blog_post

September 9, 2019

1 1. Business Understanding

- How to make a next step towards earning extra money with your property?
- What attributes increase my Airbnb listing price?
- What attributes decrease my Airbnb listing price?
- How to maximize my Airbnb profits?

2 2. Data Understanding

[46]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from itertools import chain
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
%matplotlib inline
```

[47]:
```
df = pd.read_csv("data/listings.csv")
```

2.1 Inspection of the data set

Before we start changing the data set, we have to go column after column and investigate the data types, irrelevant columns and duplicates. To achieve this, we used the following methods. Below we show only the investigating methods for one column exemplary.

The most important part of the data analysis is to achieve an understanding about data we have. To get a high validity of data, we first have to check, if the **data types** are correct. Often number are stored as strings not as numerical data types. To have a correct data types for the analysis, we have to convert the data types, often using regular expressions. Second, **irrelevant data** and **duplicates** have to be removed. Third, we have to investigate **missing data**.

[48]:
```
df["security_deposit"].shape[0] #Provide the number of rows in the column
df["security_deposit"].dtypes #Provide the data type of the column
df["security_deposit"].isnull().sum() #Count empty entries within the column
df["security_deposit"].describe() #Show simple statistics of the column
df["security_deposit"].unique() #Show variables stored in the column
```

```
df["security_deposit"].head(5) #Show first entries of the column
```

[48]:
```
0          NaN
1      $100.00
2    $1,000.00
3          NaN
4      $700.00
Name: security_deposit, dtype: object
```

Source: https://towardsdatascience.com/the-ultimate-guide-to-data-cleaning-3969843991d4

2.2 Cleaning the data set from irrelevant data and duplicates.

After we inspected our data set by looking into 92 columns, we have to drop irrelevant data: - all url columns, which does not help us with the analysis and contain no relevant information - all ids, description, irrelevant columns or duplicates - all information which does not improve the analysis like city, state etc.

[49]:
```
to_drop =[#Drop all url columns, which does not help us with the analysis and␣
 ↪contain no relevant information.
            "listing_url",
            "thumbnail_url",
            "medium_url",
            "picture_url",
            "xl_picture_url",
            "host_url",
            "host_thumbnail_url",
            "host_picture_url",
            #Drop all ids.
            "id",
            "scrape_id",
            "host_id",
            #Drop all descriptions.
            "name",
            "summary",
            "description",
            "space",
            "neighborhood_overview",
            "host_about",
            "notes",
            "transit",
            #Drop irrelevant columns, or columns with no data
            "host_name",
            "host_location",
            "host_neighbourhood",
            "host_verifications",
            "last_scraped",
            "calendar_last_scraped",
            #future data irrelevant for the past
```

```
            "has_availability",
            "availability_30",
            "availability_60",
            "availability_90",
            "availability_365",
            "requires_license",
            #drop information about the past, since we will not investigate it
            "host_since",
            "first_review",
            "last_review",
            #Drop duplicates: weekly and monthly price. We are only interested␣
↪in general price
            "weekly_price",
            "monthly_price",
            #Drop empty columns with empty values
            "experiences_offered",
            #Drop redudant information (similar to host_response_rate)
            "host_response_time",
            #Drop redudant information (similar to review_scores_location)

            #Drop redudant information (similar to neighbourhood_group_cleansed)
            "neighbourhood_cleansed",
            "neighbourhood",
            "street",
            #Drop host_listing variables, as calculated_host_listings_count is␣
↪more accurate
            "host_listings_count",
            "host_total_listings_count",
            #Drop information which does not improve the analysis
            "city",
            "state",
            "zipcode",
            "market",
            "smart_location",
            "country_code",
            "country",
            "jurisdiction_names",
            "calendar_updated",
            "latitude",
            "longitude"
        ]
```

[50]:
```
df = df.drop(columns=to_drop, axis=1)
```

2.3 Converting data types from objects to correct ones.

In the second step we transform the object data types. Unfortunately while reading the csv-file, the correct data types was not recognized. Using regular expressions, we transform the string objects, so that we can convert the data to correct types.

2.3.1 Covert to a boolean

[51]:
```
df=df.replace(to_replace=["t", "f","nan"], value=[1, 0, np.nan])
```

2.3.2 Convert to float

[52]:
```
perc_to_float = ["host_response_rate","host_acceptance_rate"]
df[perc_to_float] = df[perc_to_float].replace(regex=["%"], value="").astype(np.
 ↪float16) / 100.0
```

2.3.3 Remove commas and dollars signs from the columns and convert to float

[53]:
```
to_float = ["price",
            "security_deposit",
            "cleaning_fee",
            "extra_people"]
df[to_float] = df[to_float].replace({
                                    "\$": "",
                                     ",": ""}, regex=True)
df[to_float]=df[to_float].astype(np.float)
```

2.3.4 Convert to integer

[54]:
```
df=df.dropna(subset=["host_has_profile_pic"])
df=df.dropna(subset=["host_is_superhost"])
to_int = [
        "host_is_superhost",
        "host_has_profile_pic",
        "host_identity_verified",
        "instant_bookable",
        "require_guest_profile_picture",
        "require_guest_phone_verification",
        "is_location_exact"
            ]
df[to_int]=df[to_int].astype(np.int)
```

2.3.5 Show categorical data.

[55]:
```
df_categorical = list(df.select_dtypes(include=['object']).columns)
```

2.3.6 Amenities Encoding

Unfortunately amenities are saved as a json, but read_csv recognizes only a string. In the first step we will split the string in parts and using regular expressions we will remove unnecessary characters. Then we will encode amenities as dummy variables

[56]:
```
amenities = df["amenities"].str.split(",",expand=True)
amenities.head(3)
```

[56]:
```
     0           1                        2                      3  \
0  {TV  "Cable TV"                 Internet  "Wireless Internet"
1  {TV    Internet  "Wireless Internet"                 Kitchen
2  {TV  "Cable TV"                 Internet  "Wireless Internet"

