' '1.0' '30.0'
  '30.0' '1125.0' '1125.0' '30.0' '1125.0' '0.0' '0.0' '0.0' '0.0' '0.0'
  '0.0' '1.0' '1.0' '0.0' '0.0' 'Mission District' 'Entire home/apt']]
```

In [58]:
```
# Linear Regression
```

In [115]:
```
columns2="host_id host_listings_count host_total_listings_count accommodates bathrooms bedrooms beds weekly_price monthly_price security_deposit cleaning_fee guests_included minimum_nights mximum_nights mimimum_mimimum_nights maximum_minimum_nights minimum_maximum_nights maximum_maximum_nights minimum_nights_avg_ntm maximum_nights_avg_ntm availability_30 availability_60 availability_90 availability_365 number_of_reviews number_of_reviews_ltm review_scores_rating review_scores_accuracy review_scores_cleanliness review_scores_checkin review_scores_communication review_scores_location review_scores_value calculated_host_listings_count calculated_host_listings_count_entire_homes calculated_host_listings_count_private_rooms calculated_host_listings_count_shared_rooms neighbourhood room_type reviews_per_month".split()
```

In [117]:
```
target = df2['price']
```

In [118]:

```
DF=pd.DataFrame(df2, columns=columns2)
DF
```

Out[118]:

	host_id	host_listings_count	host_total_listings_count	accommodates	bathrooms	bedrooms	beds	weekly_price	monthly_pri
1	1169	1.0	1.0	3.0	1.0	1.0	2.0	1120.0	4200
2	8904	2.0	2.0	5.0	1.0	2.0	3.0	1600.0	5500
3	21994	10.0	10.0	2.0	4.0	1.0	1.0	485.0	1685
4	21994	10.0	10.0	2.0	4.0	1.0	1.0	490.0	1685
5	24215	2.0	2.0	5.0	1.5	2.0	2.0	0	
...	...	...	...	...	...	...	...	...	
7571	94857021	2.0	2.0	2.0	1.0	2.0	2.0	0	
7572	4430421	92.0	92.0	3.0	1.0	1.0	1.0	0	
7573	50247302	2.0	2.0	6.0	1.5	2.0	2.0	0	
7574	163879334	3.0	3.0	6.0	1.0	2.0	3.0	0	
7575	236978	3.0	3.0	4.0	2.0	2.0	2.0	0	

7575 rows × 40 columns

In [119]:
```python
DF1 = DF.dropna(axis=1)
DF1
```

Out[119]:

	host_id	host_listings_count	host_total_listings_count	accommodates	bathrooms	bedrooms	beds	weekly_price	monthly_pri
1	1169	1.0	1.0	3.0	1.0	1.0	2.0	1120.0	420
2	8904	2.0	2.0	5.0	1.0	2.0	3.0	1600.0	550
3	21994	10.0	10.0	2.0	4.0	1.0	1.0	485.0	168
4	21994	10.0	10.0	2.0	4.0	1.0	1.0	490.0	168
5	24215	2.0	2.0	5.0	1.5	2.0	2.0	0	
...	...	...	...	...	...	...	...	...	
7571	94857021	2.0	2.0	2.0	1.0	2.0	2.0	0	
7572	4430421	92.0	92.0	3.0	1.0	1.0	1.0	0	
7573	50247302	2.0	2.0	6.0	1.5	2.0	2.0	0	
7574	163879334	3.0	3.0	6.0	1.0	2.0	3.0	0	
7575	236978	3.0	3.0	4.0	2.0	2.0	2.0	0	

7575 rows × 38 columns

In [120]:
```python
y=target
```

In [121]:
```python
X_train, X_test, y_train, y_test = train_test_split(DF1, y, test_size=0.40)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(4545, 38) (4545,)
(3030, 38) (3030,)
```

In [122]:
```python
lm = linear_model.LinearRegression()
model = lm.fit(X_train, y_train)
predictions = lm.predict(X_test)
```

```
In [123]: predictions
```

```
Out[123]: array([ 71.77284909,  43.34879243, 172.73036966, ..., 382.5593061 ,
       172.3052225 , 142.59380895])
```

In [124]:
```
plt.scatter(y_test, predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.title('Predicted')
plt.xticks(rotation=45)
```

```
Out[124]: ([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30,
  31,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  39,
  40,
  41,
  42,
```

43,
44,
45,
46,
47,
48,
49,
50,
51,
52,
53,
54,
55,
56,
57,
58,
59,
60,
61,
62,
63,
64,
65,
66,
67,
68,
69,
70,
71,
72,
73,
74,
75,
76,
77,
78,
79,
80,
81,
82,
83,
84,
85,

86,
87,
88,
89,
90,
91,
92,
93,
94,
95,
96,
97,
98,
99,
100,
101,
102,
103,
104,
105,
106,
107,
108,
109,
110,
111,
112,
113,
114,
115,
116,
117,
118,
119,
120,
121,
122,
123,
124,
125,
126,
127,
128,

```
129,
130,
131,
132,
133,
134,
135,
136,
137,
138,
139,
140,
141,
142,
143,
144,
145,
146,
147,
148,
149,
150,
151,
152,
153,
154,
155,
156,
157,
158,
159,
160,
161,
162,
163,
164,
165,
166,
167,
168,
169,
170,
171,
```

172,
173,
174,
175,
176,
177,
178,
179,
180,
181,
182,
183,
184,
185,
186,
187,
188,
189,
190,
191,
192,
193,
194,
195,
196,
197,
198,
199,
200,
201,
202,
203,
204,
205,
206,
207,
208,
209,
210,
211,
212,
213,
214,

```
215,
216,
217,
218,
219,
220,
221,
222,
223,
224,
225,
226,
227,
228,
229,
230,
231,
232,
233,
234,
235,
236,
237,
238,
239,
240,
241,
242,
243,
244,
245,
246,
247,
248,
249,
250,
251,
252,
253,
254,
255,
256,
257,
```

