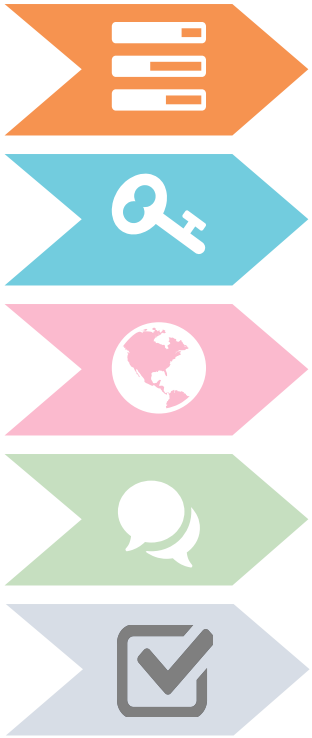
Airbnb Data Challenge

Executive Summary



Core Metrics will be **Number of Bookings** and **Booking-to-Inquiry Rate (BIR)**.
These two KPI's allow us to monitor performance of guest-host matching month over month.

Instant Book makes up **28%** of inquiries, but is responsible for **58% of Bookings**.
It is a key driver in our conversions.

Increase awareness of neighborhoods with an excess of listings
49% of all Inquiries come from Brazil. There is also an opportunity to capture more international demand.

Interactions and **reviews** are highly correlated with Bookings
Among the engagement metrics these two had the 3rd/4th highest Pearson's correlation scores.

Focus on increasing **bookings**, **awareness**, and **engagement**
And monitor campaigns on KPI's: # of Bookings, First Time Booking cohorted by country, Booking-to-Inquiry Rate

Q2-16' outperforms Q1-16' on core metrics

Rio de Janiero bookings spike in Jan and June. Jan has a low BIR while volume of Inquiries is high.

Defining metrics

$$\#\ of\ Bookings\ /Month$$

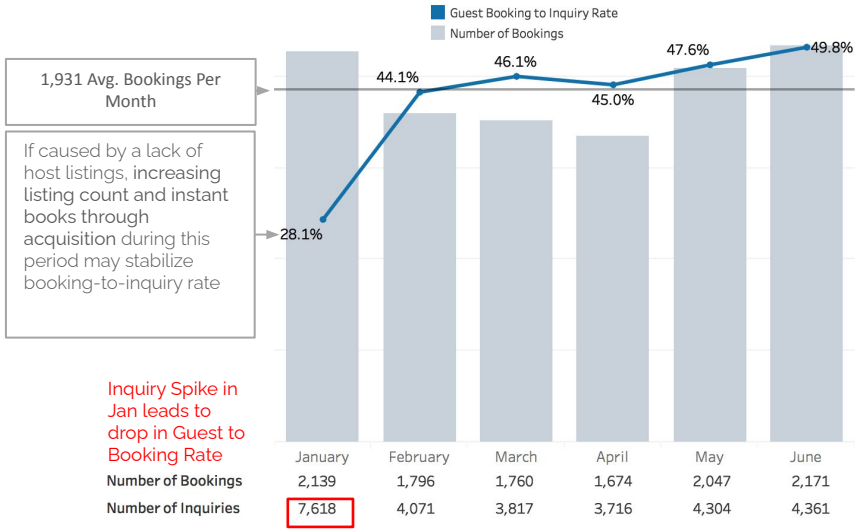
- Directly helps measure bookings and revenue.

$$\#\ of\ Inquiries\ /Month$$

- Top of funnel metric, which helps us gauge demand.

$$BIR = \frac{\#\ of\ Bookings}{\#\ of\ Inquiries}$$

- Funnel metric which shows our conversion rate from top-to-bottom.
- BIR is an actionable metric that can provide insights into improvements we make in the guest-host matching process.



Note: Metrics should be tracked on a monthly basis to compare month to month progress, since any changes would take at least a month to be realized.

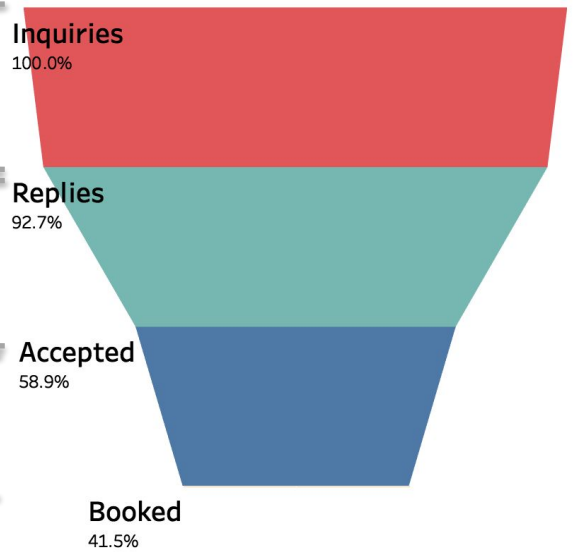
41.5% of Inquiries convert to Bookings

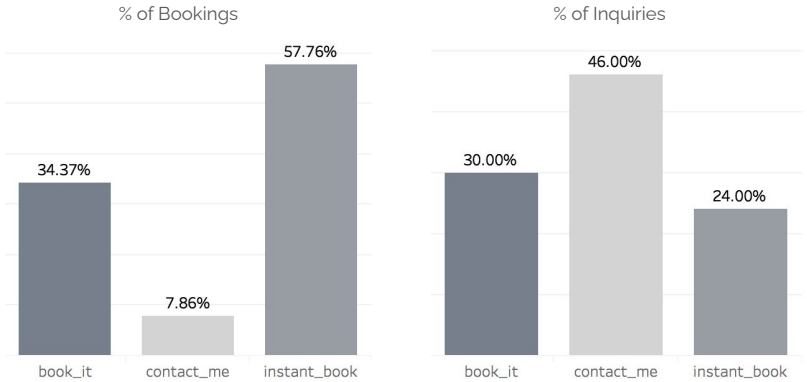
Almost **58%** of these Bookings were made through **Instant Book**, which only accounts for **24%** of all Inquiries.

