# Springboard Data Science Capstone Project  Airbnb Yearly Revenue Prediction

Ashley Jiangyang

Mar, 2020

## Table of Contents


*1. Introduction* ........ *3*

*2. Data Collection and Wrangling Summary* ........ *3*

**2.1 Drop Features** ........ **4**

**2.2 Re-code the Features** ........ **4**

**2.3 Deal with Missingness** ........ **4**

**2.4 Outliers Detection** ........ **4**

**2.5 Generate New Features** ........ **5**

**2.6 Define the Target Metric – Yearly Revenue** ........ **5**

**2.7 Data Codebook** ........ **6**

*2 Exploratory Data Analysis Summary* ........ *8*

**3.1 Yearly Revenue** ........ **8**

**3.2 Host Activeness on the Platform** ........ **9**

**3.3 Listings Accommodates & Facilities** ........ **13**

**3.4 Interactive Relationship** ........ **17**

**3.5 Geographical Information** ........ **21**

*3 Results and In-depth Analysis Using Machine Learning* ........ *22*

**3.1 Preprocess Data** ........ **22**

3.1.1 Access Collinearity ........ 22

3.1.2 Normalizing & Standardizing ........ 24

**3.2 Evaluate & Compare Machine Learning Models** ........ **24**

3.2.1 Models to Evaluate ........ 24

3.2.2 Model Optimization ........ 25

**4.3 Conclusion & Recommendation** ........ **30**

*4 Conclusion* ........ *31*

## 1. Introduction

Since its founding, Airbnb has hosted over 60 million people in 34,000 cities across the world and is continuing to grow quickly. Airbnb provides an alternative source of income for people who have otherwise vacant properties and for guests looking for affordable and convenient housing options. With any service, trying to monitor and understand the underlying attributes dynamics of the Airbnb and market trend are very important both for hosts and the entire housing market. As users continue to grow on both the supply and demand side, it becomes crucial to understand and predict how much can a listing earns on the platform.
We seek to analyze over 3,800 listings in the Seattle area in order to better understand how the use of listing attributes such as host activeness, accommodates, location, ratings, and more can be used to accurately predict the optimal earning for the host and those who want to join the platform to become hosts. With better-earning suggestion estimates, Airbnb hosts can improve their management of properties to gain better earning, also the information from the analysis would be insightful for the whole housing market. The objective of this project is to build a model that predicts the earning of a property taking into account listing features and attributes. The end goal is so hosts and the platforms can understand what features of an Airbnb listing are most important to managing and invest for greater yearly.
Answering the question would be a benefit for both potential hosts and any home-sharing platform, that info will be very useful guidance to make the investment as well as regulate the listings market.

## 2. Data Collection and Wrangling Summary

The detailed data we use listings.csv (22 February, 2020) was acquired from the Airbnb official website, check the official website for more detailed information. This data set contains 7,544 Observations and 106 features regarding the listing attributes. Each row is an entry for a listing, and the columns describe the information such as host reaction(host response rate/time), listing properties(property type, room type, bathrooms, bedrooms), and review score. Etc.

We performed data wrangling step with the following data management steps to generate a cleaned dataset for further analysis.

### 2.1 Drop Features

51 meaningless features are dropped, which includes,

- 32 *Meaningless Features* - such as ID, name, constant value, text-related values,
- *16 Redundancy features* - we will keep one features if a multiple same information exist,
- *3 Over-missingness Features* - features with over 50% missing values.

### 2.2 Re-code the Features

42 features are recoded, which are,

- *Convert the Data Type*-  24 data objects are converted to numeric/categorical/datetime/Boolean data type,
- *Re-categorize the Data – 17 features are re*grouped with less categories,
- *Extract Information from Useable Format* - extract 1 features from list format and recoded as 30 dummy features.

### 2.3 Deal with Missingness

Dropping cases and imputation were applied,

- 961 (12.7%) records was dropped due to MNAR (Missing Not At Random),
- 2 features were imputed with 0 (13.63% NAs, 5.35% NAs ),
- 2 features were imputed by median (9.90% NAs, 0.06% NAs ).

### 2.4 Outliers Detection

Sanity check for all the data range and 0 outliers were dropped.

- Data Range and quantile were calculated for continuous data,

- Percentage for each classes were calculated for categorical data.

## 2.5 Generate New Features

3 new features were created by the raw features, which are,

- *time_to_first_review*: months a listing takes from join to get the first review,
- *last_review_month*: months a listing get the last review to the time that data was scraped,
- *days_on_airbnb*: days from the first review to the last review.

## 2.6 Define the Target Metric – Yearly Revenue

This section is a step-by-step demonstration how we define the metric **Yearly Revenue** as the target features.

The San Francisco Model refers to a method created by Alex Marqusee for the San Francisco Planning Department (Brousseau, 2015) and the Budget and Legislative Analyst's Office (Rodgers, 2015) to quantify the impact of Airbnb in this city. These method given develop ways of estimate business metrics to evaluate local Airbnb listing's booking, occupancy rate and revenue.

We used the San Francisco Model as a guidance for our calculation, below we have listed the features and formulate that leads us to the final business metrics,

- **Days on Airbnb** = $last_{review} - first_{review}$
- **Minimum Booking in Year** = *reviews_per_month*12*
- **Estimated Booking in Year** = *Minimum Booking in Year/ 50%*
- **Nights Per Year CAP** = *Estimated Booking in Year * minimum_nights_avg_nt*
- **Occupancy Rate** = *Nights Per Year CAP /365*

- **Yearly Revenue** = *price * Occupancy Rate * 365*

### 2.7 Data Codebook

The final dataset contains 6,583 observation and 80 features in total. A simplified data codebook is given below,

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6583 entries, 0 to 7513
Data columns (total 80 columns):
 #   Column                                         Non-Null Count  Dtype
---  ------                                         --------------  -----
 0   host_id                                        6583 non-null   int64
 1   host_since                                     6583 non-null   datetime64[ns]
 2   host_response_time                             6583 non-null   object
 3   host_response_rate                             6583 non-null   category
 4   host_acceptance_rate                           6583 non-null   float64
 5   host_is_superhost                              6583 non-null   float64
 6   host_identity_verified                         6583 non-null   float64
 7   neighbourhood_group_cleansed                   6583 non-null   category
 8   zipcode                                        6554 non-null   object
 9   latitude                                       6583 non-null   float64
 10  longitude                                      6583 non-null   float64
 11  is_location_exact                              6583 non-null   float64
 12  property_type                                  6583 non-null   category
 13  room_type                                      6583 non-null   category
 14  accommodates                                   6583 non-null   int64
 15  bathrooms                                      6583 non-null   float64
 16  bedrooms                                       6583 non-null   float64
 17  beds                                           6583 non-null   float64
 18  price                                          6583 non-null   float64
 19  security_deposit                               6583 non-null   float64
 20  cleaning_fee                                   6583 non-null   float64
 21  guests_included                                6583 non-null   int64
 22  extra_people                                   6583 non-null   float64
 23  minimum_nights_avg_ntm                         6583 non-null   float64
 24  calendar_updated                               6583 non-null   category
```

```
 25  availability_90                              6583 non-null   int64
 26  number_of_reviews                            6583 non-null   int64
 27  number_of_reviews_ltm                        6583 non-null   int64
 28  first_review                                 6583 non-null   datetime64[ns]
 29  last_review                                  6583 non-null   datetime64[ns]
 30  review_scores_rating                         6583 non-null   category
 31  review_scores_accuracy                       6583 non-null   category
 32  review_scores_cleanliness                    6583 non-null   category
 33  review_scores_checkin                        6583 non-null   category
 34  review_scores_communication                  6583 non-null   category
 35  review_scores_location                       6583 non-null   category
 36  review_scores_value                          6583 non-null   category
 37  instant_bookable                             6583 non-null   float64
 38  cancellation_policy                          6583 non-null   object
 39  require_guest_profile_picture                6583 non-null   float64
 40  require_guest_phone_verification             6583 non-null   float64
 41  calculated_host_listings_count_entire_homes  6583 non-null   int64
 42  calculated_host_listings_count_private_rooms 6583 non-null   int64
 43  calculated_host_listings_count_shared_rooms  6583 non-null   int64
 44  reviews_per_month                            6583 non-null   float64
 45  time_to_first_review                         6583 non-null   category
 46  last_review_month                            6583 non-null   category
 47  Wifi                                         6583 non-null   float64
 48  Essentials                                   6583 non-null   float64
 49  Heating                                      6583 non-null   float64
 50  Smoke detector                               6583 non-null   float64
 51  Shampoo                                      6583 non-null   float64
 52  Hangers                                      6583 non-null   float64
 53  Hair dryer                                   6583 non-null   float64
 54  Carbon monoxide detector                     6583 non-null   float64