                          4                          5  \
0          "Air Conditioning"                    Kitchen
1  "Free Parking on Premises"  "Buzzer/Wireless Intercom"
2          "Air Conditioning"                    Kitchen

                          6                    7  \
0                    Heating  "Family/Kid Friendly"
1                    Heating  "Family/Kid Friendly"
2  "Free Parking on Premises"         "Pets Allowed"

                              8       9  ...        20    21    22    23  \
0                        Washer  Dryer}  ...      None  None  None  None
1                        Washer   Dryer  ...      None  None  None  None
2  "Pets live on this property"  Dog(s)  ...  Shampoo}  None  None  None

     24    25    26    27    28    29
0  None  None  None  None  None  None
1  None  None  None  None  None  None
2  None  None  None  None  None  None

[3 rows x 30 columns]
```

[57]:
```
amenities = amenities.replace(regex=["[^\w\s]"], value="")
```

[58]:
```
amenities_list = [amenities[item].unique().tolist() for item in amenities.
 ↪columns.values]
amenities_list = set(list(chain.from_iterable(amenities_list)))
amenities_list.remove('')
amenities_list.remove(None)
amenities_list
```

[58]:
```
{'24Hour Checkin',
 'Air Conditioning',
 'Breakfast',
 'BuzzerWireless Intercom',
 'Cable TV',
 'Carbon Monoxide Detector',
 'Cats',
 'Dogs',
 'Doorman',
 'Dryer',
 'Elevator in Building',
 'Essentials',
 'FamilyKid Friendly',
 'Fire Extinguisher',
 'First Aid Kit',
 'Free Parking on Premises',
 'Gym',
 'Hair Dryer',
 'Hangers',
 'Heating',
 'Hot Tub',
 'Indoor Fireplace',
 'Internet',
 'Iron',
 'Kitchen',
 'Laptop Friendly Workspace',
 'Lock on Bedroom Door',
 'Other pets',
 'Pets Allowed',
 'Pets live on this property',
 'Pool',
 'Safety Card',
 'Shampoo',
 'Smoke Detector',
 'Smoking Allowed',
 'Suitable for Events',
 'TV',
 'Washer',
 'Washer  Dryer',
 'Wheelchair Accessible',
 'Wireless Internet'}
```

[59]:
```
for items in amenities_list:
    df[items] = df["amenities"].apply(lambda x: 1 if items in x else 0)
```

[60]:
```
df.drop(columns="amenities",inplace=True)
```

2.3.7 Drop empty aminities

[61]:
```
drop_0 = [
            "Other pets",
            "Washer  Dryer",
            "Cats",
            "Dogs",
            "24Hour Checkin",
            "BuzzerWireless Intercom",
            "FamilyKid Friendly"
            ]
```

[62]:
```
df = df.drop(columns=drop_0, axis=1)
```

2.4 Analyze missing data

[63]:
```
#Provide a set of columns with 0 missing values.
no_nulls = set(df.columns[df.isnull().mean() == 0])
print(no_nulls)
```