Most hosts will reply to inquiries for their listings, this is not an issue.

This is where we first start to see guests drop-off, due to hosts declining inquiries.

Finally, guests may go with other listings if they have multiple inquiries further reducing the conversion rate





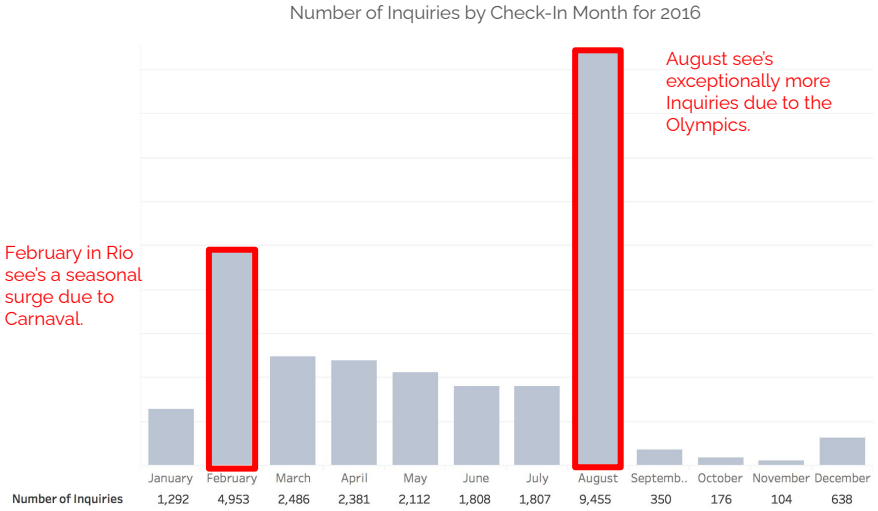
Recommendation:

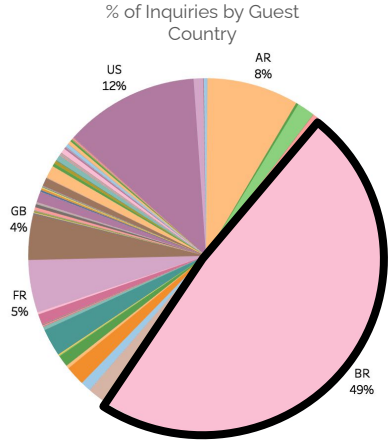
Providing incentives for our hosts to **opt-in to Instant Book**, which can drive our **BIR up**.

Note: See Appendix for additional metrics not shown.

Brazilians are looking at Rio

In a year where Rio is hosting the Olympics, one would think that there would be an influx of international guests. Instead, almost **50%** of Inquiries are from the host country.





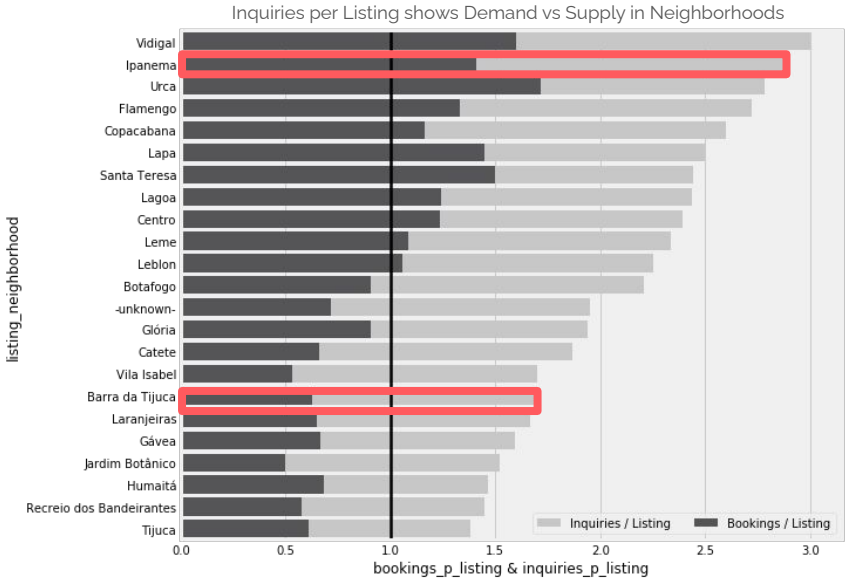
Recommendation:

Need to enlarge top of funnel, by bringing in more tourists from other countries.

Continued on Next Slide...

Some neighborhoods are filled with listing Inquiries

We should recommend guests to nearby or similar neighborhoods that are less saturated



Increasing awareness

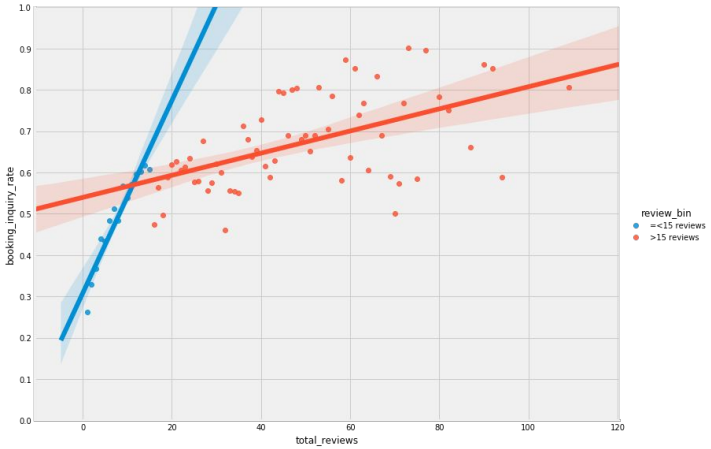
- Acquiring top-of-funnel Inquiries from other countries (US, FR, ARG, UK) by running more ads. The more Inquiries, the more possible conversions.
  - User acquisition can be monitored on the metric of First-Time-Booking (FTB).
- Search/collaborative filtering recommendation system can be used to guide guests to **less booked**, **more similar**, and **low BIR** neighborhoods: **Barra da Tijuaca**

