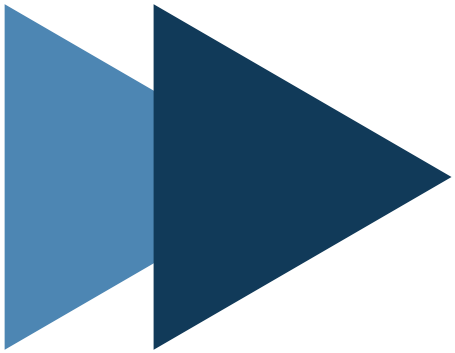
Analyzing Airbnb Seattle

Prepared by: Shantam Mogali, Jason Pastoor, Shubham Sharma, Ashish Tomar, Ruth Yang

Date: 07/30/2019



Agenda

- Business Context Overview
- Acquiring Data
- High Level Overview
- Data Processing
- Exploratory Data Analysis
- Modeling
- Conclusions

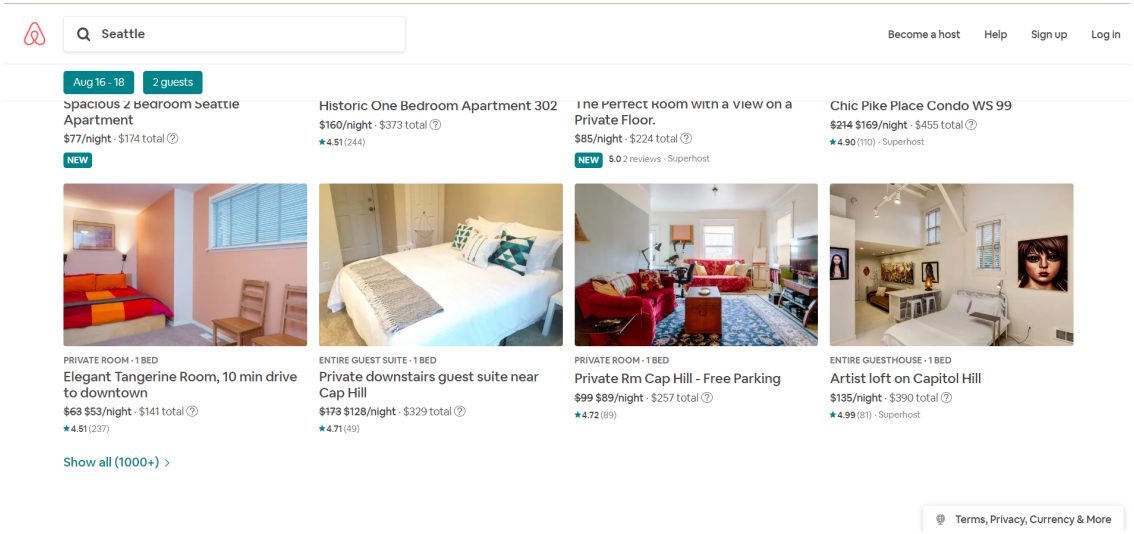
About Airbnb

Part of the ‘sharing economy’

Two-sided platform for short-term lodging

One of the fastest growing companies

A disruptive innovation



About the Dataset

Retrieved from http://insideairbnb.com/
(Seattle, Washington, United States)

Up-to-date as of July 14, 2019

Raw data scraped from Airbnb

Available for public use for individual cities

INSIGHTS ON...

What are the busiest times of each season to visit Seattle?

How do prices vary across different neighborhoods?

What other factors cause variability in prices?

What is the predicted price for each listing?

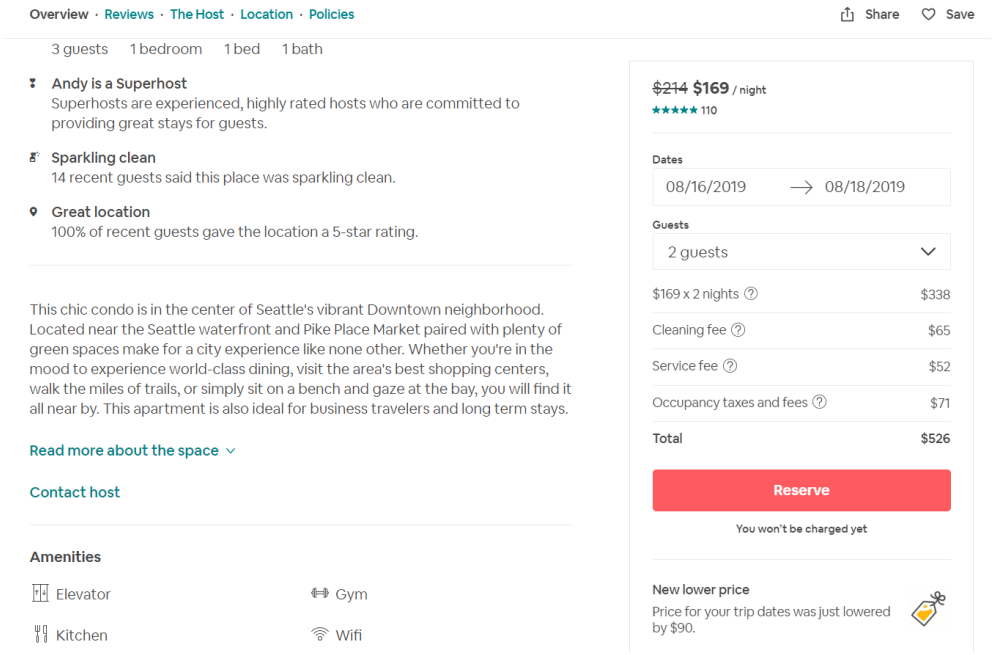
Recommendations

Dataset Overview

```
listings.columns
```

```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'summary',
       'space', 'description', 'experiences_offered', 'neighborhood_overview',
       ...
       'instant_bookable', 'is_business_travel_ready', 'cancellation_policy',
       'require_guest_profile_picture', 'require_guest_phone_verification',
       'calculated_host_listings_count',
       'calculated_host_listings_count_entire_homes',
       'calculated_host_listings_count_private_rooms',
       'calculated_host_listings_count_shared_rooms', 'reviews_per_month'],
      dtype='object', length=106)
```

- Host information
- Property information
- Booking information



Dataset Overview (Cont.)