```
258,
259,
260,
261,
262,
263,
264,
265,
266,
267,
268,
269,
270,
271,
272,
273,
274,
275,
276,
277,
278,
279,
280,
281,
282,
283,
284,
285,
286,
287,
288,
289,
290,
291,
292,
293,
294,
295,
296,
297,
298,
299,
300,
```

301,
302,
303,
304,
305,
306,
307,
308,
309,
310,
311,
312,
313,
314,
315,
316,
317,
318,
319,
320,
321,
322,
323,
324,
325,
326,
327,
328,
329,
330,
331,
332,
333,
334,
335,
336,
337,
338,
339,
340,
341,
342,
343,

344,
345,
346,
347,
348,
349,
350,
351,
352,
353,
354,
355,
356,
357,
358,
359,
360,
361,
362,
363,
364,
365,
366,
367,
368,
369,
370,
371,
372,
373,
374,
375,
376,
377,
378,
379,
380,
381,
382,
383,
384,
385,
386,

```
 387,
 388,
 389,
 390,
 391,
 392,
 393,
 394,
 395,
 396,
 397,
 398,
 399],
[Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
```

```
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
```

```
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
```

```
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
```

```
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
```

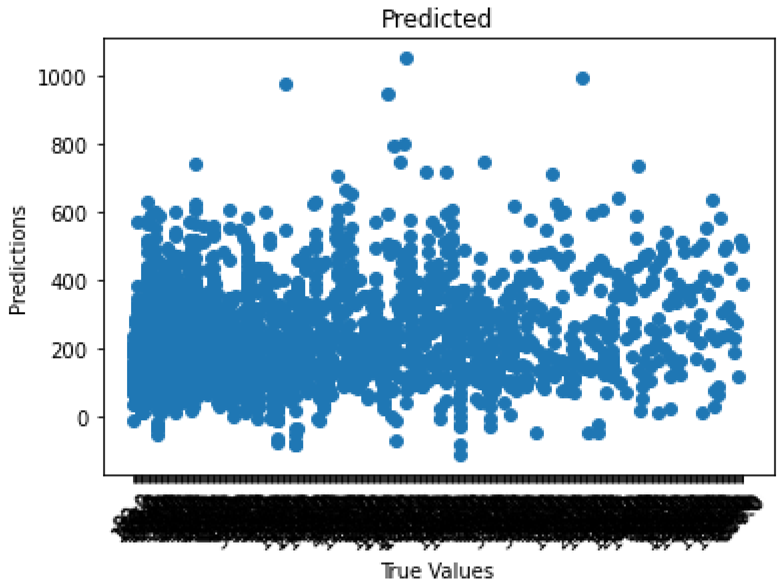
```
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
```

```
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
```

```
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
```

```
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
```

```
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, '')])
```



In [125]:
```
print('Score:', model.score(X_test, y_test))
```

Score: 0.28478289270098456

In [ ]:
```
# Linear Regression MOdel 2
```

In [136]:
```
df2 = df2.dropna(axis=1)
df2
```

Out[136]:

	host_id	host_listings_count	host_total_listings_count	accommodates	bathrooms	bedrooms	beds	price	weekly_price	mon
1	1169	1.0	1.0	3.0	1.0	1.0	2.0	170.0	1120.0	
2	8904	2.0	2.0	5.0	1.0	2.0	3.0	235.0	1600.0	
3	21994	10.0	10.0	2.0	4.0	1.0	1.0	65.0	485.0	
4	21994	10.0	10.0	2.0	4.0	1.0	1.0	65.0	490.0	
5	24215	2.0	2.0	5.0	1.5	2.0	2.0	685.0	0	
...	...	...	...	...	...	...	...	...	...	
7571	94857021	2.0	2.0	2.0	1.0	2.0	2.0	475.0	0	
7572	4430421	92.0	92.0	3.0	1.0	1.0	1.0	115.0	0	
7573	50247302	2.0	2.0	6.0	1.5	2.0	2.0	500.0	0	
7574	163879334	3.0	3.0	6.0	1.0	2.0	3.0	180.0	0	
7575	236978	3.0	3.0	4.0	2.0	2.0	2.0	300.0	0	

7575 rows × 39 columns

In [137]:
```
df2.dtypes
```

Out[137]:
```
host_id                                         object
host_listings_count                             object
host_total_listings_count                       object
accommodates                                    object
bathrooms                                       object
bedrooms                                        object
beds                                            object
price                                           object
weekly_price                                    object
monthly_price                                   object
security_deposit                                object
cleaning_fee                                    object
guests_included                                 object
minimum_nights                                  object
maximum_minimum_nights                          object
minimum_maximum_nights                          object
maximum_maximum_nights                          object
minimum_nights_avg_ntm                          object
maximum_nights_avg_ntm                          object
availability_30                                 object
availability_60                                 object
availability_90                                 object
availability_365                                object
number_of_reviews                               object
number_of_reviews_ltm                           object
review_scores_rating                            object
review_scores_accuracy                          object
review_scores_cleanliness                       object
review_scores_checkin                           object
review_scores_communication                     object
review_scores_location                          object
review_scores_value                             object
calculated_host_listings_count                  object
calculated_host_listings_count_entire_homes     object
calculated_host_listings_count_private_rooms    object
calculated_host_listings_count_shared_rooms     object
neighbourhood                                    int64
room_type                                        int64
reviews_per_month                               object
dtype: object
```

In [138]:

```
df2['host_id'] = le.fit_transform(df2.host_id.values)
df2['host_listings_count'] = le.fit_transform(df2.host_listings_count.values)
df2['host_total_listings_count'] = le.fit_transform(df2.host_total_listings_count.values)
df2['accommodates'] = le.fit_transform(df2.accommodates.values)
df2['price'] = le.fit_transform(df2.price.values)
df2['guests_included'] = le.fit_transform(df2.guests_included.values)
df2['guests_included'] = le.fit_transform(df2.guests_included.values)
df2['minimum_nights'] = le.fit_transform(df2.minimum_nights.values)
df2['maximum_minimum_nights'] = le.fit_transform(df2.maximum_minimum_nights.values)
df2['minimum_maximum_nights'] = le.fit_transform(df2.minimum_maximum_nights.values)
df2['maximum_maximum_nights '] = le.fit_transform(df2.maximum_maximum_nights .values)
df2['minimum_nights_avg_ntm'] = le.fit_transform(df2.minimum_nights_avg_ntm.values)
df2['maximum_nights_avg_ntm'] = le.fit_transform(df2.maximum_nights_avg_ntm.values)
df2['availability_30'] = le.fit_transform(df2.availability_30.values)
df2['availability_60'] = le.fit_transform(df2.availability_60.values)
df2['availability_90'] = le.fit_transform(df2.availability_90.values)
df2['availability_365'] = le.fit_transform(df2.availability_365.values)
df2['number_of_reviews'] = le.fit_transform(df2.number_of_reviews.values)
df2['number_of_reviews_ltm'] = le.fit_transform(df2.number_of_reviews_ltm.values)
df2['calculated_host_listings_count'] = le.fit_transform(df2.calculated_host_listings_count.values)
df2['calculated_host_listings_count_entire_homes '] = le.fit_transform(df2.calculated_host_listings_count_ent
ire_homes .values)
df2['calculated_host_listings_count_private_rooms'] = le.fit_transform(df2.calculated_host_listings_count_pri
vate_rooms.values)
df2['calculated_host_listings_count_shared_rooms'] = le.fit_transform(df2.calculated_host_listings_count_shar
ed_rooms.values)
```

In [139]:

```
df2.dtypes
```

Out[139]:
```
host_id                                          int32
host_listings_count                              int32
host_total_listings_count                        int32
accommodates                                     int32
bathrooms                                       object
bedrooms                                        object
beds                                            object
price                                            int32
weekly_price                                    object
monthly_price                                   object
security_deposit                                object
cleaning_fee                                    object
guests_included                                  int64
minimum_nights                                   int32
maximum_minimum_nights                           int32
minimum_maximum_nights                           int32
maximum_maximum_nights                          object
minimum_nights_avg_ntm                           int32
maximum_nights_avg_ntm                           int32
availability_30                                  int32
availability_60                                  int32
availability_90                                  int32
availability_365                                 int32
number_of_reviews                                int32
number_of_reviews_ltm                            int32
review_scores_rating                            object
review_scores_accuracy                          object
review_scores_cleanliness                       object
review_scores_checkin                           object
review_scores_communication                     object
review_scores_location                          object
review_scores_value                             object
calculated_host_listings_count                   int32
calculated_host_listings_count_entire_homes     object
calculated_host_listings_count_private_rooms     int32
calculated_host_listings_count_shared_rooms      int32
neighbourhood                                    int64
room_type                                        int64
reviews_per_month                               object
maximum_maximum_nights                           int32
calculated_host_listings_count_entire_homes      int32
dtype: object
```

In [140]:
```
X = df2.drop('price', axis = 1)
lm=LinearRegression()
lm
```

Out[140]:
```
LinearRegression()
```

In [141]:
```
lm.fit(X,df2.price)
```

Out[141]:
```
LinearRegression()
```

In [142]:
```
lm.predict(X)
```

Out[142]:
```
array([162.42074521, 204.01203772, 321.55025648, ..., 178.39610041,
       170.63379757, 185.55723681])
```

In [143]:
```
print('Estimated intercept coefficient:',lm.intercept_)
```

```
Estimated intercept coefficient: 154.43218419155556
```

In [144]:
```
print("Number of coeffiients:", len(lm.coef_))
```