```
{'Wheelchair Accessible', 'host_is_superhost', 'cancellation_policy', 'Lock on
Bedroom Door', 'room_type', 'Washer', 'accommodates', 'Free Parking on
Premises', 'Indoor Fireplace', 'Doorman', 'Smoke Detector', 'Dryer', 'Shampoo',
'is_location_exact', 'require_guest_profile_picture', 'Carbon Monoxide
Detector', 'Fire Extinguisher', 'Heating', 'Pets Allowed', 'TV',
'instant_bookable', 'require_guest_phone_verification', 'extra_people',
'Kitchen', 'Suitable for Events', 'maximum_nights', 'number_of_reviews', 'Cable
TV', 'host_identity_verified', 'Gym', 'price', 'Air Conditioning', 'Wireless
Internet', 'neighbourhood_group_cleansed', 'Pets live on this property', 'First
Aid Kit', 'Laptop Friendly Workspace', 'Safety Card', 'bed_type', 'Essentials',
'host_has_profile_pic', 'Pool', 'Internet', 'Breakfast', 'Hair Dryer', 'Iron',
'Smoking Allowed', 'calculated_host_listings_count', 'Elevator in Building',
'minimum_nights', 'Hot Tub', 'Hangers', 'guests_included'}
```

[64]:
```
drop_75 = [col for col in df.columns if df[col].isnull().sum()/  df.shape[0] > 0.
 ↪75]
drop_75
```

[64]:
```
['square_feet', 'license']
```

2.4.1 Drop columns with more than 75 % empty values

[65]:
```
df = df.drop(columns=drop_75, axis=1)
```

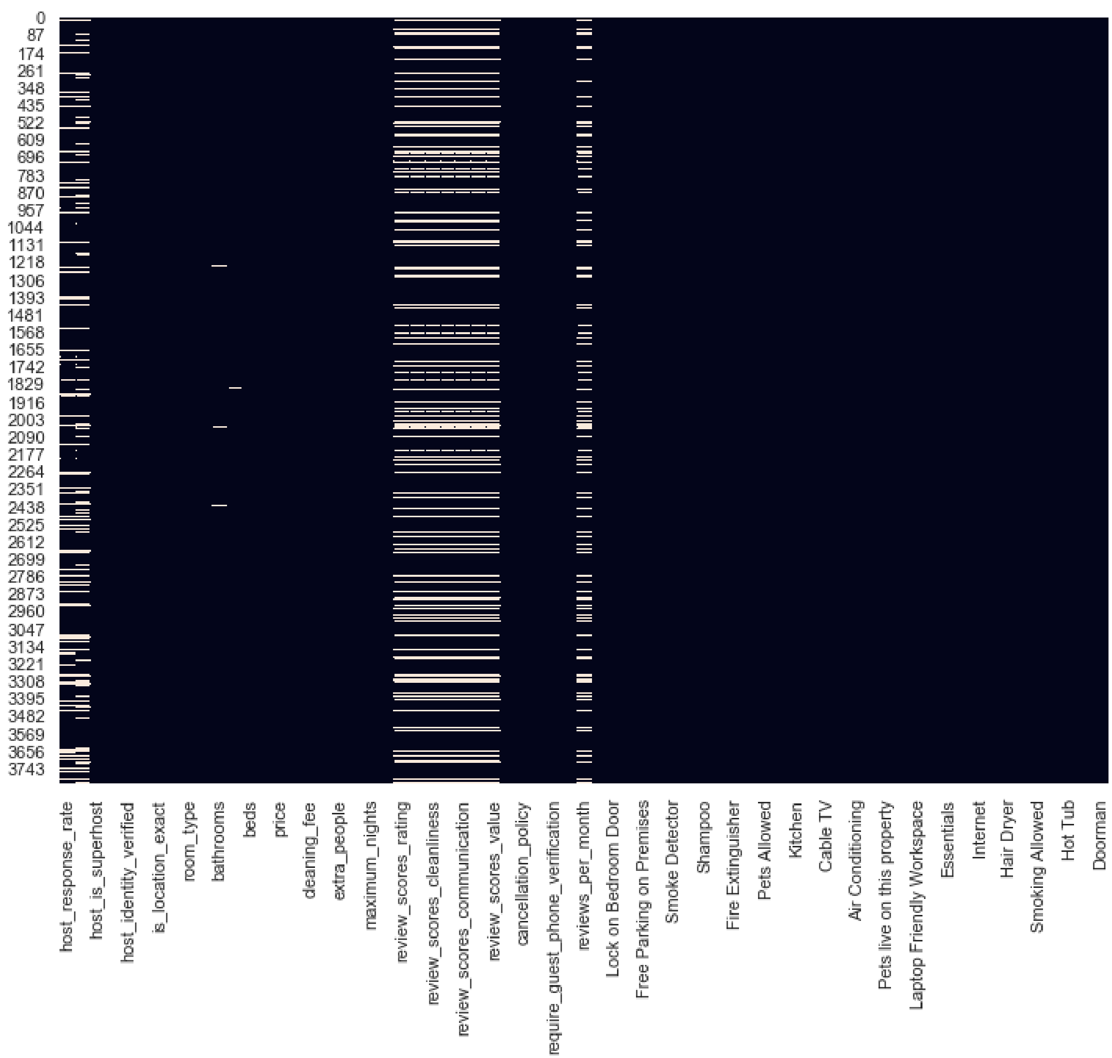
2.4.2 Fill missing data with entries

[66]:
```
nan_to_zero = ["security_deposit","cleaning_fee"]