Checking for nulls in the overall dataset

```
thumbnail_url              8921
medium_url                 8921
host_acceptance_rate       8921
xl_picture_url             8921
square_feet                8565
                           ...
minimum_nights                0
maximum_nights                0
minimum_minimum_nights        0
maximum_minimum_nights        0
id                            0
Length: 106, dtype: int64
```

```
listings.isna().sum().sort_values(ascending=False)
```

executed in 74ms, finished 16:32:46 2019-07-27

- Dropping multiple columns which have nothing but nulls
- Removing columns which would not help in evaluating the result such URL links, picture sizes, descriptions, etc.

Data Cleaning & Preparation Overview

1. Cleaning price values & other columns
2. Handling date conversions and new columns
3. Managing missing data
4. Understanding and removing outliers
5. Feature engineering: Amenities, Property Type & Room Type
6. Feature engineering: Distance from Downtown

DATA CLEANING

```
cal['price']=cal['price'].str.replace("$","").str.replace(",","").astype('float')
listings['price']=listings['price'].str.replace("$","").str.replace(",","").astype('float')
listings['extra_people']=listings['extra_people'].str.replace("$","").str.replace(",","").astype('float')
listings['cleaning_fee']=listings['cleaning_fee'].str.replace("$","").str.replace(",","").astype('float')
listings['security_deposit']=listings['security_deposit'].str.replace("$","").str.replace(",","").astype('float')
listings['host_acceptance_rate']=listings['host_acceptance_rate'].replace("%","").replace(",","").astype('float')
listings['host_response_rate']=listings['host_response_rate'].str.replace("%","").str.replace(",","").astype('float')
listings.amenities = listings.amenities.str.replace("{","").str.replace("}","").str.replace('"',"")
```

- Remove $ sign from price column
- Remove % sign from rate columns
- Converting strings to floats
- Format amenities column
- Converting strings (‘t’ and ‘f’) to boolean

DATA CLEANING (CONT.)

Formatting dates

```
#To maintain order for month by names
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun","Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
#converting date column to proper date format
cal['date']=pd.to_datetime(cal['date'])
cal['year']=cal['date'].dt.year
cal['month']=cal['date'].dt.month.apply(lambda x: calendar.month_abbr[x])
cal['month'] = pd.Categorical(cal['month'], categories=months, ordered=True) #using months variable here to categorize
cal['day']=cal['date'].dt.day
```

executed in 33.6s, finished 16:32:46 2019-07-27

	listing_id	date
0	2318	2019-07-14
1	564428	2019-07-14
2	564428	2019-07-15
3	564428	2019-07-16
4	564428	2019-07-17

year	month	day
2019	Jul	14
2019	Jul	14
2019	Jul	15
2019	Jul	16
2019	Jul	17

Data Cleaning (Cont.)

Null Value Imputation

```
#listings.host_acceptance_rate.fillna(listings['host_acceptance_rate'].median(), inplace=True)
listings.review_scores_rating.fillna(listings['review_scores_rating'].median(), inplace=True)
listings.reviews_per_month.fillna(listings['reviews_per_month'].median(), inplace=True)
listings.host_response_rate.fillna(listings['host_response_rate'].median(), inplace=True)
```

executed in 13ms, finished 16:32:46 2019-07-27

- Replace null values by median
- Avoid impact of outliers

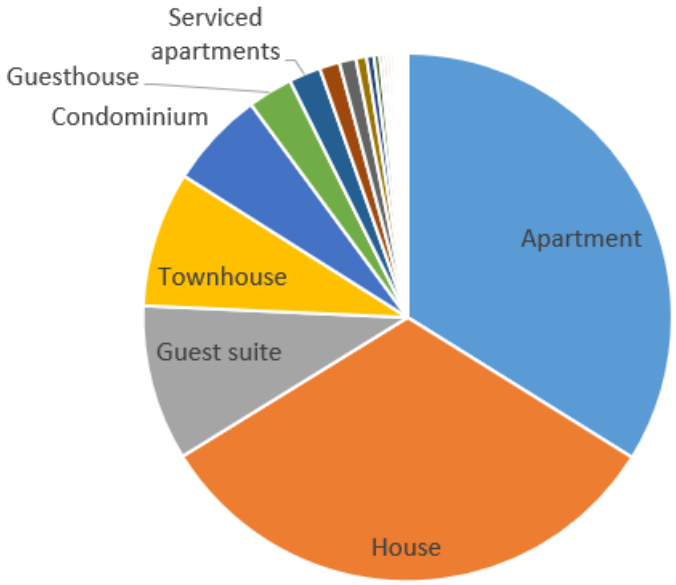
Data Cleaning (Cont.)