	Grand Total	January	February	March	April	May	June
Santa Teresa	**0.6095**	0.5105	0.6625	0.6667	0.6164	0.6762	0.5761
Ipanema	**0.4726**	0.3150	0.5177	0.5607	0.4939	0.5137	0.5980
Leblon	**0.4643**	0.2882	0.5439	0.4783	0.4800	0.5519	0.6250
Copacabana	**0.4412**	0.3025	0.4727	0.4848	0.4657	0.5156	0.5483
-unknown-	**0.3641**	0.2279	0.3920	0.4031	0.4117	0.4185	0.4395
Barra da Tijuca	**0.3657**	0.2128	0.3932	0.4148	0.4226	0.4176	0.4104

Messaging and reviews are highly correlated with bookings

BIR R^2 regression line climbs fastest from 0-10 reviews

Total Reviews vs BIR Linear Model



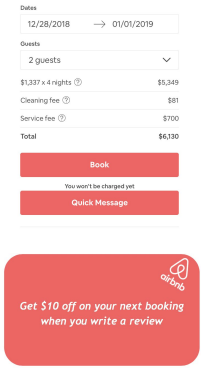
Pearson's Correlation Matrix

Attribute	r
Booked	1.000000
contact_channel_instant_book	0.666519
contact_channel_contact_me	0.645164
m_interactions	0.412032
total_reviews	0.291051

Increasing engagement

Drive guests to interact with hosts.

- **Quick Message Button**: a feature that takes a combination of user profile info and simple input to generate an inquiry message to hosts.
- **Tooltips** and **Hotspots** to encourage users to take specific actions such as send inquiry messages and leave reviews.
- **Provide incentives** to guests for writing reviews



Note: Although our findings show correlated relationships, they do not infer causation. Instead, A/B testing experiments should be performed to tease out true causality.

Note: r ranges from +1 to -1. 0 indicated no association. >0 indicates positives association. <0 indicates negative association.

Q3-16' Priorities

Number of Bookings

- Instant Book is responsible for 58% of all Bookings, but only makes up 28% of inquiries. We need to eliminate blockers and incentivize hosts to opt-in to Instant Book.

Awareness

- 50% of Inquiries are domestic. Get Rio on the map internationally(US, FR, ARG, UK) by increasing acquisition spend, effectively run more ads.
- Through the use of recommendation systems we should guide guests towards low inquiry per listing and low booking per listing neighborhoods

Engagement

- Implement A/B test for interactions and reviews and track using the BIR metric to determine causality. Once determined, build tools to encourage users to send messages and get listings to 10+ reviews.

Additional research/experiments/approaches that would be helpful

Search and Web/App Data

- With our current data, our top-of-funnel is Inquiries into listings, but with web/app and search data we can see how many distinct users are on the platform, and how many of these distinct visitors are looking at Rio as a destination. This will help us understand the demand of the market.

Causal inference through A/B testing

For reviews:

- Problem: Do reviews cause more bookings?
- Variants: reviews not displayed (treatment) and reviews shown (control)
- Experimental unit: visitors at listings page (sampled randomly)
- Power Analysis: Effect size, sample size, sig level, power
- Results: Observe significance and confidence interval
- Remember not to peak

Recommendation System Model

- In my iPython Notebook I have already experimented with predicting which inquiries ultimately led to bookings. Although, this is useful for predicting conversions in the funnel, it does not affect how guests are brought in at the top-of-funnel.
- There is an opportunity here to use collaborate filtering to enable better matching listings based on guest search queries. The goal would be to create a model that outputs a probability of acceptance, and display listings ranked on this probability output, thereby increasing the probability of conversions with higher quality inquiries from guests.

Appendix

Jupyter Notebook/Additional Metrics/Graphs

iPython Notebook

```
import os
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.style as style
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.ensemble import GradientBoostingClassifier

import warnings
%matplotlib inline
# Set styles globally
plt.rcParams['font.family'] = 'serif'
plt.rcParams['font.serif'] = 'Ubuntu'
plt.rcParams['font.monospace'] = 'Ubuntu Mono'
plt.rcParams['font.size'] = 10
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['axes.labelweight'] = 'bold'
plt.rcParams['axes.titlesize'] = 10
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 8
plt.rcParams['legend.fontsize'] = 10
plt.rcParams['figure.titlesize'] = 12
plt.rcParams.update(plt.rcParamsDefault)
style.use('ggplot')
```

Part I

What key metrics would you propose to monitor over time the success of the team's efforts in improving the guest host matching process and why? Clearly define your metric(s) and explain how each is computed.

What areas should we invest in to increase the number of successful bookings in Rio de Janeiro? What segments are doing well and what could be improved? Propose 2-3 specific recommendations (business initiatives and product changes) that could address these opportunities. Demonstrate rationale behind each recommendation AND prioritize your recommendations in order of their estimated impact.

There is also interest from executives at Airbnb about the work you are doing, and a desire to understand the broader framing of the challenge of matching supply and demand, thinking beyond the data provided. What other research, experiments, or approaches could help the company get more clarity on the problem?