df[nan_to_zero]=df[nan_to_zero].fillna(0)
```

2.4.3 Investigate missing data

[67]:
```
plt.figure(figsize=(12, 9))
sns.heatmap(df.isnull(), cbar=False)
```

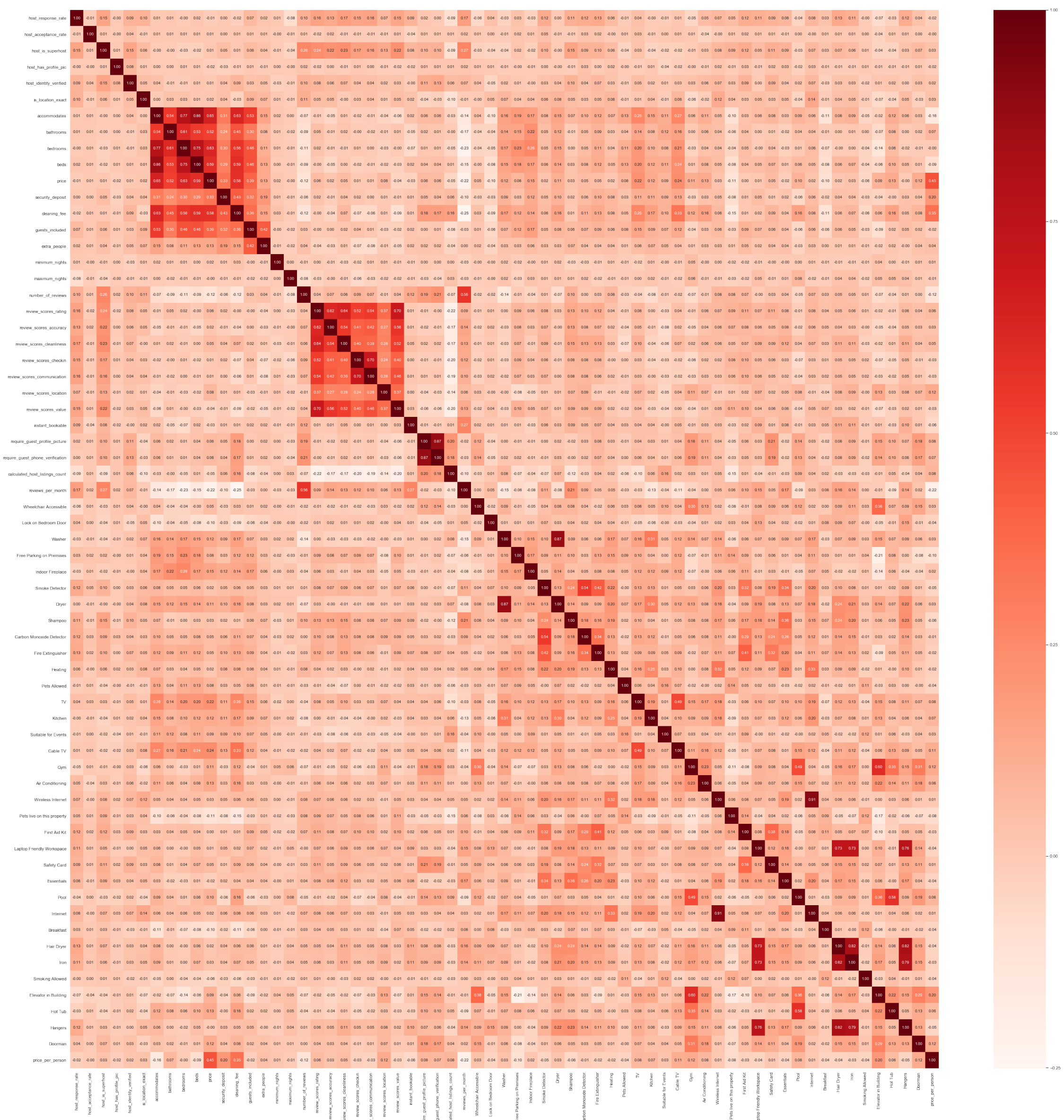
[67]: `<matplotlib.axes._subplots.AxesSubplot at 0x1a25a40d68>`



2.5 Analyze Data

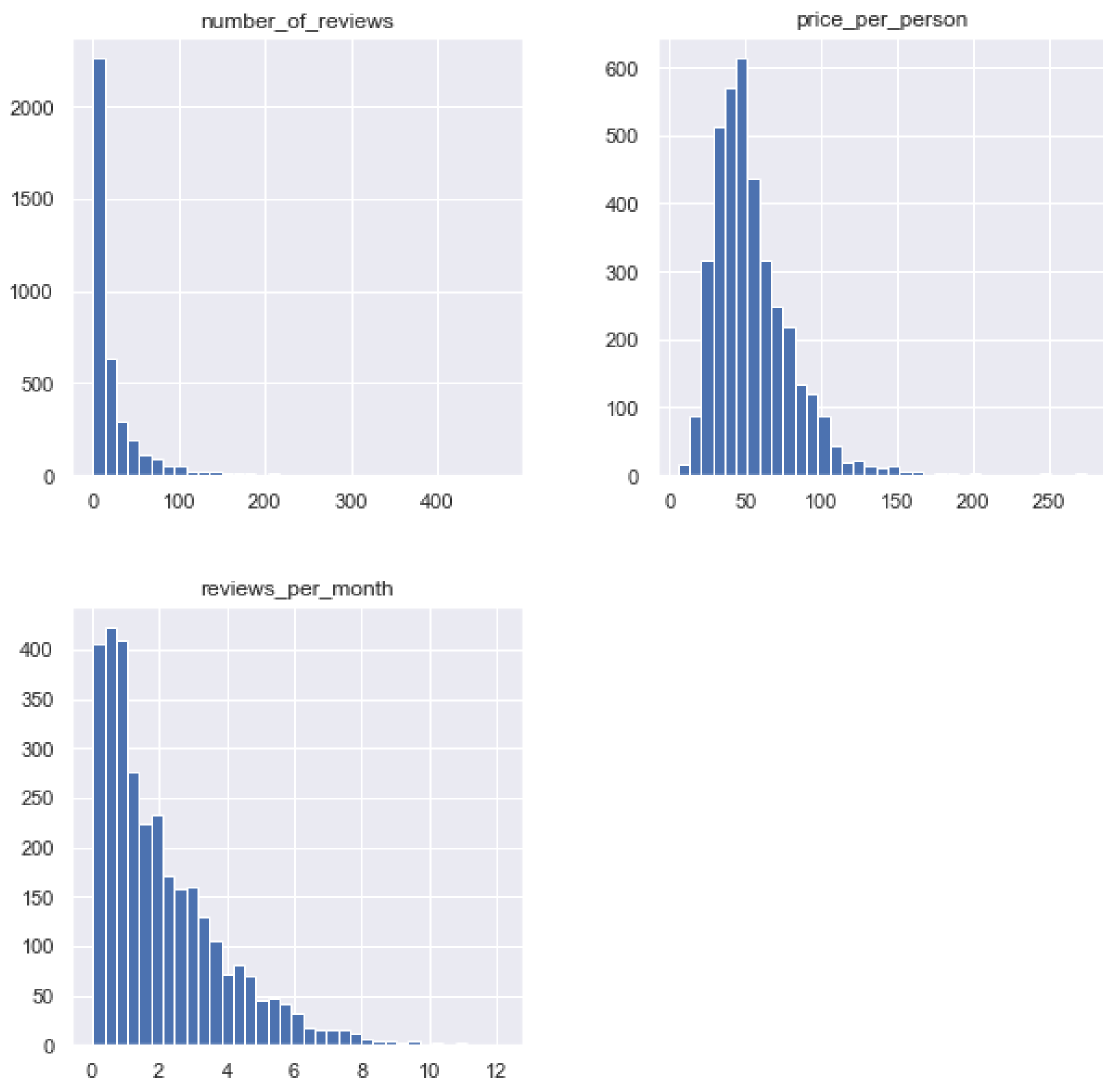
[68]:
```
df["price_per_person"] = (df["price"]+ df["cleaning_fee"])/ df["accommodates"]
df["price_per_person"].fillna(0, inplace = True)
```

[69]:
```
plt.figure(figsize=(50,50))
cor = df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds, fmt=".2f")
plt.show()
```



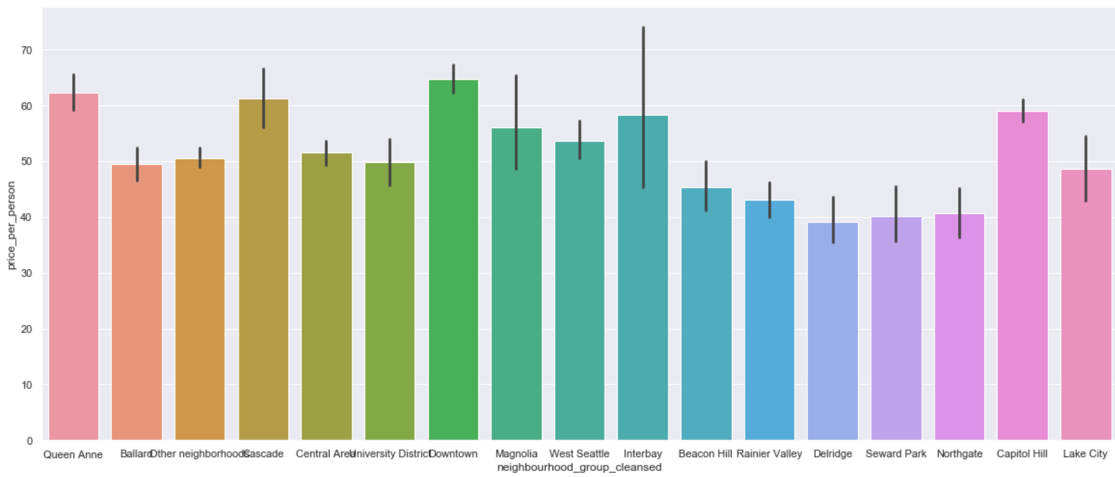
[70]:
```
```