```
Number of coeffiients: 40
```

In [145]:

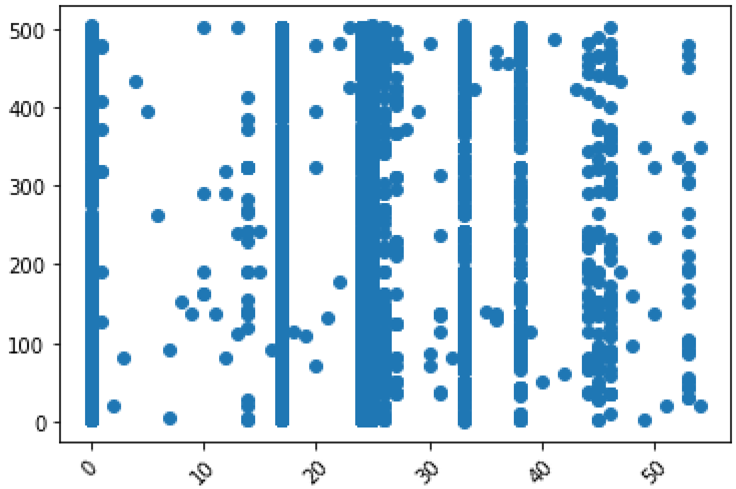
```
pd.DataFrame(zip(X.columns, lm.coef_), columns = ['features', 'estimatedCoefficients'])
```

Out[145]:

	features	estimatedCoefficients
0	host_id	0.002568
1	host_listings_count	0.218014
2	host_total_listings_count	0.218014
3	accommodates	-5.235693
4	bathrooms	16.185030
5	bedrooms	20.502325
6	beds	7.485358
7	weekly_price	0.003900
8	monthly_price	-0.002278
9	security_deposit	-0.010565
10	cleaning_fee	0.092709
11	guests_included	-3.305059
12	minimum_nights	0.683907
13	maximum_minimum_nights	0.129070
14	minimum_maximum_nights	-0.103230
15	maximum_maximum_nights	0.000002
16	minimum_nights_avg_ntm	-0.223200
17	maximum_nights_avg_ntm	0.302968
18	availability_30	-0.198812
19	availability_60	0.056494
20	availability_90	-0.110744
21	availability_365	0.032356
22	number_of_reviews	0.004312
23	number_of_reviews_ltm	-0.010657
24	review_scores_rating	-1.098829
25	review_scores_accuracy	11.628345

	features	estimatedCoefficients
26	review_scores_cleanliness	-3.602554
27	review_scores_checkin	11.602738
28	review_scores_communication	9.640296
29	review_scores_location	-13.917118
30	review_scores_value	-4.477288
31	calculated_host_listings_count	0.404228
32	calculated_host_listings_count_entire_homes	-0.307295
33	calculated_host_listings_count_private_rooms	1.704368
34	calculated_host_listings_count_shared_rooms	-0.853777
35	neighbourhood	-0.039407
36	room_type	41.415688
37	reviews_per_month	-4.036389
38	maximum_maximum_nights	-0.151252
39	calculated_host_listings_count_entire_homes	-1.175656

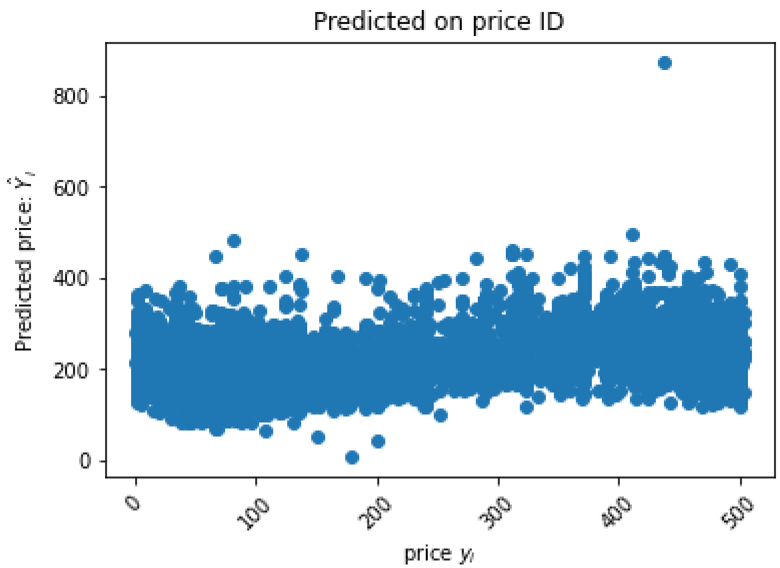
In [146]:
```
# True PLot
plt.scatter(df2.minimum_nights, df2.price)
plt.xlabel('')
plt.ylabel('')
plt.title('')
plt.xticks(rotation=45)
plt.show()
```



In [147]:
```
lm.predict(X)[0:5]
```

Out[147]:
```
array([162.42074521, 204.01203772, 321.55025648, 353.99479439,
       210.71661718])
```

In [148]:
```
# Prediction PLot
plt.scatter(df2.price, lm.predict(X))
plt.xlabel('price $y_i$')
plt.ylabel('Predicted price: $\hat{Y}_i$')
plt.title('Predicted on price ID')
plt.xticks(rotation=45)
plt.show()
```



In [149]:
```
mseFull = np.mean((df2.price - lm.predict(X)) ** 2)
print(mseFull)
```

```
22110.75385550316
```

In [150]:
```
v = np.var(X)
print("variance", v)
```

```
variance host_id                                      1.442344e+06
host_listings_count                              3.565887e+02
host_total_listings_count                        3.565887e+02
accommodates                                     1.003424e+01
guests_included                                  6.480794e+00
minimum_nights                                   1.206550e+02
maximum_minimum_nights                           1.706717e+03
minimum_maximum_nights                           1.867289e+03
minimum_nights_avg_ntm                           2.352391e+03
maximum_nights_avg_ntm                           4.375798e+03
availability_30                                  1.228940e+02
availability_60                                  4.484524e+02
availability_90                                  9.087368e+02
availability_365                                 1.466309e+04
number_of_reviews                                1.988508e+04
number_of_reviews_ltm                            1.350129e+03
calculated_host_listings_count                   1.325670e+02
calculated_host_listings_count_private_rooms     3.688758e+01
calculated_host_listings_count_shared_rooms      3.260231e+00
neighbourhood                                    2.516769e+02
room_type                                        3.040613e-01
maximum_maximum_nights                           5.788197e+03
calculated_host_listings_count_entire_homes      4.803761e+01
dtype: float64
```

In [151]:
```
lm = LinearRegression()
lm.fit(X[['minimum_nights']], df2.price)
```

Out[151]:
```
LinearRegression()
```

In [155]:
```
mse2 = np.mean((df2.price - lm.predict(X[['minimum_nights']])) ** 2)
mse2
```

Out[155]:
```
25394.1741190435
```

In [156]:
```
X_train, X_test, Y_train, Y_test = train_test_split(
    X, df2.price, test_size=.33, random_state=1000)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(5075, 40)
(2500, 40)
(5075,)
(2500,)
```

In [157]:
```
lm=LinearRegression()
lm.fit(X_train, Y_train)
pred_train = lm.predict(X_train)
pred_test = lm.predict(X_test)
```

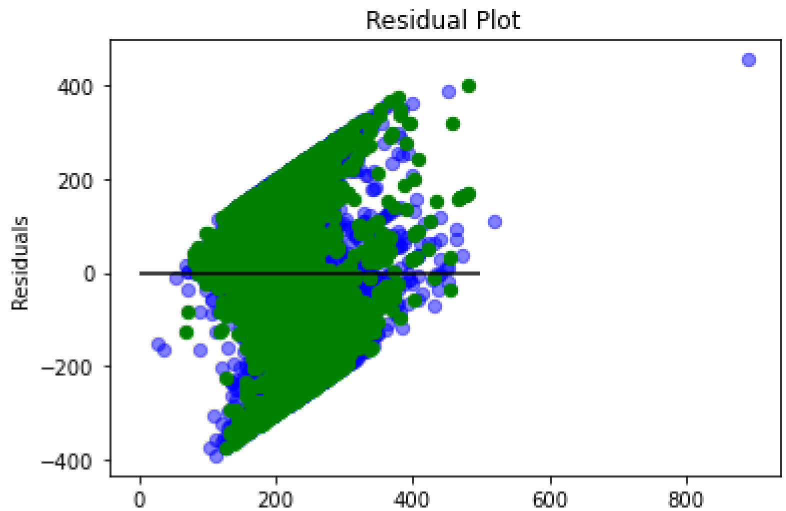
In [158]:
```
print("Fit a model X_train, and calculate MSE with Y_train:"), np.mean((Y_train-lm.predict(X_train)) ** 2)
print("Fit a model X_train, and calculate MSE with X_test, Y_test:"), np.mean((Y_test-lm.predict(X_test)) **
2)
```