Exploratory Analysis

Lets load the csv's and take a peak first.

```
contacts = pd.read_csv('contacts.csv')
listings = pd.read_csv('listings.csv')
users = pd.read_csv('users.csv')
```

```
contacts.head()
```

	id_guest_anon	id_host_anon	id_listing_anon	ts_interaction_first	ts_reply_at_first	ts_accepted_at_first	ts_booking_at	ds_checkin_first	ds_c
0	da8656a1-51af-4f38-b1c4-94be1f585157	5426897d-960d-4013-9e38-606ae746793c	a408a8b2-0d44-4513-a611-3736d0409bb2	2016-04-21 02:55:53	2016-04-21 03:15:00.0	2016-04-21 03:15:00.0	2016-04-21 03:15:00.0	2016-08-02	2016
1	8590d6f1-8bc9-4e8b-bdfb-de78f69fcf37	f30417c5-6df4-45ac-bfc2-6ad1cce398ab	e387c705-0aeb-464c-9375-ece63be6f006	2016-02-16 22:14:01	2016-02-16 23:37:36.0	NaN	NaN	2016-08-11	2016
2	ebcd83ba-bda1-47eb-9680-2dd04ccf3988	13cbf50a-3272-45d4-9866-a06b6ea1b99a	d1eb1960-938f-4305-a353-51e224414dd4	2016-01-27 23:33:38	2016-01-28 02:12:47.0	NaN	NaN	2016-03-14	2016
3	b0af8848-fe2a-4ef1-991e-26ab3066feb3	01614601-d5a4-4776-ab9b-c10d3b865bf0	855f6779-346c-45fc-a64b-ea133250ca7a	2016-05-05 14:42:52	2016-05-05 15:17:40.0	2016-05-05 15:17:40.0	NaN	2016-05-27	2016
4	5ddbbcc3-ac1a-4d8a-b6f8-0f675b76d1b2	f2fed6f3-4c5c-453d-9e64-37c62b8bd06d	f2928a59-c5e7-42b2-9c37-240a40f38dc0	2016-06-23 03:09:25.0	2016-06-23 03:09:26.0	2016-06-23 03:09:33.0	2016-06-23 03:09:33.0	2016-08-19	2016

```
# Lets check the column data types
contacts.dtypes
```

```
id_guest_anon                          object
id_host_anon                           object
id_listing_anon                        object
ts_interaction_first                   object
ts_reply_at_first                      object
ts_accepted_at_first                   object
ts_booking_at                          object
ds_checkin_first                       object
ds_checkout_first                      object
m_guests                              float64
m_interactions                          int64
m_first_message_length_in_characters  float64
contact_channel_first                  object
guest_user_stage_first                 object
dtype: object
```

```
# Looks like we can change the timestamps to datetimes
contacts.ts_interaction_first = pd.to_datetime(contacts.ts_interaction_first, utc=True)
contacts.ts_reply_at_first = pd.to_datetime(contacts.ts_reply_at_first, utc=True)
contacts.ts_accepted_at_first = pd.to_datetime(contacts.ts_accepted_at_first, utc=True)
contacts.ts_booking_at = pd.to_datetime(contacts.ts_booking_at, utc=True)
contacts.ds_checkin_first = pd.to_datetime(contacts.ds_checkin_first, utc=True)
contacts.ds_checkout_first = pd.to_datetime(contacts.ds_checkout_first, utc=True)
contacts.dtypes
```

```
id_guest_anon                                object
id_host_anon                                 object
id_listing_anon                              object
ts_interaction_first            datetime64[ns, UTC]
ts_reply_at_first               datetime64[ns, UTC]
ts_accepted_at_first            datetime64[ns, UTC]
ts_booking_at                   datetime64[ns, UTC]
ds_checkin_first                datetime64[ns, UTC]
ds_checkout_first               datetime64[ns, UTC]
m_guests                                    float64
m_interactions                                int64
m_first_message_length_in_characters        float64
contact_channel_first                        object
guest_user_stage_first                       object
dtype: object
```

```
print("Size of the contacts table")
print(contacts.shape)

# Unique listings that were contacted
grouped_contacts_df = contacts.groupby(['id_listing_anon']).size()
print("Unique listings that were contacted: {}".format(grouped_contacts_df.shape[0]))
```

```
Size of the contacts table
(27887, 14)
Unique listings that were contacted: 12819
```

```
# On first glance it looks like guests and hosts will book listings repeatedly, lets make sure these arent dups
grouped_contacts_df = contacts.groupby(['id_guest_anon','id_host_anon','id_listing_anon']).size().sort_values(ascending=False).reset_index()
grouped_contacts_df.head()
```

	id_guest_anon	id_host_anon	id_listing_anon	0
0	757d72a2-fefd-4dee-9798-5a03500e32af	d08e0a70-1d4d-4138-b3d5-0023bd29edec	bdbe88db-10f5-4930-a16d-572bae857b4c	4
1	96cbb3e0-65bb-4c4a-bd8e-842621d44035	7a6ac972-d38c-4f71-82de-ed63c451257c	27c45eef-e706-4738-ac17-e60e503659e0	3
2	fa7be6b9-85bd-492d-8c56-5f30c24629da	3429cbcd-ab0d-43ec-8be2-40941f58098a	9e8c83c6-091d-40fc-8bb4-95fde8e0e15c	3
3	3b9a1f6e-3488-4300-af56-1f17b064a5d7	0b3b65e6-3242-46a0-b89f-19b3b5d291f3	3528efe8-f999-4d59-988e-973467c6369c	2
4	ab6834ad-ef03-4727-b028-8e0be4fd7e5e	54d4c1b1-3832-484e-b257-a86760eb5240	7fc389bb-ee0d-488f-82ce-8a7c1409ef78	2

```
# If we group by checkin and checkout dates we should get the dups, lets investigate whats going on here
grouped_contacts_df = contacts.groupby(['id_guest_anon','id_host_anon','id_listing_anon','ds_checkin_first','ds_checkout_first']).size().sort_values(ascending=False).reset_index()
grouped_contacts_df.rename(columns={0: 'dup_bookings'}, inplace=True)
grouped_contacts_df.head()
```

	id_guest_anon	id_host_anon	id_listing_anon	ds_checkin_first	ds_checkout_first	dup_bookings
0	6d40f3b2-a3f7-4f3e-97c3-3e88398a58d9	3353c76c-2eaa-4353-89aa-152ff7f4e70b	7754738b-1742-4d4d-aa33-32b7375d3ade	2016-03-30 00:00:00+00:00	2016-04-03 00:00:00+00:00	2
1	5378478e-f946-41d1-b4b8-e58e50243a05	690d5847-acb0-4333-a6c2-28a5a113fcfe	e0f0c068-a8e0-47ae-996e-99cef820a7f7	2016-08-01 00:00:00+00:00	2016-08-22 00:00:00+00:00	2
2	fffbce5a-d9b6-40aa-9028-e1257f6fa5e7	9ebc8a6c-f3d8-4141-825e-a635e9a823d4	2ddb4121-9cd6-453c-89af-f2a97dc74dce	2016-08-10 00:00:00+00:00	2016-08-17 00:00:00+00:00	1
3	556274fd-dae7-4af1-bf15-0843871f6c7d9	371377fb-57b2-473b-8078-276422180068	af92ad93-37b1-4dac-bd36-3bd9e22d4b1d	2016-08-07 00:00:00+00:00	2016-08-14 00:00:00+00:00	1
4	553db8ae-cbff-44e8-8446-	6459c2fa-35a3-4097-bf81-	8b864df5-4745-4e1d-a1bf-	2016-08-02	2016-08-15	1

```
# For both dups here we found that there was some issue that caused the guest to double book
contacts[(contacts['id_guest_anon'] == "6d40f3b2-a3f7-4f3e-97c3-3e88398a58d9") & (contacts['id_host_anon'] == "3353c76
c-2eaa-4353-89aa-152ff7f4e70b")]
```

	id_guest_anon	id_host_anon	id_listing_anon	ts_interaction_first	ts_reply_at_first	ts_accepted_at_first	ts_booking_at	ds_checkin_first
8355	6d40f3b2-a3f7-4f3e-97c3-3e88398a58d9	3353c76c-2eaa-4353-89aa-152ff7f4e70b	7754738b-1742-4d4d-aa33-32b7375d3ade	2016-03-24 20:39:50+00:00	2016-03-24 20:46:24+00:00	NaT	NaT	2016-03-30 00:00:00+00:00
20569	6d40f3b2-a3f7-4f3e-97c3-3e88398a58d9	3353c76c-2eaa-4353-89aa-152ff7f4e70b	7754738b-1742-4d4d-aa33-32b7375d3ade	2016-01-13 02:33:04+00:00	2016-01-13 10:00:45+00:00	2016-01-13 10:11:37+00:00	2016-01-13 10:11:37+00:00	2016-03-30 00:00:00+00:00

```
# Since there are only two issues here, lets just ignore the errors for the sake of this exercise
grouped_contacts_df[grouped_contacts_df['dup_bookings'] == 2]['id_guest_anon']
```

```
0     6d40f3b2-a3f7-4f3e-97c3-3e88398a58d9
1     5378478e-f946-41d1-b4b8-e58e50243a05
Name: id_guest_anon, dtype: object
```

```
listings.head()
```

	id_listing_anon	room_type	listing_neighborhood	total_reviews
0	71582793-e5f8-46d7-afdf-7a31d2341c79	Private room	-unknown-	0.0
1	a1a3f728-e21f-4432-96aa-361d28e2b319	Entire home/apt	Copacabana	0.0
2	353a68be-ecf9-4b7b-9533-c882dc2f0760	Entire home/apt	Barra da Tijuca	3.0
3	b9ae1908-0486-40ac-bbcf-bd63ffebfe63	Entire home/apt	Lapa	4.0
4	fa0290ef-7881-4482-8981-8ebb1ce5dbfd	Entire home/apt	-unknown-	0.0

```
print("Total Listings: {}".format(listings.size))
# Check if listings id_listing_anon is unique
print("Are listing id's unique?: {}".format(listings['id_listing_anon'].is_unique))
```

```
Total Listings: 52152
Are listing id's unique?: True
```

```
#There are many missing neighborhoods in the listing dataset
listings.listing_neighborhood.value_counts().reset_index().head()
```

	index	listing_neighborhood
0	-unknown-	6221
1	Copacabana	2531
2	Ipanema	1041
3	Barra da Tijuca	593
4	Leblon	458

```
users.head()
```

	id_user_anon	country	words_in_user_profile
0	1d16a001-31a2-494c-a101-1f7b308adc62	FR	0
1	42607e0a-86c0-472e-b633-9e192114e93c	AR	0
2	25f85eb5-a700-44e1-b142-4c076222198d	BR	0
3	55abeba0-18ef-4c58-80f4-3c278b706aca	BR	1
4	5d62d35a-7d6d-45dd-ae9b-a5d2f82a7d7b	BR	98

```
# Check if id_user_anon is unique
print("Are user_id's unique?: {}".format(users['id_user_anon'].is_unique))
```

```
Are user_id's unique?: False
```

```
dup_count  = users.id_user_anon.value_counts().reset_index()
print("There are {} duplicates of id_user_anon".format(dup_count[dup_count['id_user_anon'] == 2].shape[0]))
```

```
There are 68 duplicates of id_user_anon
```

```
users[users['id_user_anon'] == 'f371e503-9ebf-4ba9-a6a9-2f7c11279c73']
```

	id_user_anon	country	words_in_user_profile
29097	f371e503-9ebf-4ba9-a6a9-2f7c11279c73	BR	11
31522	f371e503-9ebf-4ba9-a6a9-2f7c11279c73	BR	11

```
users.drop_duplicates('id_user_anon', inplace=True)
```

```
# Lets double check to make sure there are no more duplicates left
dup_count  = users.id_user_anon.value_counts().reset_index()
print("Number of duplicates now: {}".format(dup_count[dup_count['id_user_anon'] == 2].shape[0]))
```

```
Number of duplicates now: 0
```

```
def denormalize_contacts(contacts, users, listings):
    '''
    Function creates an unique_id column as well as a booked column
    to act as a label for successful/unsuccessful booking.
    '''

    # Denormalize column `viewed_user_id`
    dn_merged_df = contacts.merge(users, how='left', left_on='id_guest_anon', right_on='id_user_anon')
    dn_merged_df = dn_merged_df.merge(users, how='left', left_on='id_host_anon', right_on='id_user_anon',
                                      suffixes=('_guest', '_host'))
    dn_merged_df = dn_merged_df.merge(listings, how='left', left_on='id_listing_anon', right_on='id_listing_anon')

    #Pre-feature engineering columns
    dn_merged_df['interaction_id'] = dn_merged_df.index + 1
    dn_merged_df['booked'] = dn_merged_df['ts_booking_at'].apply(lambda x: 0 if pd.isnull(x) else 1)

    cols = dn_merged_df.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    dn_merged_df = dn_merged_df.loc[:, cols]

    return dn_merged_df

def write_output_to_csv(df, output_file='bookings.csv'):
    '''
    Helper function to output resulting data into a .csv file
    '''
    if os.path.exists(output_file):
        os.remove(output_file)
    df.to_csv(output_file, encoding = "utf-8", index = False)
    print("File Written")

#write_output_to_csv(dn_merged_df)
```

```
dn_merged_df = denormalize_contacts(contacts, users, listings)
dn_merged_df.head()
```

	booked	id_guest_anon	id_host_anon	id_listing_anon	ts_interaction_first	ts_reply_at_first	ts_accepted_at_first	ts_booking_at	ds_checkin_fi
0	1	da8656a1-51af-4f38-b1c4-94be1f585157	5426897d-960d-4013-9e38-606ae746793c	a408a8b2-0d44-4513-a611-3736d0409bb2	2016-04-21 02:55:53	2016-04-21 03:15:00.0	2016-04-21 03:15:00.0	2016-04-21 03:15:00.0	2016-08-02
1	0	8590d6f1-8bc9-4e8b-bdfb-de78f69fcf37	f30417c5-6df4-45ac-bfc2-6ad1cce398ab	e387c705-0aeb-464c-9375-ece63be6f006	2016-02-16 22:14:01	2016-02-16 23:37:36.0	NaN	NaN	2016-08-11
2	0	ebcd83ba-bda1-47eb-9680-2dd04ccf3988	13cbf50a-3272-45d4-9866-a06b6ea1b99a	d1eb1960-938f-4305-a353-51e224414dd4	2016-01-27 23:33:38	2016-01-28 02:12:47.0	NaN	NaN	2016-03-14
3	0	b0af8848-fe2a-4ef1-991e-26ab3066feb3	01614601-d5a4-4776-ab9b-c10d3b865bf0	855f6779-346c-45fc-a64b-ea133250ca7a	2016-05-05 14:42:52	2016-05-05 15:17:40.0	2016-05-05 15:17:40.0	NaN	2016-05-27
4	1	5ddbbcc3-ac1a-4d8a-b6f8-0f675b76d1b2	f2fed6f3-4c5c-453d-9e64-37c62b8bd06d	f2928a59-c5e7-42b2-9c37-240a40f38dc0	2016-06-23 03:09:25.0	2016-06-23 03:09:26.0	2016-06-23 03:09:33.0	2016-06-23 03:09:33.0	2016-08-19

airbnb

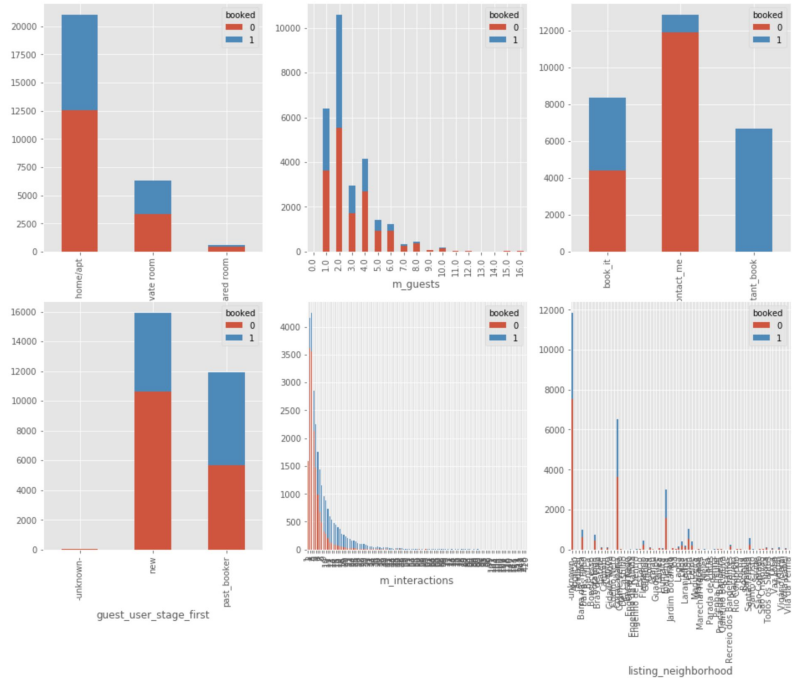
```
def make_features(dn_merged_df):
    '''
    Function which one-hot encodes categorical variables in binary features
    '''
    dn_merged_df_onehot = dn_merged_df.copy()

    dn_merged_df_onehot = pd.get_dummies(dn_merged_df_onehot, columns=['room_type'], prefix = ['room_type'])
    dn_merged_df_onehot = pd.get_dummies(dn_merged_df_onehot, columns=['listing_neighborhood'], prefix = ['neighborhood'])
    dn_merged_df_onehot = pd.get_dummies(dn_merged_df_onehot, columns=['country_host'], prefix = ['country_host'])
    dn_merged_df_onehot = pd.get_dummies(dn_merged_df_onehot, columns=['country_guest'], prefix = ['country_guest'])
    dn_merged_df_onehot = pd.get_dummies(dn_merged_df_onehot, columns=['guest_user_stage_first'], prefix = ['user_stage'])
    dn_merged_df_onehot = pd.get_dummies(dn_merged_df_onehot, columns=['contact_channel_first'], prefix = ['contact_channel'])

    return dn_merged_df_onehot
```

```
feature_eng_df = make_features(dn_merged_df)
feature_eng_df.head()
```

	booked	id_guest_anon	id_host_anon	id_listing_anon	ts_interaction_first	ts_reply_at_first	ts_accepted_at_first	ts_booking_at	ds_checkin_f
0	1	da8656a1-51af-4f38-b1c4-94be1f585157	5426897d-960d-4013-9e38-606ae746793c	a408a8b2-0d44-4513-a611-3736d0409bb2	2016-04-21 02:55:53+00:00	2016-04-21 03:15:00+00:00	2016-04-21 03:15:00+00:00	2016-04-21 03:15:00+00:00	2016-08-02 00:00:00+00:(
1	0	8590d6f1-8bc9-4e8b-bdfb-de78f69fcf37	f30417c5-6df4-45ac-bfc2-6ad1cce398ab	e387c705-0aeb-464c-9375-ece63be6f006	2016-02-16 22:14:01+00:00	2016-02-16 23:37:36+00:00	NaT	NaT	2016-08-11 00:00:00+00:(
2	0	ebcd83ba-bda1-47eb-9680-2dd04ccf3988	13cbf50a-3272-45d4-9866-a06b6ea1b99a	d1eb1960-938f-4305-a353-51e224414dd4	2016-01-27 23:33:38+00:00	2016-01-28 02:12:47+00:00	NaT	NaT	2016-03-14 00:00:00+00:(
3	0	b0af8848-fe2a-4ef1-991e-26ab3066feb3	01614601-d5a4-4776-ab9b-c10d3b865bf0	855f6779-346c-45fc-a64b-ea133250ca7a	2016-05-05 14:42:52+00:00	2016-05-05 15:17:40+00:00	2016-05-05 15:17:40+00:00	NaT	2016-05-27 00:00:00+00:(
4	1	5ddbbcc3-ac1a-4d8a-b6f8-0f675b76d1b2	f2fed6f3-4c5c-453d-9e64-37c62b8bd06d	f2928a59-c5e7-42b2-9c37-240a40f38dc0	2016-06-23 03:09:25+00:00	2016-06-23 03:09:26+00:00	2016-06-23 03:09:33+00:00	2016-06-23 03:09:33+00:00	2016-08-19 00:00:00+00:(

5 rows × 243 columns

```
# Take a look at how Room Type, Guests, Contact Channel, User Stage, Interactions, and Listing Neighborhood are related to Bookings
fig, axes = plt.subplots(2,3, figsize=(16,12))
attr_ls = ['room_type', 'm_guests', 'contact_channel_first', 'guest_user_stage_first', 'm_interactions','listing_neighborhood']
for attr, ax in zip(attr_ls[:3], axes[0]):
    dn_merged_df.pivot_table("interaction_id", attr, "booked", "count").plot(kind="bar", stacked=True, ax=ax)
for attr, ax in zip(attr_ls[3:], axes[1]):
    dn_merged_df.pivot_table("interaction_id", attr, "booked", "count").plot(kind="bar", stacked=True, ax=ax)
```



Looks like there are significantly more bookings for homes/apts and there is an optimal amount of guests for successful bookings as well(around 1-4). It makes sense that instant-book accounts for a significant amount of bookings. For past_bookers there is a slight difference in the proportion that are booked. As for interactions it is hard to tell here, how many interactions is indictive of a booking, but it looks like there is a low-middle range that resulted in more bookings. It's hard to read the neighborhoods here, but it is clear there are a few that are much more in demand than the rest.

airbnb

```
feature_eng_df['m_guests'].fillna((feature_eng_df['m_guests'].median()), inplace=True)
# check which columns have null values
# pd.isnull(feature_eng_df).sum() > 0
```

Lets scale our numerical variables, so we normalize the values among different features.

```
scaler = MinMaxScaler()

scaled_feature_df = feature_eng_df.copy()
scaled_feature_df['m_guests'].fillna((scaled_feature_df['m_guests'].median()), inplace=True)

scaled_feature_df[['m_guests','m_interactions','total_reviews',
                   'm_first_message_length_in_characters','words_in_user_profile_host',
                   'words_in_user_profile_guest']] = scaler.fit_transform(scaled_feature_df[['m_guests','m_interaction
s',
                                                                                            'total_reviews',
                                                                                            'm_first_message_length_i

n_characters',
                                                                                            'words_in_user_profile_ho

st',
                                                                                            'words_in_user_profile_gu

est']])
```

Here we verify our observations using a correlation matrix, in which we can see that of course instant_book is highly correlated with bookings, followed by contacts. Then, messages exchanged and listing reviews were the next highest indicators of a booking. Surprisingly, being a returning user held very little weight against new users. Lastly, it looks like being from Brazil or the US were also moderate indicators of successful bookings.

```
corr = scaled_feature_df.corr()
print(np.abs(corr["booked"]).sort_values(ascending=False).head(20))
```

```
booked                                1.000000
contact_channel_instant_book          0.666519
contact_channel_contact_me            0.645164
m_interactions                        0.412032
total_reviews                         0.291051
country_guest_BR                      0.209647
user_stage_past_booker                0.194693
user_stage_new                        0.194543
country_guest_US                      0.130752
m_guests                              0.128144
m_first_message_length_in_characters  0.106471
words_in_user_profile_host            0.099738
neighborhood_-unknown-                0.089594
contact_channel_book_it               0.080495
country_guest_FR                      0.079517
country_guest_GB                      0.072715
country_guest_AR                      0.069857
country_guest_DE                      0.068223
room_type_Private room                0.061122
words_in_user_profile_guest           0.059185
Name: booked, dtype: float64
```

As an experiment, lets take the current data and see if we can extract the most important features both categorical and numerical to try and create a binary classification model, which can predict whether each interaction resulted in a booking.