```
hist = df.
↪hist(column=["reviews_per_month","number_of_reviews","price_per_person"],␣
↪figsize=(10,10), bins=35)
```



[71]:
```
import seaborn as sns
%matplotlib inline
sns.set(rc={'figure.figsize':(20,8.27)})
sns.barplot(x="neighbourhood_group_cleansed",y="price_per_person",data=df)
```

[71]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2dc3de48>
```



[72]:
```python
#Correlation with output variable
cor_Y = cor["price_per_person"]
#Showing all correlated features
cor_Y.sort_values(ascending=False)
```

[72]:
```
price_per_person                      1.000000
price                                 0.454125
cleaning_fee                          0.350421
security_deposit                      0.203517
Elevator in Building                  0.198643
Doorman                               0.121563
review_scores_location                0.121196
Gym                                   0.120789
Cable TV                              0.113371
calculated_host_listings_count        0.083392
Pool                                  0.083153
require_guest_profile_picture         0.077501
Air Conditioning                      0.076503
TV                                    0.075863
Kitchen                               0.071869
require_guest_phone_verification      0.069068
bathrooms                             0.067893
Hot Tub                               0.060124
review_scores_rating                  0.057456
review_scores_cleanliness             0.047834
extra_people                          0.043730
host_is_superhost                     0.033979
Washer                                0.033724
is_location_exact                     0.030891
Wheelchair Accessible                 0.030126
Suitable for Events                   0.027487
Dryer                                 0.027284
```

```
review_scores_accuracy              0.025629
host_identity_verified              0.018371
Indoor Fireplace                    0.016748
                                      ...
bedrooms                           -0.004667
Carbon Monoxide Detector           -0.006559
maximum_nights                     -0.008081
host_has_profile_pic               -0.016607
guests_included                    -0.020732
host_response_rate                 -0.022906
Fire Extinguisher                  -0.023408
Heating                            -0.023520
review_scores_checkin              -0.029164
Iron                               -0.030975
review_scores_communication        -0.031730
First Aid Kit                      -0.032686
review_scores_value                -0.034253
Smoking Allowed                    -0.036158
Laptop Friendly Workspace          -0.039142
Breakfast                          -0.040326
Lock on Bedroom Door               -0.041744
Essentials                         -0.042807
Hair Dryer                         -0.043097
Pets Allowed                       -0.044563
Hangers                            -0.046562
Smoke Detector                     -0.049897
Shampoo                            -0.058516
instant_bookable                   -0.063909
Pets live on this property         -0.078505
beds                               -0.087425
Free Parking on Premises           -0.098190
number_of_reviews                  -0.121019
accommodates                       -0.159946
reviews_per_month                  -0.219961
Name: price_per_person, Length: 65, dtype: float64
```

3 3. Prepare Data

[73]:
```
def create_cat_encodings(dataframe, categorical_list, dummy_na):
    '''
    Convert categorical variables into dummy variables.

    INPUT:
    dataframe - input dataframe
    categorical_list - list of categorical variables
    dummy_na - add a column to indicate NaNs, if False NaNs are ignored.
```

```
    OUTPUT:
    dataframe - a new dataframe, that contains all previous columns except the␣
↪categoricals,
    '''
    dummies =pd.
↪get_dummies(dataframe[categorical_list],drop_first=True,dummy_na=dummy_na)
    dataframe = pd.concat([dataframe.drop(categorical_list,axis=1), dummies],␣
↪axis=1, join='inner')

    return dataframe
```

[74]:
```
df_categorical = list(df.select_dtypes(include=['object']).columns)
df_extended = create_cat_encodings (df,df_categorical, dummy_na=False )
```