 55  Laptop friendly workspace                    6583 non-null   float64
 56  Iron                                         6583 non-null   float64
 57  TV                                           6583 non-null   float64
 58  Washer                                       6583 non-null   float64
 59  Dryer                                        6583 non-null   float64
 60  Hot water                                    6583 non-null   float64
 61  Fire extinguisher                            6583 non-null   float64
 62  Dishes and silverware                        6583 non-null   float64
 63  Refrigerator                                 6583 non-null   float64
 64  Microwave                                    6583 non-null   float64
 65  Coffee maker                                 6583 non-null   float64
 66  Self check-in                                6583 non-null   float64
 67  Free street parking                          6583 non-null   float64
 68  Cooking basics                               6583 non-null   float64
 69  Stove                                        6583 non-null   float64
 70  Bed linens                                   6583 non-null   float64
 71  First aid kit                                6583 non-null   float64
 72  Oven                                         6583 non-null   float64
 73  Private entrance                             6583 non-null   float64
 74  Extra pillows and blankets                   6583 non-null   float64
 75  Free parking on premises                     6583 non-null   float64
 76  Dishwasher                                   6583 non-null   float64
 77  days_on_airbnb                               6583 non-null   float64
 78  occupancy_rate                               6583 non-null   float64
 79  yearly_revenue                               6583 non-null   float64
dtypes: category(14), datetime64[ns](3), float64(51), int64(9), object(3)
memory usage: 3.5+ MB
```

More details and source codes on Data Wrangling are available at

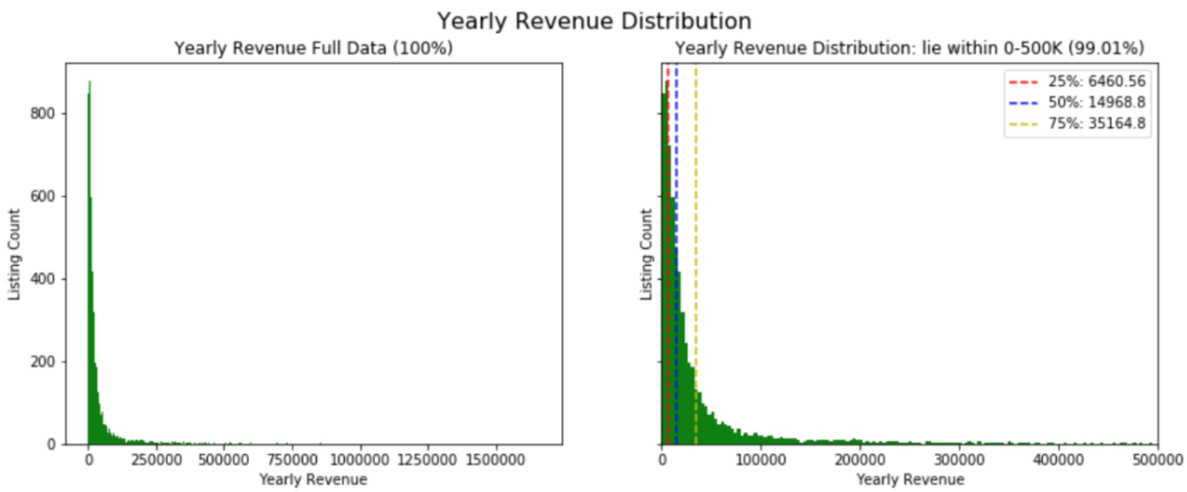
Capstone_Project_Data_Wrangling.

## 2 Exploratory Data Analysis Summary

In this section, we perform the Exploratory Data Analysis on the cleaned data generated from the Data Wrangling step. We will go through most of the features in the data to explore their relationship with the Yearly Revenue. And the EDA is organized based on different topics on the related information, features regarding the same topic information will be  analysis together.

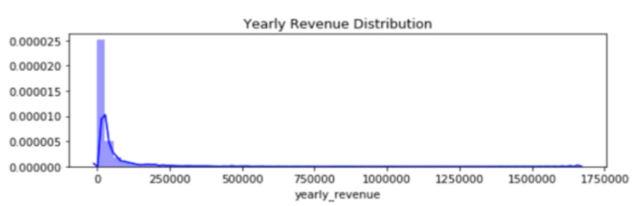
### 3.1 Yearly Revenue

The Yearly Revenue is the target business metric we want to evaluate for all the listing population from our data, the definition was given in 2.7. Let’s evaluate the distribution and statistics summary for it.




*Yearly Revenue Statistics*

| | |
|---|---|
| Mean | 42788.59 |
| Std | 92133.01 |
| Min | 96.00 |
| 25% | 6460.56 |
| 50% | 14968.80 |
| 75% | 35164.80 |
| Max | 1659240.00 |

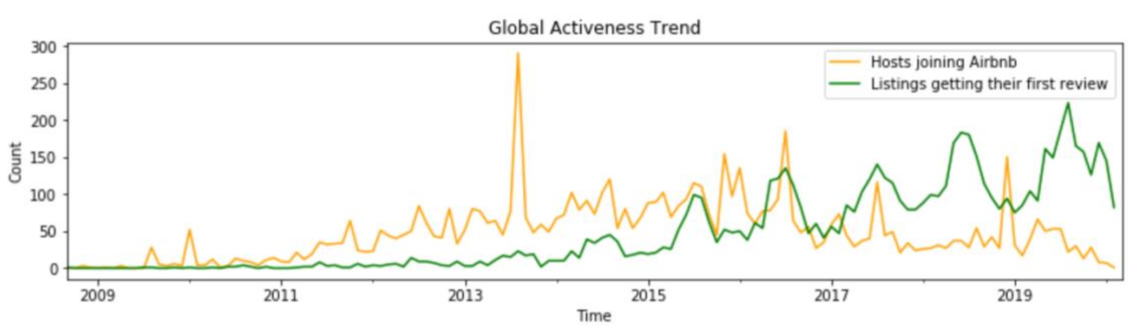


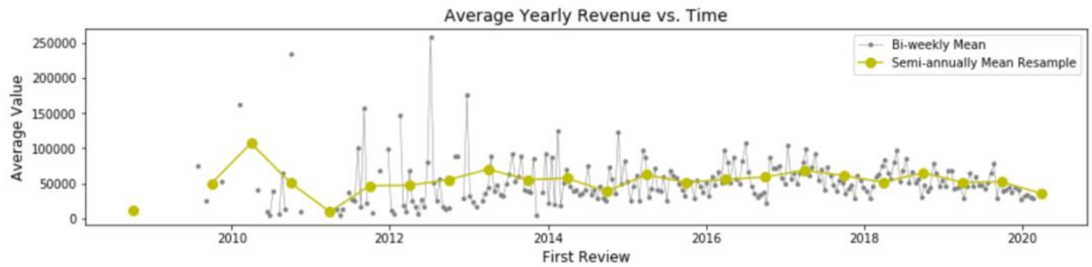
