ANALYSIS OF AIRBNB OPEN DATA

NAME: MEENAKSHI RAJGOPAL

Table of Content

ANALYSIS OF AIRBNB OPEN DATA

COMPANY OVERVIEW:

This American home rental platform is based in San Francisco, which allows people to find an accommodation or rent their place for short term housing in more than 191 countries and 81,000 cities. Airbnb was founded in 2008 by Brian Chesky, Joe Gebbia, and Nathan Blecharczyk. Recently, Airbnb launched open home program which enables hosts to open their house to refugees, travelers and people who are looking for shelter after natural calamity. They have also expanded to business travelers along with multiple amenities. The types of accommodation they offer are for different purpose like vacation budget rooms, luxury rooms and long term rental accommodations. There are multiple new segments started by Airbnb in the last few years making them one of the fastest growing companies. Also leading in the art of storytelling with their variety of data. There are various factors like location, ratings etc. which influence the rate and availability of these rental accommodation.

Introduction

For this project, we have selected Airbnb’s Seattle region data. It contains multiple variables related to Airbnb’s Seattle rental accommodations. Data set is originally retrieved from Kaggle.com. This project will focus on the analysis of correlation between the price of Airbnb rental accommodation, availability and different factors like location, type of house, amenities provided using Python. The report will also attempt to identifying patterns in Time series analysis over the year, box plots, bar charts which will tell us the average price of the different types of accommodation and few more.

By the end of this analysis, we look forward to understanding how price for the Airbnb rentals varies and what type of accommodations, during what period are more popular. Also, which neighborhoods in Seattle are more popular. The ANOVA and T tests will talk about which of the variables are close or they have significantly different mean.

Data Collection

We start with importing the libraries in python required for this project. Some of them are added as and when required during the analysis.

```
In [74]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as pyplt
         from matplotlib import pyplot, pylab
         import statistics
         from matplotlib.pyplot import *
         from matplotlib import pyplot
         from pandas.tools.plotting import scatter_matrix
         from pandas import Series
         from scipy.stats import pearsonr
         import statsmodels.api as sm
         from statsmodels.formula.api import ols
         from statsmodels.stats.anova import anova_lm
         from sklearn import linear_model as lm
```

After importing libraries, read the csv file.

```
#read the csv file

listings = pd.read_csv("listings.csv")
listings.head()
```

	id	listing_url	scrape_id	last_scraped	name	summary	space	description	experiences_offered	neighbor
0	241032	https://www.airbnb.com/rooms/241032	20160104002432	2016-01-04	Stylish Queen Anne Apartment	NaN	Make your self at home in this charming one-be...	Make your self at home in this charming one-be...	none	
1	953595	https://www.airbnb.com/rooms/953595	20160104002432	2016-01-04	Bright & Airy Queen Anne Apartment	Chemically sensitive? We've removed the irrita...	Beautiful, hypoallergenic apartment in an extr...	Chemically sensitive? We've removed the irrita...	none	wonderfu

1. We must replace the characters in the price ($,) before checking the correlation between the variables [using str.replace()], otherwise it will not give us the result including price.
2. Concatenate and create a data frame of all the variables that we are using in this analysis, to understand Airbnb's strategy for price and availability.

Results of data analysis using Python

Descriptive Statistics:

Correlation: The below image shows that:

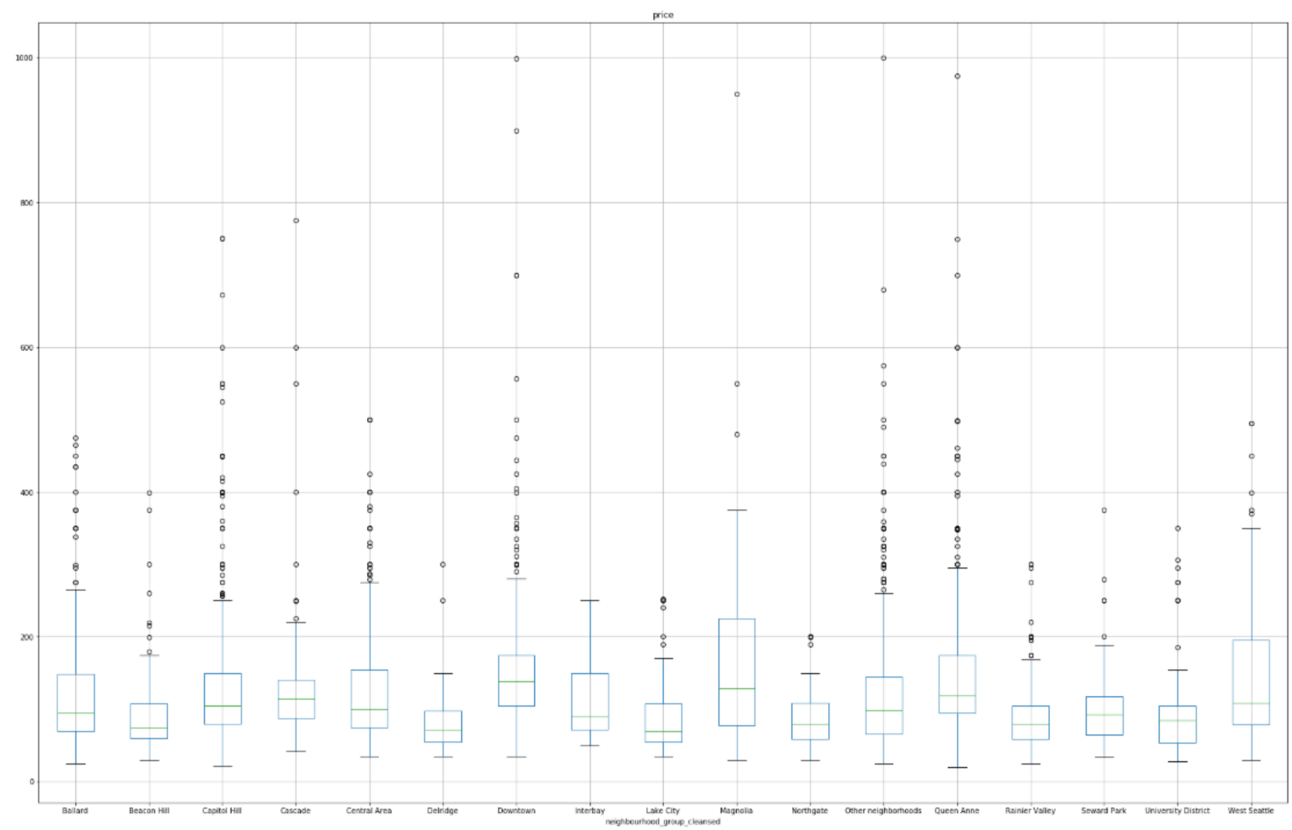
- ⇨ Beds (no. of beds) and Accommodates (no. of people accommodates) have high correlation.
- ⇨ Price and Bedrooms have high correlation.

This shows variables price and accommodates have the highest correlation with availability of number of beds and bedrooms.

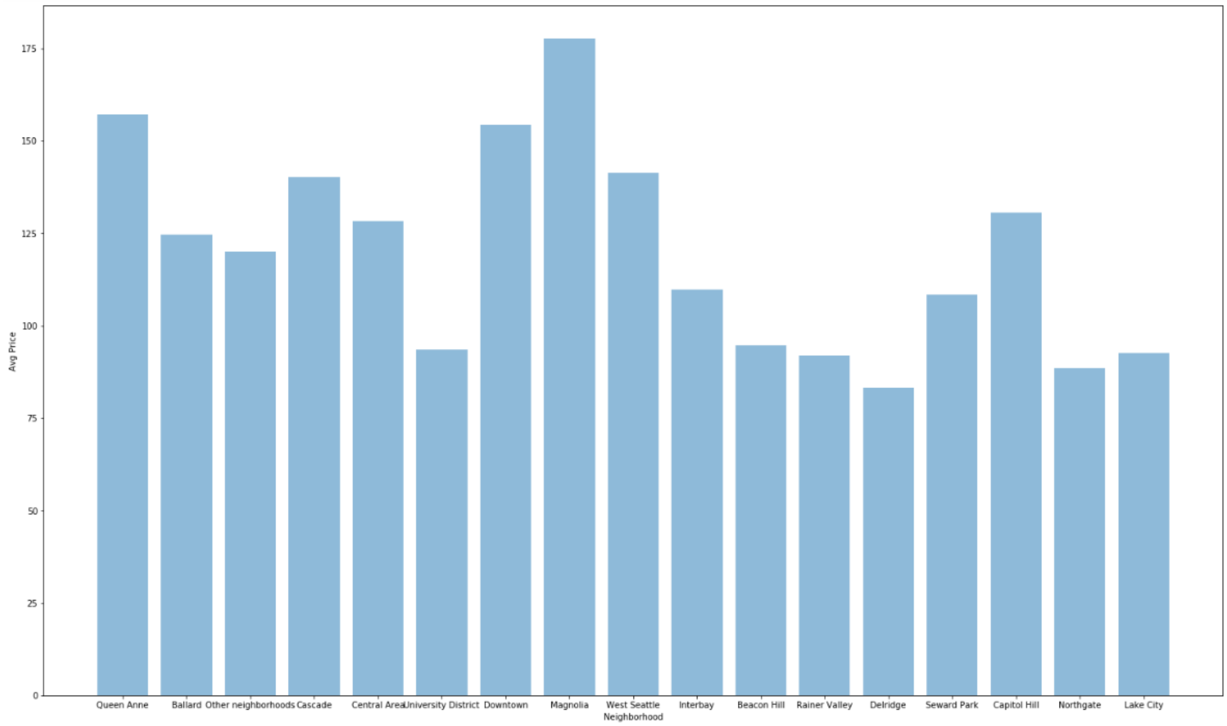
```
list_cor.corr(method='pearson')
```

	accommodates	price	availability_365	number_of_reviews	review_scores_rating	bedrooms	bathrooms	beds
accommodates	1.000000	0.652218	-0.031535	-0.072978	-0.013101	0.770974	0.538439	0.861119
price	0.652218	1.000000	-0.015550	-0.124695	0.055551	0.627720	0.516424	0.589525
availability_365	-0.031535	-0.015550	1.000000	0.094273	-0.038600	-0.049788	-0.002326	-0.009773
number_of_reviews	-0.072978	-0.124695	0.094273	1.000000	0.036242	-0.105555	-0.092147	-0.089077
review_scores_rating	-0.013101	0.055551	-0.038600	0.036242	1.000000	0.023257	0.045101	-0.000720
bedrooms	0.770974	0.627720	-0.049788	-0.105555	0.023257	1.000000	0.610937	0.753167
bathrooms	0.538439	0.516424	-0.002326	-0.092147	0.045101	0.610937	1.000000	0.532838
beds	0.861119	0.589525	-0.009773	-0.089077	-0.000720	0.753167	0.532838	1.000000

Box Plot for different neighborhoods: The boxplot grouped by “Neighborhood_group_cleaned” shows the mean for different neighborhoods in Seattle region. The below boxplot indicates that means are different from each other. Also, there are multiple outliers.



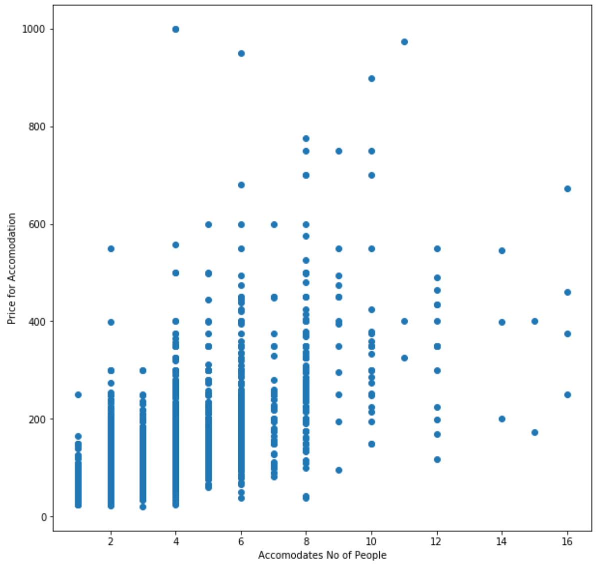
Bar Chart for different neighborhood: Below chart shows us that neighborhood Magnolia has the highest rates and Delridge being the cheapest.

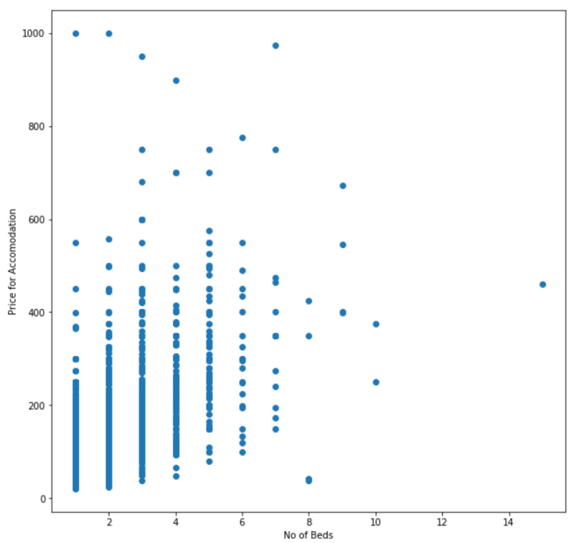


Scatter Plot:

To understand further from correlation that which of the two variables is better correlated we do the scatter plot.

- ⇨ It shows the relationship between variables Price – Accommodates no of people and Price – No of Beds. Price-Accommodates no of people shows a better correlation in scatter plot than No of beds.

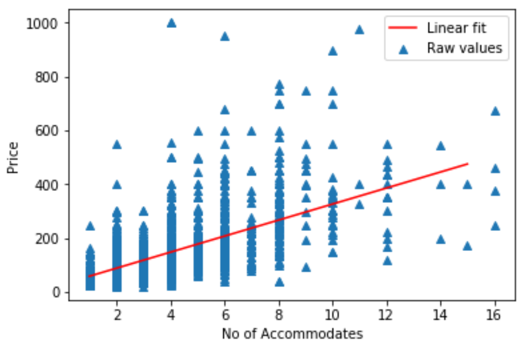




Fitting linear model:

We see there is a better correlation between Price & Accommodates hence we can now fit the linear model and predict price based on the No. of Accommodates. We are using statsmodel’s ols estimator to fit the data and create the best fit line.

```python
# Fitting the model with Price vs Accommodates

mod_linear = ols('price ~ accommodates',listings).fit()

minaccommodates = min(listings['accommodates'])
maxaccommodates = max(listings['accommodates'])
xfit = pd.Series(np.arange(minaccommodates, maxaccommodates, 1),name = 'accommodates')
y_linearfit = mod_linear.predict(xfit)

pyplt.scatter(x,y, marker = "^", label = 'Raw values')
pyplt.plot(xfit,y_linearfit, color='r', label = 'Linear fit')


pyplt.xlabel('No of Accommodates')
pyplt.ylabel('Price')

pyplt.legend()
pyplt.show()

print(mod_linear.summary())
```




```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.425
Model:                            OLS   Adj. R-squared:                  0.425
Method:                 Least Squares   F-statistic:                     2825.
Date:                Wed, 05 Dec 2018   Prob (F-statistic):               0.00
Time:                        18:33:55   Log-Likelihood:                -21550.
No. Observations:                3818   AIC:                         4.310e+04
Df Residuals:                    3816   BIC:                         4.312e+04
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       28.2823      2.178     12.985      0.000      24.012      32.553
accommodates    29.7647      0.560     53.151      0.000      28.667      30.863
==============================================================================
Omnibus:                     2904.778   Durbin-Watson:                   1.848
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           114947.627
Skew:                           3.235   Prob(JB):                         0.00
Kurtosis:                      29.090   Cond. No.                         8.03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

R^2 here accounts for 42.5% variance accounted by the linear model. The lower the variance means farther the data points fall to the fitted line which we see from the scatter plot. Also, the model explains that none or very few of its variability of the response (dependent variable) data is around its mean.

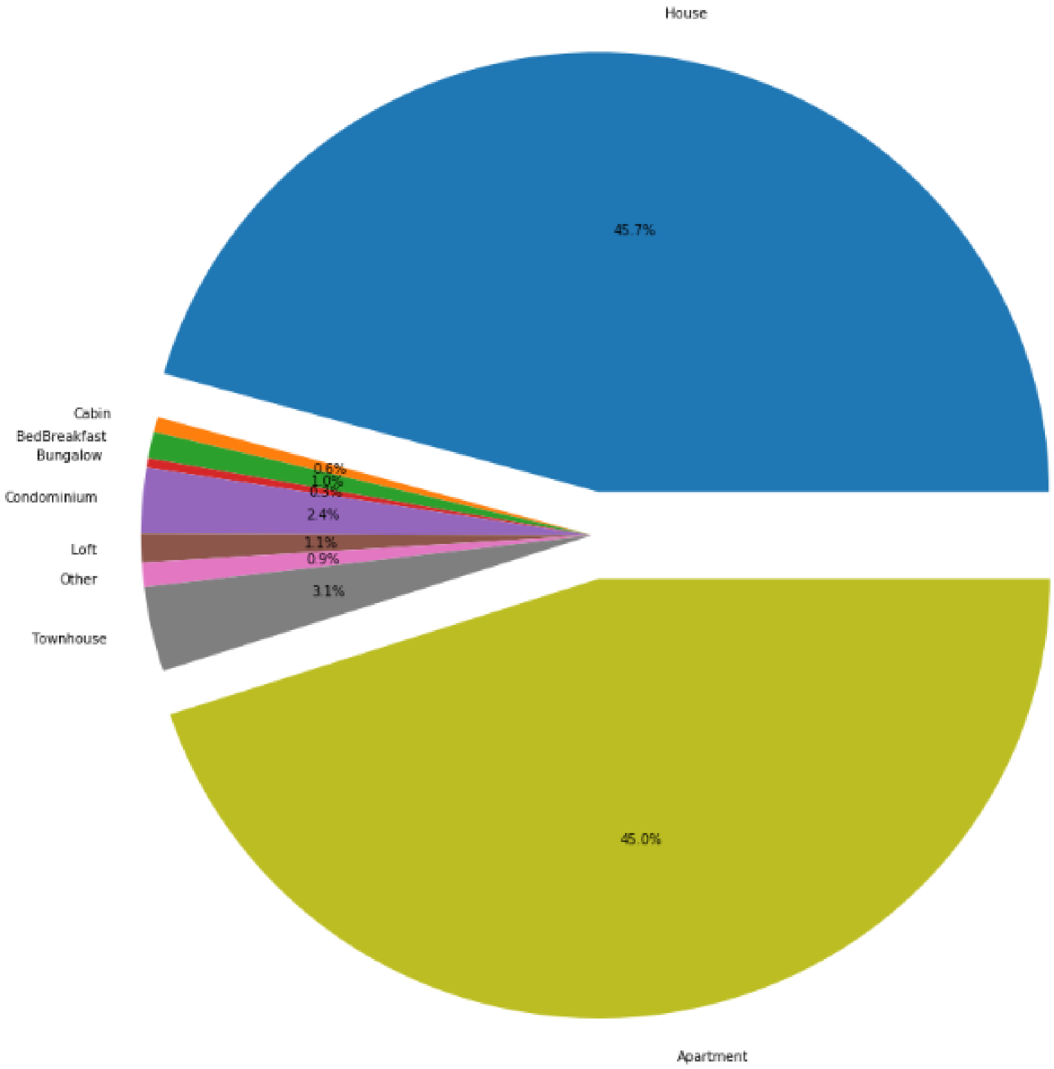
Null hypothesis - H_0: Slope of the fitted line is equal to zero.

Alternate hypothesis - H_1: Slope of the fitted line is not equal to zero.

The low p -value here suggests that slope is not equal to zero and the reject the null hypothesis. So, changes in response variable (y) is related to changes in independent variable (x). Price increases as number of accommodates increases.

Pie Chart: The pie chart explains the type and the proportions of these type of accommodations offered in Airbnb. Here it indicates that House and Apartments are offered the most as compared to Cabin, Loft etc.

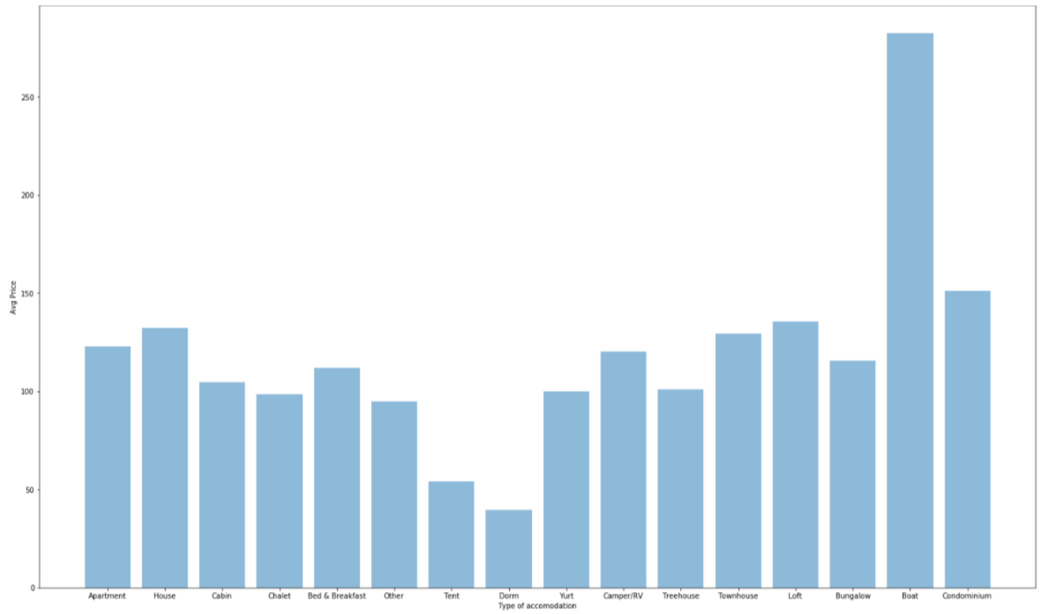
Note: “Other” includes boats, camper/rv, dorm, treehouse, tent, chalet, yurt due to small sample size.



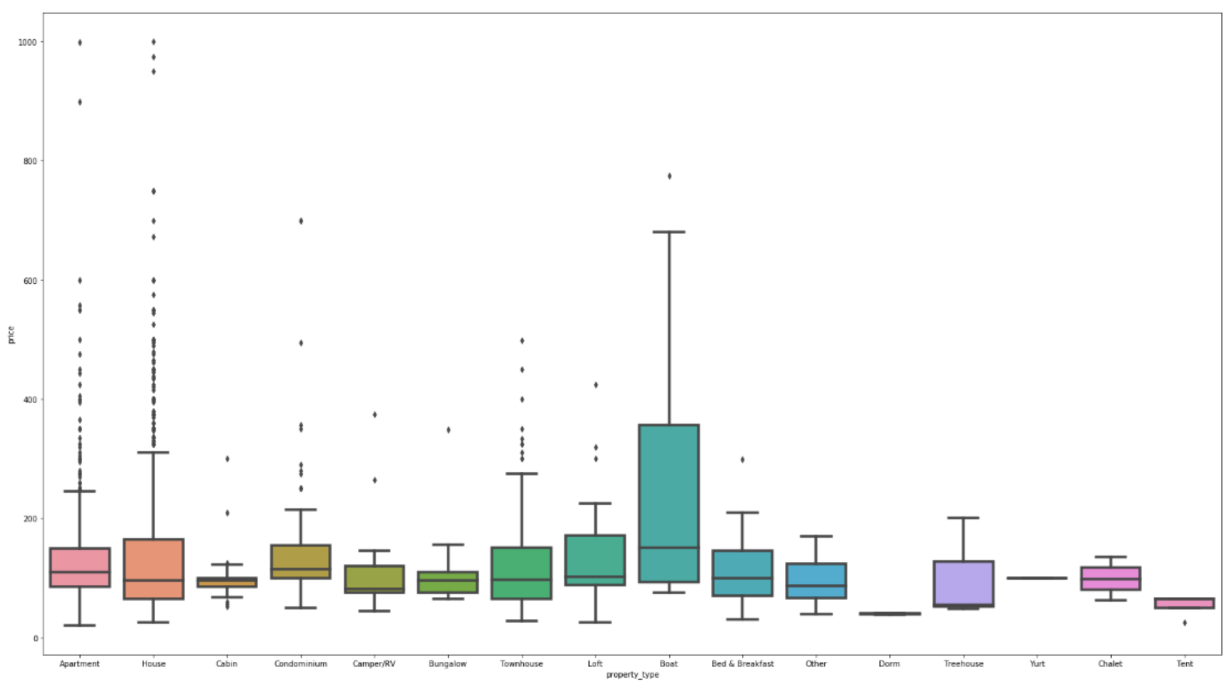
Bar Chart for different property type:

The below bar chart explains the average price for all the types of accommodations available at Airbnb.

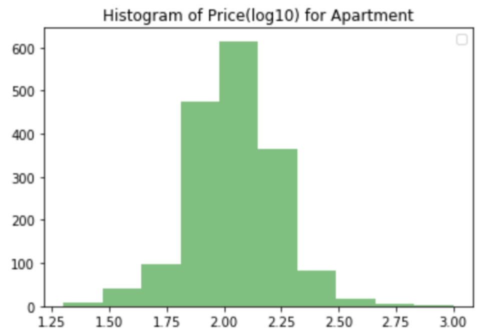
Dorms cost less than $50 per night whereas Boat cost more than $250 per night.

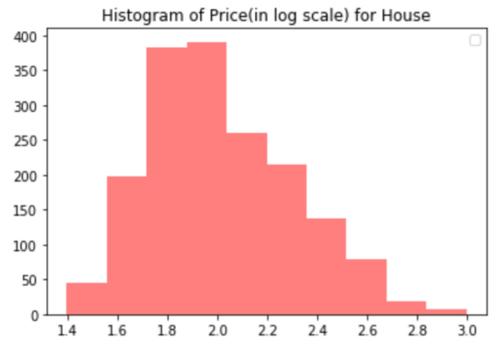


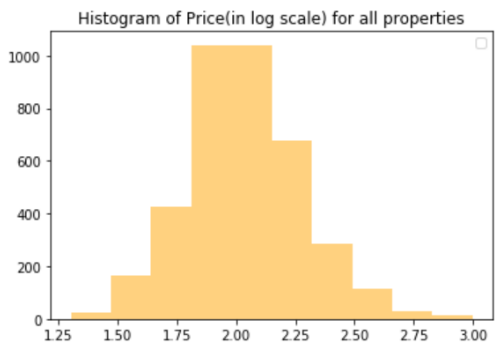
Boxplot for different property type: The boxplot below as shows the mean for all the different accommodations are different from one another.



Histogram:







ANOVA:

We create a data frame of all the variables which are needed to run the ANOVA analysis. Note that we are taking log10 of these variables to make the highly skewed variables less skewed. Axis=1, indicates the axis to concatenate along.

In [186]:
```
dfaov = pd.concat([np.log10(listings['price']),listings['neighbourhood_group_cleansed'],listings['property_type'],
                   listings['accommodates'],listings['room_type']], axis=1)
dfaov.head()
```

Out[186]:

	price	neighbourhood_group_cleansed	property_type	accommodates	room_type
0	1.929419	Queen Anne	Apartment	4	Entire home/apt
1	2.176091	Queen Anne	Apartment	4	Entire home/apt
2	2.989005	Queen Anne	House	11	Entire home/apt
3	2.000000	Queen Anne	Apartment	3	Entire home/apt
4	2.653213	Queen Anne	House	6	Entire home/apt

Now, run the one-way ANOVA with price against “Property_type” and “Neighbourhood_group_cleaned”

```
In [187]: # One way Anova

          model = ols('price ~ C(property_type)+C(neighbourhood_group_cleansed)',dfaov).fit()

          aov = sm.stats.anova_lm(model, typ=2)
          print(aov)
          model.params
          print(model.summary())
```

```
                                     sum_sq      df          F        PR(>F)
C(property_type)                   2.571807    15.0   3.144111  3.749576e-05
C(neighbourhood_group_cleansed)   20.928664    16.0  23.986800  4.254200e-68
Residual                         206.402566  3785.0        NaN           NaN
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.104
Model:                            OLS   Adj. R-squared:                  0.096
Method:                 Least Squares   F-statistic:                     14.10
Date:                Mon, 26 Nov 2018   Prob (F-statistic):           8.31e-69
Time:                        23:40:55   Log-Likelihood:                 151.75
No. Observations:                3817   AIC:                            -239.5
Df Residuals:                    3785   BIC:                            -39.60
Df Model:                          31
Covariance Type:            nonrobust
```

As per the ANOVA results of low p-value and large F value, variation in the means for these two variables are large and significantly different from each other. However, it still does not tell us how price is affected with these two variables.

```
                                                            coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------------------
Intercept                                                 1.9899      0.017    118.711      0.000       1.957       2.023
C(property_type)[T.Bed & Breakfast]                       0.0091      0.039      0.232      0.817      -0.068       0.086
C(property_type)[T.Boat]                                  0.2615      0.084      3.122      0.002       0.097       0.426
C(property_type)[T.Bungalow]                             -0.0005      0.065     -0.008      0.994      -0.128       0.127
C(property_type)[T.Cabin]                                 0.0088      0.052      0.170      0.865      -0.092       0.110
C(property_type)[T.Camper/RV]                             0.0009      0.065      0.014      0.989      -0.127       0.129
C(property_type)[T.Chalet]                               -0.0511      0.166     -0.309      0.758      -0.376       0.273
C(property_type)[T.Condominium]                           0.0312      0.025      1.236      0.217      -0.018       0.081
C(property_type)[T.Dorm]                                 -0.5439      0.165     -3.288      0.001      -0.868      -0.220
C(property_type)[T.House]                                 0.0385      0.009      4.133      0.000       0.020       0.057
C(property_type)[T.Loft]                                  0.0377      0.038      1.005      0.315      -0.036       0.111
C(property_type)[T.Other]                                -0.0781      0.050     -1.554      0.120      -0.177       0.020
C(property_type)[T.Tent]                                 -0.2007      0.106     -1.901      0.057      -0.408       0.006
C(property_type)[T.Townhouse]                             0.0213      0.023      0.930      0.353      -0.024       0.066
C(property_type)[T.Treehouse]                            -0.0781      0.135     -0.578      0.564      -0.343       0.187
C(property_type)[T.Yurt]                                 -0.0500      0.234     -0.213      0.831      -0.509       0.409
C(neighbourhood_group_cleansed)[T.Beacon Hill]           -0.0983      0.027     -3.693      0.000      -0.150      -0.046
C(neighbourhood_group_cleansed)[T.Capitol Hill]           0.0396      0.019      2.130      0.033       0.003       0.076
C(neighbourhood_group_cleansed)[T.Cascade]                0.0671      0.030      2.257      0.024       0.009       0.125
C(neighbourhood_group_cleansed)[T.Central Area]           0.0284      0.020      1.443      0.149      -0.010       0.067
C(neighbourhood_group_cleansed)[T.Delridge]              -0.1369      0.030     -4.489      0.000      -0.197      -0.077
C(neighbourhood_group_cleansed)[T.Downtown]               0.1504      0.020      7.694      0.000       0.112       0.189
C(neighbourhood_group_cleansed)[T.Interbay]              -0.0457      0.073     -0.629      0.529      -0.188       0.097
C(neighbourhood_group_cleansed)[T.Lake City]             -0.1165      0.032     -3.588      0.000      -0.180      -0.053
C(neighbourhood_group_cleansed)[T.Magnolia]               0.1180      0.034      3.504      0.000       0.052       0.184
C(neighbourhood_group_cleansed)[T.Northgate]             -0.1002      0.030     -3.299      0.001      -0.160      -0.041
C(neighbourhood_group_cleansed)[T.Other neighborhoods]   -0.0149      0.018     -0.852      0.394      -0.049       0.019
C(neighbourhood_group_cleansed)[T.Queen Anne]             0.1161      0.021      5.590      0.000       0.075       0.157
C(neighbourhood_group_cleansed)[T.Rainier Valley]        -0.1123      0.024     -4.640      0.000      -0.160      -0.065
C(neighbourhood_group_cleansed)[T.Seward Park]           -0.0597      0.038     -1.552      0.121      -0.135       0.016
C(neighbourhood_group_cleansed)[T.University District]   -0.0969      0.026     -3.674      0.000      -0.149      -0.045
C(neighbourhood_group_cleansed)[T.West Seattle]           0.0601      0.023      2.664      0.008       0.016       0.104
==============================================================================
Omnibus:                      100.320   Durbin-Watson:                   1.841
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              113.297
Skew:                           0.365   Prob(JB):                     2.50e-25
Kurtosis:                       3.423   Cond. No.                         72.4
```

To get a further clear picture of the analysis, we can run the ANOVA again using price against "property_type". Creating data frames for price and the types of property which shows similar yet different average price in the bar chart. (House & Apartment, Condo & Townhouse)

```
dfaovtownhouse = pd.concat([np.log10(propth['price']),propth['property_type']], axis=1)

dfaovcondo = pd.concat([np.log10(propCondo['price']),propCondo['property_type']], axis=1)

dfaovapart = pd.concat([np.log10(propapart['price']),propapart['property_type']], axis=1)

dfaovhouse = pd.concat([np.log10(prophouse['price']),prophouse['property_type']], axis=1)


dfaovtownhousecondo = pd.concat([dfaovtownhouse,dfaovcondo], axis = 0)

dfaovaparthouse = pd.concat([dfaovapart,dfaovhouse], axis = 0)
```

ANOVA with Price and property type – Townhouse and Condominium

```
# One way Anova between townhouse and condo

model1 = ols('price ~ C(property_type)',dfaovtownhousecondo).fit()

aov = sm.stats.anova_lm(model1, typ=2)
print(aov)
model1.params
print(model1.summary())
```

```
                    sum_sq     df         F    PR(>F)
C(property_type)   0.390048    1.0  6.048101  0.014742
Residual          13.349620  207.0       NaN       NaN
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.028
Model:                            OLS   Adj. R-squared:                  0.024
Method:                 Least Squares   F-statistic:                     6.048
Date:                Mon, 26 Nov 2018   Prob (F-statistic):             0.0147
Time:                        23:42:38   Log-Likelihood:                -9.0947
No. Observations:                 209   AIC:                             22.19
Df Residuals:                     207   BIC:                             28.87
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================================
                                     coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
Intercept                          2.1096      0.027     79.245      0.000       2.057       2.162
C(property_type)[T.Townhouse]     -0.0871      0.035     -2.459      0.015      -0.157      -0.017
==============================================================================
Omnibus:                       10.222   Durbin-Watson:                   1.800
Prob(Omnibus):                  0.006   Jarque-Bera (JB):               10.687
Skew:                           0.554   Prob(JB):                      0.00478
Kurtosis:                       3.040   Cond. No.                         2.80
==============================================================================
```

ANOVA with Price and property type – House and Apartment

```
# One way Anova between apartment and house

model2 = ols('price ~ C(property_type)',dfaovaparthouse).fit()

aov = sm.stats.anova_lm(model2, typ=2)
print(aov)
model2.params
print(model2.summary())
```

```
                      sum_sq      df         F    PR(>F)
C(property_type)    0.497538     1.0  8.314709  0.003957
Residual          205.784035  3439.0       NaN       NaN
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     8.315
Date:                Mon, 26 Nov 2018   Prob (F-statistic):            0.00396
Time:                        23:43:13   Log-Likelihood:                -36.452
No. Observations:                3441   AIC:                             76.90
Df Residuals:                    3439   BIC:                             89.19
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
Intercept                     2.0439      0.006    345.322      0.000       2.032       2.056
C(property_type)[T.House]    -0.0240      0.008     -2.884      0.004      -0.040      -0.008
==============================================================================
Omnibus:                      108.233   Durbin-Watson:                   1.650
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              122.063
Skew:                           0.412   Prob(JB):                     3.12e-27
Kurtosis:                       3.416   Cond. No.                         2.63
==============================================================================
```

As we already know the means are different for all the variables, the above ANOVA result indicates that means of Apartment and House is different than the means of Townhouse and Condominium.

T – Tests:

We can also run t tests (for population group of only two variables) to check further the means of variables price and different property types. The result shows that, means for the price of the property types house, apartments, townhouse and condominiums are significantly different with lower p-value like our ANOVA result.

```
In [64]: #t test

         sm.stats.ttest_ind(np.log10(prophouse['price']),np.log10(propapart['price']))

Out[64]: (-2.88352368234328, 0.003956948167749751, 3439.0)
```

```
In [66]: #t test

         sm.stats.ttest_ind(np.log10(propth['price']),np.log10(propCondo['price']))

Out[66]: (-2.4592887313841367, 0.01474167440200136, 207.0)
```

Time Series Analysis:

We start with the time series analysis reading and using the other set of data. Again replace the characters in price ($,) to run the analysis accurately. Also, replacing "t" as 1 and "f" as 0 to separate and check the means of only the listing id's which are available.

```
calendar = pd.read_csv("calendar.csv",parse_dates=["date"],index_col="date")

calendar['price'] = calendar['price'].str.replace("$","")
calendar['price'] = calendar['price'].str.replace(",","")
calendar['price'] = calendar['price'].astype(float)

calendar['available']=calendar['available'].str.replace("t","1")
calendar['available']=calendar['available'].str.replace("f","0")
calendar['available'] = calendar['available'].astype(int)


calendar.head()
```

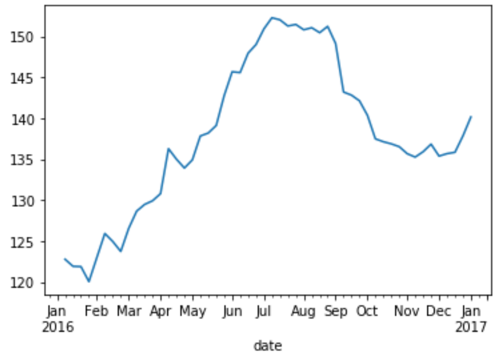
	listing_id	available	price
date			
2016-01-04	241032	1	85.0
2016-01-05	241032	1	85.0
2016-01-06	241032	0	NaN
2016-01-07	241032	0	NaN
2016-01-08	241032	0	NaN

We are only including the data from entire 2016 by specifying the range in the script and ignore Jan 2017 since we only have the data for one month in 2017. 'D' = daily

```
#avg price for 2016
calendarnew = calendar["2016-01-01":"2016-12-31"]
calendarnew

calendarnew.price.resample('D').mean()
```

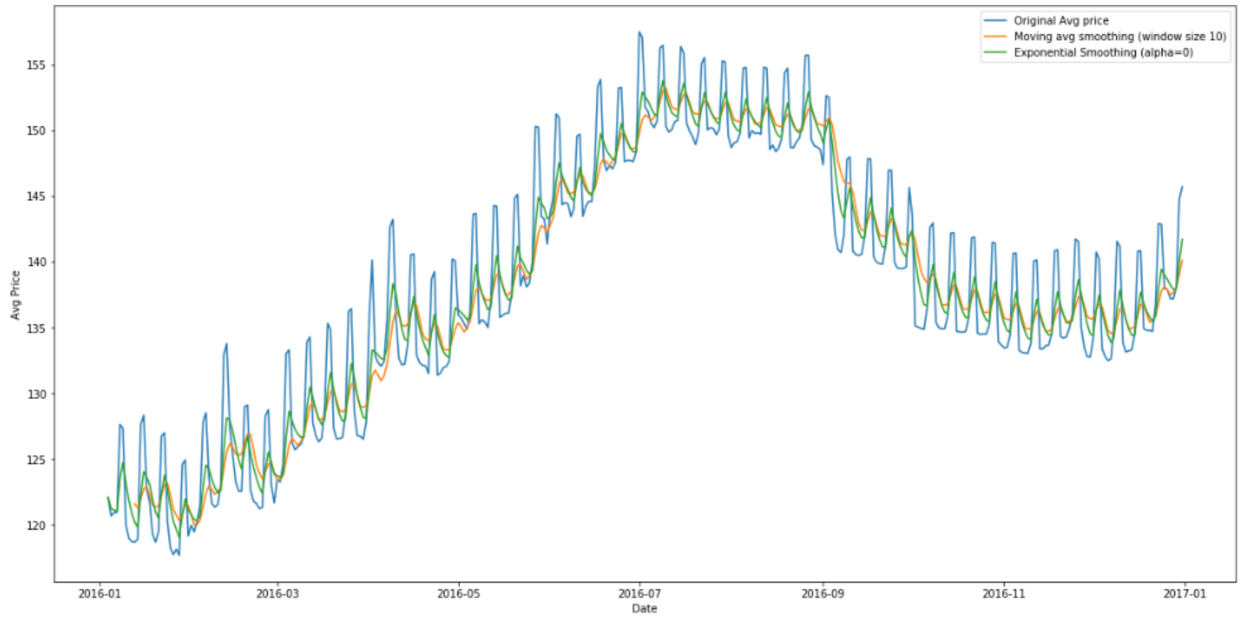
```
#Plotting the avg price over time (Weekly)

%matplotlib inline
calendarnew.price.resample('W').mean().plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23ffac77da0>
```



The above time series plot shows the ‘W’ weekly average price trend for the year 2016. Price is high between July-September. We can also run the ‘D’ daily avg price data and smooth the trend so get a clear idea.

```
#Plotting the avg price over time (Daily)

priceseries = calendarnew.price.resample('D').mean()

pyplt.figure(figsize=(20,10))

pyplt.plot(priceseries, label='Original Avg price')

# Moving avg filter - smoothing original avg price over time
smoothpriceseries = priceseries.rolling(10).mean()

pyplt.plot(smoothpriceseries, label='Moving avg smoothing (window size 10)')


#Exponential Smoothing
expsmoothpriceseries = priceseries.ewm(alpha=0.3).mean()

pyplt.plot(expsmoothpriceseries, label='Exponential Smoothing (alpha=0)')

pyplt.xlabel('Date')
pyplt.ylabel('Avg Price')


addlegend = pyplt.legend()
```



We are using ‘Moving average smoothing’ and ‘Exponential smoothing’ to filter the noise from the data and smooth the time series plot, so that we can look at the trend better. It shows a better trend and increase in average price from end of July 2016 till the beginning of September 2016. Resampling is used when we want the time series in Daily, Weekly or Monthly frequency like what we used here.

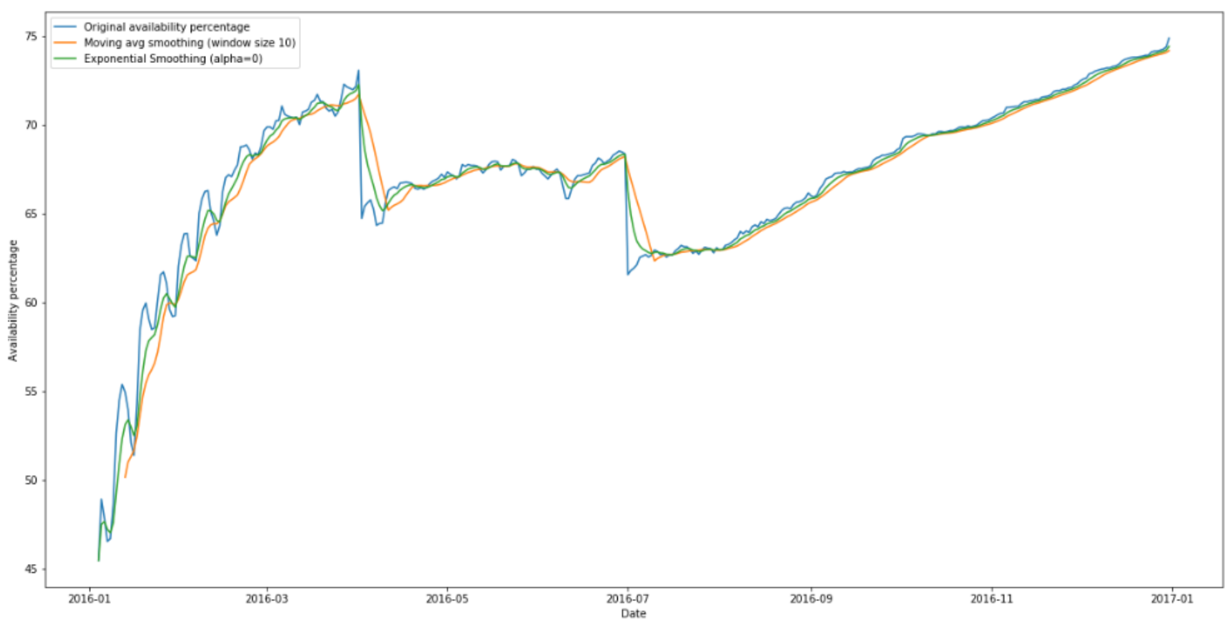
For moving average, we have used window size = 10 i.e. Number of observations used to calculate the moving average value. Higher window has an advantage for less noise in the time series. Alpha for the exponential smoothing should be between 0 to 1 and it is the smoothing factor.

We did the Time series for the average price. Now running the time series for availability.

```
In [78]: calendarnew.available.resample('W').mean()

Out[78]: date
         2016-01-10    0.480843
         2016-01-17    0.537753
         2016-01-24    0.591671
         2016-01-31    0.606264
         2016-02-07    0.633353
         2016-02-14    0.651575
         2016-02-21    0.673501
         2016-02-28    0.685475
         2016-03-06    0.701003
         2016-03-13    0.704333
```

```
#Plotting availability over time

availabilitypercentage = 100*calendarnew.available.resample('D').mean()

pyplt.figure(figsize=(20,10))

pyplt.plot(availabilitypercentage, label='Original availability percentage')

# Moving avg filter - smoothing original availability percentage over time
smoothavailpercent = availabilitypercentage.rolling(10).mean()

pyplt.plot(smoothavailpercent, label='Moving avg smoothing (window size 10)')


#Exponential Smoothing
expsmoothavailpercent = availabilitypercentage.ewm(alpha=0.3).mean()

pyplt.plot(expsmoothavailpercent, label='Exponential Smoothing (alpha=0)')

pyplt.xlabel('Date')
pyplt.ylabel('Availability percentage')


addlegend = pyplt.legend()
```

We run the similar Moving average and Exponential smoothing on the time series for availability. The availability is increased during April 2016 and then declined during July. We already saw prices are high during July-Aug and availability is low, which explains the trend for the year 2016.



Conclusion

After looking at Airbnb's data analysis for the year 2016, we can say no. of bedrooms and no. of beds has the highest correlation with the Airbnb price and accommodates (no. of people accommodating). Price changes as and when no. of accommodates changes or there is high or a smaller number of beds and bedrooms.

As per the pie-chart, House (45.7%) and Apartments (45%) are the largest number of property types that are rented out. Bar chart shows that, average price for the house stands at $132.3 and for the apartments is $123. Dorms are cheapest with average price of $39.5 and Boats are the most expensive at $282.38 per night. Boxplot shows the means for property types and the neighborhoods are different and they have outliers. Bar chart for neighborhoods points out that, Magnolia has the highest rates at $177.6 and Delridge has cheapest accommodations at $83. ANOVA result indicates that, the means of Apartment and House is different than the means of Townhouse and Condominium. Similar results can be seen from T tests as well.

Time series analysis indicates the trend for the availability and the trend for price in Airbnb. End of July and beginning of September looks to be the most expensive duration along with the availability percentage dropping during the same time of the year and picking from September.

Reference

Dataquest. (2017, December 13). Pandas concatenation tutorial. Retrieved from https://www.dataquest.io/blog/pandas-concatenation-tutorial/

Fast Company. (n.d.). Most innovative companies Airbnb. Retrieved from https://www.fastcompany.com/company/airbnb

Jinka, P. (2017, July 22). Exponential smoothing for time series forecasting. Retrieved from https://www.vividcortex.com/blog/exponential-smoothing-for-time-series-forecasting

Kaggle. (n.d.). Seattle Airbnb open data. Retrieved from https://www.kaggle.com /airbnb/seattle/home

Perktold, J., Seabold, S., & Taylor, J. (n.d.). Welcome to statsmodels's documentation. Retrieved from https://www.statsmodels.org/stable/index.html

Python Spot. (n.d.) Python tutorials. Retrieved from https://pythonspot.com/matplotlib-bar-chart/

The Minitab Blog. (2013, July 01). How to interpret regression analysis results: P-values and coefficients. Retrieved from http://blog.minitab.com/blog/adventures-in-statistics-2/how-to-interpret-regression-analysis-results-p-values-and-coefficients

Coding:

```
# coding: utf-8

# In[375]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as pyplt
from matplotlib import pyplot, pylab
import statistics
from matplotlib.pyplot import *
from matplotlib import pyplot
from pandas.tools.plotting import scatter_matrix
from pandas import Series
from scipy.stats import pearsonr
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
from sklearn import linear_model as lm
import seaborn
from matplotlib import rcParams



# In[376]:
#read the csv file

listings = pd.read_csv("listings.csv")
listings.head()
```

```
# In[377]:
#replace the $ to calculate further.

listings['price'] = listings['price'].str.replace("$","")
listings['price'] = listings['price'].str.replace(",","")
listings['price'] = listings['price'].astype(float)
listings['price'].head()


# In[378]:
#Concatinate

data1 = pd.concat([listings['accommodates'],listings['price'],listings['availability_365'],
listings['number_of_reviews'],listings['review_scores_rating'],listings['bedrooms'],listings
['bathrooms'],listings['beds']],axis =1)
data1.head()

# In[379]:
list_cor = pd.DataFrame(data = data1)
list_cor.head()

# In[380]:
type(list_cor['price'])

# In[381]:
list_cor.corr(method='pearson')

# In[382]:
#boxplot for different neighborhood.

listings.boxplot(column='price',by= 'neighbourhood_group_cleansed', figsize=(30,20))
```

```
# In[501]:
#box plot using library seaborn

seaborn.boxplot(x="neighbourhood_group_cleansed", y="price", data=listings,
linewidth=2.5)
rcParams['figure.figsize'] = 23.5,15
pyplt.show()

# In[384]:
# Multivariate plots to check relation between the variables

newlist = list_cor.iloc[:,[0,1,2,3,4,5,6,7]]
axes = pd.plotting.scatter_matrix(newlist, alpha=0.90)
pyplt.tight_layout()
pyplt.show()

# In[385]:
#scatter plot
y=listings['price']
x=listings['accommodates']
pyplt.figure(figsize=(10,10))

pyplt.scatter(x,y)
pyplt.xlabel("Accomodates No of People")
pyplt.ylabel("Price for Accomodation")
pyplt.show()
```

```
# In[386]:
#scatter plot
y1=listings['price']
x1=listings['beds']
pyplt.figure(figsize=(10,10))
pyplt.scatter(x1,y1)
pyplt.xlabel("No of Beds")
pyplt.ylabel("Price for Accomodation")
pyplt.show()

# In[451]:
# Fitting the model with Price vs Accommodates


mod_linear = ols('price ~ accommodates',listings).fit()
minaccommodates = min(listings['accommodates'])
maxaccommodates = max(listings['accommodates'])
xfit = pd.Series(np.arange(minaccommodates, maxaccommodates, 1),name = 
'accommodates')
y_linearfit = mod_linear.predict(xfit)
pyplt.scatter(x,y, marker = "^", label = 'Raw values')
pyplt.plot(xfit,y_linearfit, color='r', label = 'Linear fit')

pyplt.xlabel('No of Accommodates')
pyplt.ylabel('Price')

pyplt.legend()
pyplt.show()
print(mod_linear.summary())
```

```
# In[388]:

proptype = listings['property_type']
total = proptype.size
total



# In[389]:


noofapart = np.sum(proptype.str.count('Apartment'))
print(noofapart)

# In[390]:
noofcabin = np.sum(proptype.str.count('Cabin'))

# In[391]:
noofhouse = np.sum(proptype.str.count('House'))

# In[392]:
noofLoft = np.sum(proptype.str.count('Loft'))

# In[393]:
noofBedBreakfast = np.sum(proptype.str.count('Bed & Breakfast'))

# In[394]:
noofOther = np.sum(proptype.str.count('Other')) +
np.sum(proptype.str.count('Dorm'))+np.sum(proptype.str.count('Treehouse'))+np.sum(pro
ptype.str.count('Tent'))+np.sum(proptype.str.count('Chalet'))
```

```
+np.sum(proptype.str.count('Camper/RV'))+np.sum(proptype.str.count('Boat'))+np.sum(p
roptype.str.count('Yurt'))


# In[395]:
noofBungalow = np.sum(proptype.str.count('Bungalow'))


# In[396]:
noofTownhouse = np.sum(proptype.str.count('Townhouse'))


# In[397]:
noofCondominium = np.sum(proptype.str.count('Condominium'))


# In[398]:
#other includes boats,camper/rv,dorm,treehouse,tent,chalet,yurt due to small sample size.


sizes =
[noofhouse,noofcabin,noofBedBreakfast,noofBungalow,noofCondominium,noofLoft,noo
fOther,
    noofTownhouse,noofapart]


Label =
['House','Cabin','BedBreakfast','Bungalow','Condominium','Loft','Other','Townhouse','Ap
artment']
explode = (0.1,0,0,0,0,0,0,0,0.1)


pyplt.figure(figsize=(15,15))


pyplt.pie(sizes ,labels = Label, explode=explode, autopct='%1.1f%%')
pyplt.show()


# In[399]:
#Calculating mean for thhe price of different property type
```

```
prophouse = listings[listings['property_type'] == 'House']


# In[400]:
np.mean(prophouse['price'])

# In[401]:
propapart = listings[listings['property_type'] == 'Apartment']

# In[402]:
np.mean(propapart['price'])

# In[403]:
proptent = listings[listings['property_type'] == 'Tent']

# In[404]:
np.mean(proptent['price'])

# In[405]:
propbedbrk = listings[listings['property_type'] == 'Bed & Breakfast']


# In[406]:
np.mean(propbedbrk['price'])

# In[407]:
propboat = listings[listings['property_type'] == 'Boat']


# In[408]:
```

```
np.mean(propboat['price'])


# In[409]:
propbungalow = listings[listings['property_type'] == 'Bungalow']


# In[410]:
np.mean(propbungalow['price'])


# In[411]:
propcabin = listings[listings['property_type'] == 'Cabin']


# In[412]:
np.mean(propcabin['price'])


# In[413]:
propchalet = listings[listings['property_type'] == 'Chalet']


# In[414]:
np.mean(propchalet['price'])


# In[415]:
propCamperRV = listings[listings['property_type'] == 'Camper/RV']

# In[416]:
```

```
np.mean(propCamperRV['price'])


# In[417]:
propCondo = listings[listings['property_type'] == 'Condominium']


# In[418]:
np.mean(propCondo['price'])

# In[419]:
propth = listings[listings['property_type'] == 'Townhouse']


# In[420]:
np.mean(propth['price'])


# In[421]:
proploft = listings[listings['property_type'] == 'Loft']

# In[422]:
np.mean(proploft['price'])


# In[423]:
propdorm = listings[listings['property_type'] == 'Dorm']


# In[424]:
np.mean(propdorm['price'])
```

In[425]:

```
proptreeh = listings[listings['property_type'] == 'Treehouse']
```

In[426]:

```
np.mean(proptreeh['price'])
```

In[427]:

```
propyurt = listings[listings['property_type'] == 'Yurt']
```

In[428]:

```
np.mean(propyurt['price'])
```

In[429]:

```
propoth = listings[listings['property_type'] == 'Other']
```

In[430]:

```
np.mean(propoth['price'])
```

```
# In[431]:
# Bar chart for different property type.

typeaccomodation = ('Apartment', 'House', 'Cabin', 'Chalet', 'Bed & Breakfast', 'Other',
'Tent','Dorm','Yurt','Camper/RV','Treehouse',
          'Townhouse','Loft','Bungalow','Boat','Condominium')
series = np.arange(len(typeaccomodation))

meanprice =
[np.mean(propapart['price']),np.mean(prophouse['price']),np.mean(propcabin['price']),np.
mean(propchalet['price']),

np.mean(propbedbrk['price']),np.mean(propoth['price']),np.mean(proptent['price']),np.me
an(propdorm['price']),

np.mean(propyurt['price']),np.mean(propCamperRV['price']),np.mean(proptreeh['price']),
np.mean(propth['price']),

np.mean(proploft['price']),np.mean(propbungalow['price']),np.mean(propboat['price']),np.
mean(propCondo['price'])]

pyplt.figure(figsize=(25,15))

pyplt.bar(series, meanprice, align='center', alpha=0.5)
pyplt.xticks(series, typeaccomodation)

pyplt.xlabel('Type of accomodation')
pyplt.ylabel('Avg Price')
pyplt.show()
```

```
# In[468]:
queenan = listings[listings['neighbourhood_group_cleansed']=='Queen Anne']
np.mean(queenan['price'])


# In[469]:
ballard = listings[listings['neighbourhood_group_cleansed']=='Ballard']
np.mean(ballard['price'])

# In[470]:
others = listings[listings['neighbourhood_group_cleansed']=='Other neighborhoods']
np.mean(others['price'])


# In[471]:

cascade = listings[listings['neighbourhood_group_cleansed']=='Cascade']
np.mean(cascade['price'])


# In[472]:

central = listings[listings['neighbourhood_group_cleansed']=='Central Area']
np.mean(central['price'])


# In[473]:

university = listings[listings['neighbourhood_group_cleansed']=='University District']
np.mean(university['price'])
```

In[474]:

```
downtown = listings[listings['neighbourhood_group_cleansed']=='Downtown']
np.mean(downtown['price'])
```

In[475]:

```
magnolia = listings[listings['neighbourhood_group_cleansed']=='Magnolia']
np.mean(magnolia['price'])
```

In[476]:

```
wseattle = listings[listings['neighbourhood_group_cleansed']=='West Seattle']
np.mean(wseattle['price'])
```

In[477]:

```
Ibay = listings[listings['neighbourhood_group_cleansed']=='Interbay']
np.mean(Ibay['price'])
```

In[478]:

```
bhill = listings[listings['neighbourhood_group_cleansed']=='Beacon Hill']
np.mean(bhill['price'])
```

In[479]:

```
rainier = listings[listings['neighbourhood_group_cleansed']=='Rainier Valley']
np.mean(rainier['price'])


# In[480]:

delr = listings[listings['neighbourhood_group_cleansed']=='Delridge']
np.mean(delr['price'])


# In[481]:

seward = listings[listings['neighbourhood_group_cleansed']=='Seward Park']
np.mean(seward['price'])


# In[482]:

chill = listings[listings['neighbourhood_group_cleansed']=='Capitol Hill']
np.mean(chill['price'])


# In[483]:

ngate = listings[listings['neighbourhood_group_cleansed']=='Northgate']
np.mean(ngate['price'])



# In[484]:
```

```
lake = listings[listings['neighbourhood_group_cleansed']=='Lake City']
np.mean(lake['price'])


# In[485]:

# Bar chart for different neighbourhoods

area = ('Queen Anne', 'Ballard','Other neighborhoods','Cascade', 'Central Area', 'University District', 'Downtown', 'Magnolia','West Seattle','Interbay',
        'Beacon Hill','Rainer Valley','Delridge','Seward Park','Capitol Hill','Northgate','Lake City')

series = np.arange(len(area))

meanpricearea = [np.mean(queenan['price']),np.mean(ballard['price']),np.mean(others['price']),np.mean(cascade['price']),

np.mean(central['price']),np.mean(university['price']),np.mean(downtown['price']),np.mean(magnolia['price']),

np.mean(wseattle['price']),np.mean(Ibay['price']),np.mean(bhill['price']),np.mean(rainier['price']),

np.mean(delr['price']),np.mean(seward['price']),np.mean(chill['price']),np.mean(ngate['price']),np.mean(lake['price'])]

pyplt.figure(figsize=(25,15))

pyplt.bar(series, meanpricearea, align='center', alpha=0.5)
pyplt.xticks(series, area)
```

```
pyplt.xlabel('Neighborhood')
pyplt.ylabel('Avg Price')
pyplt.show()
# In[498]:

#Boxplot for property type

seaborn.boxplot(x="property_type", y="price", data=listings, linewidth=3.5)
rcParams['figure.figsize'] = 27.5,15.5
pyplt.show()



# In[433]:


dfaov = 
pd.concat([np.log10(listings['price']),listings['neighbourhood_group_cleansed'],listings['p
roperty_type'],
                 listings['accommodates'],listings['room_type']], axis=1)
dfaov.head()



# In[434]:


# One way Anova


model = ols('price ~ C(property_type)+C(neighbourhood_group_cleansed)',dfaov).fit()


aov = sm.stats.anova_lm(model, typ=2)
print(aov)
model.params
print(model.summary())
```

```
# In[435]:

dfaovtownhouse = pd.concat([np.log10(propth['price']),propth['property_type']], axis=1)

dfaovcondo = pd.concat([np.log10(propCondo['price']),propCondo['property_type']],
axis=1)

dfaovapart = pd.concat([np.log10(propapart['price']),propapart['property_type']], axis=1)

dfaovhouse = pd.concat([np.log10(prophouse['price']),prophouse['property_type']],
axis=1)

dfaovtownhousecondo = pd.concat([dfaovtownhouse,dfaovcondo], axis = 0)

dfaovaparthouse = pd.concat([dfaovapart,dfaovhouse], axis = 0)


# In[436]:

# One way Anova between townhouse and condo

model1 = ols('price ~ C(property_type)',dfaovtownhousecondo).fit()

aov = sm.stats.anova_lm(model1, typ=2)
print(aov)
model1.params
print(model1.summary())
```

```
# In[437]:

# One way Anova between apartment and house
model2 = ols('price ~ C(property_type)',dfaovaparthouse).fit()
aov = sm.stats.anova_lm(model2, typ=2)
print(aov)
model2.params
print(model2.summary())


# In[438]:

#t test
sm.stats.ttest_ind(np.log10(prophouse['price']),np.log10(propapart['price']))

# In[439]:

#t test
sm.stats.ttest_ind(np.log10(propth['price']),np.log10(propCondo['price']))

# In[459]:

nbins = 5
n, bins, patches = pyplt.hist(dfaovapart['price'], facecolor='green', alpha=0.5)
title("Histogram of Price(in log scale) for Apartment")
legend()
pyplt.show()
```

```
# In[458]:

nbins = 5
n, bins, patches = pyplt.hist(dfaovhouse['price'], facecolor='red', alpha=0.5)
title("Histogram of Price(in log scale) for House")
legend()
pyplt.show()


# In[456]:

nbins = 5
n, bins, patches = pyplt.hist(np.log10(listings['price']), facecolor='orange', alpha=0.5)
title("Histogram of Price(in log scale) for all properties")
legend()
pyplt.show()


# In[444]:

calendar = pd.read_csv("calendar.csv",parse_dates=["date"],index_col="date")
calendar['price'] = calendar['price'].str.replace("$","")
calendar['price'] = calendar['price'].str.replace(",","")
calendar['price'] = calendar['price'].astype(float)

calendar['available']=calendar['available'].str.replace("t","1")
calendar['available']=calendar['available'].str.replace("f","0")
calendar['available'] = calendar['available'].astype(int)

calendar.head()
```

```
# In[445]:

#avg price for 2016
calendarnew = calendar["2016-01-01":"2016-12-31"]
calendarnew

calendarnew.price.resample('D').mean()


# In[446]:
#Plotting the avg price over time (Weekly)

get_ipython().run_line_magic('matplotlib', 'inline')
calendarnew.price.resample('W').mean().plot()


# In[447]:

#Plotting the avg price over time (Daily)

priceseries = calendarnew.price.resample('D').mean()

pyplt.figure(figsize=(20,10))

pyplt.plot(priceseries, label='Original Avg price')

# Moving avg filter - smoothing original avg price over time
smoothpriceseries = priceseries.rolling(10).mean()

pyplt.plot(smoothpriceseries, label='Moving avg smoothing (window size 10)')
```

```
#Exponential Smoothing
expsmoothpriceseries = priceseries.ewm(alpha=0.3).mean()

pyplt.plot(expsmoothpriceseries, label='Exponential Smoothing (alpha=0)')

pyplt.xlabel('Date')
pyplt.ylabel('Avg Price')
addlegend = pyplt.legend()


# In[448]:

calendarnew.available.resample('W').mean()


# In[449]:

get_ipython().run_line_magic('matplotlib', 'inline')
availabilitypercentage = 100*calendarnew.available.resample('W').mean()
availabilitypercentage.plot()


# In[450]:

#Plotting availability over time

availabilitypercentage = 100*calendarnew.available.resample('D').mean()

pyplt.figure(figsize=(20,10))
```

```
pyplt.plot(availabilitypercentage, label='Original availability percentage')

# Moving avg filter - smoothing original availability percentage over time
smoothavailpercent = availabilitypercentage.rolling(10).mean()

pyplt.plot(smoothavailpercent, label='Moving avg smoothing (window size 10)')


#Exponential Smoothing
expsmoothavailpercent = availabilitypercentage.ewm(alpha=0.3).mean()

pyplt.plot(expsmoothavailpercent, label='Exponential Smoothing (alpha=0)')

pyplt.xlabel('Date')
pyplt.ylabel('Availability percentage')

addlegend = pyplt.legend()
```