[75]:
```
df_extended.head()
```

[75]:
```
   host_response_rate  host_acceptance_rate  host_is_superhost  \
0            0.959961                   1.0                  0
1            0.979980                   1.0                  1
2            0.669922                   1.0                  0
3                 NaN                   NaN                  0
4            1.000000                   NaN                  0

   host_has_profile_pic  host_identity_verified  is_location_exact  \
0                     1                       1                  1
1                     1                       1                  1
2                     1                       1                  1
3                     1                       1                  1
4                     1                       1                  1

   accommodates  bathrooms  bedrooms  beds  ...  property_type_Treehouse  \
0             4        1.0       1.0   1.0  ...                        0
1             4        1.0       1.0   1.0  ...                        0
2            11        4.5       5.0   7.0  ...                        0
3             3        1.0       0.0   2.0  ...                        0
4             6        2.0       3.0   3.0  ...                        0

   property_type_Yurt  room_type_Private room  room_type_Shared room  \
0                   0                       0                      0
1                   0                       0                      0
2                   0                       0                      0
3                   0                       0                      0
4                   0                       0                      0

   bed_type_Couch  bed_type_Futon  bed_type_Pull-out Sofa  bed_type_Real Bed  \
0               0               0                       0                  1
1               0               0                       0                  1
```

```
2               0               0                       0                  1
3               0               0                       0                  1
4               0               0                       0                  1

   cancellation_policy_moderate  cancellation_policy_strict
0                             1                           0
1                             0                           1
2                             0                           1
3                             0                           0
4                             0                           1

[5 rows x 104 columns]
```

3.0.1 Select highly correlated features with price per person

[76]:
```
cor = df_extended.corr()
cor_Y = cor["price_per_person"]
cor_Y.sort_values(ascending=False)
```

[76]:
```
price_per_person                                1.000000
price                                           0.454125
cleaning_fee                                    0.350421
security_deposit                                0.203517
Elevator in Building                            0.198643
neighbourhood_group_cleansed_Downtown           0.162595
cancellation_policy_strict                      0.130720
Doorman                                         0.121563
review_scores_location                          0.121196
Gym                                             0.120789
Cable TV                                        0.113371
neighbourhood_group_cleansed_Queen Anne         0.090053
calculated_host_listings_count                  0.083392
Pool                                            0.083153
property_type_Boat                              0.079727
neighbourhood_group_cleansed_Capitol Hill       0.078682
require_guest_profile_picture                   0.077501
Air Conditioning                                0.076503
TV                                              0.075863
Kitchen                                         0.071869
require_guest_phone_verification                0.069068
bathrooms                                       0.067893
bed_type_Real Bed                               0.063524
Hot Tub                                         0.060124
review_scores_rating                            0.057456
property_type_Condominium                       0.051757
review_scores_cleanliness                       0.047834
extra_people                                    0.043730
```

```
neighbourhood_group_cleansed_Cascade                   0.042380
property_type_Bed & Breakfast                          0.039962
                                                         ...
Smoking Allowed                                       -0.036158
bed_type_Pull-out Sofa                                -0.038353
Laptop Friendly Workspace                             -0.039142
Breakfast                                             -0.040326
room_type_Shared room                                 -0.040489
cancellation_policy_moderate                          -0.041405
Lock on Bedroom Door                                  -0.041744
bed_type_Futon                                        -0.042067
Essentials                                            -0.042807
property_type_Dorm                                    -0.042850
Hair Dryer                                            -0.043097
Pets Allowed                                          -0.044563
Hangers                                               -0.046562
Smoke Detector                                        -0.049897
neighbourhood_group_cleansed_Seward Park              -0.057727
Shampoo                                               -0.058516
neighbourhood_group_cleansed_Beacon Hill              -0.059966
instant_bookable                                      -0.063909
neighbourhood_group_cleansed_Other neighborhoods      -0.070834
neighbourhood_group_cleansed_Northgate                -0.075342
Pets live on this property                            -0.078505
neighbourhood_group_cleansed_Delridge                 -0.083484
beds                                                  -0.087425
neighbourhood_group_cleansed_Rainier Valley           -0.088460
Free Parking on Premises                              -0.098190
number_of_reviews                                     -0.121019
property_type_House                                   -0.126896
accommodates                                          -0.159946
room_type_Private room                                -0.168101
reviews_per_month                                     -0.219961
Name: price_per_person, Length: 104, dtype: float64
```

4 4. Data Modeling

4.0.1 Fill all missing values with 0

[77]:
```
df_extended = df_extended.fillna(0)
df_extended.isnull().count().mean() == df_extended.shape[0]
df_extended.head()
```

[77]:
```
   host_response_rate  host_acceptance_rate  host_is_superhost  \
0            0.959961                   1.0                  0
1            0.979980                   1.0                  1
2            0.669922                   1.0                  0
```

```
3            0.000000                    0.0                0
4            1.000000                    0.0                0

   host_has_profile_pic  host_identity_verified  is_location_exact  \
0                     1                       1                  1
1                     1                       1                  1
2                     1                       1                  1
3                     1                       1                  1
4                     1                       1                  1

   accommodates  bathrooms  bedrooms  beds  ...  property_type_Treehouse  \
0             4        1.0       1.0   1.0  ...                        0
1             4        1.0       1.0   1.0  ...                        0
2            11        4.5       5.0   7.0  ...                        0
3             3        1.0       0.0   2.0  ...                        0
4             6        2.0       3.0   3.0  ...                        0

   property_type_Yurt  room_type_Private room  room_type_Shared room  \
0                   0                       0                      0
1                   0                       0                      0
2                   0                       0                      0
3                   0                       0                      0
4                   0                       0                      0

   bed_type_Couch  bed_type_Futon  bed_type_Pull-out Sofa  bed_type_Real Bed  \
0               0               0                       0                  1
1               0               0                       0                  1
2               0               0                       0                  1
3               0               0                       0                  1
4               0               0                       0                  1

   cancellation_policy_moderate  cancellation_policy_strict
0                             1                           0
1                             0                           1
2                             0                           1
3                             0                           0
4                             0                           1

[5 rows x 104 columns]
```

