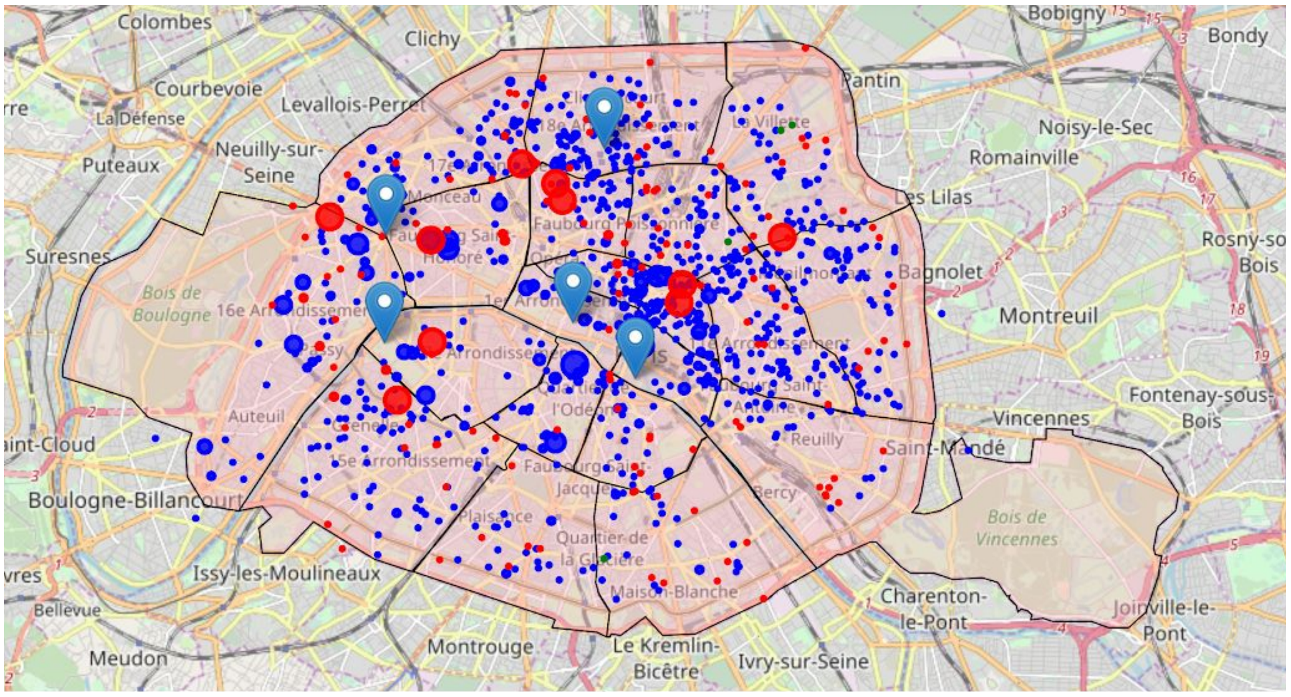
AirBnB Listings in Paris - Identifying and Predicting a Good Deal



Paris is a city of love, or at least it is one like this for me as my own love story started there... <3

Thousands of people from all over the world are traveling to France's capital each year. Some come experience outstanding french cuisine, some are diving deep into history, art and architecture, others just arrive "to see Paris and die". Whatever the reason is behind the visit, all these people need places to stay. Among traditional hotels, family and friends stays there is a new popular way of finding accommodation: to rent a place through AirBnB. Therefore I took a deeper look at the data related to AirBnB listings in Paris.

I. Problem Statement

The idea behind AirBnB is that people rent out their actual residences. This means that each place offers a unique experience which makes it a pretty difficult choice for a traveler to pick one. Traditional hotels have star ratings that give a good idea of what the stay will look like. It would be beneficial to give a similar type of rating to the listings on AirBnB. It may, however, not be enough to make an informed decision as rating similar to the hotel star rating will only give an idea of comfort level. The overall experience is more important in AirBnB case and definitely everyone is looking to get a good deal. I decided to explore listings in Paris to classify them accordingly and try to predict if a certain listing will represent a “good deal”. The results can be useful to all involved parties:

- Travellers - it is great to know you are getting a good deal without reading tons of reviews and comparing uncomparable places
- Hosts - diligent ones will be rewarded for offering great places, being nice and honest hosts

- AirBnB itself - can a new feature to the application and the website.

II. Obtaining Data

For this project I have downloaded four data sets from open data source on http://insideairbnb.com/get-the-data.html These data sets are:

- Listings - over 60K rows - information about each individual listing at the download time
- Calendar - over 22MM rows - information about listing availability throughout the year and corresponding prices at each date
- Reviews - over 1.1MM rows - information with transcript of all reviews to date
- Neighbourhoods - geojson file containing geographical lines of Paris neighbourhoods

Inside AirBnB website scraps data for AirBnB listings periodically and offers the newest sets on their site. The data I used was scrapped on 2019-06-05 and has listing information on that date, availability calendar for the year after that, and information on reviews posted before that.
All files were in .gz compressed format because of the volume. The datasets are too large to upload to github but all the calculations can be transferred to similar data currently available at above website.

III. Data Cleaning and Wrangling

There were several steps performed for each dataset. No cleaning was needed for neighbourhoods data.

- Steps for Listings dataset
    - *Loading data* - read data from the file. There were originally 106 columns in data set.
    - *Check data content* - I performed pandas_profiling on the data to identify columns that cannot be used in further analysis. Half of the columns were dropped after this step. Some of the columns did not contain relevant for price prediction information, like 'host_id', 'scrapped_date' etc. Others contained the same information for each listing, like 'host_acceptance_rate', 'has_availability' etc, or were highly correlated with remaining in data set columns, like 'availability_60', 'minimum_minimum_nights' etc.
    - *Data transformation* -
        a. Explore 'country' column entry for Switzerland - confirming that it is in fact Paris listing, dropping 'country column'
        b. Transform 'neighbourhood' and 'host_neighbourhood' columns into a new one 'is_host_near' indicating 't' if entries in both columns match. Both original columns were dropped afterwards (there is also 'neighbourhood_cleansed' column that contains generalized entries for neighbourhoods)
        c. Transform 'host_verifications' and 'amenities' columns from list like strings to actual lists. 'host_verifications' was further transformed to the numbers of verification counts. I kept 'amenities' column as list of amenities for each listing

and created a new column ‘amenities_count’ with the same transformation as for host verifications

  d. Transform price containing columns (‘price’, ‘security_deposit’, ‘cleaning_fee’) from strings to floats

- *Filling Missing Values* - divided columns with missing values into several categories(based on the data to be filled in each of them) and with the help of defined function filled NaNs as below
  e. Empty string ‘’ for columns 'name', 'summary', 'space', 'description', 'transit', 'house_rules'
  f. ‘f’ indicating False for columns 'host_is_superhost', 'host_identity_verified'
  g. ‘unknown’ for 'host_response_time'
  h. Mean or median for 'host_response_rate', 'host_since', 'review_scores_rating', 'review_scores_accuracy', 'review_scores_cleanliness', 'review_scores_checkin', 'review_scores_communication', 'review_scores_location', 'review_scores_value'
  i. Maximum of ('accommodates'/2) and 1 for ‘bedrooms’
  j. Maximum of ('bedrooms' - 1) and 1 for ‘bathrooms’
  k. Maximum of 'bedrooms' and 1 for ‘beds’
  l. Mode for 'cancellation_policy' and 'zipcode'
- *Exploring outliers* - the most important exploration for outliers was done for ‘price’ column. The original data was extremely skewed and therefore I transformed it to log prices and distribution curve looked closer to normal distribution one. There are three types of listings offered: ‘entire home/apt’, ‘private room’ and ‘shared room’. It makes sense to explore corresponding prices separately. I defined a function that identifies outliers based on their distance from the mean (3*standard deviation(std) for left outliers and 4*std for right ones as data is still skewed) and removes outliers from the dataset if they make less than 1% of the data. After this function was applied on prices for each room type category, outliers in ‘entire home/apt’ and ‘private room’ (about 0.4% of the total data combined) were removed. ‘Shared room’ outliers were slightly over 1%(just 5 actual listings) and further exploration allowed to drop them as well.
  A more general function (with lower outliers acceptance of 0.5%) was defined to explore outliers in 'accommodates', 'bedrooms', 'bathrooms', 'beds', 'minimum_nights', 'maximum_nights' columns. As output it would provide information about outliers quantities and range and made suggestions on keeping or removing outliers. 'bedrooms', 'bathrooms', 'minimum_nights', 'maximum_nights' showed to have less than 1% of outliers combined (each not more than 0.5%) and they were dropped. 'accommodates' and ‘beds’ columns had more outliers and further exploration showed them to be valid entries, thus they were kept

- Steps for Calendar and Reviews data
  - Loading data
  - Dropping missing values as their amounts were insignificant given datasets volumes
  - Data transformation

a. Price columns: from strings to floats
b. Date columns: from strings to datetime objects

- Merging with listing_ids from Listings data set to ensure that dropped values from main data set are not used in these ones.

IV. Exploratory and Statistical Data Analysis

A. Occupancy Rate Estimation for All Listings

When thinking about listings the first question that comes to mind is "What takes the place to get booked?". It would have been natural to look at historical renting data and compare it with characteristics of the listings. Unfortunately AirBnB does not reveal past booking information. Instead the webpage I downloaded the data from http://insideairbnb.com/get-the-data.html offers a disclaimer that allows to estimate past occupancy rate and earned income for each listing based on number of reviews left.

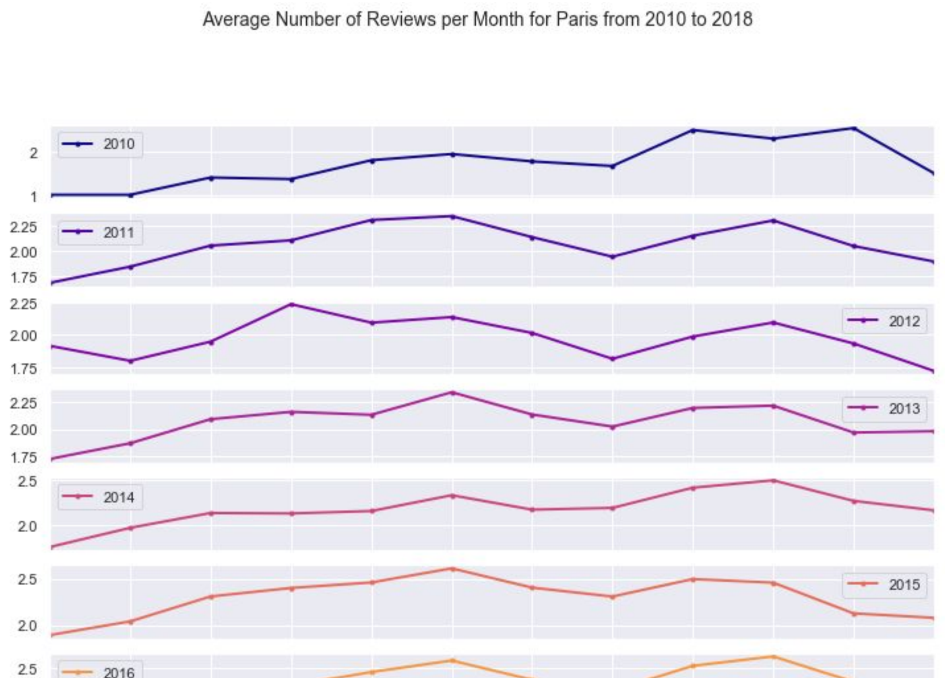
So called "San Francisco model" is used for this purpose. It uses review rate of 50% (meaning each second guest leaves a review) to convert reviews to estimated bookings and average length of stay calculated by AirBnB for each city. In case of Paris it equals to 5.2 nights.

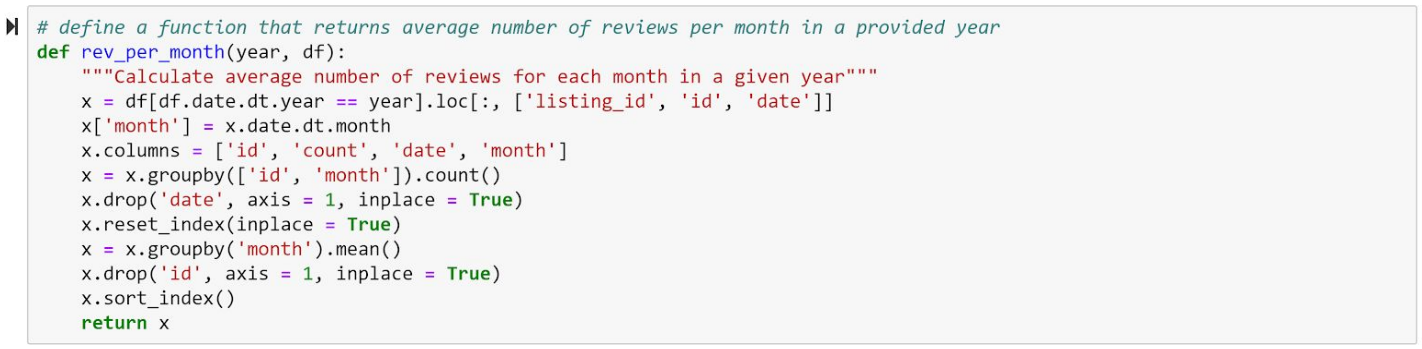
Occupancy rate and income are then calculated by below formulas:

Occupancy rate = Average Length of Stay * Reviews per Month / Review Rate

Income = Occupancy Rate * Price * 12

Since the only variable in calculating occupancy rate is an average number of reviews per month, I have explored it first. The review data starts from 2010. With below defined function I have calculated average number of reviews for each month in each year from 2010 to 2018 (last full calendar year).





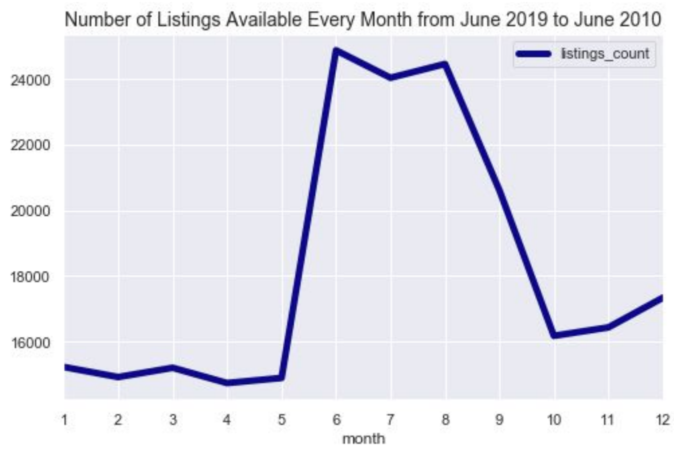
```
# define a function that returns average number of reviews per month in a provided year
def rev_per_month(year, df):
    """Calculate average number of reviews for each month in a given year"""
    x = df[df.date.dt.year == year].loc[:, ['listing_id', 'id', 'date']]
    x['month'] = x.date.dt.month
    x.columns = ['id', 'count', 'date', 'month']
    x = x.groupby(['id', 'month']).count()
    x.drop('date', axis = 1, inplace = True)
    x.reset_index(inplace = True)
    x = x.groupby('month').mean()
    x.drop('id', axis = 1, inplace = True)
    x.sort_index()
    return x
```

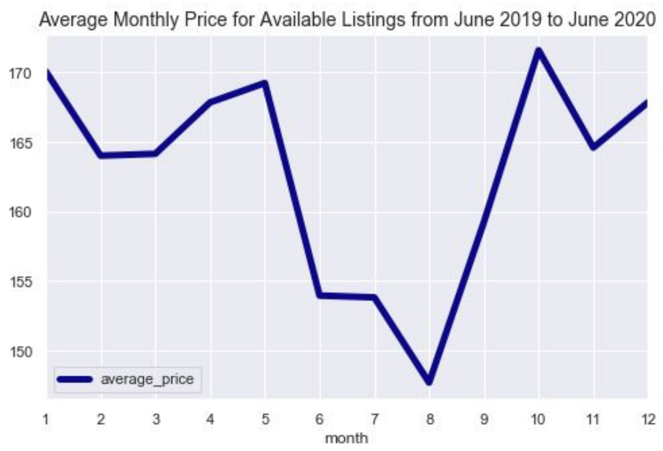
The results were plotted on below graph.
Almost all years (the earlier data for 2010 and 2011 is slightly different to the possible facts of fewer listings available in general and less people reviewing anything back then) show similar pattern:

- demand is the lowest in the winter
- people seem to travel to Paris mostly during spring and fall
- demand drops throughout the summer reaching minimum in August as tourists in general prefer being on the beach than sightseeing during hot months

Listings on AirBnB are primarily offered by private hosts that are renting their own places when they travel themselves. It is interesting to see if the supply corresponds to the demand and when most listings are being offered. For this purpose I used data from Calendar dataset. I built two lists:

I. A list containing number of listings offered each month
II. A list with average price for each month.





Interestingly supply is not fully correlated with demand with most listings offered during summer months. It supports the fact that hosts usually list their primary residences and do so when they travel themselves. Summer is the most popular time for travels therefore the supply is high. Winter and early spring offers the least amount of available places.

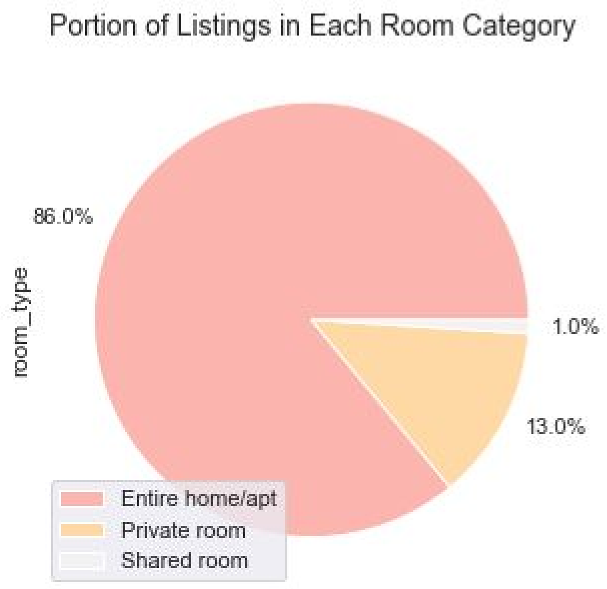
Also there is a slight increase in supply during winter holidays that can also be explained by host travelling to see the family during those times.

Average price is definitely correlated with the demand with high picks in late spring and early fall and big dive in August. Lower price during summer can also be explained by more listings offered during that time.

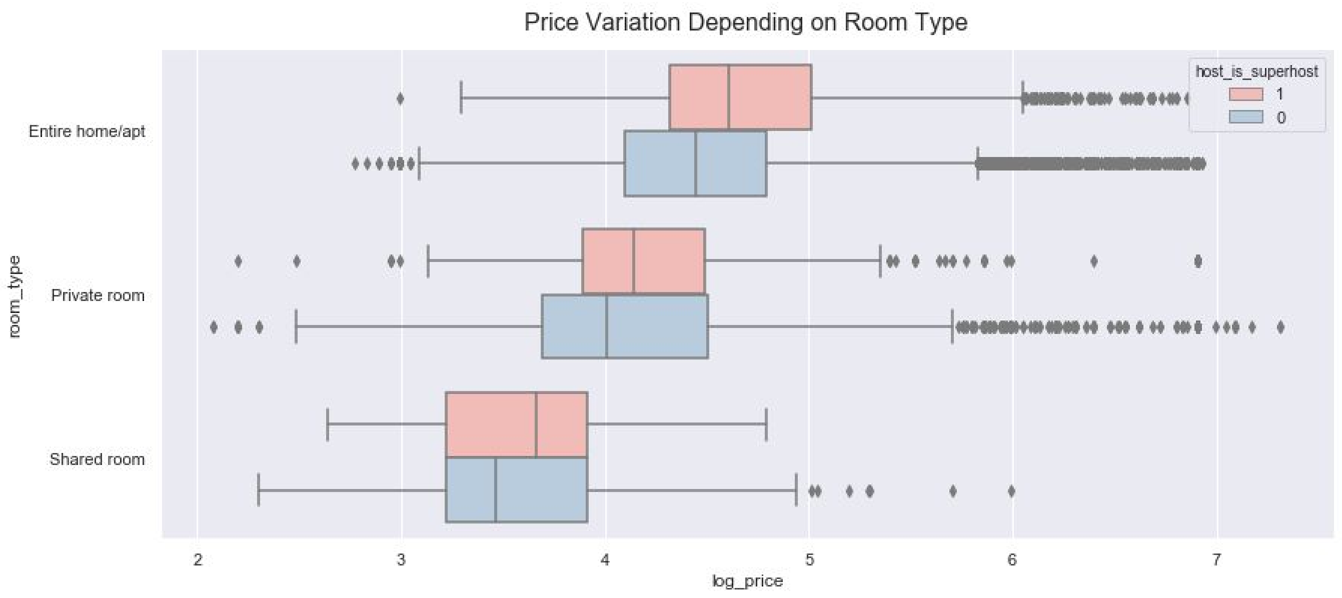
B. Listing Distribution and Statistical Testing

The above analysis was done for all of the listings. However, there are three types of listings. Depending on the room type they fall into: Entire home, Private room and Shared room. The first category represent the places that are being rented entirely without the host on premises so that the tourists have more privacy. Private and shared room mean sharing either common spaces in the house or the room itself.

These different categories represent very different experiences and intuitively I wanted to explore them separately. A slightly deeper look at the composition of listings and their prices. Showed that most of the listings (86%) are entire home ones.



And log-prices of each category seemed to have different medians as seen in below graph.

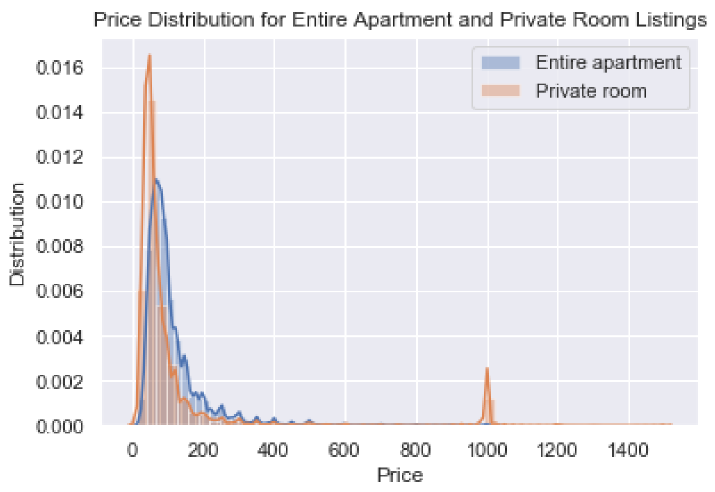


To support my decision to only explore Entire home listings I performed statistical testing to see if there is indeed price difference between different listing categories. Below hytothesis testing was done:

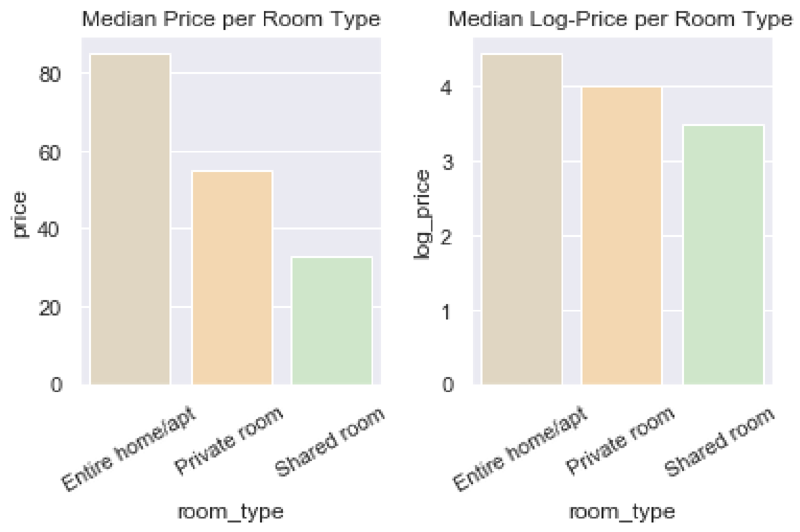
H0: Average price of entire home is the same as average price for private and/or shared rooms

H1: Average prices are different

Testing with the help of a built in function ttest_ind() from the package stats showed significant difference between the average prices of Entire homes and Shared rooms, as well as significant difference between the average prices for Shared and Private rooms. However, the result was not significant to reject the null hypothesis for Entire homes and Private rooms. By further looking at price distributions for these two categories(image below) I have noticed I have noticed a portion of outliers around $1000 price for private rooms, although most of them were not priced over $200. Also the distributions looked different.



Given above, I decided to test median prices instead. Graphical analysis proved I was moving in the right direction:



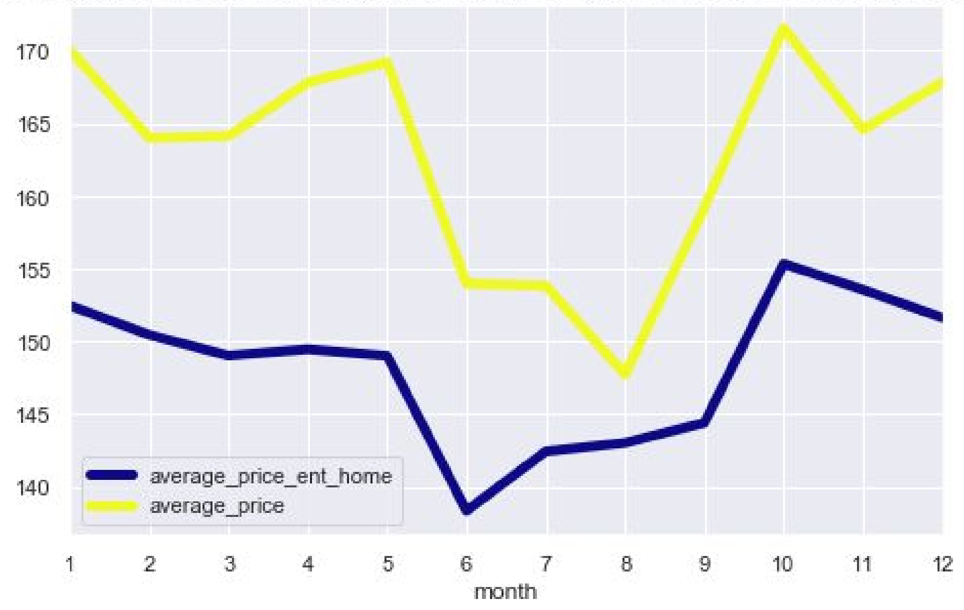
To test median price difference I have performed bootstrap simulation under the null hypothesis that there is no difference between median prices. I have done 10000 simulations. In each step I randomly rearranged the combined (entire homes and private rooms) dataset of prices and divided it into two sets according to the sizes of initial datasets. Then I compared simulated differences to an actually observed result. Computed p-value was zero, meaning that none of 10000 simulated values was at least as high as observed. This means null hypothesis can be rejected and there is significant difference in median prices for two room categories. In further analysis, I only focused on Entire home listings.

C. Occupancy Rate and Income for Entire Home Listings

Similar to all listings, I have calculated average number of reviews for each month for the years 2010-218 for Entire home listings. The results are consistent with the findings for all listings together (after all the vast majority of listings are entire home ones). The interesting fact is that the average monthly price for Entire home listings available in the year after the scrap date is smaller than the average monthly price for all of the listings. It might be explained by the large number of expensive outliers in Private room category that was discussed in a previous section.

Also the general trend is different. The best prices for entire home listings are offered in June vs. August for all listings.
However, the average price is in line with previous demand with higher prices in busier seasons (late spring and fall).

Average Monthly Price for Entire home(purple) and All(yellow) Available Listings from June 2019 to June 2020

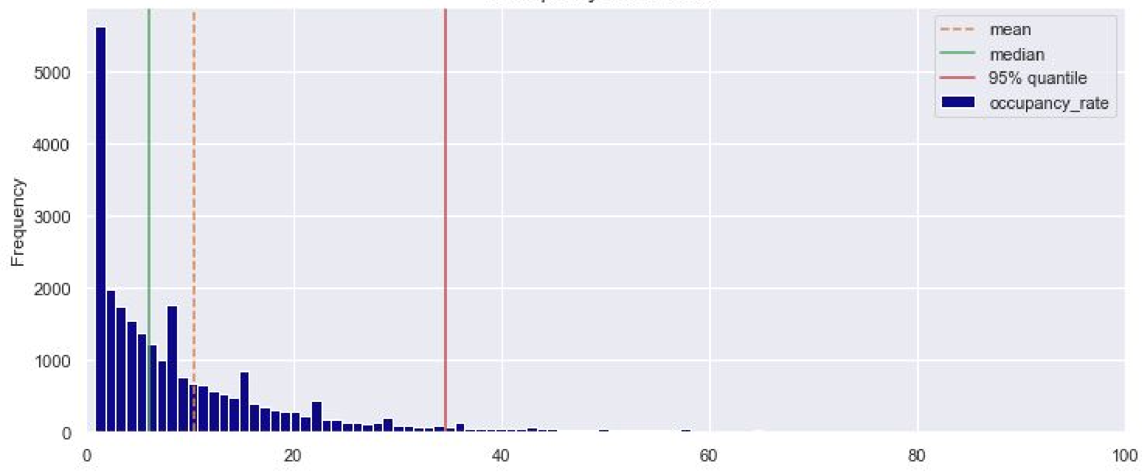


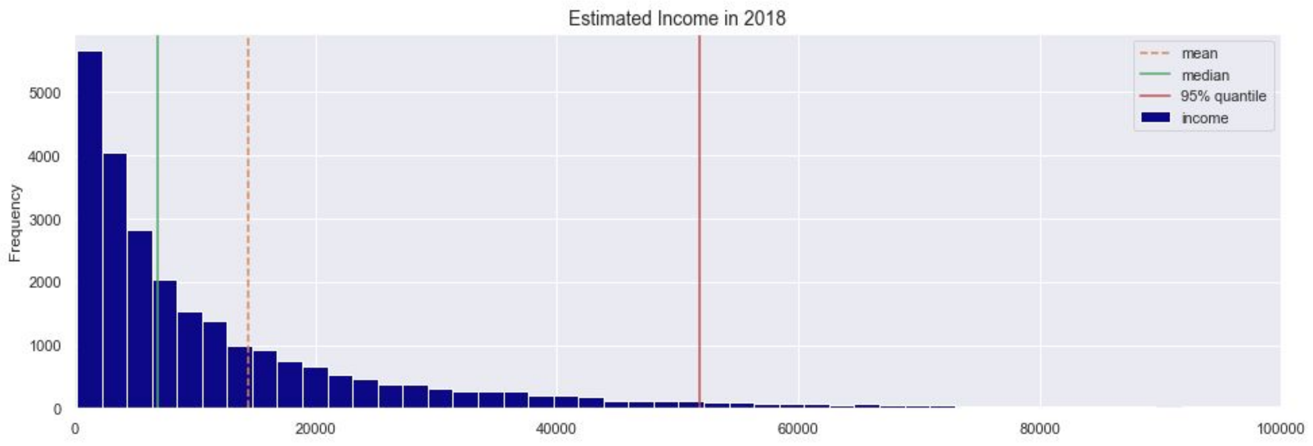
Using previously discussed “San Francisco” model, occupancy rate and income for 2018 were calculated through below code:

```
In [61]:  # calculate occupancy rate and estimated income
           avg_stay_paris = 5.2
           review_rate = 0.5
           rev_home_2018 = rev_home[rev_home.date.dt.year == 2018].loc[:, ['listing_id', 'price', 'id']]
           rev_home_2018 = rev_home_2018.groupby(['listing_id', 'price']).count().loc[:,['id']]/12
           rev_home_2018.reset_index(inplace = True)
           rev_home_2018.rename({'id': 'reviews_per_month'}, axis = 1, inplace = True)

           # add columns for occupancy rate and income
           rev_home_2018['occupancy_rate'] = avg_stay_paris * rev_home_2018['reviews_per_month'] / review_rate
           rev_home_2018['income'] = rev_home_2018.occupancy_rate * rev_home_2018.price *12

           # occupancy rate over 100 is impossible, therefore will drop those values
           rev_home_2018.drop(rev_home_2018[rev_home_2018.occupancy_rate > 100].index, inplace = True)
```

The average occupancy rate for the year 2018 was slightly over 10% for Entire home listings in Paris. And the hosts earned around $14,000 by renting out their places.

Occupancy rate in 2018

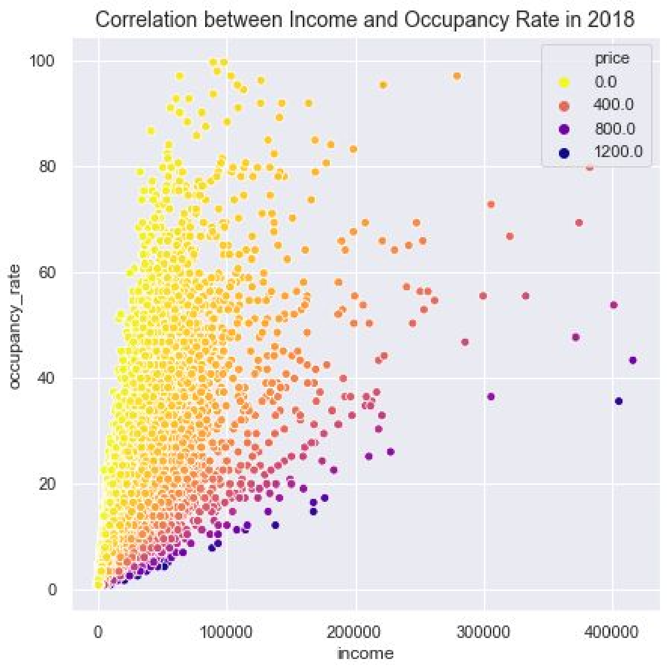




The above findings and graphs support the idea that most apartments listed on AirBnB are private residences rented to tourists when hosts are traveling themselves. Thus half of the listings have occupancy rate of 6% or less, which corresponds to approximately 22 calendar days. Average occupancy rate is higher and is slightly over 10% (36 calendar days). Also 95% of listings don't exceed the rate of 35% (about 128 calendar days ~ third of the year).

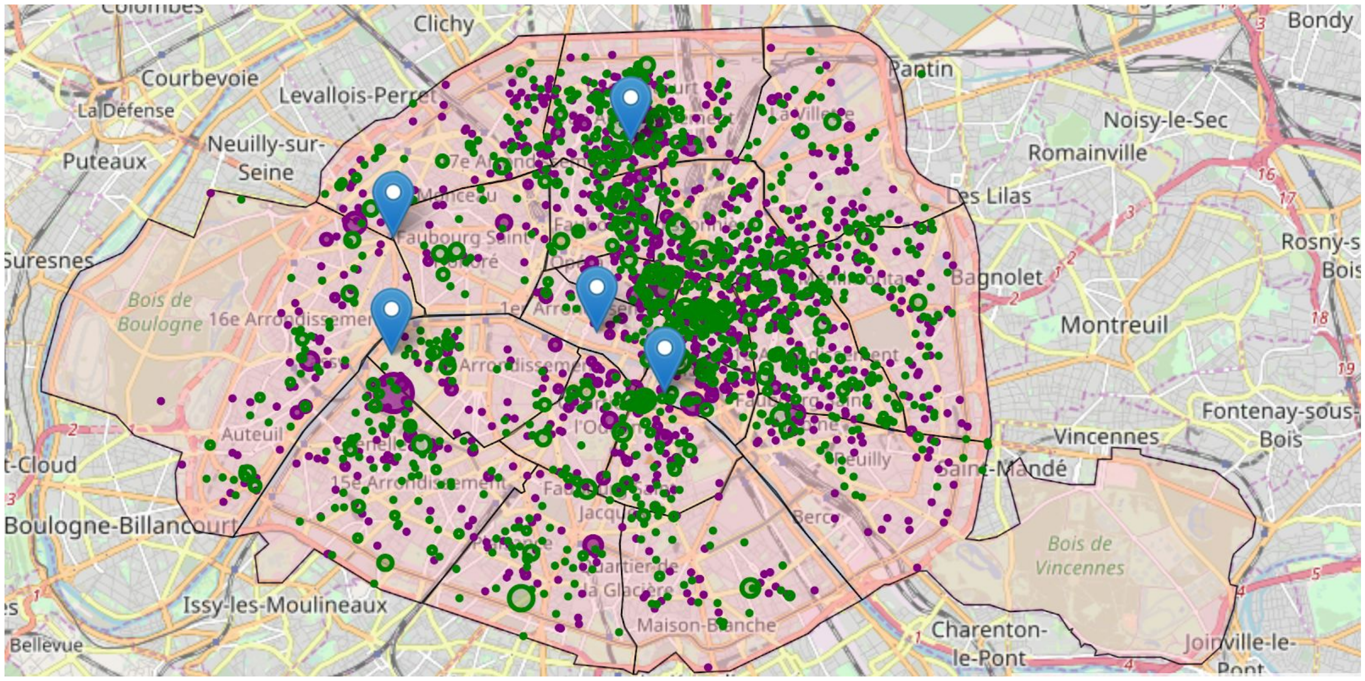
Income distribution is also highly skewed with half of the listing generating under $ 7,000 yearly income in 2018. Average income is about $14,000 and 95% listings did not make over $55,000. The maximum income is over $400,000 though and it will be interesting to take a look at the listings that generated high income.

Income is a function of occupancy rate and price and below graph shows this relationship clearly:



However, the listings that generated the most income were not the ones with the highest occupancy rate. The price level of listings was high for high income places. Also we can see that some listings with really high occupancy rates did generate high income, but for most of them prices were on the lower side.

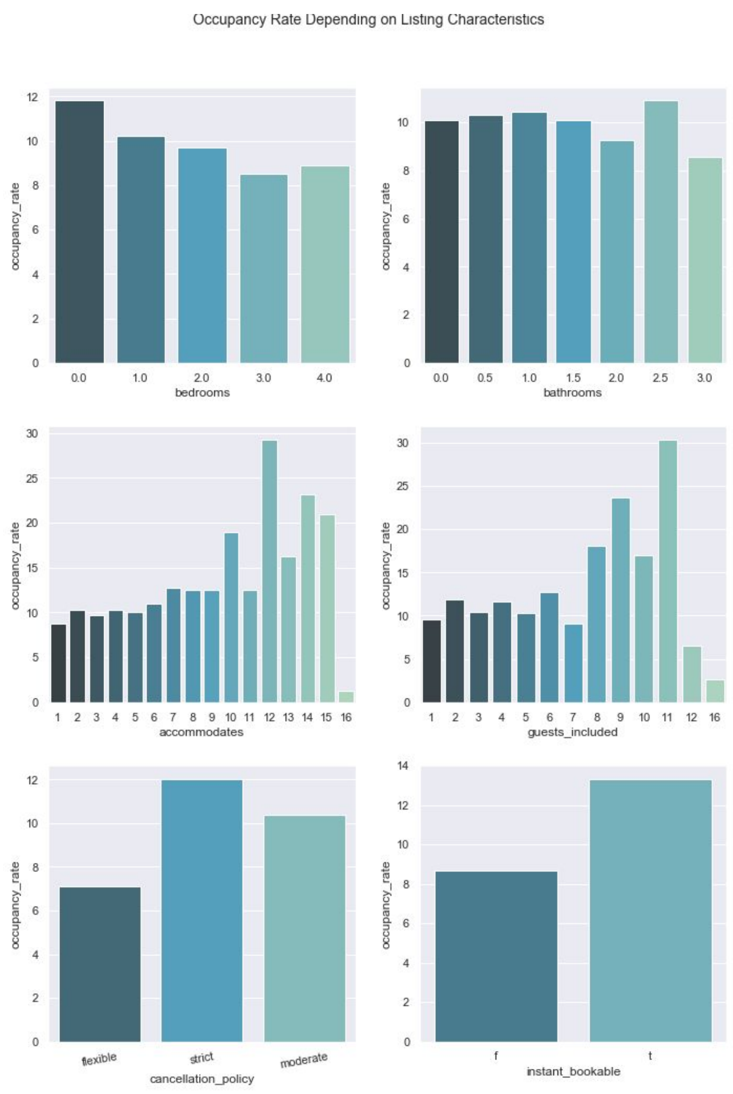
It is also beneficial to see geographical distribution of listings. With the help of the package folium, I have plotted listings on the map. Green represent occupancy rates and purple - income. Size of the bubbles corresponds to the value.



The map above shows a sample distribution of 2000 listings across neighbourhoods. We can see that more central places that are located near the main attractions. Especially Notre Dame and Sacre Coeur. While most higher income generating places follow this finding as prices in central parts tend to be higher, some of the most rented places are located further out and might represent a good deal that renters on AirBnB are looking for. It is definitely worth exploring what can define a good deal and therefore get apartment rented more often.

Before that let's explore what factors are influencing income. As comes from the formula we used to calculate income, it is proportional to price and occupancy rate. Price forming depends on location, size and other characteristics. Obviously, the higher price will mean the higher income. But what if the place does not get rented? Therefore, in my opinion, it is more important to look at the factors that influence occupancy rate.

D. Factors Influencing Occupancy Rate



There are many listing features that can contribute to forming occupancy rate. Roughly they can be divided into three categories:

- apartment features (number of bedrooms, bathrooms, guests the place can accommodate, is it instant bookable etc.)
- host characteristics (is host a superhost, host response rate and time, is he near the listing location etc.)
- what people think about the listing (review ratings based on location, accuracy, communication etc.)

I explored each of these categories and the results were as follows:

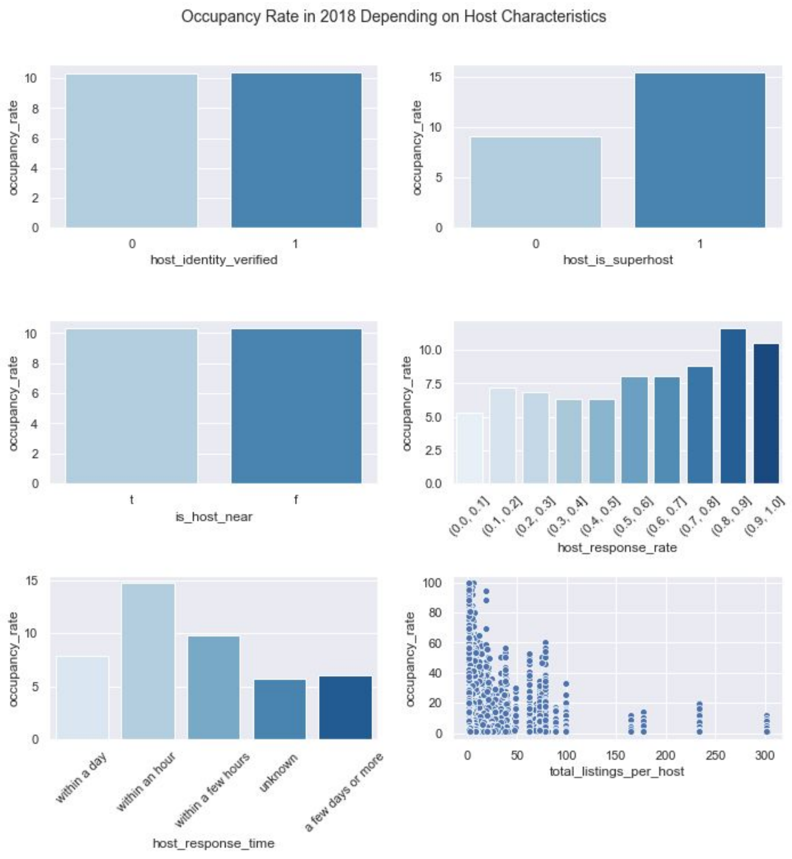
Apartment Characteristics:

There are some interesting findings from these graphs:

- apartments of all sizes (number of bedrooms and bathrooms) are being rented approximately equally (around 10% occupancy rate - which is an average occupancy rate across all listings). This fact might be supported by assumption that people travel in groups of different sizes and everybody is able to find a place to stay corresponding to their needs.
- however, places that can accommodate more guests regardless of their size are getting rented more often.
- it is surprising that the places with stricter cancellation policy are being rented more often. It might be due to the fact that with stricter cancellation policy it is more difficult to make changes to travel plans;
- similar understanding can be behind the finding that places that are instant bookable (no pre-approval from host to rent the place is needed) have higher occupancy rates. Also the fact that people tend to minimize human interactions can contribute to this outcome.

Overall it looks like the features of the listing itself have a big impact on forming the occupancy rate. These features are coming with the apartment as well as its location that also influences if the place gets rented. If the place is not located near the main attractions or is not very large, what can the host do to increase the probability of renting out his or her place?

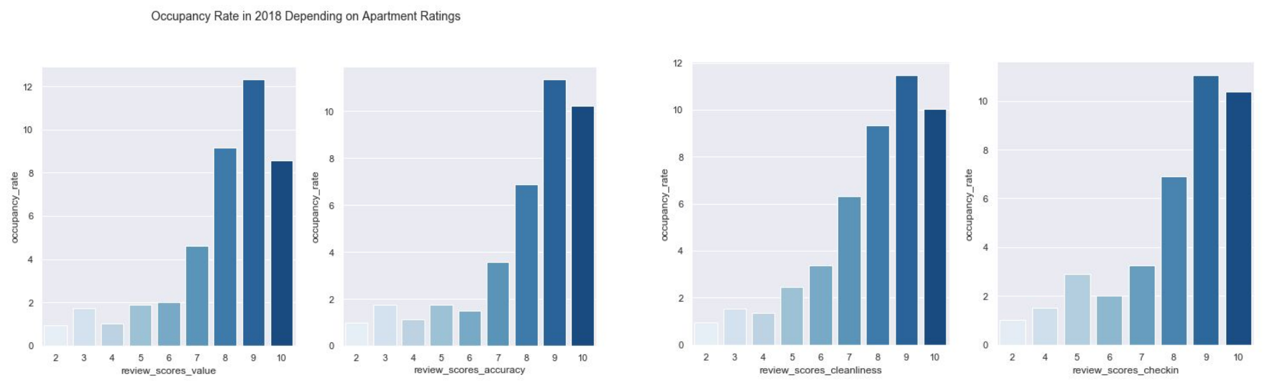
Host Characteristics:



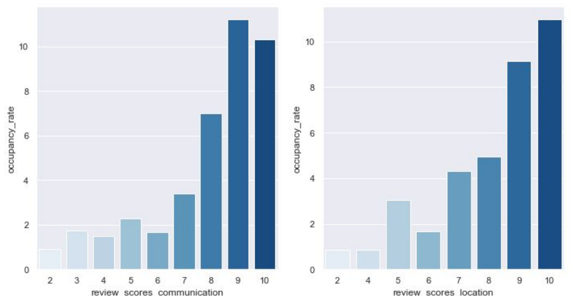
Some core conclusions from these graphs:

- it is beneficial to be a superhost as you place will be rented on average 6% more often
- it is important to answer to as many queries as possible and do it promptly: hosts that answer within an hour have on average 5% higher occupancy rates
- identified host identity and host proximity to the listing don't seem to matter and might need further exploration
- the most occupied places belong to hosts with fewer listings as it might get difficult to provide good services if need to manage over 100 listings. For further exploration it might be interesting to look at 4 hosts that have over 100 listings each.

Review Ratings:

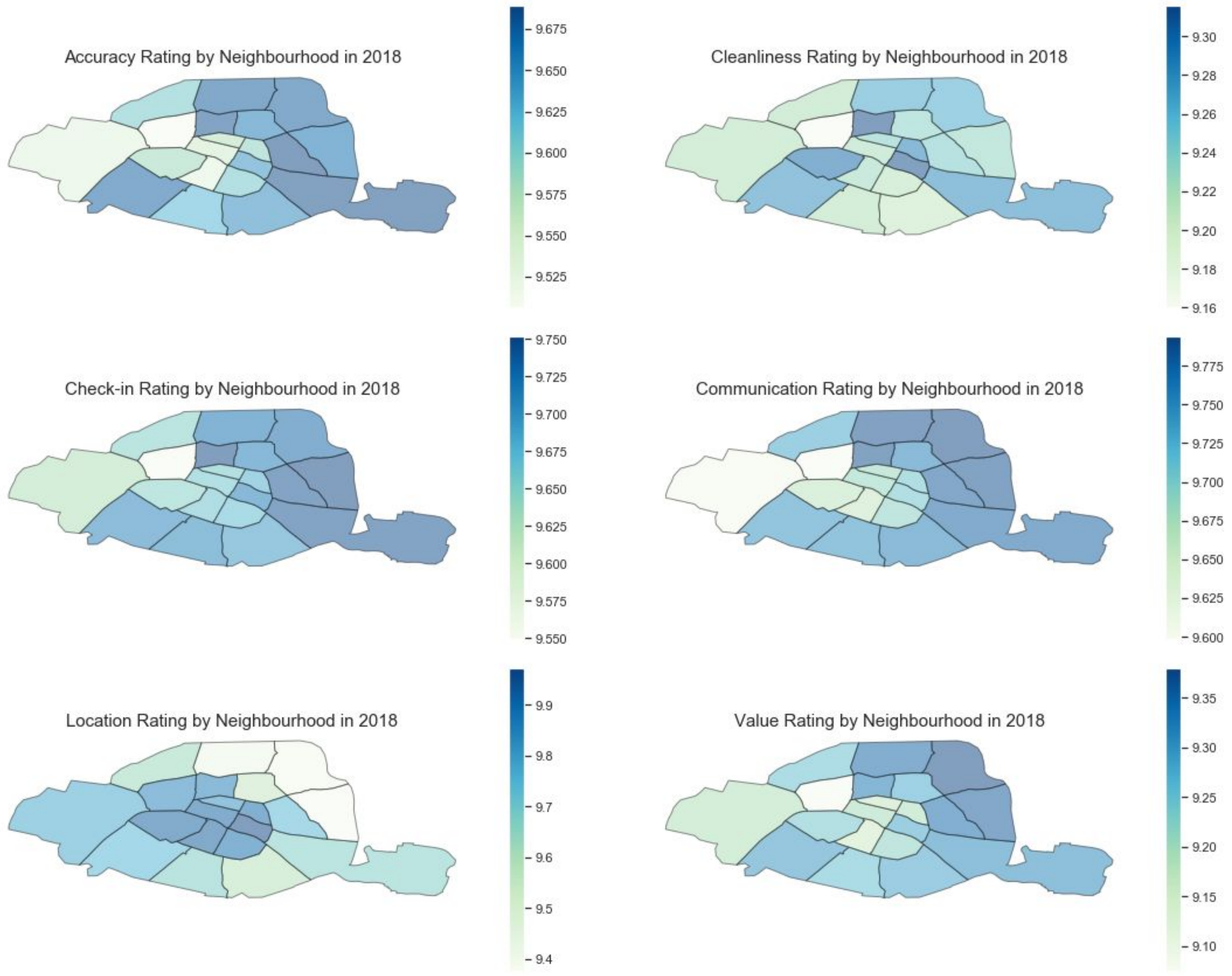


It is very clear from the graphs that reviews are important and places with ratings 9 and higher are being rented more often. Especially it's noticeable in



categories accuracy (how realistic internet representation is), check-in (comfort of initial access) and location.

We have seen previously seen on the map that apartments with higher income have usually more centered location, while high occupancy rate does not necessary mean the place is in the center. As above findings specify, occupancy rate is highly correlated with the ratings people give to different apartment characteristics. High ratings lead to more bookings. This means that previously booked place can be rented again and again if people are satisfied with it.
To show how ratings are distributed across neighbourhoods six small maps with choropleth maps for six ratings in different categories were plotted with geopandas.


Accuracy Rating by Neighbourhood in 2018
9.675
9.650
9.625
9.600
9.575
9.550
9.525
Cleanliness Rating by Neighbourhood in 2018
9.30
9.28
9.26
9.24
9.22
9.20
9.18
9.16
Check-in Rating by Neighbourhood in 2018
9.750
9.725
9.700
9.675
9.650
9.625
9.600
9.575
9.550
Communication Rating by Neighbourhood in 2018
9.775
9.750
9.725
9.700
9.675
9.650
9.625
9.600
Location Rating by Neighbourhood in 2018
9.9
9.8
9.7
9.6
9.5
9.4
Value Rating by Neighbourhood in 2018
9.35
9.30
9.25
9.20
9.15
9.10

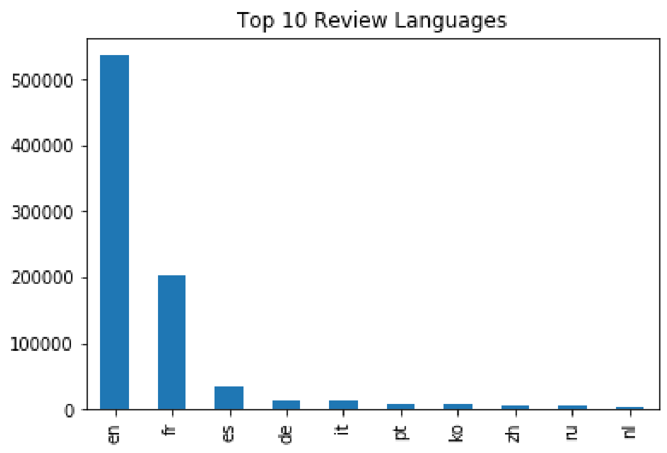
All of the neighbourhoods have ratings above 9. However, the central parts tend to have smaller ratings (except for location category). It is a very interesting finding that opens up a new reason for exploration. Looking deeper into what people think about each place and their experience in it and with the host can help identify more characteristics that help to get the place rented. Typically, travelers renting through

AirBnB are looking for a good deal. What are the components of a good deal and how to know that you are getting it? So far I have found out some factors that affect occupancy rate:
- possibility to accommodate larger groups
- ability to rent the place right away without host's approval
- appropriate amount of amenities
- fast and clear communication with host
- getting a place from superhost
- location of the place
- experiences of previous renters.

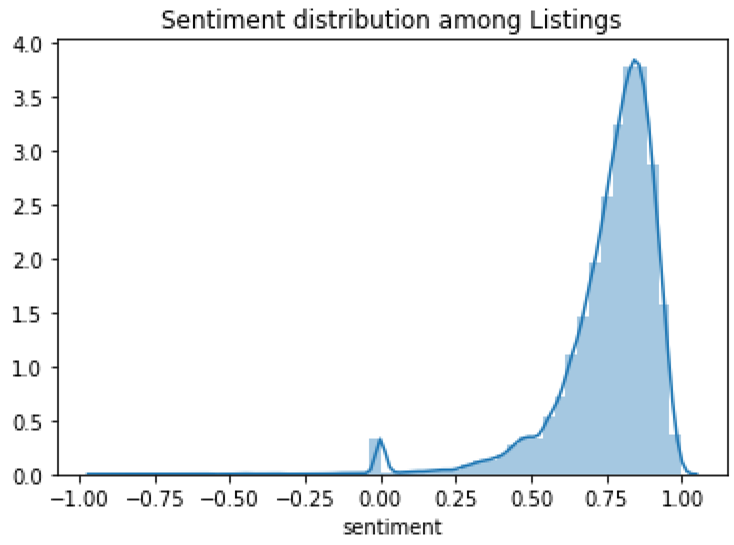
Further I will try to classify listings in terms "does the place represent a good deal or not" and see if I can predict whether a listing is a good one.

E. Sentiment Analysis

As discovered during exploratory data analysis, previous experience of other renters is important in making a decision about booking or not booking certain place. The data I am working with has six rating scores for each individual aspect of apartment and one overall rating score. While these represent a great source of information and definitely help to make a decision, there is another variable that can influence the booking.
Think of how you are looking for a new restaurant to go out, or a new movie to watch. Of course, you are looking at the scores, but also you read reviews left by others to get the overall idea. I did the same for the AirBnB listings in Paris. After selecting reviews for listings in question, I had over 800,000 reviews. Obviously reading each of them would require an enormous amount of time. Instead, I aimed to get so called sentiment analysis of each of them.
Sentiment analysis is a score (between -1 and 1 in the analysis I conducted) that tells an overall vibe of the text. There are multiple instruments that allow to get the sentiment analysis of the text already implemented in Python. I decided to use lexicon and rule-based VaderSentiment as it works best with social media style of text, which reviews for AirBnB can be classified as. This tool gives really great results for English comments and text as it is built over English language lexicon. It does analysis for other languages, but the results are often incorrect.



Out of almost 800,000 reviews over 300,000 were not in English. Language distribution can be seen in the graph above. In order to get more accurate sentiment scores, I first translated all of these listings into English using googletrans package. It was the most time consuming step and it might be beneficial to search for or create alternative sentiment analysis tools that can work directly with other languages.
Translated ratings together with original English ratings were analyzed with Vader and assigned a sentiment score. The results for the reviews about the same listing were then averaged and the score was added as a new variable to the listing database.



As seen from the above graph, most of the listings have positive sentiment analysis. Relatively large amount of 0 sentiment listings represent the ones without reviews.

F. Final Data Preparation for Machine Learning

After calculating sentiment analysis average review scores for each listing, there is just one step left before building models. Final preparation of predictors and target variable was needed. I performed the following steps for that:

- I defined a function that based on selection parameters (75% top of ratings and sentiment and also below neighbourhood average price) separated listings into *'good deal'* (label 1) and *non-'good deal'* (label 0) ones
- Feature selection was performed based on my findings from EDA
- Categorical features were transformed to columns with 0-1 values
- Numerical features were scaled using **StandardScaler** so that they all have a mean of 0 and a variance of 1.

G. Machine Learning

1. 1. Challenges and Decision Metrics

The data was ready for building models at this point. It was split in training and testing sets. Training sets were reserved for final evaluation of the models.
However, my data was highly imbalanced. The ‘good deal’ listings represented below 5% of the data. In the case of imbalanced data standard metric of success, accuracy score, does not deliver needed results. Even with a poor model of identifying all data points as majority class it will be high. In my case, at least 0.95. So different decision metrics were required. I settled on two of them:

a) Precision = positive predictive value = fraction of correctly identified ‘good deal’ listings out of all listings labeled by the model as a ‘good deal’
b) Recall = sensitivity = fraction of correctly identified ‘good deal’ listings out of all actual ‘good deal’ listings.

The main idea behind my model building is not only to identify all of the ‘good deal’ listings correctly, but also not to mistakenly label regular listings as a ‘good deal’. Therefore, I decided to pay more attention to precision score (make sure the model doesn’t label too many irrelevant listings as a ‘good deal’), but to keep a balance with recall as well (identify as many ‘good deal’ listings as possible).

2. Methods to deal with imbalanced data

As mentioned above, the collected data was highly imbalanced. I made a decision to apply several techniques to create balanced data for training models and to compare results. On the base of the simple model using logistic regression I tested the following techniques divided into three groups:

1) **Under-sampling** techniques - the main idea to balance the data by keeping the amount of minority class representatives the same and by sampling majority class to get the number of representatives in each class.
    a) *Random Under-sampling* - sample over majority class randomly
    b) *NearMiss* - a 2-steps algorithm based on nearest-neighbor algorithm. First, for each 'good deal' sample, their M nearest-neighbors will be kept. Then, the non-good deal samples selected are the ones for which the average distance to the N nearest-neighbors is the largest.
    c) *ENN* (Edited Nearest Neighbor) - also applies the nearest-neighbors algorithm to “edit” the dataset by removing samples which do not agree “enough” with their neighborhood. For each sample in the majority class, the nearest-neighbors are computed and if the selection criterion(can be either majority or all) is not fulfilled, the sample is removed.

2) **Over-sampling** techniques - to balance data by resampling minority class
    a) *Random Over-sampling* - randomly sampling over minority class with replacement
    b) *SMOTE* - Synthetic Minority Oversampling Technique - synthesising new minority instances between existing minority instances and generating samples by interpolating new points between marginal outliers and inliers.
    c) *SMOTENC* - a modification of SMOTE where categorical variables are treated differently (the majority rule applies)
    d) *ADASYN* - similar to the regular SMOTE. However, the number of samples generated for each minority class member is proportional to the number of samples which are not from the same class than this member in a given neighborhood.

3) **Combination** techniques - apply over-sampling to even out number of class representatives and then under-sampling to reduce noise. - *SMOTEENN*

The results of testing these eight techniques are in below table:

Type of data balancing	**F1 Score**	**Precision Score**	**Recall Score**
Imbalanced data	0.69	0.77	0.63
Random Under-Sampling	0.53	0.36	0.98
NearMiss	0.63	0.52	0.82
ENN	0.71	0.70	0.72
Random Over-Sampling	0.58	0.41	0.98
SMOTE	0.60	0.44	0.97
SMOTENC	0.60	0.45	0.93
ADASYN	0.58	0.41	0.99
SMOTEENN	0.56	0.39	0.98

Most of the techniques had significant improvement on identifying as many ‘good deal’ listings as possible, as seen in high recall scores. However, the models trained over balanced data by these techniques tend to mistakenly label lots of irrelevant listings as good deals. Precision scores are really low in these cases. For further exploration I have picked three sets of training data showing the best results:

1. Original imbalanced data
2. Under-sampled data with ENN
3. SMOTENC - this technique did not show much improvement on precision score, but I wanted to use at least one over-sampling method.

3. Supervised Learning

As I have previously labeled the data manually, I built several models based on supervised learning algorithms to classify data. Each model was separately trained on three of the above discussed training sets. Model performance was evaluated on the original test data set reserved in the beginning. I have trained models based on the following algorithms:

a) **Logistic Regression** - classification algorithm that is used to model the probability of certain class existing.
b) **K-Nearest Neighbors** - another classification algorithms that defines the class based on the majority class of sample's k (hyperparameter to be tuned) nearest neighbours.
c) **SVM** - Support Vector Machines - finds the hyper-plane in n-dimensional feature space. This hyperplane separates two classes.
d) **Naive Bayes** - a classification algorithm for binary and multiclass classification problems, in which the calculations of the probabilities for each class are simplified to make their calculations tractable. Rather than attempting to calculate the probabilities of each attribute value, they are assumed to be conditionally independent given the class value. Neither of balanced data sets produced better results for this algorithm. I used principal component analysis (PCA) instead to identify that dimension reduction to 2 or 3 dimensions brought better results for this algorithm.
e) **Decision Tree** - a supervised learning algorithm that separates the data at each node based on certain criteria it learns from the training data.
f) **Random Forest** - an ensemble algorithm that uses multiple decision trees and combines the results from them to make a classification decision. Each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias.
g) **AdaBoost** -an ensemble method that fits a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.
   The data modifications at each so-called boosting iteration consist of applying weights to each of the training samples. Initially, those weights are all set to 1/N, so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive

ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence

h) **Gradient Boosting** - another ensemble model that uses weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

For each model I have calculated precision, recall and f1-score and saved them in comparison table. Please take a look at below top 10 models (sorted by scores in the following order: precision, f1, recall).

Classifier Name	**F1 score**	**Precision score**	**Recall**
Random Forest	0.836	0.846	0.827
AdaBoost	0.866	0.834	0.901
Gradient Boosting	0.835	0.831	0.838
SVM	0.717	0.815	0.639
Random Forest ENN	0.850	0.799	0.906
Decision Tree	0.790	0.794	0.787
Decision Tree SMOTENC	0.823	0.791	0.858
AdaBoost ENN	0.849	0.784	0.926
Decision Tree ENN	0.826	0.782	0.875
Gradient Boosting ENN	0.840	0.773	0.920

The three first places were taken by ensemble methods (Random Forest, AdaBoost and Gradient Boosting) trained on imbalanced data. Same algorithms trained on Edited Nearest Neighbours balanced data showed slightly lower precision scores but higher recall scores (identified correctly larger number of actual 'good deal' listings than the ones trained on imbalanced data).

All of the 'winning' models represent ensemble methods where training data is resampled within algorithm to train separate models. It might explain why these algorithms perform better when working with original imbalanced data.

4. Performance Improvement

After identifying the best performing algorithms I explored the ways to improve their performance. As the first step for improvement I tried combining three ‘winning’ models in the **Voting Classifier.** It yielded about the same result for precision score, but improved recall and hence f1 score slightly.

Another technique to try to improve prediction power of the model will be to try to reduce dimensionality by principal component analysis (**PCA**). The set of predictors contained 20 features. I ran 18 PCA models to reduce the dimensions of data to 2, 3,..., 19 components. Then the model based on Random Forest was trained in the new dimensional space. Results were compared to Random Forest trained on original imbalanced data. The original classifier showed better results across three scores (precision, recall and f1 score).

Removing some of the **features** also did not bring any improvements.

Also there was no improvement when introducing **interaction between features** as well as **adding squared terms** of variables.

Hyperparameter tuning for **Random Forest** model found a set of optimal parameters. I trained the model with discovered parameters and it provided a slightly better score for precision and f1.

Hyperparemeter tuning for AdaBoost was not that successful. It did increase precision score, but recall and f1 scores suffered a lot.

5. Results

After building and testing over 20 models, I have concluded that the best one is using Random Forest classifier with tuned hyperparameters. The best results were also achieved on original imbalanced data. With 87% accuracy it correctly labels good deal listings and identifies about 85% of good deals.