```
Fit a model X_train, and calculate MSE with Y_train:
Fit a model X_train, and calculate MSE with X_test, Y_test:
```

Out[158]:
```
(None, 22895.8325980217)
```

In [159]:
```
plt.scatter(lm.predict(X_train), lm.predict(X_train) - Y_train, c='b', s=40, alpha=0.5)
plt.scatter(lm.predict(X_test), lm.predict(X_test) - Y_test, c='g', s=40)
plt.hlines(y=0, xmin=0, xmax=500, color="black")
plt.title('Residual Plot')
plt.ylabel('Residuals')
```

Out[159]:
```
Text(0, 0.5, 'Residuals')
```



In [ ]:
```
# Decision Tree
```

In [37]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import metrics
from sklearn.tree import export_graphviz
from IPython.display import Image
from sklearn import tree
```

In [67]:
```
df4 = pd.read_csv("sf_airbnb_listings.csv", sep=",", header=None, engine='python', encoding="utf-8-sig")
```

In [68]:

```
header = df4.iloc[0]
# take the rest of your data minus the header row
df4 = df4[1:]
# set the header row as the df header
df4.columns = header
pd.set_option('display.max_rows', 7500)
pd.set_option('display.max_columns', 106)
df4
```

Out[68]:

	id	listing_url	scrape_id	last_scraped	name	summary	space	description
1	958	https://www.airbnb.com/rooms/958	2.02E+13	6/2/2019	Bright, Modern Garden Unit - 1BR/1B	New update: the house next door is under const...	Newly remodeled, modern, and bright garden uni...	New update: the house next door is under const...
2	5858	https://www.airbnb.com/rooms/5858	2.02E+13	6/2/2019	Creative Sanctuary	NaN	We live in a large Victorian house on a quiet ...	We live in a large Victorian house on a quiet ...
3	7918	https://www.airbnb.com/rooms/7918	2.02E+13	6/2/2019	A Friendly Room - UCSF/USF - San Francisco	Nice and good public transportation. 7 minute...	Room rental-sunny view room/sink/Wi Fi (inner ...	Nice and good public transportation. 7 minute...
4	8142	https://www.airbnb.com/rooms/8142	2.02E+13	6/2/2019	Friendly Room Apt. Style - UCSF/USF - San Franc...	Nice and good public transportation. 7 minute...	Room rental Sunny view Rm/Wi-Fi/TV/sink/large ...	Nice and good public transportation. 7 minute...
5	8339	https://www.airbnb.com/rooms/8339	2.02E+13	6/2/2019	Historic Alamo Square Victorian	Pls email before booking. Interior featured i...	Please send us a quick message before booking ...	Pls email before booking. Interior featured i...
...	...	...	...	...	...	...	...	...
7571	35284961	https://www.airbnb.com/rooms/35284961	2.02E+13	6/2/2019	Brand New Designer 2 BR SF Condo	Luxury spacious 2 bedroom condo located in SF,...	Private dedicated entrance NEST temperature c...	Luxury spacious 2 bedroom condo located in SF,...
7572	35285751	https://www.airbnb.com/rooms/35285751	2.02E+13	6/2/2019	Beautiful 1x1 in Historic Mission Tudor Building	A beautifully remodeled one bedroom in a great...	NaN	A beautifully remodeled one bedroom in a great...

	id	listing_url	scrape_id	last_scraped	name	summary	space	description
7573	35286441	https://www.airbnb.com/rooms/35286441	2.02E+13	6/2/2019	Beautiful Queen Victorian in the heart of Mission	Our place is a charming Victorian located in t...	The house is a quintessential remodeled Victor...	Our place is a charming Victorian located in t...
7574	35288483	https://www.airbnb.com/rooms/35288483	2.02E+13	6/2/2019	New comfortable, convenient place for family	This new place is comfortable, with easy commu...	NaN	This new place is comfortable, with easy commu...
7575	35291911	https://www.airbnb.com/rooms/35291911	2.02E+13	6/2/2019	Spacious 2bdrm/2bath in the heart of SF	Freshly remodeled in May 2019, 2 bedroom 2 ba...	- Centrally located - Steps away from 16th and...	Freshly remodeled in May 2019, 2 bedroom 2 ba...

7575 rows × 106 columns

In [225]:

```
columns4="host_listings_count accommodates host_response_rate bedrooms beds guests_included maximum_nights mi
nimum_nights availability_30 availability_365 number_of_reviews  review_scores_rating price weekly_price mont
hly_price security_deposit cleaning_fee room_type reviews_per_month".split()
columns5="host_id host_listings_count accommodates bathrooms bedrooms beds guests_included minimum_nights ava
ilability_30 availability_365 number_of_reviews  review_scores_rating  neighbourhood room_type reviews_per_mo
nth".split()
df6=pd.DataFrame(df4, columns=columns4)
df6.fillna(0, inplace=True)
df6
#df6.to_csv('df6.csv')
```

Out[225]:

	host_listings_count	accommodates	host_response_rate	bedrooms	beds	guests_included	maximum_nights	minimum_nights	a
1	1.0	3.0	100%	1.0	2.0	2.0	30.0	1.0	
2	2.0	5.0	100%	2.0	3.0	2.0	60.0	30.0	
3	10.0	2.0	100%	1.0	1.0	1.0	60.0	32.0	
4	10.0	2.0	100%	1.0	1.0	1.0	90.0	32.0	
5	2.0	5.0	100%	2.0	2.0	2.0	1125.0	4.0	
...	...	...	...	...	...	...	...	...	
7571	2.0	2.0	100%	2.0	2.0	1.0	1125.0	3.0	
7572	92.0	3.0	99%	1.0	1.0	1.0	150.0	30.0	
7573	2.0	6.0	100%	2.0	2.0	4.0	10.0	1.0	
7574	3.0	6.0	100%	2.0	3.0	4.0	1125.0	30.0	
7575	3.0	4.0	100%	2.0	2.0	4.0	1125.0	1.0	

7575 rows × 19 columns

In [226]:
```
#df6['neighbourhood'] = le.fit_transform(df6.neighbourhood.values)
df6['room_type'] = le.fit_transform(df6.room_type.values)
df6['host_response_rate'] = df6['host_response_rate'].str.rstrip('%').astype('float') / 100.0
# df6['host_response_rate'] = le.fit_transform(df6.host_response_rate.values)
df6
```

Out[226]:

	host_listings_count	accommodates	host_response_rate	bedrooms	beds	guests_included	maximum_nights	minimum_nights	a
1	1.0	3.0	1.00	1.0	2.0	2.0	30.0	1.0	
2	2.0	5.0	1.00	2.0	3.0	2.0	60.0	30.0	
3	10.0	2.0	1.00	1.0	1.0	1.0	60.0	32.0	
4	10.0	2.0	1.00	1.0	1.0	1.0	90.0	32.0	
5	2.0	5.0	1.00	2.0	2.0	2.0	1125.0	4.0	
...	...	...	...	...	...	...	...	...	
7571	2.0	2.0	1.00	2.0	2.0	1.0	1125.0	3.0	
7572	92.0	3.0	0.99	1.0	1.0	1.0	150.0	30.0	
7573	2.0	6.0	1.00	2.0	2.0	4.0	10.0	1.0	
7574	3.0	6.0	1.00	2.0	3.0	4.0	1125.0	30.0	
7575	3.0	4.0	1.00	2.0	2.0	4.0	1125.0	1.0	

7575 rows × 19 columns

In [227]:
```
df6 = df6.dropna(axis=1)
df6
```

Out[227]:

	host_listings_count	accommodates	bedrooms	beds	guests_included	maximum_nights	minimum_nights	availability_30	availab
1	1.0	3.0	1.0	2.0	2.0	30.0	1.0	1.0	
2	2.0	5.0	2.0	3.0	2.0	60.0	30.0	0.0	
3	10.0	2.0	1.0	1.0	1.0	60.0	32.0	30.0	
4	10.0	2.0	1.0	1.0	1.0	90.0	32.0	11.0	
5	2.0	5.0	2.0	2.0	2.0	1125.0	4.0	30.0	
...	...	...	...	...	...	...	...	...	
7571	2.0	2.0	2.0	2.0	1.0	1125.0	3.0	26.0	
7572	92.0	3.0	1.0	1.0	1.0	150.0	30.0	23.0	
7573	2.0	6.0	2.0	2.0	4.0	10.0	1.0	7.0	
7574	3.0	6.0	2.0	3.0	4.0	1125.0	30.0	16.0	
7575	3.0	4.0	2.0	2.0	4.0	1125.0	1.0	19.0	

7575 rows × 18 columns

In [228]:
```
main_columns = columns4
X = df6 # main data
y = df.neighbourhood # Target variable
```

In [229]:
```
df6.shape
```

Out[229]: (7575, 18)

In [230]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.40, random_state=2000)
```

In [231]:
```
classifier = DecisionTreeClassifier(max_depth=2) # chooses the depth
classifier.fit(X_train, y_train)
```

Out[231]: DecisionTreeClassifier(max_depth=2)

In [232]:
```
y_pred = classifier.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred)) # returns the accuracy
```

```
Accuracy: 0.14785478547854786
```

In [233]:

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

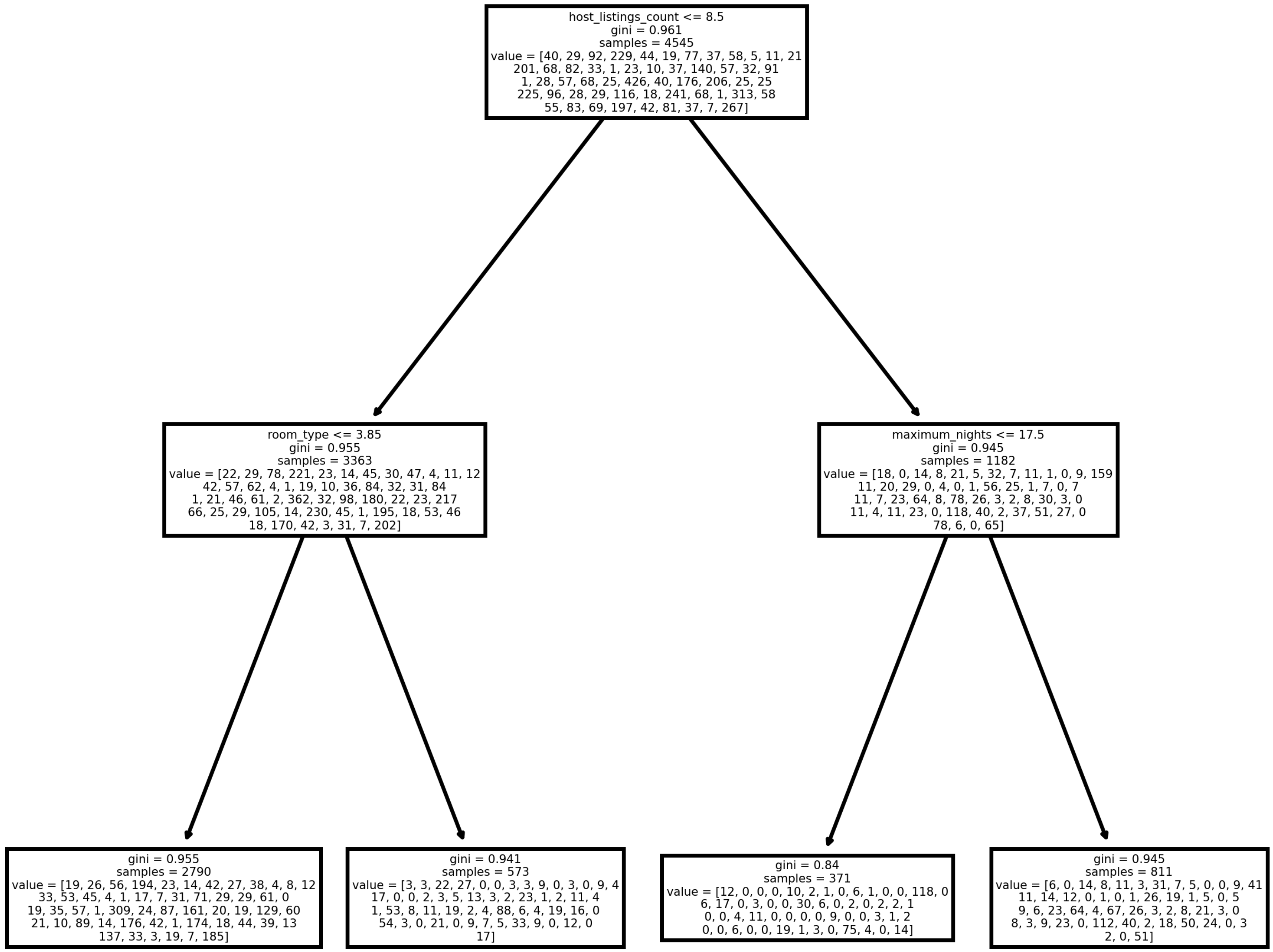
```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
                      precision    recall  f1-score   support

         Alamo Square       0.00      0.00      0.00        23
       Balboa Terrace       0.00      0.00      0.00        16
              Bayview       0.00      0.00      0.00        78
       Bernal Heights       0.00      0.00      0.00       165
            Chinatown       0.00      0.00      0.00        22
         Civic Center       0.00      0.00      0.00         6
          Cole Valley       0.00      0.00      0.00        48
           Cow Hollow       0.00      0.00      0.00        25
       Crocker Amazon       0.00      0.00      0.00        45
            Daly City       0.00      0.00      0.00         1
      Diamond Heights       0.00      0.00      0.00         8
             Dogpatch       0.00      0.00      0.00        15
             Downtown       0.34      0.58      0.43       146
      Duboce Triangle       0.00      0.00      0.00        47
            Excelsior       0.00      0.00      0.00        45
   Financial District       0.00      0.00      0.00        32
   Fisherman''s Wharf       0.00      0.00      0.00        24
          Forest Hill       0.00      0.00      0.00         5
            Glen Park       0.00      0.00      0.00        34
       Haight-Ashbury       0.00      0.00      0.00        70
         Hayes Valley       0.00      0.00      0.00        32
            Ingleside       0.00      0.00      0.00        19
         Inner Sunset       0.00      0.00      0.00        59
            Japantown       0.00      0.00      0.00         5
            Lakeshore       0.00      0.00      0.00        25
         Lower Haight       0.00      0.00      0.00        32
               Marina       0.00      0.00      0.00        37
          Mission Bay       0.00      0.00      0.00        19
     Mission District       0.12      0.74      0.20       296
      Mission Terrace       0.00      0.00      0.00        30
             Nob Hill       0.00      0.00      0.00       133
           Noe Valley       0.00      0.00      0.00       158
          North Beach       0.00      0.00      0.00        17
            Oceanview       0.00      0.00      0.00        19
```

```
          Outer Sunset       0.13      0.38      0.20       136
      Pacific Heights       0.00      0.00      0.00        83
             Parkside       0.00      0.00      0.00        16
              Portola       0.00      0.00      0.00        17
         Potrero Hill       0.00      0.00      0.00        88
             Presidio       0.00      0.00      0.00         1
     Presidio Heights       0.00      0.00      0.00        13
    Richmond District       0.00      0.00      0.00       121
         Russian Hill       0.00      0.00      0.00        40
            Sea Cliff       0.00      0.00      0.00         2
                 SoMa       0.17      0.43      0.24       213
          South Beach       0.00      0.00      0.00        42
            Sunnyside       0.00      0.00      0.00        40
       Telegraph Hill       0.00      0.00      0.00        51
           Tenderloin       0.00      0.00      0.00        39
           The Castro       0.00      0.00      0.00       116
           Twin Peaks       0.00      0.00      0.00        25
         Union Square       0.00      0.00      0.00        54
    Visitacion Valley       0.00      0.00      0.00        22
          West Portal       0.00      0.00      0.00         4
Western Addition/NOPA       0.00      0.00      0.00       171

             accuracy                           0.15      3030
            macro avg       0.01      0.04      0.02      3030
         weighted avg       0.05      0.15      0.07      3030