Taking a quick glimpse on how the data is split up across property type

```
Apartment                0.338863
House                    0.323058
Guest suite              0.095169
Townhouse                0.082838
Condominium              0.059298
Guesthouse               0.027463
Serviced apartment       0.019392
Loft                     0.012330
Bungalow                 0.010201
Cottage                  0.006502
Aparthotel               0.004484
Bed and breakfast        0.003475
Tiny house               0.002242
Other                    0.002242
Boat                     0.002130
Camper/RV                0.002018
Cabin                    0.001681
Houseboat                0.001457
Villa                    0.000897
Hotel                    0.000897
Dome house               0.000897
Tent                     0.000673
Hostel                   0.000448
Boutique hotel           0.000336
Yurt                     0.000224
Resort                   0.000224
Treehouse                0.000112
Barn                     0.000112
Chalet                   0.000112
Nature lodge             0.000112
Farm stay                0.000112
Name: property_type, dtype: float
```

```
listings.property_type.value_counts(normalize=True)
```

executed in 40ms, finished 16:32:05 2019-07-27



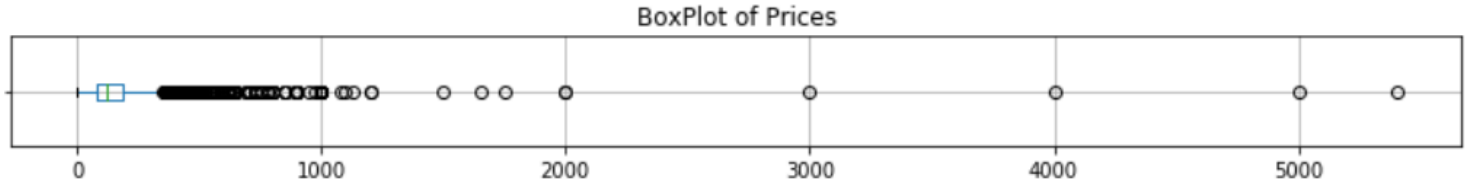
Top 5 property types constitute 90% of listings

```
property_filter = ['Apartment','House','Guest suite','Townhouse','Condominium']
```

DATA CLEANING (CONT.)

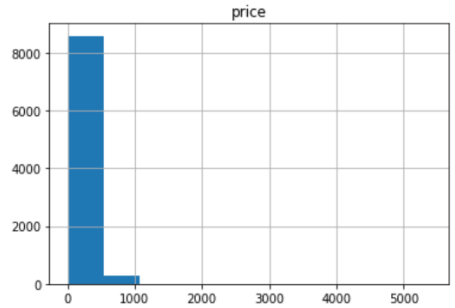
Removing Outliers

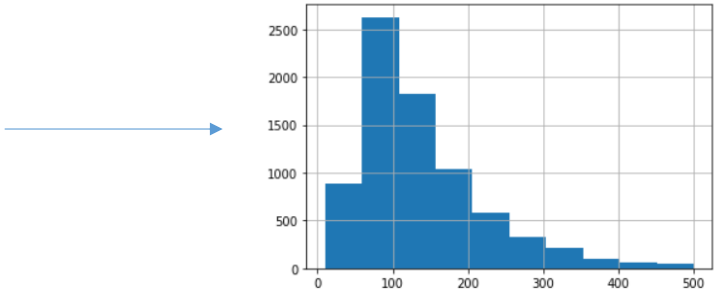
```
fig = plt.figure(figsize=(16,6))
ax = fig.gca()
listings.boxplot(column='price', ax=ax,vert=False)
```




```
listings.query('price<=500 & price>0', inplace=True)
listings.query('property_type in @property_filter', inplace=True)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002042F5A9B70>]],
      dtype=object)
```





DATA PREPARATION

Feature Engineering on Amenities

```
amenities = listings.amenities.str.split(',').tolist()
unqiue_values = np.unique(np.concatenate(amenities))
binarised = [[1 if unique in x else 0 for unique in unqiue_values] for x in amenities]
binarised = pd.DataFrame(binarised, columns=unqiue_values)
```

amenities
Internet,Wifi,Kitchen,Free parking on premises...
TV,Internet,Wifi,Free street parking,Heating,F...
TV,Internet,Wifi,Air conditioning,Kitchen,Free...
Internet,Wifi,Air conditioning,Kitchen,Free pa...
TV,Cable TV,Internet,Wifi,Air conditioning,Kit...

One Hot Encoding →

...	Wheelchair accessible	Wide clearance to shower	Wide doorway to guest bathroom	Wide entrance	Wide entrance for guests	Wide entryway	Wide hallways	Wifi	Window guards
...	0	0	0	0	0	0	0	1	0
...	0	0	0	0	0	0	0	1	0
...	0	0	0	0	0	0	0	1	0
...	0	0	0	0	0	0	0	1	0
...	1	0	0	0	0	0	0	1	0

Same for property type, bed type, neighborhood, room type

Data Preparation (Cont.)

Feature Engineering on Distance to Center of City

```
def dist_to_downtown(lat, lon):
    centre = (47.605034, -122.334765)
    accommodation = (lat, lon)
    return great_circle(centre, accommodation).km

listings['distance'] = listings.apply(lambda x: dist_to_downtown(x.latitude, x.longitude), axis=1)
```

execution queued 15:52:14 2019-07-28

	latitude	longitude	distance
0	47.61082	-122.29082	3.356692
1	47.52398	-122.35989	9.207843
2	47.55062	-122.32014	6.149217
3	47.60801	-122.32874	0.559939
4	47.55539	-122.38474	6.672593

Data Preparation (Cont.)

Normalization

```
x = listings_feature.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
listings_feature = pd.DataFrame(x_scaled,index=listings_feature.index, columns=listings_feature.columns)
```

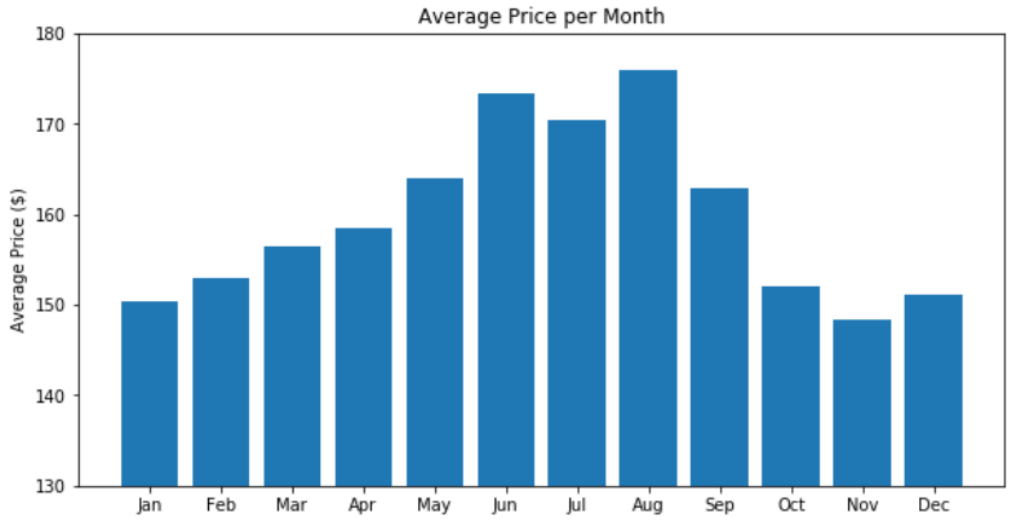
executed in 71ms, finished 16:32:52 2019-07-27

host_response_rate	host_is_superhost	host_listings_count	host_total_listings_count	host_has_profile_pic	accommodates	bathrooms	bedrooms	beds
1.0	1.0	0.001142	0.001142	1.0	0.421053	0.416667	0.571429	0.081633
1.0	1.0	0.000571	0.000571	1.0	0.052632	0.166667	0.000000	0.020408
1.0	1.0	0.004569	0.004569	1.0	0.052632	0.500000	0.142857	0.020408
1.0	1.0	0.002284	0.002284	1.0	0.052632	0.166667	0.142857	0.020408
1.0	1.0	0.001142	0.001142	1.0	0.157895	0.166667	0.285714	0.102041

DESCRIPTIVE ANALYSIS

Price Variation by Time

```
#price variations over the months
price = round(cal.groupby(['month']).mean()["price"],2)
#Plotting graph on price variations over the month
x_pos = np.arange(len(price))
plt.figure(figsize=(10,5))
plt.bar(x_pos,price,align='center')
#plt.plot(price['price'])
plt.ylabel('Average Price ($)')
axes = plt.gca()
axes.set_ylim([130,180])
plt.title('Average Price per Month')
plt.xticks(x_pos,months)
plt.show()
```

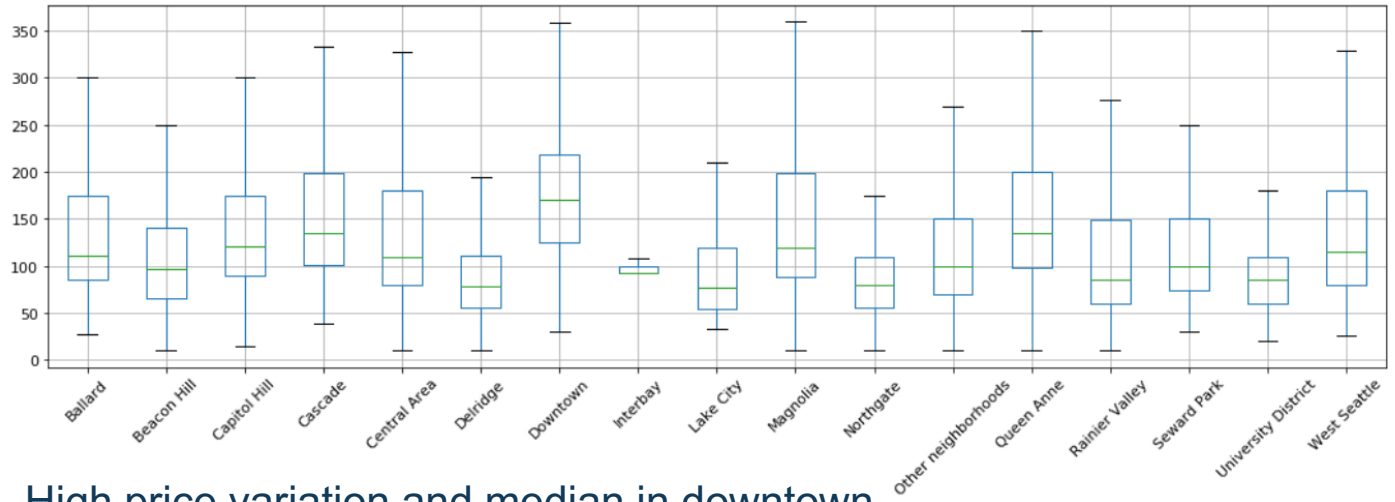


- Demand is highest during the summer months
- People are less likely to visit Seattle during the winters

DESCRIPTIVE ANALYSIS (CONT.)

Price Variation by Neighborhood

```
price_area=listings[['neighbourhood_group_cleansed','price']]
price_area_mean=price_area.groupby('neighbourhood_group_cleansed').mean()
price_area1=price_area.pivot(columns='neighbourhood_group_cleansed',values='price')
plt.figure(figsize=(18,5))
plot3=price_area1.boxplot(showfliers=False,rot=45,fontsize=11)
```



- High price variation and median in downtown
- Low variation in Interbay is because of fewer data points
- Overall median price is $120

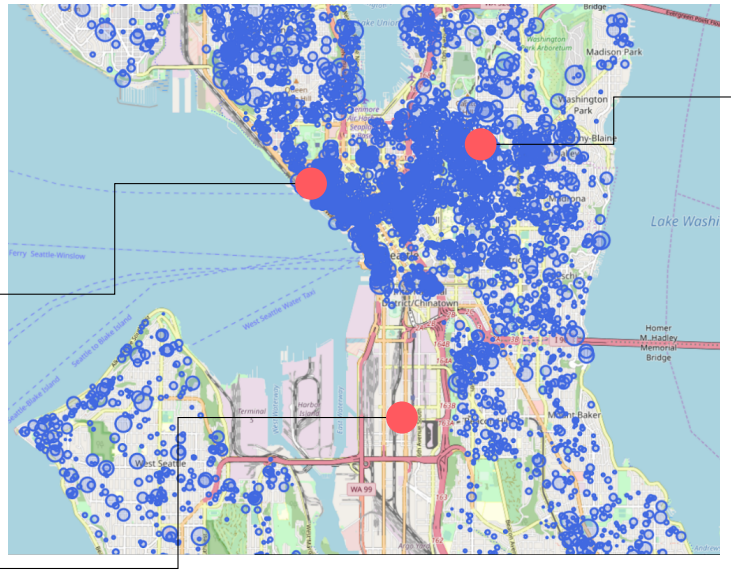
DESCRIPTIVE ANALYSIS (CONT.)

Price Variation by Area & Listing Density

```
#map seattle airbnb
mymap = folium.Map(location = [47.6,-122.3],zoom_start = 14)

for index, row in listings.iterrows():
    folium.Circle(location = [row['latitude'], row['longitude']],
                          radius= row['price']/3,
                          popup=row['price'],
                          fill = True,
                          fill_color = 'royalblue',
                          color="royalblue",
                         ).add_to(mymap)
```

Waterfront: Due to high number of summer events as well as proximity to downtown.(Seattle Great Wheel)

Georgetown: Almost negligible listings in spite of proximity to downtown area, due to it being an industrial area.



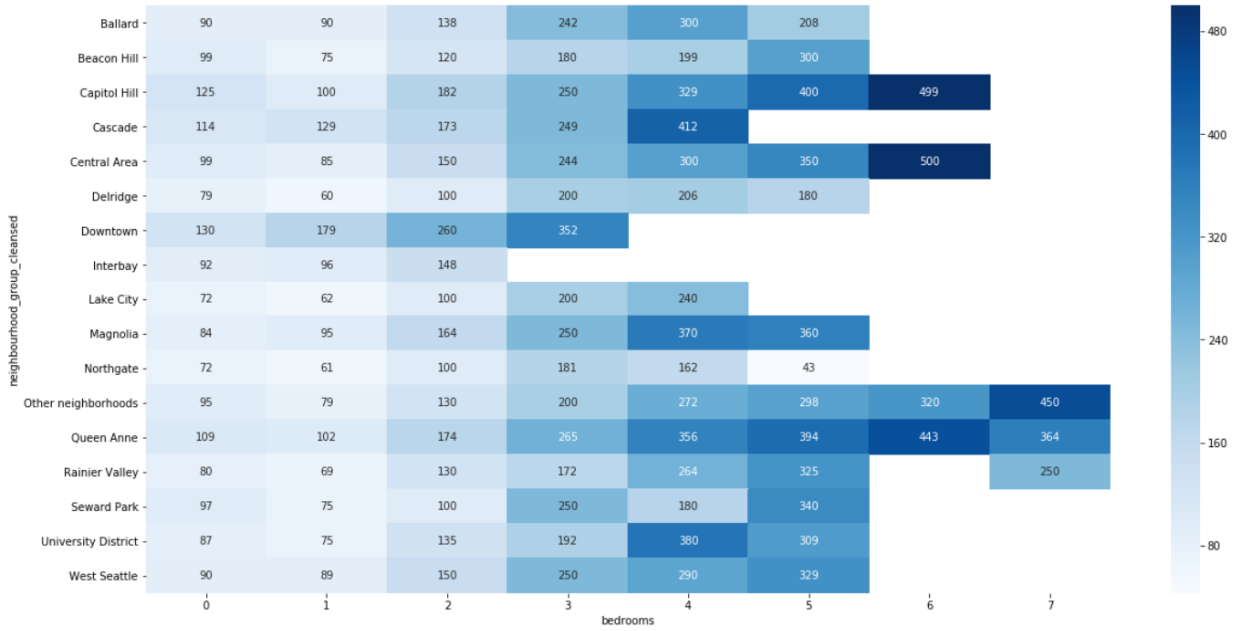
Capitol Hill: Seattle's coolest neighborhood with an abundance of brunch spots, bars and commercial thoroughfares.

Descriptive Analysis (Cont.)

Price variation per bedroom per neighborhood

```
plt.figure(figsize=(20,10))
sn.heatmap(listings.groupby([
        'neighbourhood_group_cleansed', 'bedrooms']).price.median().unstack(),annot=True, fmt=".0f",cmap="Blues")
```

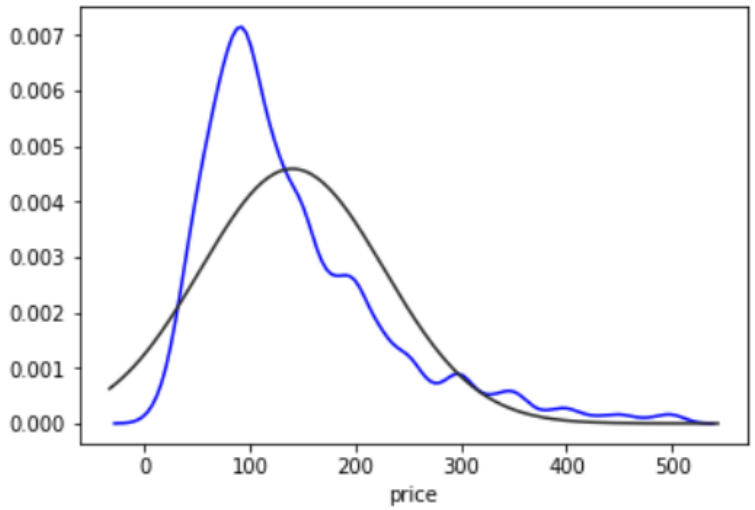
- In Downtown max bedroom available is of 3 bedrooms
- Downtown has high population density & small property size
- Price increases as number of bedrooms increases



Descriptive Analysis (Cont.)