From the summary statistics above, we can see that a listing can have no revenue at all among the year, also a listing can gain a very high revenue up to $1,659,240. However , the distribution of the yearly revenue is highly right-skewed, we can see that most of the data are centered within $100,000, and 99% of our data lies within $500,000. The median is about $ 14,968 and the average revenue goes up to $42,788. The density plot is given alongside. Putting those numbers into perspective, the 2019 Airbnb official sites reported that On average globally, Experience hosts earn $10,000 a year, our data reveals a info that indicating a healthy premium for actively managed listings for the Seattle Airbnb Market.

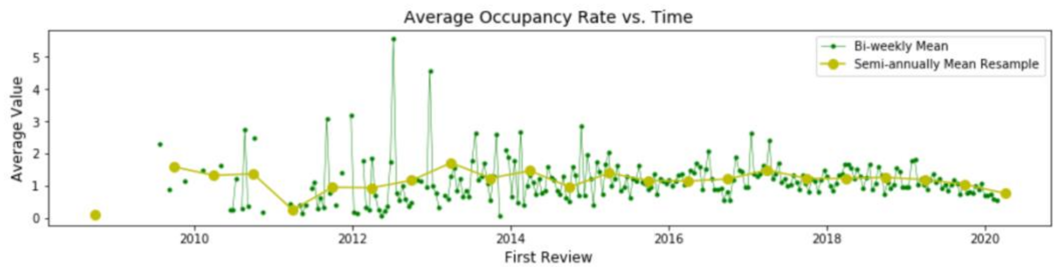
### 3.2 Host Activeness on the Platform

In this section, we will exam the host activeness from different perspectives, which includes the global active on the platform, the ability to attract the guest and stay on the market, and it's longevity on the platform. And also explore the their relationship with the yearly revenue.

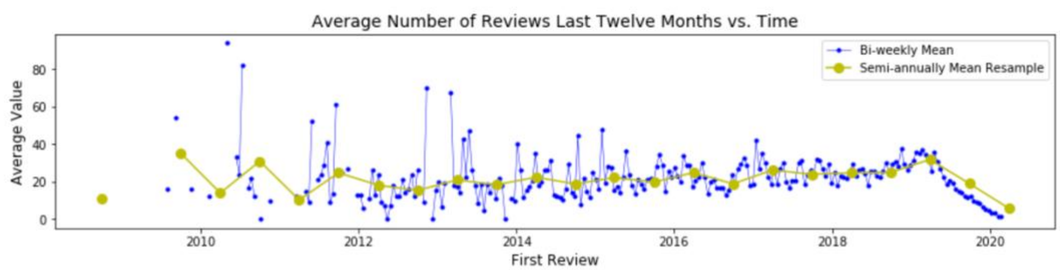
**Global Business Trend**

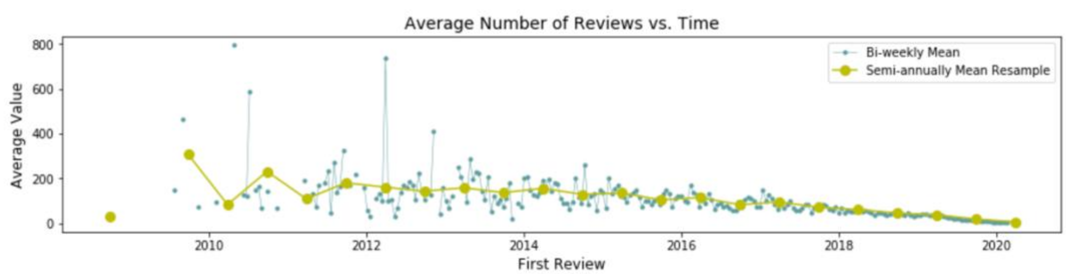




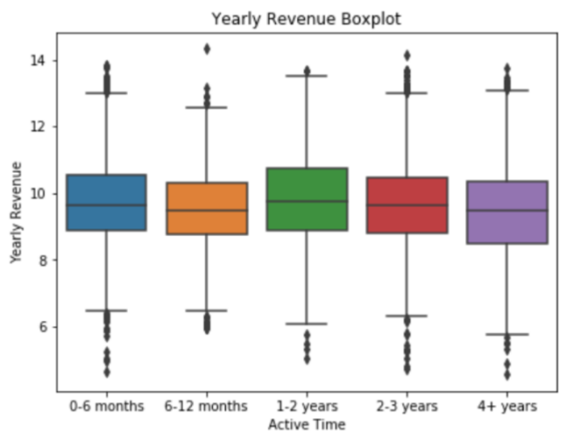


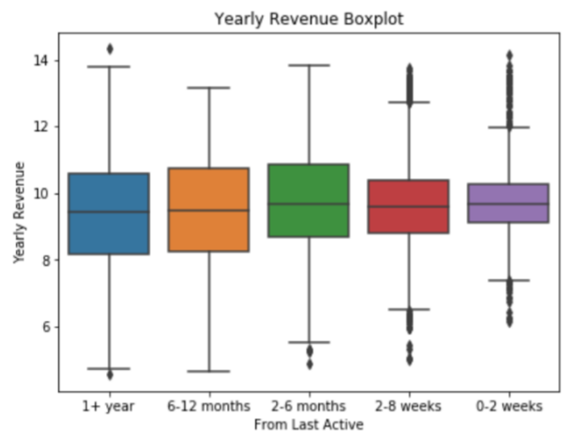



The first plot indicates that the business is keep boosting all these years – more host join Airbnb and the number of host get its first review is increasing. In general, during the decades of 2010 to 2020, there are not tremendous increase among all the business metrics. The yearly revenue, price, and occupancy rate look flat with a slightly increasing tendency. Noticed that the both the average review number and that feature in the last twelve month have a slightly decrease trend could cause by delay of getting the review data coming in.

**Power of Attraction & Staying**



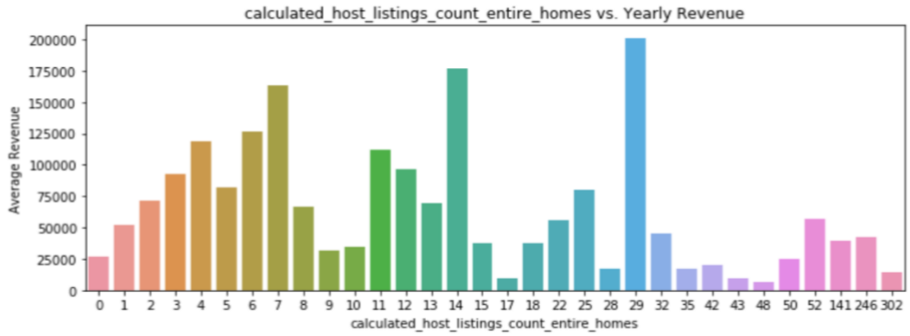



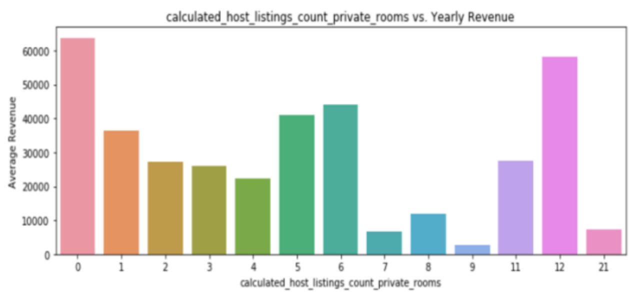

*Statistical summary table*

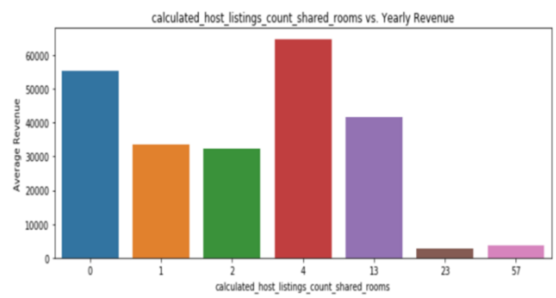
| **active_time** | | | | **from_last_active** | | | |
|---|---|---|---|---|---|---|---|
| | Count | Mean | Median | | Count | Mean | Median |
| **0-6 months** | 1,603 | $47,808 | $15,585 | **0-2 weeks** | 1,655 | $33,775 | $16,236 |
| **6-12 months** | 630 | $34,332 | $12,994 | **2-8 weeks** | 1,987 | $43,877 | $14,515 |
| **1-2 years** | 981 | $51,735 | $17,280 | **2-6 months** | 1,463 | $54,541 | $16,245 |
| **2-3 years** | 1,743 | $43,588 | $15,552 | **6-12 months** | 480 | $41,594 | $13,212 |
| **4+ years** | 1,626 | $34,860 | $13,058 | **1+ years** | 998 | $38,912 | $12,391 |

The created feature active_time is the months values from join to get the first review, which indicated power of attraction with shorter time. It looks like the among the groups of different months length to get the first review, the yearly revenue doesn't change much. The statistics summary indicates that the 1-2 years have the highest mean ($51,735) and median ($17,280). To embody the staying power, we created feature from_last_active, which is the length of last review to the data was scrapped time. The plots indicates that as the better staying power (get the last review as recent as it can), the higher the average yearly revenue.

**Host Listing Counts**

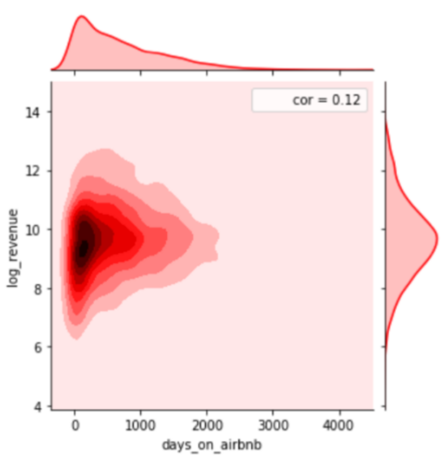



The Host Listing Count features are specified for entire homes/private rooms/shared rooms, the average yearly revenue varies and show no specific patterns among different counts.

**Days on Airbnb**

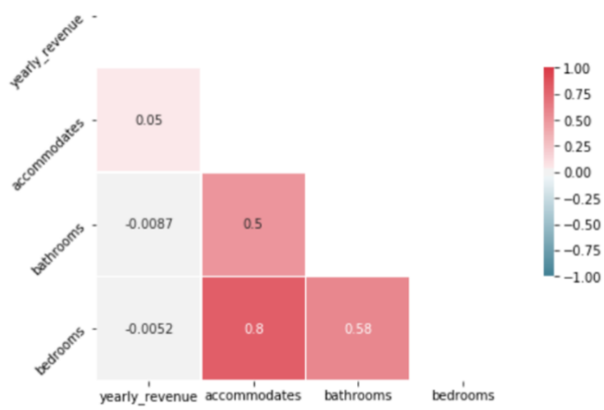



The features days_on_airbnb is calculated length from first review to last review, which potentially indicates the activeness length on the platform. Original pearson correlation is weak

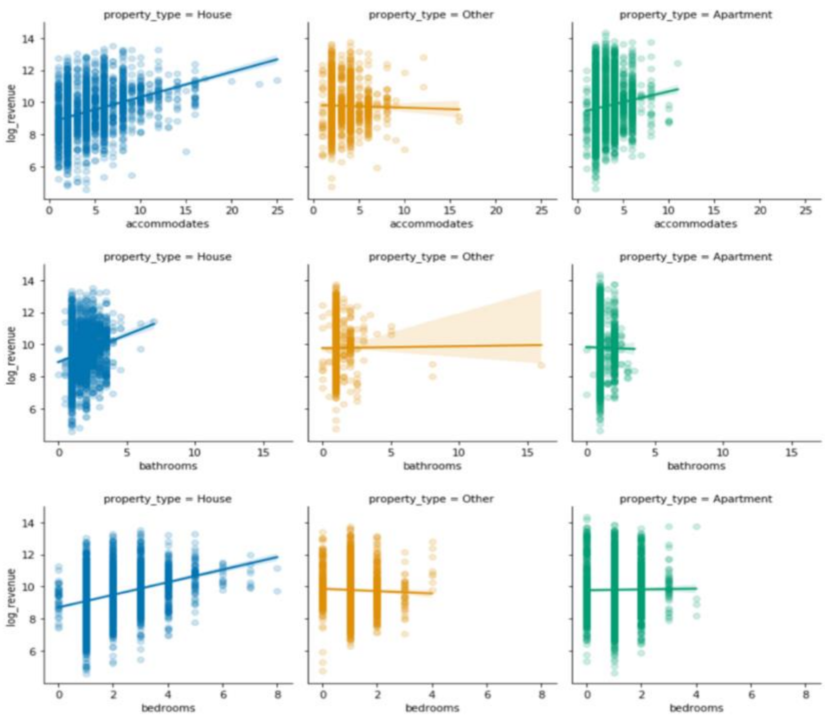
0.035 between the yearly revenue and days on Airbnb. The KDE plot above show the log transformed revenue and correlation. We can see that most intense data are within 0-1000 days, which is about 3 years, there's no indication that longer days is associated with higher revenue.

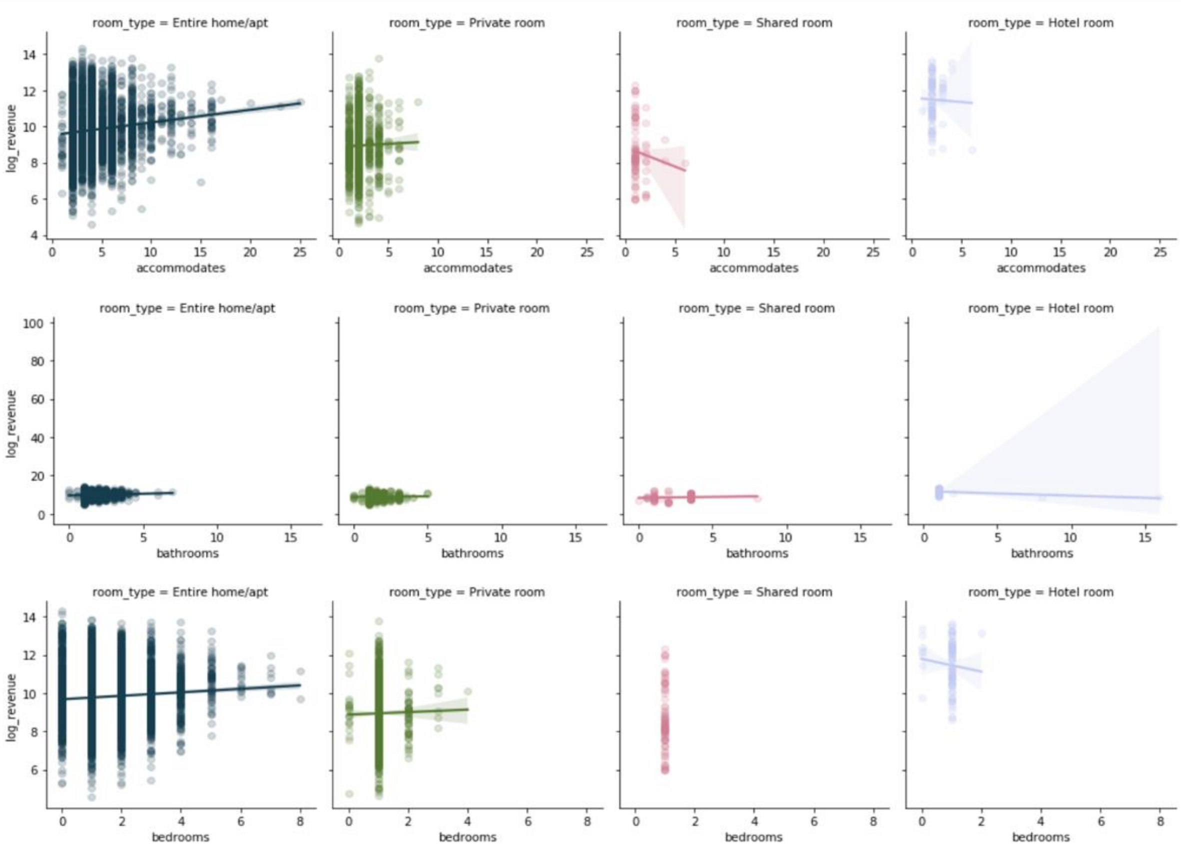
### 3.3 Listings Accommodates & Facilities

**Bedrooms & Bathrooms & Accommodates**



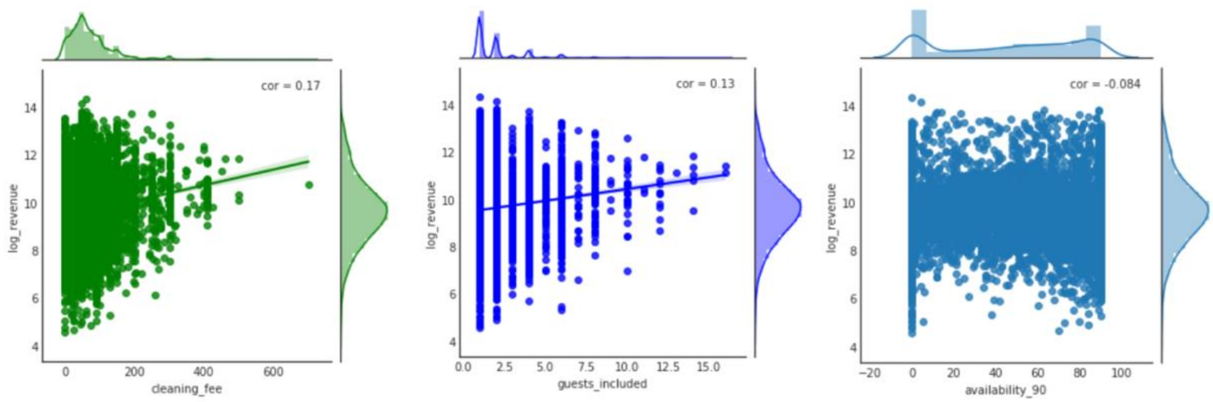

The correlation matrix above shows negligible correlation between the revenue and bedrooms/bathrooms, let's take a further examination this accommodates/bedrooms/bathrooms features within two features - property type and room type.


property_type = House
property_type = Other
property_type = Apartment
log_revenue
accommodates
bathrooms
bedrooms


room_type = Entire home/apt
room_type = Private room
room_type = Shared room
room_type = Hotel room
log_revenue
accommodates
bathrooms
bedrooms

The above detailed plots the relationship between yearly revenue with each class in property type and room type. Only under the property type is house and apartment, there's slightly positive relationship indicates that as the number of accommodates increases, the yearly revenue increases. And the same for revenue and accommodates within property type is apartment.

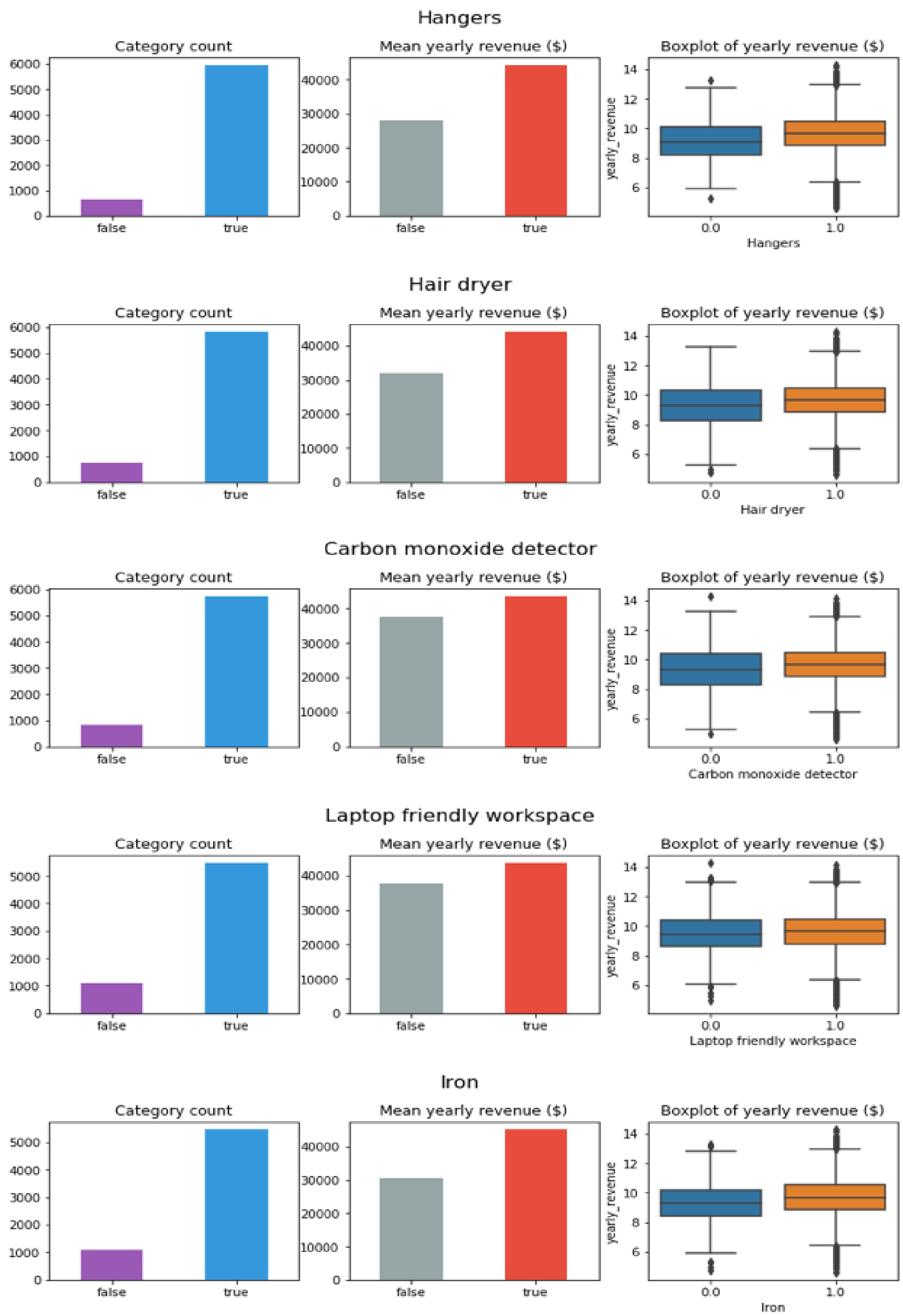
**Clean Fee & Guest Included & Availability**




The above features are examined with log transformed yearly revenue. There is minor positive relationship between the yearly revenue and features of cleaning fee and guests included.

**Amenities**

We have 30 single amenity features to check, they are essential facilities like WIFI, TV, Hairdryer, etc. We combine the plots of the counts with and without the single amenities, the mean revenue for each class along with the boxplot, we display the 5 out of 10 single amenities here, check the Jupyter notebook for more details.



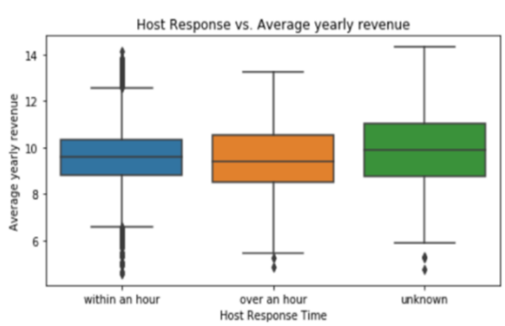

For better information, we implement bootstrap hypothesis testing across the 30 amenities. Univariate test reveals statistical significance for five amenities - **Fire extinguisher, Free street parking, First aid kit, private entrance and Free parking on premises,** which indicates with those amenities provided, the yearly revenue are statistically different. Further examination brings in more interesting results - the mean yearly revenue are lower with those amenities.

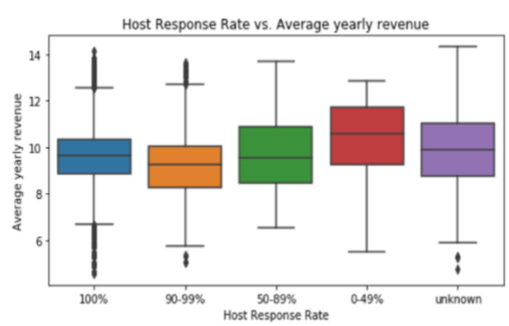
```
Bootstrap hypothesis test for difference of means yearly revenue:

Hypothesis Test of Wifi                       | p-value = 0.4641
Hypothesis Test of Essentials                 | p-value = 0.9673
Hypothesis Test of Heating                    | p-value = 0.4547
Hypothesis Test of Smoke detector             | p-value = 0.9892
Hypothesis Test of Shampoo                    | p-value = 0.9614
Hypothesis Test of Hangers                    | p-value = 1.0
Hypothesis Test of Hair dryer                 | p-value = 1.0
Hypothesis Test of Carbon monoxide detector   | p-value = 0.9708
Hypothesis Test of Laptop friendly workspace  | p-value = 0.9898
Hypothesis Test of Iron                       | p-value = 1.0
Hypothesis Test of TV                         | p-value = 1.0
Hypothesis Test of Washer                     | p-value = 1.0
Hypothesis Test of Dryer                      | p-value = 1.0
Hypothesis Test of Hot water                  | p-value = 0.7468
Hypothesis Test of Fire extinguisher          | p-value = 0.0
Hypothesis Test of Dishes and silverware      | p-value = 0.2256
Hypothesis Test of Refrigerator               | p-value = 0.0344
Hypothesis Test of Microwave                  | p-value = 0.0519
Hypothesis Test of Coffee maker               | p-value = 0.4128
Hypothesis Test of Self check-in              | p-value = 1.0
Hypothesis Test of Free street parking        | p-value = 0.0
Hypothesis Test of Cooking basics             | p-value = 0.9986
Hypothesis Test of Stove                      | p-value = 0.9977
Hypothesis Test of Bed linens                 | p-value = 0.9245
Hypothesis Test of First aid kit              | p-value = 0.0
Hypothesis Test of Oven                       | p-value = 0.9998
Hypothesis Test of Private entrance           | p-value = 0.0
Hypothesis Test of Extra pillows and blankets | p-value = 0.798
Hypothesis Test of Free parking on premises   | p-value = 0.0
Hypothesis Test of Dishwasher                 | p-value = 0.9967
```

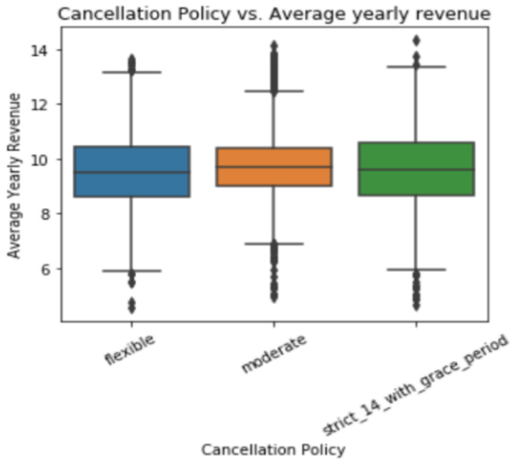
## 3.4 Interactive Relationship

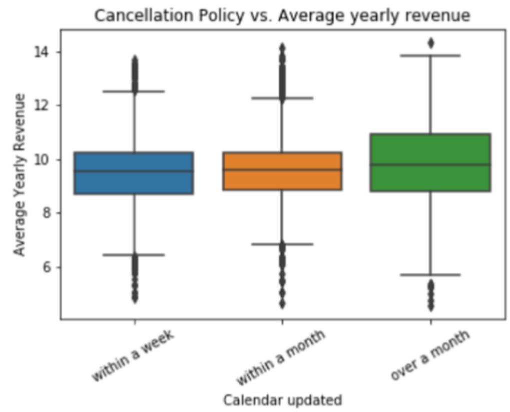
The section we group the features into two perspectives and the exploration analysis will be performed from two sides.

**From Host's Side**



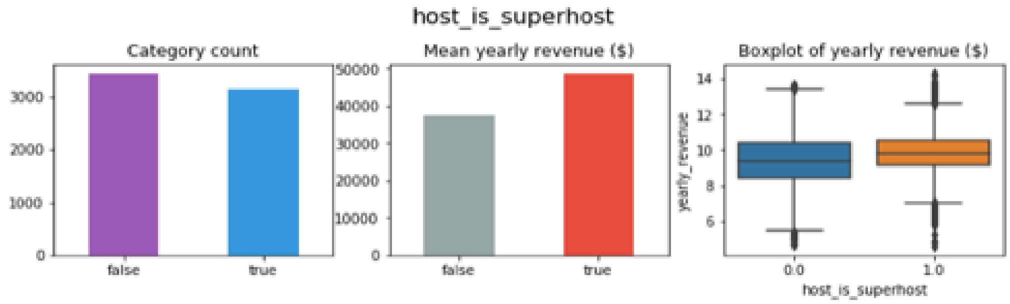


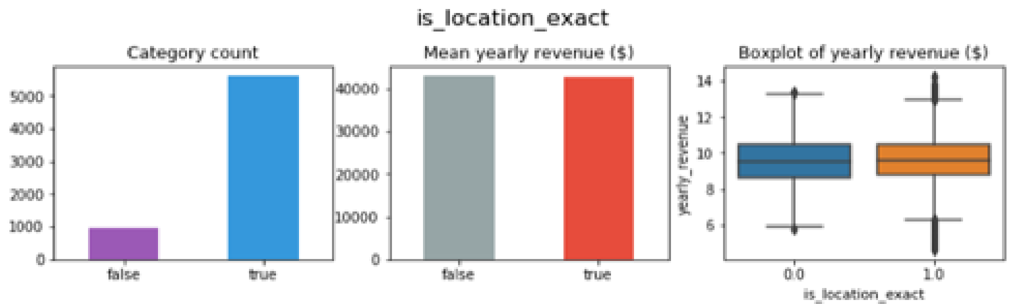



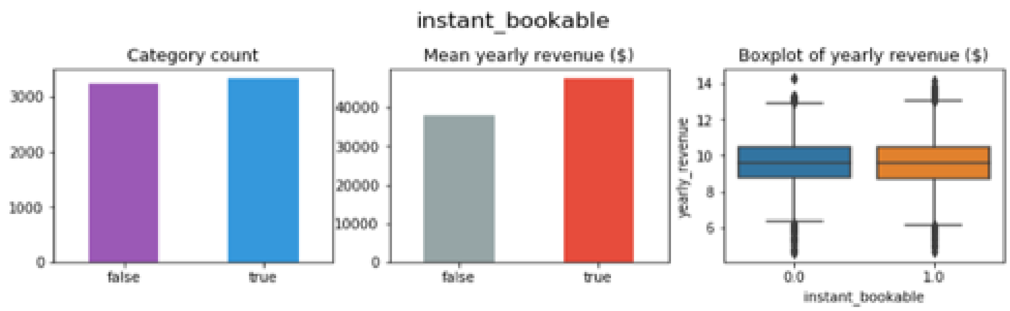

The more active response rate and time are not appear to have a better revenue than we expected. Neither are the features as cancellation policy and calendar updated time.

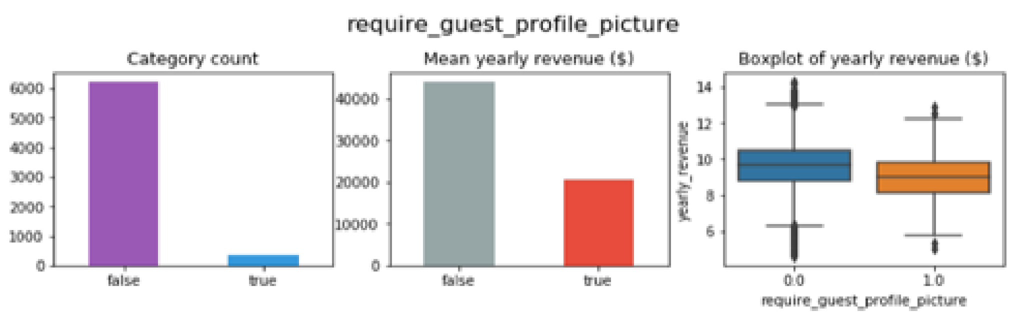
We explore the five features – *host_is_superhost, is_location_excat, instant-bookable, require_guest_profile_picture and require_guest_phone_verification* below. The visualization and bootstrap hypothesis testing were applied for the
the It looks like only require guest profile picture and require quest phone verification are associated with lower yearly revenue – maybe the guests prefer no verification and skip these listing when making a choice.

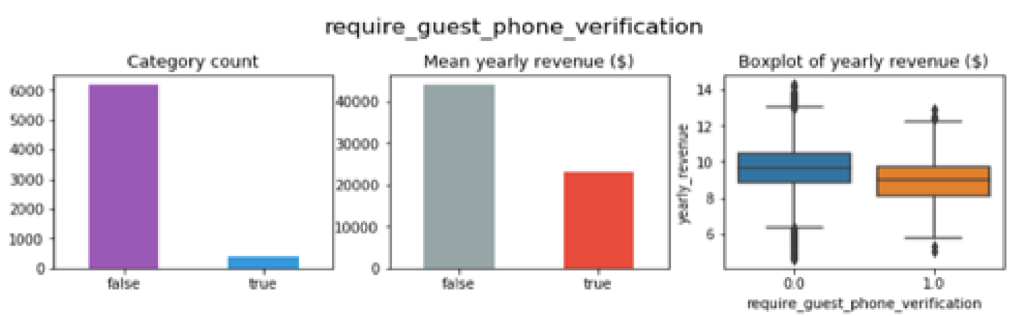
```
Bootstrap hypothesis test for difference of means yearly revenue:

Hypothesis Test of host_is_superhost                | p-value = 1.0
Hypothesis Test of is_location_exact                | p-value = 0.4659
Hypothesis Test of instant_bookable                 | p-value = 1.0
Hypothesis Test of require_guest_profile_picture    | p-value = 0.0
Hypothesis Test of require_guest_phone_verification | p-value = 0.0
```


host_is_superhost
Category count
Mean yearly revenue ($)
Boxplot of yearly revenue ($)
false
true
yearly_revenue
0.0
1.0
host_is_superhost


is_location_exact
Category count
Mean yearly revenue ($)
Boxplot of yearly revenue ($)
false
true
yearly_revenue
0.0
1.0
is_location_exact

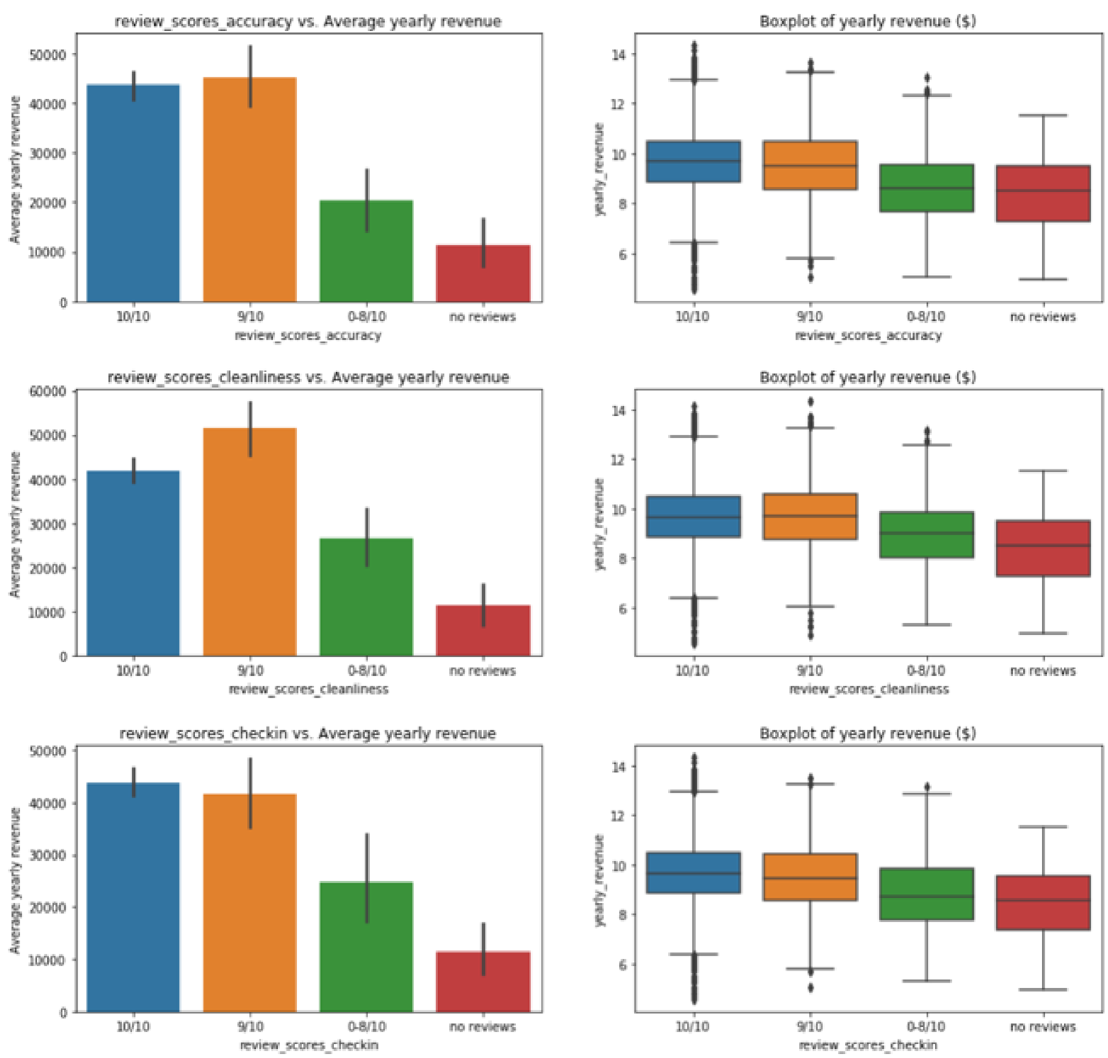

instant_bookable
Category count
Mean yearly revenue ($)
Boxplot of yearly revenue ($)
false
true
yearly_revenue
0.0
1.0
instant_bookable


require_guest_profile_picture
Category count
Mean yearly revenue ($)
Boxplot of yearly revenue ($)
false
true
yearly_revenue
0.0
1.0
require_guest_profile_picture


require_guest_phone_verification
Category count
Mean yearly revenue ($)
Boxplot of yearly revenue ($)
false
true
yearly_revenue
0.0
1.0
require_guest_phone_verification

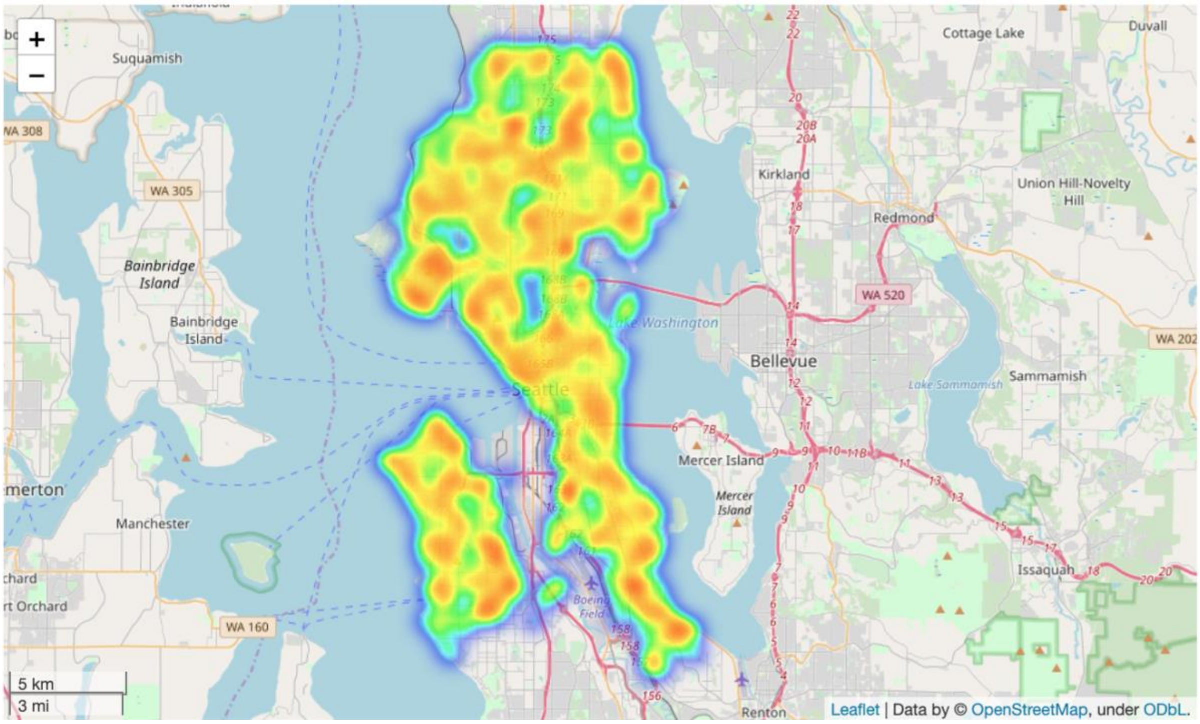
**From Guest's Side**

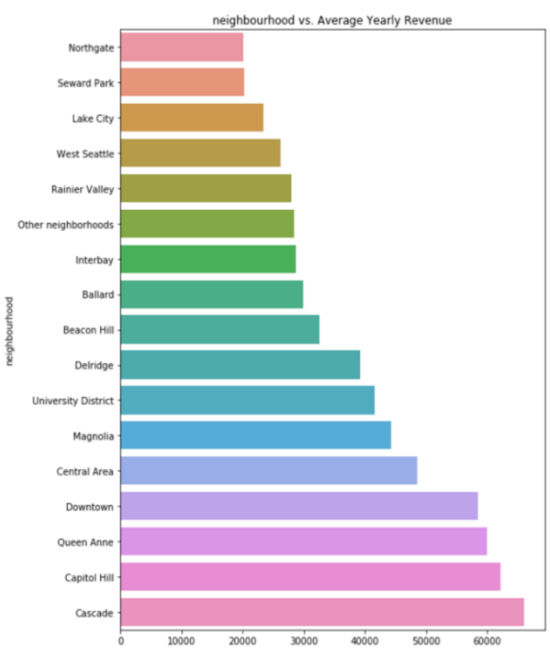
The features we can get info from the guest's side are their rating scores. There are 7 review scores regarding accuracy, cleanliness, checking, communication, location and value. As we expected – the higher score group earn more revenue in general, and the yearly revenue trend almost monotonic decreasing along with the review score go down. Noted that 3 features are given here, check the Jupyter notebook for more details.

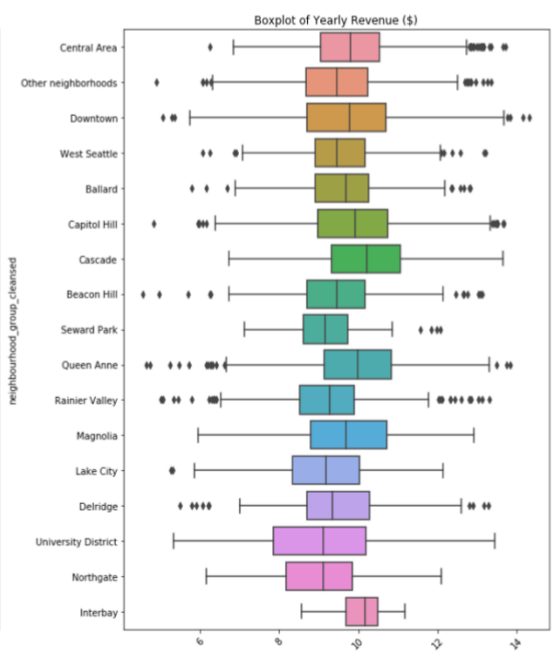


## 3.5 Geographical Information

We mapped the median revenue that group by the geographical information (latitude, longitude). It’s obvious that more close to the central part, the revenue is getting higher. When zoom in the plot, it’s more clear some region such as Queen Anne, and Capitol Hill.



We group the data by the neighborhood classes and use bar plot to display the average revenue for each area, the trends align with we get from the geographical data, the top regions are Cascade, Capitol Hill, Queen Anne, Downtown, and etc.

## 3 Results and In-depth Analysis Using Machine Learning

This section consisted of two parts, we will start from the **Data Preprocessing**, in which we will access the collinearity to drop the highly correlated features, and also apply the normalizing and standardizing on the dataset. After that, we will apply the machine learning algorithms in the **Modeling** part with the processed data.

### 3.1 Preprocess Data

#### 3.1.1 Access Collinearity

We use the heatmap as a tool to access the correlation between all the features we have.

Figure. Heatmap on features before dropping

The heatmap show's before processing, the correlation range is (-0.75, 0.75). Some features are highly correlated,

- *Accommodates* and *bathrooms, bedrooms, bed, guest_included,*
- *require_guest_phone_verification* and *require_guest_profile_profile_picture,*
- *Dryer* and *Washer, Dishes and silverware* and *Refrigerator etc.,*
- All the *'_No Reviews' features,* all the *'review_score_9/10'*

For all of the correlated features, we will only keep one from them. The heatmap for the reduced data are given below, the correlation range go down to (-0.6, 0.6).

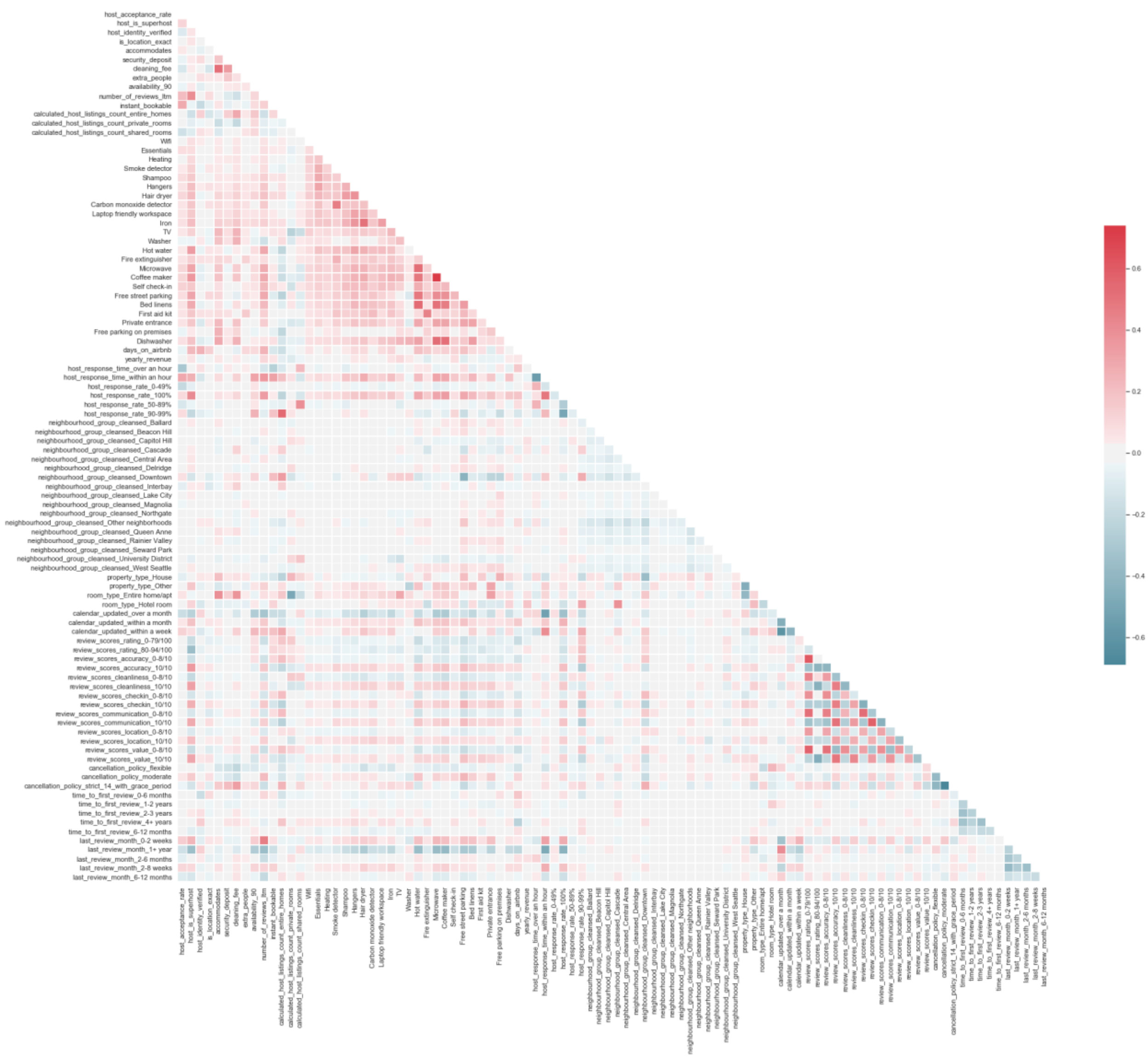


Figure. Heatmap on features after dropping

### 3.1.2 Normalizing & Standardizing

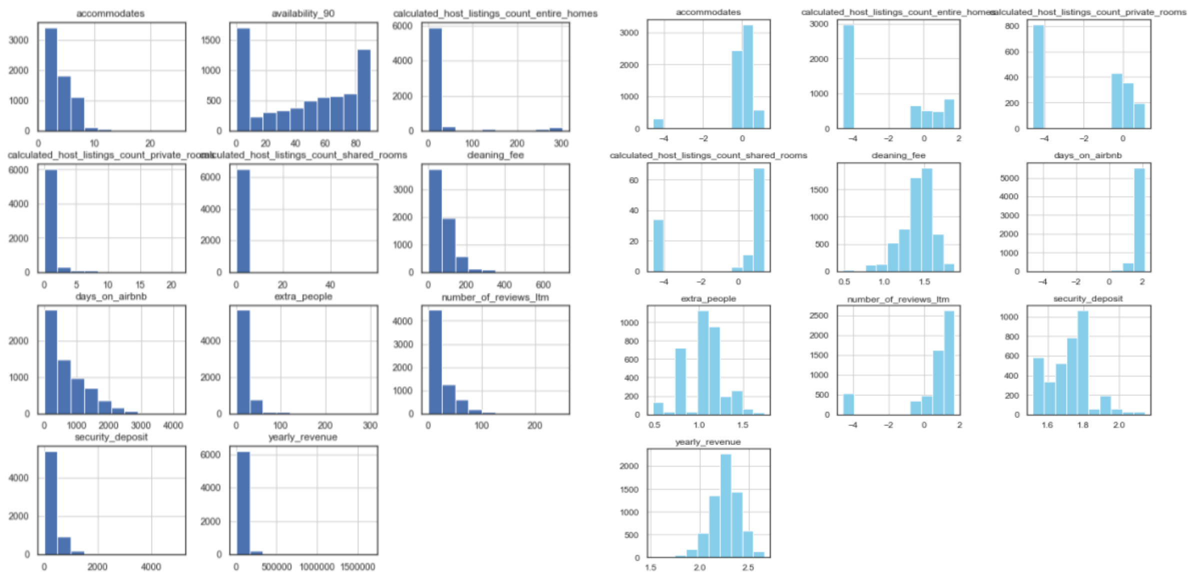



Figure. Before vs. after normalization

Then for all the continuous features, we applied the log transform. We still see after that not every single features approached the normal distribution perfectly, but all of them improved a lot.

Finally, we splitted the data into train (80%) and test (20%) dataset, and the standard scaler is applied to the train data, the train scaler the applied to test data to prevent the leaking problem.

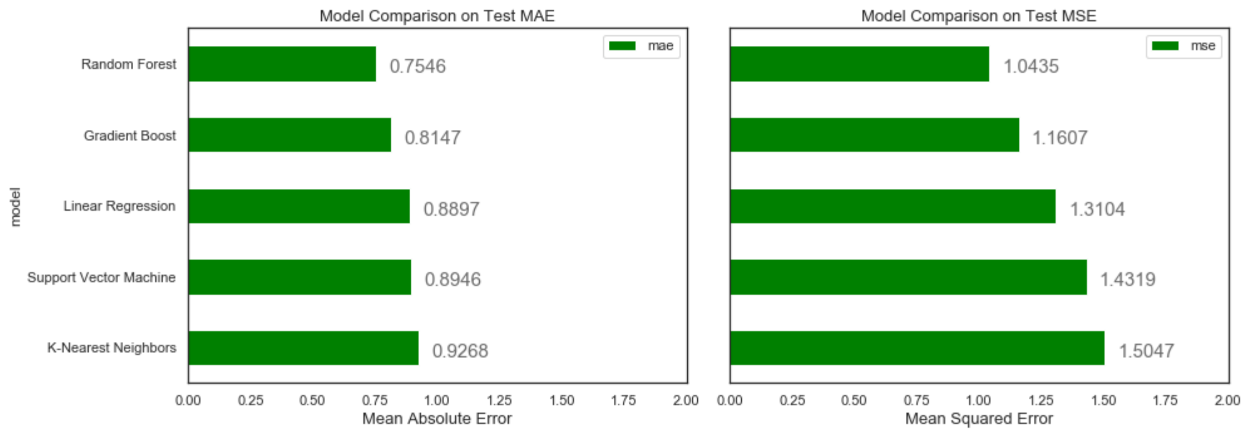
## 3.2 Evaluate & Compare Machine Learning Models

### 3.2.1 Models to Evaluate

We will compare five different machine learning models using the Scikit-Learn library:

- Linear Regression
- Support Vector Machine Regression
- Random Forest Regression
- Gradient Boosting Regression
- K-Nearest Neighbors Regression

To compare the models, we are going to be mostly using the Scikit-Learn defaults for the model hyperparameters. And in this section, we will focus on these models' baseline performance on our dataset generally instead of optimizing the model with a determined model to use. Then we can select the best performing model for further optimization using hyperparameter tuning.




We use the testing Mean Absolute Error (MAE) and Mean Standard Error (MSE). The Base model comparison shows Random Forest, Gradient Boost and Linear Regression have the best performance.

### 3.2.2 Model Optimization

#### *3.2.2.1 Hyperparameter Tuning*

In machine learning prediction, optimizing a model will find us the best set of hyperparameters for to achieve the best performance. we will start with the best base model random forest first.

In our case of a random forest, hyperparameters include,

- **n_estimators** - number of trees in the foreset
- **max_features** - max number of features considered for splitting a node
- **max_depth** - max number of levels in each decision tree
- **min_samples_split** - min number of data points placed in a node before the node is split
- **min_samples_leaf** - min number of data points allowed in a leaf node
- **bootstrap** - method for sampling data points (with or without replacement)

The best hyperparameters are usually impossible to determine before training, which means the best practice to determine the optimal settings is to try many different combinations to evaluate the performance of each model. However, evaluating each model only on the training set can cause overfitting - which means our model will score very well on the training whereas it will not be able to generalize to new data as good as we expected.

Hyperparameter Tuning with Random Search within Cross Validation will generate more reliable results for our reference.

In our case, we will use Random Search with cross validation to explore where the model can have relatively good performance, once we narrow down the options and then use the grid search with a more limited range of options for the best model.

The K-Fold Cross Validation technique dividing the training data into K folds, and then going through an iterative process on each K-1 fold data set and evaluate performance on the kth fold. Average error on each of the K iterations as the final performance will be computed as the given results.

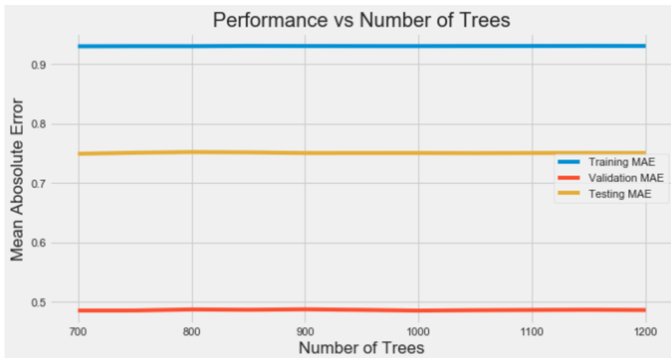
The next section will demonstrate how we use **random search** to narrow down the scope and use **grid search** to find our best model, with the combination of cross validation.

**Optimization Parameters**

| Optimization | Search Range | Best parameters | Cross - Validation |
|---|---|---|---|
| Random Search | n_estimators: (200, 2000) by 10<br>max_features: [auto, sqrt]<br>max_depth: (10, 110)<br>min_samples_split: [2, 5, 10]<br>min_samples_leaf: [1, 2, 4]<br>bootstrap: [True, Flase] | n_estimators: 400<br>max_features: sqrt<br>max_depth: None<br>min_samples_split: 2<br>min_samples_leaf: 1<br>bootstrap: Flase | 3 fold |
| Grid Search | n_estimators: (200, 700) by 50<br>max_features: sqrt<br>max_depth: None | n_estimators: 400<br>max_features: sqrt<br>max_depth: None | 3 fold |

| | |
|---|---|
| min_samples_split: 2 | min_samples_split: 2 |
| min_samples_leaf: 1 | min_samples_leaf: 1 |
| bootstrap: [True, Flase] | bootstrap: [True, Flase] |

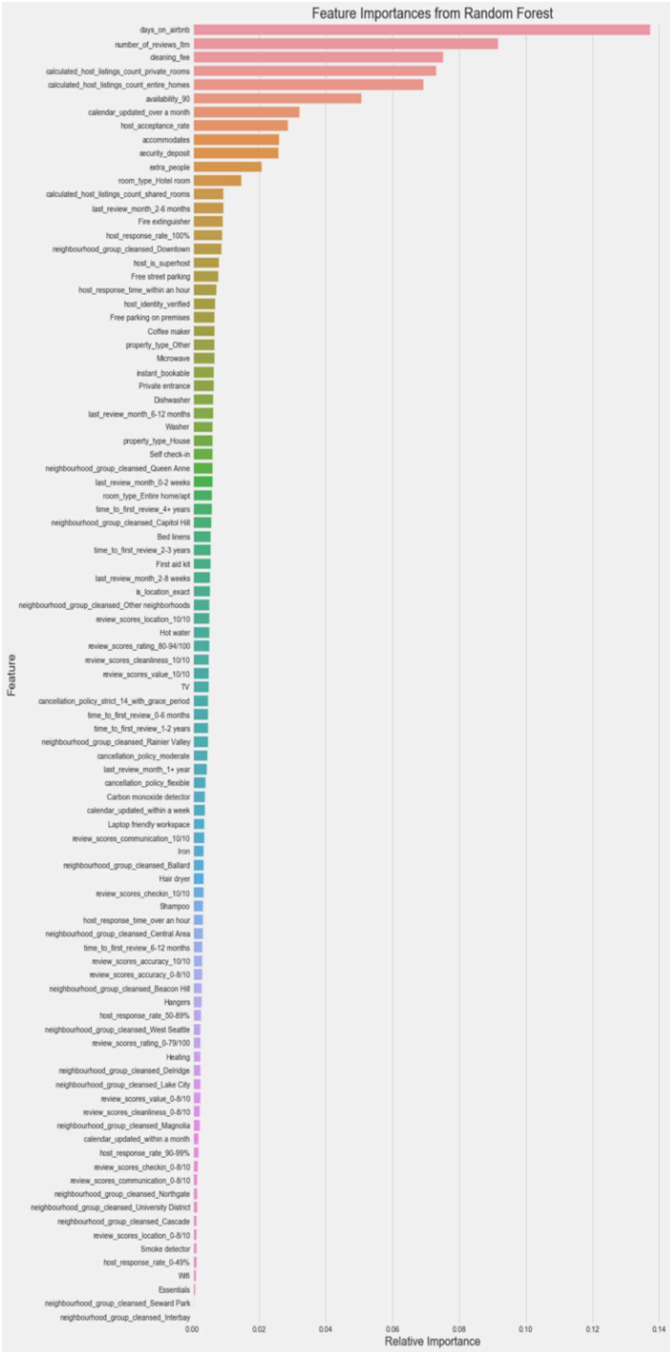
To proceed forward the grid search best model, we also plot the training error, validation error and testing error to see if the model have overfitting issue.



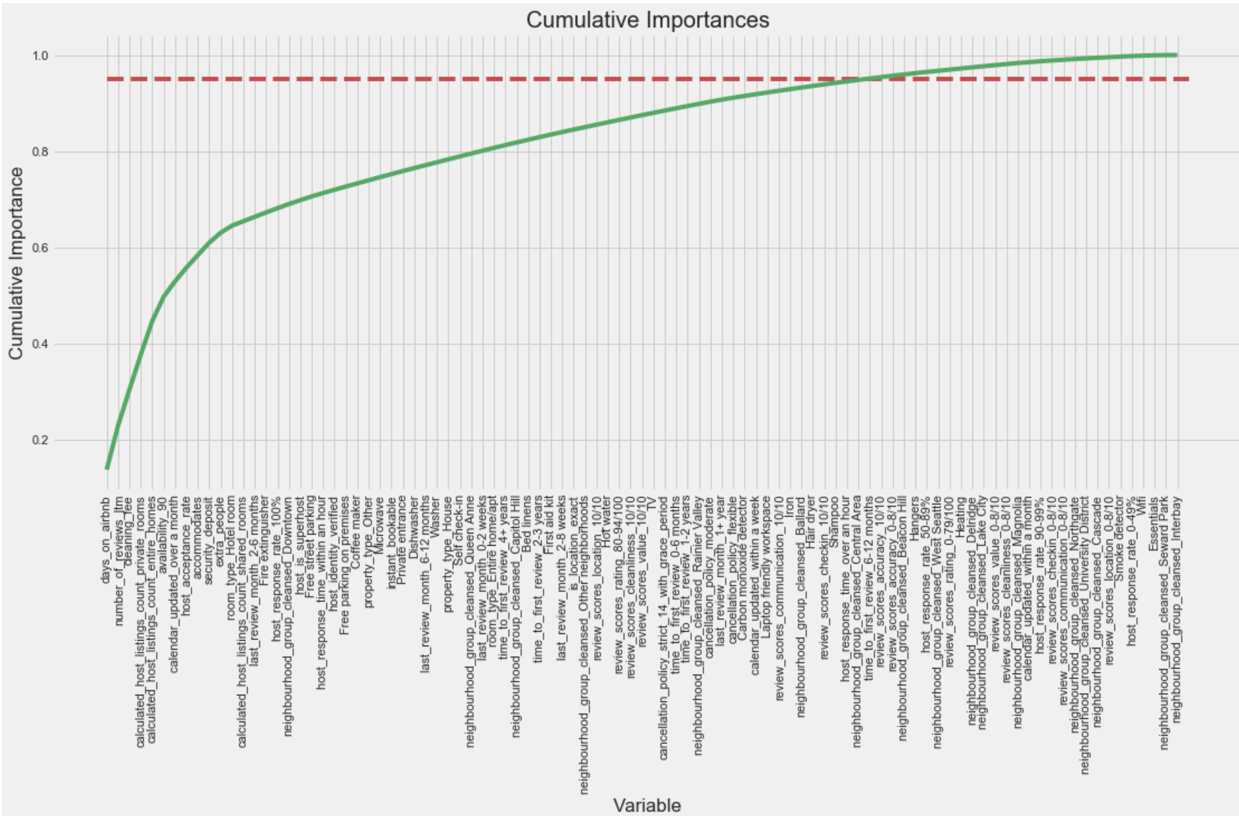

Within cross validation, we use 2 batch of data to train the model and 1 data set for validation. So the cross validation results are good – the validation error are lower than the training error. However, when the model is applied to the testing data, the model is involved with **overfitting** issue.

#### *4.2.2.2 Feature Reduction*

To better reduce the overfitting issue – we think of the curse of dimensionality – our model have 93 features! It’s time to consider the feature reduction for less complex model.




We plot the top 50 in our model, we can see down to the very bottom, some features have minimum vote for the model.




Further, we plot the accumulative importance by descending sorted features, and we added a 95% cutoff line, all the features will be used in the new model with the best parameter in precious session as the re-fitting model.

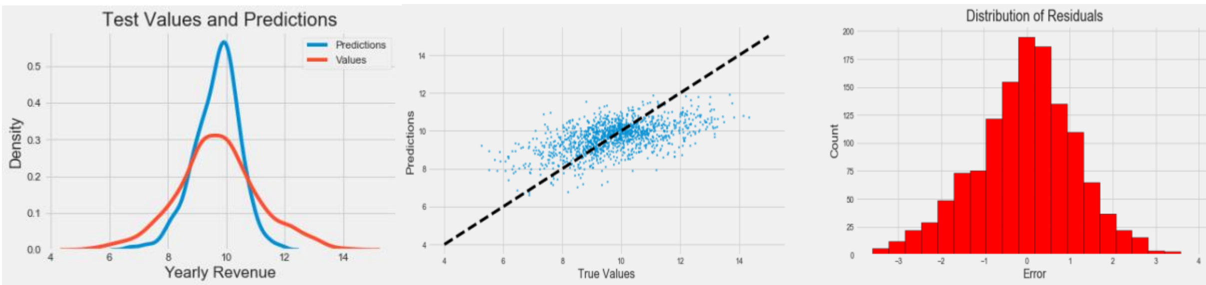
After refitting, the model contains 68 features, and the performance is as good as the model with 96 features.

Model Optimization Results

| Model | MAE | MSE | Accuracy |
|---|---|---|---|
| Base Model | 0.75 | 1.04 | 90.27% |
| Random Search best model | 0.65 | 0.98 | 92.07% |
| Grid Search best model | 0.76 | 0.98 | 92.01% |
| Feature Reduced model | 0.75 | 1.04 | 92.06% |

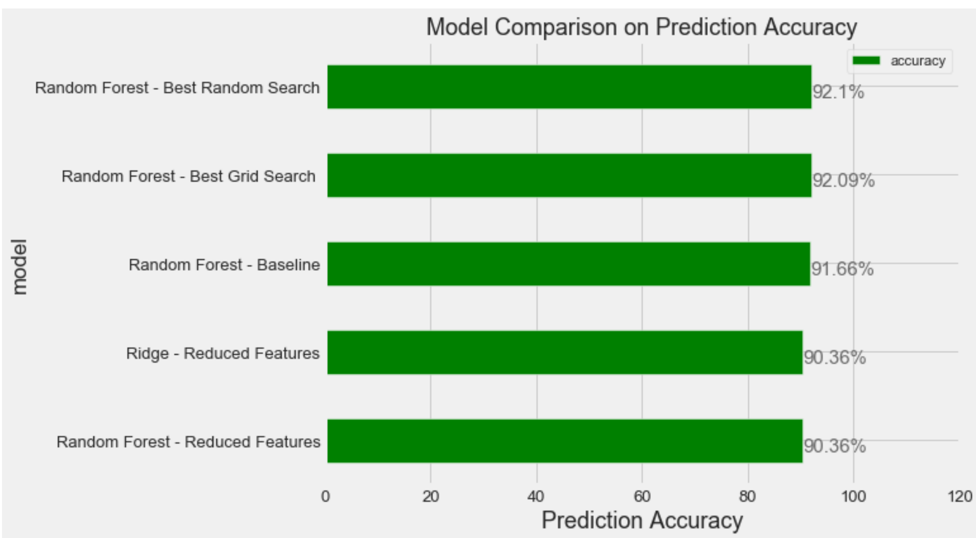
Since we already know that from the first beginning that the linear regression has a relatively good performance, we want to keep the model as simple as it can be, so we will use the reduced model to fit the linear regression.

The fitted linear model has MAE as 0.91, MSE as 1.38 and Test Accuracy as 90.36, which is very close to our featured reduced random forest model. Let’s check the model assumption.




In general our model have good performance, however, it have underprediction for the peak revenue value and overprediction for the tail in the two sides.

### 4.3 Conclusion & Recommendation




Compared to the best random forest model with an accuracy of 89.38%, the ridge regression would be a better choice for predicting the revenue.

| Model | Feature Numbers | Prediction Accuracy | Accuracy- CV | Overfitting |
|---|---|---|---|---|
| **Ridge - Reduced Features** | **58** | **96.21%** | **89.6%** | **No** |
| Random Forest - Best Random Search | 102 | 89.64% | 95.84% | Yes |
| Random Forest - Reduced Features | 58 | 89.64% | - | - |
| Random Forest - Best Grid Search | 102 | 89.61% | 95.9% | Yes |
| Random Forest - Baseline | 102 | 88.68% | - | - |

Based on the results above, the linear would be our final model, which has,

- best Prediction accuracy
- no overfitting
- a relatively simple algorithm for a further system implementing

# 4 Conclusion