4.0.2 Create X and y data sets for the model. After that split the data into training and testing data set.

[78]:
```
#create X and y data set
df_categorical = list(df_extended.select_dtypes(include=['object']).columns)
df_new = df_extended.drop(columns=df_categorical)
df_new = df_extended
```

```
# df_new = df_new.fillna(0)
y  = df_new["price_per_person"]
X  = df_new.
 ↪drop(columns=["price_per_person","price","accommodates","cleaning_fee"],␣
 ↪axis=0)
```

[79]:
```
#split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.30,
                                                    random_state=42)
```

[86]:
```
#fit and transform training data
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train_imp=scaler.fit_transform(X_train)

#transform test data
X_test_imp=scaler.transform(X_test)
```

4.0.3 Instantiate and fit the LR Model

[87]:
```
lm_model = LinearRegression(normalize=True) # Instantiate
lm_model.fit(X_train_imp, y_train) #Fit

#Predict using your model
y_test_preds = lm_model.predict(X_test_imp)
y_train_preds = lm_model.predict(X_train_imp)

#Score using your model
test_score = r2_score(y_test, y_test_preds)
train_score = r2_score(y_train, y_train_preds)
```

5 5. Evaluate the Results

[88]:
```
#Print training and testing score
print("The rsquared on the training data was {}.  The rsquared on the test data␣
 ↪was {}.".format(train_score, test_score))
```

```
The rsquared on the training data was 0.3038236069286634.  The rsquared on the
test data was 0.2605096588046998.
```

[89]:
```
#extract coefficients
feature_importances = pd.DataFrame(lm_model.coef_,
                                   index = X_train.columns,
                                    columns=["coefficient"]).
 ↪sort_values("coefficient", ascending=False)
```

[90]:
```
feature_importances
```

[90]:

	coefficient
review_scores_location	7.544330
review_scores_rating	6.475103
security_deposit	3.576273
Elevator in Building	3.324963
calculated_host_listings_count	3.315824
host_is_superhost	2.994642
neighbourhood_group_cleansed_Downtown	2.496304
review_scores_cleanliness	2.323117
review_scores_accuracy	2.194100
bathrooms	2.192843
cancellation_policy_strict	2.147447
property_type_Boat	2.080549
bedrooms	1.770069
Dryer	1.764811
property_type_House	1.750359
Cable TV	1.734323
neighbourhood_group_cleansed_Queen Anne	1.684141
is_location_exact	1.587605
neighbourhood_group_cleansed_Capitol Hill	1.557012
cancellation_policy_moderate	1.442020
property_type_Loft	1.435507
First Aid Kit	1.418802
extra_people	1.270648
Indoor Fireplace	1.197013
property_type_Townhouse	1.148427
property_type_Bed & Breakfast	1.127077
Hot Tub	0.934668
Wireless Internet	0.881583
Suitable for Events	0.763465
property_type_Bungalow	0.670478
...	...
Gym	-0.644510
host_response_rate	-0.713641
Wheelchair Accessible	-0.731218
Pool	-0.748613
neighbourhood_group_cleansed_Other neighborhoods	-0.823905
property_type_Tent	-0.879996
maximum_nights	-0.917630
Heating	-0.924868
Smoke Detector	-0.992063
Pets Allowed	-0.996075
neighbourhood_group_cleansed_Rainier Valley	-1.056265
bed_type_Pull-out Sofa	-1.182675
neighbourhood_group_cleansed_Seward Park	-1.330456

```
TV                                                    -1.383315
bed_type_Futon                                        -1.446032
neighbourhood_group_cleansed_University District      -1.518379
Hangers                                               -1.558251
neighbourhood_group_cleansed_Northgate                -1.559431
neighbourhood_group_cleansed_Delridge                 -1.573123
bed_type_Real Bed                                     -1.591788
Washer                                                -1.648584
guests_included                                       -1.731221
room_type_Shared room                                 -3.533984
host_acceptance_rate                                  -3.802856
reviews_per_month                                     -4.261347
review_scores_value                                   -5.165779
room_type_Private room                                -5.511402
review_scores_communication                           -6.060912
review_scores_checkin                                 -6.756385
beds                                                  -8.177140

[100 rows x 1 columns]
```

[ ]: