Analytics Case study

**Business Case**

Over the last five years, client have witnessed an intriguing trend that suggests a correlation between the number of property images associated with a listing and the number of bookings it attracts. Client has also noticed an overwhelming number of listings being redundant, due to lack of any associated images.

1. **Management needs help to decide upon a** **minimum number of images to be made mandatory for a listing that would ensure bookings.**

**2. Also, come up with an optimal number of images that we can suggest the host to post**

**along with a listing that would attract the most bookings and ensure success.**

**Assumptions**

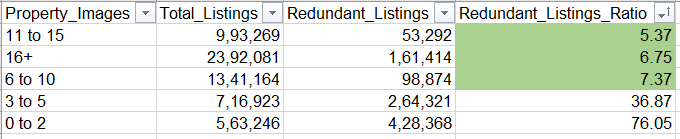
* Provided sample data is representative of population
* Data holds true for statistical analysis
* Key criteria’s for model building are Location, No of Booking, Posting Age
* Extraneous variables are not affecting our decision making

**Solution**

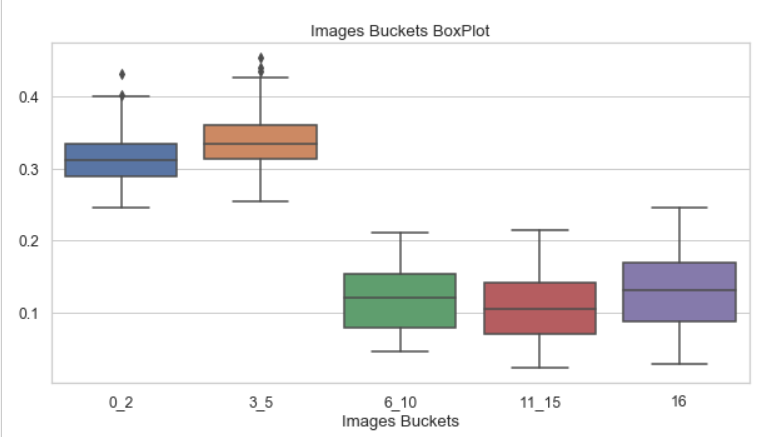
**Part 1:** Finding minimum number of images to be made mandatory for a listing that would ensure bookings

**Analysing Dataset2 & Dataset3**

1. From Dataset3, Calculated Redundant\_Listings/Total\_Listings ratio to find the images bucket with lowest ratio that didn’t attract any booking in last 1 year.
2. Categorized Images Bucket as: 0-2, 3-5, 6-10, 11-15 and 16+ in Dataset3. This categorization is done on the basis of data availability in Dataset2. Also, intuitively speaking, it doesn’t make sense to set minimum no of images more than 16.
3. Finding on Dataset3, as shown in image below, Redundant Listing Ratio significantly dropped when images bucket was: 6 to 10, 11 to 15 or 16+



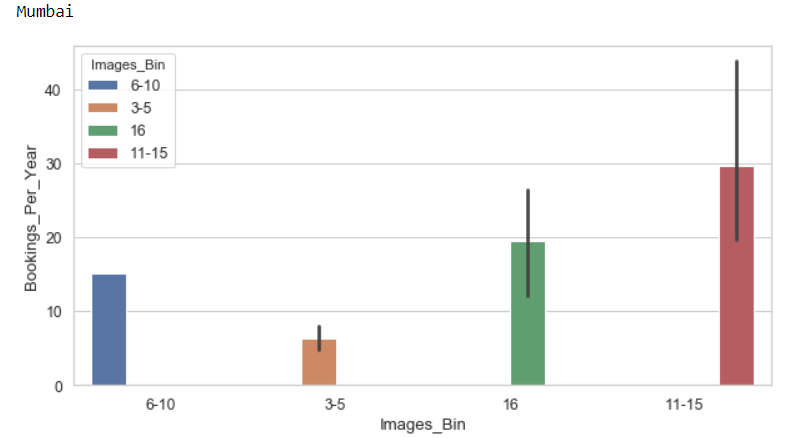
1. **Hypothesis**: Image bucket 11 to 15 is the best for ensured bookings. Used Dataset2 to validate the hypothesis.
2. In Dataset2, normalized all the Open\_Listing columns by dividing it with sum of total no of listings for each row. Boxplot all the columns as shown in image below:

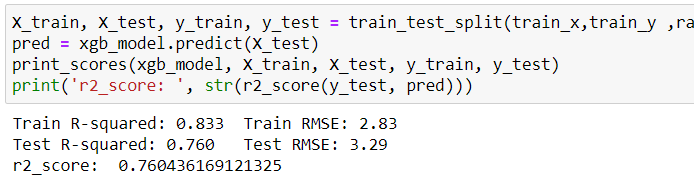


1. Although Image bucket 11 to 15 has the lowest range for no of days that didn’t attract any booking in last 1 year, which indeed proves our hypothesis, but we can see that there is only slight difference in range of image bucket 6-10 and 11-15.
2. Also, from figure 1 above, we can see that difference in Redundant listing ratio for bucket 6-10 and 11-15 is not much.
3. So from host’s point of view and also to ensure higher bookings, I belive it’s better to keep **6 images as mandatory for a listing.** Also, for smaller properties such as 1 room apartment/flat, I don’t think that it would be advisable to ask for minimum of 11 images from host. We will further see in Dataset3 analysis that keeping 6 images as mandatory makes sense.

**Part 2:** Finding an optimal number of images that we can suggest the host to post along with a listing that

would attract the most bookings and ensure success

1. To analyse Dataset1, let’s create some new features that would help us to identify optimal image buckets for new listings and model building for new existing listings.
2. Posting\_Age: No of years since listing was posted.
3. Bookings\_Per\_Year: No of avg booking made per year (Bookings/Posting\_Age)
4. Images\_Bin: Categorised images in bins: 0-2, 3-5, 6-10, 11-15, 16+
5. Removed outliers, by creating distplot on Bookings\_Per\_Year feature and removed all the rows having value above 200
6. Find **ideal images bucket for new listings as per location** - by plotting bar graph on Images\_Bin and Bookings\_Per\_Year feature for different locations. Sample image for Mumbai below: 
7. So for any new listing in Mumbai, it’s advisable to put images in range of 11 to 15 for high booking ensurity.
8. Similarly found best images bucket for different locations and suggest the same to host of new lsiting as per location. I didn’t create a ML model for new hosts/listings because parameters such as Posting\_Age, No of Bookings will be NIL for any new listing.
9. Also, we found that all the ideal images buckets for different locations are falling in 3 categories: 6-10, 11-15, 16+, this again validates our hypothesis in Part1.
10. To find optimal no of images for existing listings, we will create a regression model. First, we need to create a training data which represents **high success rate for listings**.
11. We defined any **listing as sucessful** as follow:
    1. Minimum Bookings per year > 8 (25 percentile of Booking per year)
    2. Listings which falls in ideal images bucket only, that is, 6-10, 11-15 or 16+
12. Almost 40% data falls in this category, this is the training data for regression model.
13. Cross verified above Success Measure on the basis of Superhost, all the existing superhost falls under training data, which ensures that we are not missing on any significant data.
14. Checked with heat map for any dependeny in existing data, found high dependency in between Bookings and Bookings\_Per\_Year feature, dropped Bookings feature.
15. Final set of features: Location, Host\_Type, Posting\_Age, Bookings\_Per\_Year
16. Target Variable: Images
17. Encoded categorical features, and then created various baseline models such as LinearRegression, LogisticRegression, RandomForestRegressor, XGBRegressor, LGBMRegressor
18. Hypertuned XGBRegressor and got **R-Squared value to be 0.76**, which means that we can account for around 76% of the variance using our features using XGBRegressor



1. Model performance can further be improved by adding additional features such as Property Type, Price, host neighborhood, accommodates, bathrooms, bedrooms, amnesties etc.

**Software Packages**

* Python – (Pandas, Numpy, Matplotlib, Seaborn, datetime, sklearn (LogisticRegression, LinearRegression, RandomForestRegressor, model\_selection, r2\_score, mean\_squared\_error, mean\_absolute\_error ), xgboost, lightgbm), Jupyter Notebook
* Excel

Select a.User\_id, a.start\_date, a.end\_date, 1 from users a self join users b where (a.end\_date >= b.start\_date AND a.end\_date <= b.end\_date) ;

Users\_df = pd.read\_csv(“filename.csv”)

Users\_df[‘flag’] = np.nan

def user\_flag(user\_id):

u\_df = User\_df[~(df[“user\_id”]==user\_id)]

start\_date = df[df[“user\_id”]==user\_id][‘start\_date’]

end\_date = df[df[“user\_id”]==user\_id][‘end\_date’]

sub\_df = u\_df[“start\_date”] <= end\_date

if sub\_df.shape[0]>=1 :

return 1

else:

return 0

for idx, data in enumerate(user\_df.iter\_rows()):

flag = user\_flag(user\_df, data[0][‘user\_id’])

user\_df[‘flag’].iloc[idx] = flag

Users\_df[‘flag’] = Users\_df[‘user\_id’].apply(user\_flag)

KPI’s

No of successful calls per quarter

Avg Customer Rating

Duration of Call

No of Complaints

Successful Trainings

Avg Queries solved per month