C:\Users\kevin\Anaconda3\envs\r-tutorial\lib\site-packages\sklearn\metrics\_classification.py:1221: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sample
s. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [234]:

```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (6,6), dpi=500)
tree.plot_tree(classifier, feature_names = main_columns, filled=False) # plots the tree
```

```
Out[234]: [Text(1162.5, 1887.5, 'host_listings_count <= 8.5\ngini = 0.961\nsamples = 4545\nvalue = [40, 29, 92, 229, 4
4, 19, 77, 37, 58, 5, 11, 21\n201, 68, 82, 33, 1, 23, 10, 37, 140, 57, 32, 91\n1, 28, 57, 68, 25, 426, 40, 17
6, 206, 25, 25\n225, 96, 28, 29, 116, 18, 241, 68, 1, 313, 58\n55, 83, 69, 197, 42, 81, 37, 7, 267]'),
 Text(581.25, 1132.5, 'room_type <= 3.85\ngini = 0.955\nsamples = 3363\nvalue = [22, 29, 78, 221, 23, 14, 45,
30, 47, 4, 11, 12\n42, 57, 62, 4, 1, 19, 10, 36, 84, 32, 31, 84\n1, 21, 46, 61, 2, 362, 32, 98, 180, 22, 23,
217\n66, 25, 29, 105, 14, 230, 45, 1, 195, 18, 53, 46\n18, 170, 42, 3, 31, 7, 202]'),
 Text(290.625, 377.5, 'gini = 0.955\nsamples = 2790\nvalue = [19, 26, 56, 194, 23, 14, 42, 27, 38, 4, 8, 12\n
33, 53, 45, 4, 1, 17, 7, 31, 71, 29, 29, 61, 0\n19, 35, 57, 1, 309, 24, 87, 161, 20, 19, 129, 60\n21, 10, 89,
14, 176, 42, 1, 174, 18, 44, 39, 13\n137, 33, 3, 19, 7, 185]'),
 Text(871.875, 377.5, 'gini = 0.941\nsamples = 573\nvalue = [3, 3, 22, 27, 0, 0, 3, 3, 9, 0, 3, 0, 9, 4\n17,
0, 0, 2, 3, 5, 13, 3, 2, 23, 1, 2, 11, 4\n1, 53, 8, 11, 19, 2, 4, 88, 6, 4, 19, 16, 0\n54, 3, 0, 21, 0, 9, 7,
5, 33, 9, 0, 12, 0\n17]'),
 Text(1743.75, 1132.5, 'maximum_nights <= 17.5\ngini = 0.945\nsamples = 1182\nvalue = [18, 0, 14, 8, 21, 5, 3
2, 7, 11, 1, 0, 9, 159\n11, 20, 29, 0, 4, 0, 1, 56, 25, 1, 7, 0, 7\n11, 7, 23, 64, 8, 78, 26, 3, 2, 8, 30, 3,
0\n11, 4, 11, 23, 0, 118, 40, 2, 37, 51, 27, 0\n78, 6, 0, 65]'),
 Text(1453.125, 377.5, 'gini = 0.84\nsamples = 371\nvalue = [12, 0, 0, 0, 10, 2, 1, 0, 6, 1, 0, 0, 118, 0\n6,
17, 0, 3, 0, 0, 30, 6, 0, 2, 0, 2, 2, 1\n0, 0, 4, 11, 0, 0, 0, 0, 9, 0, 0, 3, 1, 2\n0, 0, 6, 0, 0, 19, 1, 3,
0, 75, 4, 0, 14]'),
 Text(2034.375, 377.5, 'gini = 0.945\nsamples = 811\nvalue = [6, 0, 14, 8, 11, 3, 31, 7, 5, 0, 0, 9, 41\n11,
14, 12, 0, 1, 0, 1, 26, 19, 1, 5, 0, 5\n9, 6, 23, 64, 4, 67, 26, 3, 2, 8, 21, 3, 0\n8, 3, 9, 23, 0, 112, 40,
2, 18, 50, 24, 0, 3\n2, 0, 51]')]
```


host_listings_count <= 8.5
gini = 0.961
samples = 4545
value = [40, 29, 92, 229, 44, 19, 77, 37, 58, 5, 11, 21
201, 68, 82, 33, 1, 23, 10, 37, 140, 57, 32, 91
1, 28, 57, 68, 25, 426, 40, 176, 206, 25, 25
225, 96, 28, 29, 116, 18, 241, 68, 1, 313, 58
55, 83, 69, 197, 42, 81, 37, 7, 267]
room_type <= 3.85
gini = 0.955
samples = 3363
value = [22, 29, 78, 221, 23, 14, 45, 30, 47, 4, 11, 12
42, 57, 62, 4, 1, 19, 10, 36, 84, 32, 31, 84
1, 21, 46, 61, 2, 362, 32, 98, 180, 22, 23, 217
66, 25, 29, 105, 14, 230, 45, 1, 195, 18, 53, 46
18, 170, 42, 3, 31, 7, 202]
maximum_nights <= 17.5
gini = 0.945
samples = 1182
value = [18, 0, 14, 8, 21, 5, 32, 7, 11, 1, 0, 9, 159
11, 20, 29, 0, 4, 0, 1, 56, 25, 1, 7, 0, 7
11, 7, 23, 64, 8, 78, 26, 3, 2, 8, 30, 3, 0
11, 4, 11, 23, 0, 118, 40, 2, 37, 51, 27, 0
78, 6, 0, 65]
gini = 0.955
samples = 2790
value = [19, 26, 56, 194, 23, 14, 42, 27, 38, 4, 8, 12
33, 53, 45, 4, 1, 17, 7, 31, 71, 29, 29, 61, 0
19, 35, 57, 1, 309, 24, 87, 161, 20, 19, 129, 60
21, 10, 89, 14, 176, 42, 1, 174, 18, 44, 39, 13
137, 33, 3, 19, 7, 185]
gini = 0.941
samples = 573
value = [3, 3, 22, 27, 0, 0, 3, 3, 9, 0, 3, 0, 9, 4
17, 0, 0, 2, 3, 5, 13, 3, 2, 23, 1, 2, 11, 4
1, 53, 8, 11, 19, 2, 4, 88, 6, 4, 19, 16, 0
54, 3, 0, 21, 0, 9, 7, 5, 33, 9, 0, 12, 0
17]
gini = 0.84
samples = 371
value = [12, 0, 0, 0, 10, 2, 1, 0, 6, 1, 0, 0, 118, 0
6, 17, 0, 3, 0, 0, 30, 6, 0, 2, 0, 2, 2, 1
0, 0, 4, 11, 0, 0, 0, 0, 9, 0, 0, 3, 1, 2
0, 0, 6, 0, 0, 19, 1, 3, 0, 75, 4, 0, 14]
gini = 0.945
samples = 811
value = [6, 0, 14, 8, 11, 3, 31, 7, 5, 0, 0, 9, 41
11, 14, 12, 0, 1, 0, 1, 26, 19, 1, 5, 0, 5
9, 6, 23, 64, 4, 67, 26, 3, 2, 8, 21, 3, 0
8, 3, 9, 23, 0, 112, 40, 2, 18, 50, 24, 0, 3
2, 0, 51]

In [ ]:

```

```

In [ ]:

```

```