Price Skewness

```
sn.distplot(listings['price'],hist=False,fit=norm,color="blue")
```



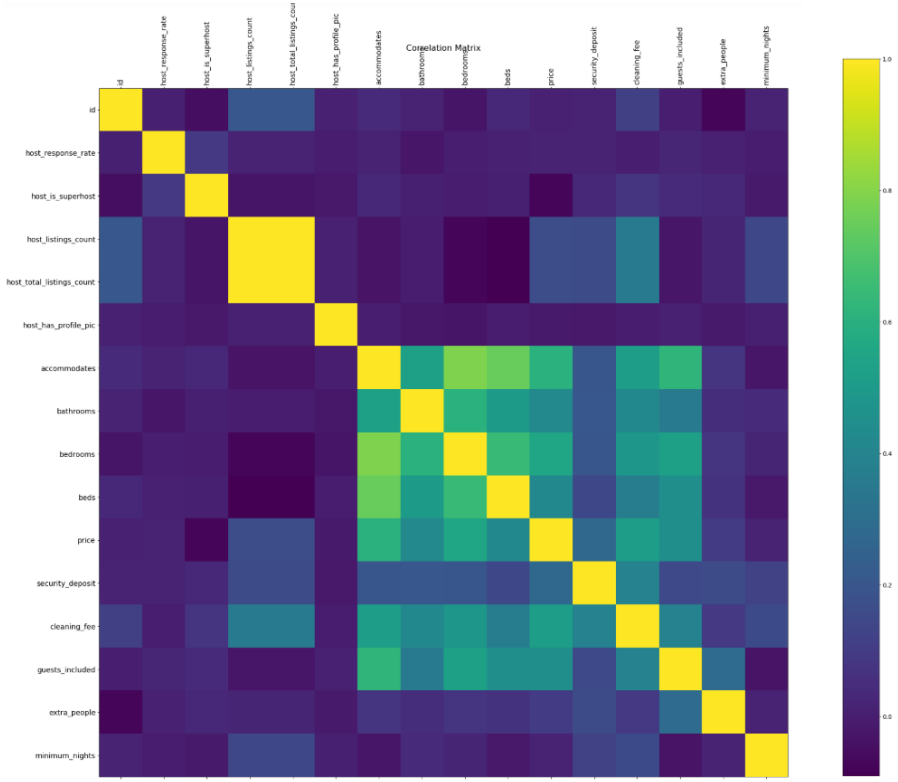
- Right skewed graph
- Most observations occur on the low-medium end, but price can reach very high values.
- Abrupt peaks at common prices such as 200,300,350, etc.

DESCRIPTIVE ANALYSIS (CONT.)

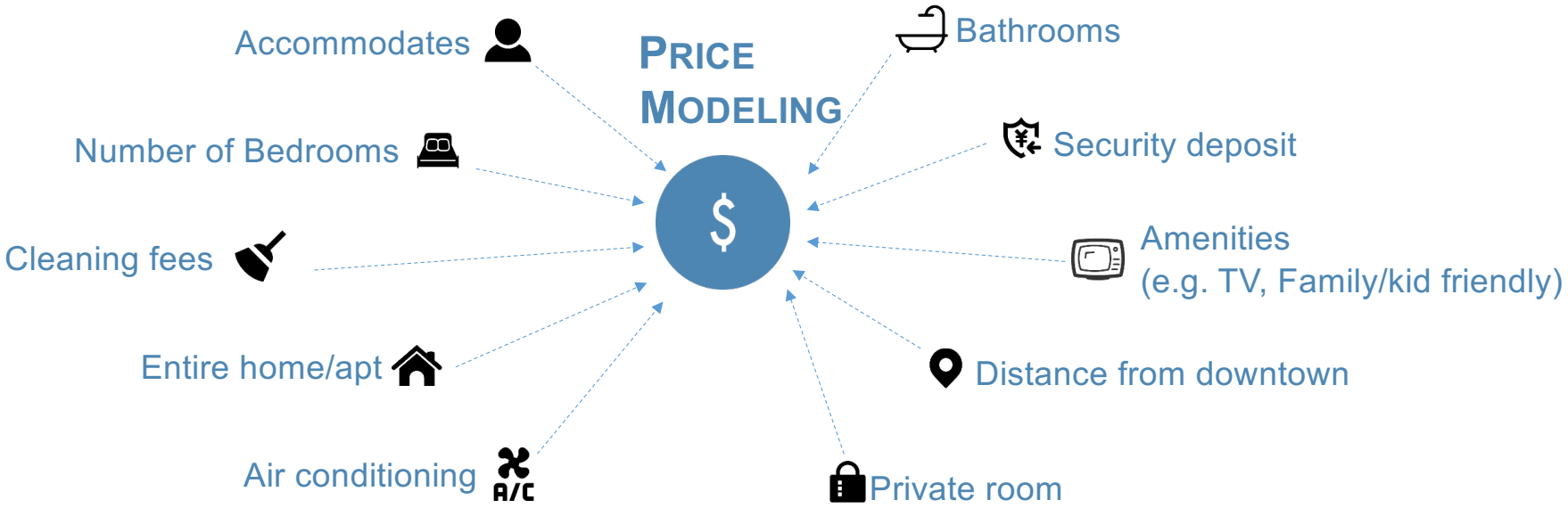
Correlation Matrix

```
listings_feature_corr = listings_feature.loc[:,:'minimum_nights']

f=plt.figure(figsize=(30,30))
plt.matshow(listings_feature_corr.corr(),fignum=f.number)

plt.xticks(range(listings_feature_corr.shape[1]), listings_feature_corr.columns, fontsize=14, rotation=90)
plt.yticks(range(listings_feature_corr.shape[1]), listings_feature_corr.columns, fontsize=14)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16)
plt.show()
```

- Green-Yellow part indicates high correlation
- Bedroom, accommodates, bathroom, security deposit, cleaning fee, etc. are variables that highly correlated with price



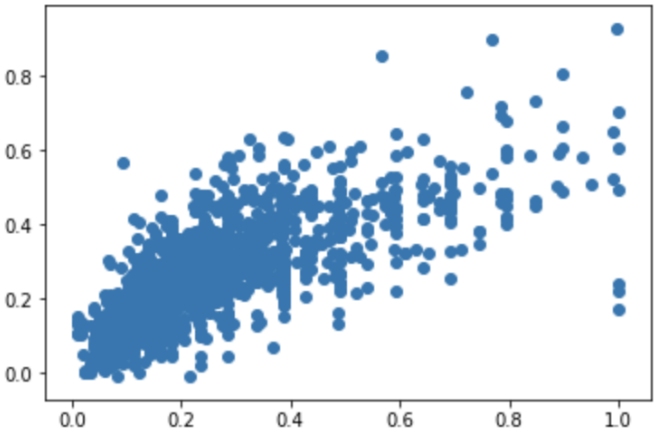
PREDICTIVE ANALYSIS

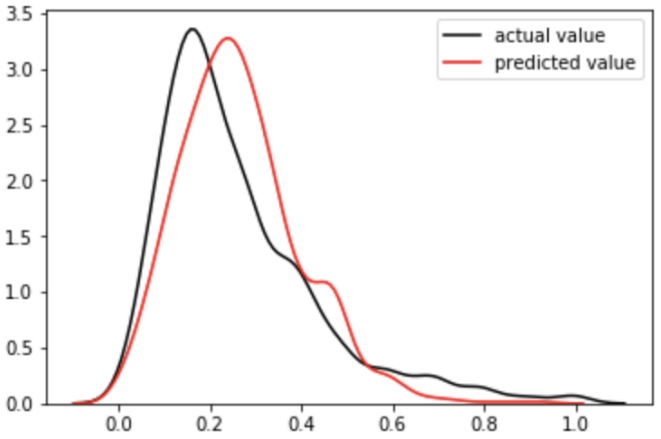


PREDICTIVE ANALYSIS

Linear Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
lm = LinearRegression()
lm.fit(X_train,y_train)
y_pred = lm.predict(X_test)
plt.scatter(y_test,y_pred)
```





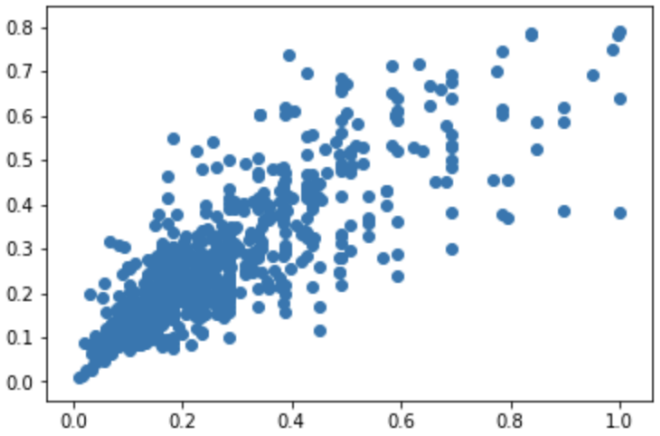
Adjusted R Square is 85.9%
Mean Absolute Error is .083

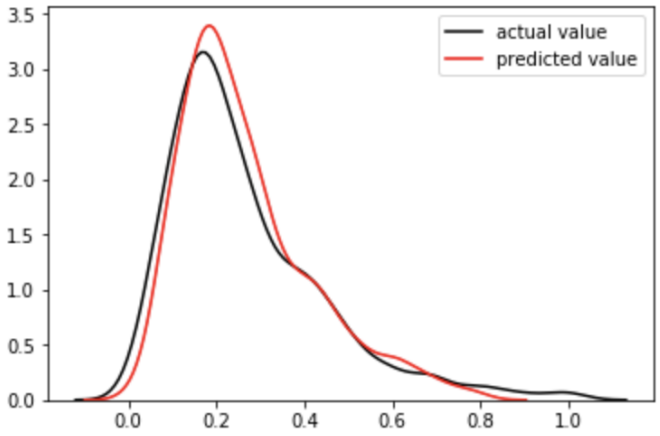
PREDICTIVE ANALYSIS

Random Forest

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.1, random_state = 42)
```

```
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```





Adjusted R Square is 88.2%
Mean absolute error is .067

Recommendations Based on Model

Price = 0.1927*accommodates + 0.4517*bedrooms + 0.2121*cleaning_fee + 0.1559*Entirehome/apt + 0.0168*Air_conditioning + 0.1476*bathrooms + 0.0238*TV + 0.1368*security_deposit + 0.0113*Family/kid_friendly +0.0862*Private_room – 0.2057*distance_from_citycenter

- The model predicts the price based on the above variables.
- The price recommended follows distribution of all the listings across Seattle.
- The price will be in the middle of all the prices for listings with similar variables.

CONCLUSION

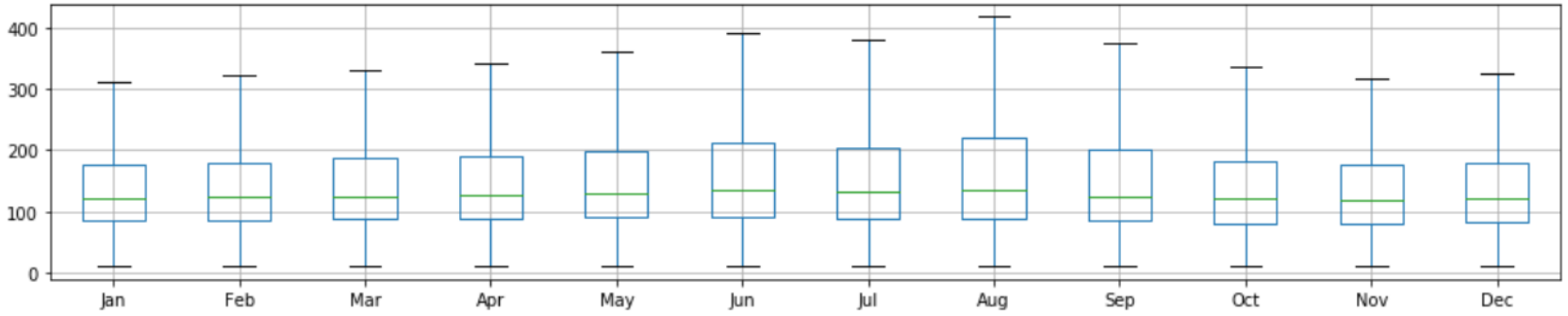
- Adding amenities like AC, family kid friendly and TV can increase the price.
- These results can be used by both sides of this marketplace: to help hosts set prices competitively, and to help guests find the best value listings.
- Based on these variables, hosts can set their listing prices according to the model.
- Model can also be used to advise on whether investing in a new amenity is worth it based on predicted price increase.
- Listings with the greatest negative residuals are the best values based on size, location, amenities, etc.

Questions

APPENDIX A

Price Variation by Time

```
price_month=cal[['month','price']]
price_month_mean=price_month.groupby('month').mean()
price_month1=price_month.pivot(columns='month',values='price')
plt.figure(figsize=(16,3))
plot2=price_month1.boxplot(showfliers=0)
```



APPENDIX B

Linear Regression OLS

OLS Regression Results

Dep. Variable:	price	R-squared:	0.860
Model:	OLS	Adj. R-squared:	0.859
Method:	Least Squares	F-statistic:	850.6
Date:	Sat, 27 Jul 2019	Prob (F-statistic):	0.00
Time:	19:52:52	Log-Likelihood:	1101.9
No. Observations:	1540	AIC:	-2182.
Df Residuals:	1529	BIC:	-2123.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
accommodates	0.1308	0.047	2.789	0.005	0.039	0.223
bedrooms	0.4094	0.040	10.363	0.000	0.332	0.487
cleaning_fee	0.0914	0.044	2.069	0.039	0.005	0.178
Entire home/apt	0.1501	0.010	14.422	0.000	0.130	0.170
Air conditioning	0.0315	0.007	4.376	0.000	0.017	0.046
bathrooms	0.2237	0.033	6.679	0.000	0.158	0.289
TV	0.0258	0.008	3.155	0.002	0.010	0.042
security_deposit	0.1811	0.046	3.900	0.000	0.090	0.272
Family/kid friendly	0.0241	0.006	3.737	0.000	0.011	0.037
distance	-0.1948	0.014	-13.939	0.000	-0.222	-0.167
Private room	0.0681	0.011	6.176	0.000	0.046	0.090

Omnibus:	423.421	Durbin-Watson:	1.915
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1837.733
Skew:	1.248	Prob(JB):	0.00
Kurtosis:	7.734	Cond. No.	25.8

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

APPENDIX C

Random Forest OLS

OLS Regression Results

Dep. Variable:	price	R-squared:	0.882
Model:	OLS	Adj. R-squared:	0.880
Method:	Least Squares	F-statistic:	513.6
Date:	Sat, 27 Jul 2019	Prob (F-statistic):	0.00
Time:	19:53:07	Log-Likelihood:	601.79
No. Observations:	770	AIC:	-1182.
Df Residuals:	759	BIC:	-1130.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
accommodates	0.1927	0.063	3.063	0.002	0.069	0.316
bedrooms	0.4517	0.055	8.183	0.000	0.343	0.560
cleaning_fee	0.2121	0.059	3.600	0.000	0.096	0.328
Entire home/apt	0.1559	0.015	10.588	0.000	0.127	0.185
Air conditioning	0.0168	0.009	1.825	0.068	-0.001	0.035
bathrooms	0.1476	0.051	2.887	0.004	0.047	0.248
TV	0.0238	0.011	2.168	0.030	0.002	0.045
security_deposit	0.1368	0.052	2.639	0.008	0.035	0.239
Family/kid friendly	0.0113	0.009	1.309	0.191	-0.006	0.028
distance	-0.2057	0.018	-11.358	0.000	-0.241	-0.170
Private room	0.0862	0.014	6.028	0.000	0.058	0.114

Omnibus:	141.888	Durbin-Watson:	2.034
Prob(Omnibus):	0.000	Jarque-Bera (JB):	329.389
Skew:	0.988	Prob(JB):	2.98e-72
Kurtosis:	5.522	Cond. No.	26.5

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