Date columns, timestamps, and anonymous id's were removed from our data

```
# Drop date columns with missing values, these feature at not that relevant so we will omit them
ml_df = scaled_feature_df.copy()
ml_df.drop(['id_guest_anon','id_host_anon','id_listing_anon','id_user_anon_guest',
                      'id_user_anon_host','ts_reply_at_first','ts_accepted_at_first','ts_booking_at',
                      'ts_interaction_first','ds_checkin_first','ds_checkout_first','interaction_id'], axis = 1, inp
lace = True)
```

```
# Initialize the label we are trying to predict
y = ml_df.booked
X = ml_df.drop(['booked'], axis=1)
```

```
# Split data into training and test (80% training)
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2)
```

```
XTrain = X_train.values
YTrain = y_train
XTest = X_test.values
```

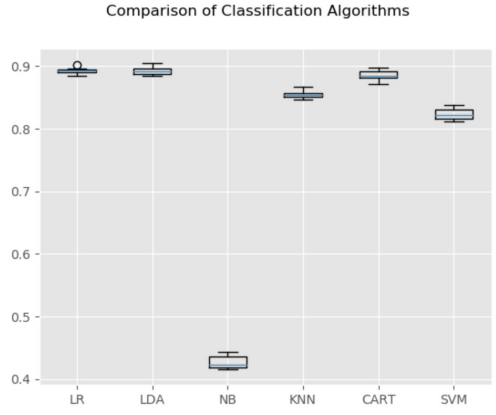
```
# Feature Selection using chi2 test

# Selecting the 5 better features that can be used in prediction model
selector = SelectKBest(score_func = chi2, k = 5)
fit = selector.fit(XTrain, YTrain)
```

```
# Summarizing selected Features
mask = selector.get_support(indices=True)
new_features = ml_test_df.columns[mask]
print(new_features)
```

```
Index(['country_guest_BR', 'user_stage_new', 'user_stage_past_booker',
       'contact_channel_contact_me', 'contact_channel_instant_book'],
      dtype='object')
```

```
warnings.filterwarnings("ignore")
# Defining number of folds
num_folds = 10
num_instances = len(XTrain)
seed = 10

# Preparing models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('NB', GaussianNB()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('SVM', SVC()))

# Model Evaluation
results = []
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits = num_folds, random_state = seed)
    cv_results = model_selection.cross_val_score(model, XTrain, YTrain, cv = kfold, scoring = 'accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

# Boxplot to compare algorithms
fig = plt.figure()
fig.suptitle('Comparison of Classification Algorithms')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
LR: 0.892644 (0.004647)
LDA: 0.892420 (0.005635)
NB: 0.426421 (0.010361)
KNN: 0.854857 (0.005769)
CART: 0.886190 (0.007398)
SVM: 0.823748 (0.008212)
```



airbnb

```
# Creating logistic regression model
modelLR = LogisticRegression()

# Training model and checking the score
modelLR.fit(XTrain, YTrain)
modelLR.score(XTrain, YTrain)

# Colecting coefficients
#print('Coefficient: \n', modelLR.coef_)
#print('Intercept: \n', modelLR.intercept_)

# Predictions
YPred = modelLR.predict(XTest)
```

```
#Checking accuracy for training
acc_log = round(modelLR.score(XTrain, YTrain) * 100, 2)
print("Model Accuracy on Training Data: {}".format(acc_log))
#Checking accuracy for test
acc_log = round(modelLR.score(X_test, y_test) * 100, 2)
print("Model Accuracy on Test Data: {}".format(acc_log))
```

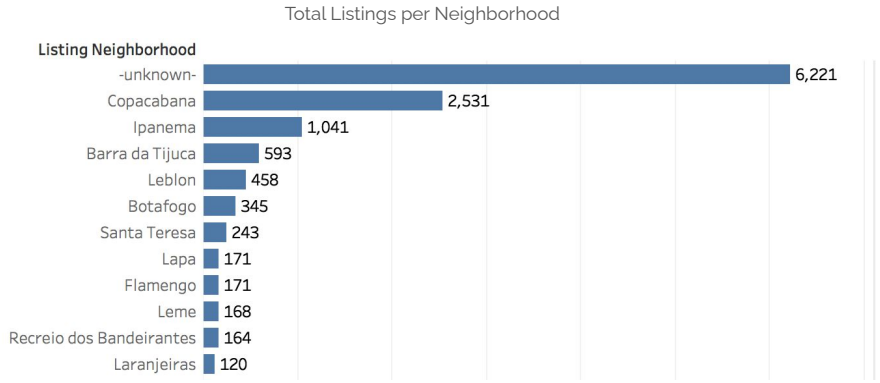
```
Model Accuracy on Training: 89.72
Model Accuracy on Training: 90.19
```

```
# Gradient Boosting Classifier

ModelCLF = GradientBoostingClassifier(n_estimators = 650, learning_rate = 1.0, max_depth = 1, random_state = 0)

# Training model and checking the score
ModelCLF.fit(XTrain, YTrain)
ModelCLF.score(XTrain,YTrain)

# Predictions
YPredGBC=ModelCLF.predict(XTest)
```

```
#Checking accuracy for training
acc_log = round(ModelCLF.score(XTrain,YTrain) * 100, 2)
print("Model Accuracy on Training Data: {}".format(acc_log))
acc_log = round(ModelCLF.score(X_test,y_test) * 100, 2)
print("Model Accuracy on Test Data: {}".format(acc_log))
```

```
Model Accuracy on Training Data: 92.31
Model Accuracy on Test Data: 92.29
```

Tableau Visualizations



Monthly Conversion Rates for Booking Funnel Stages (Q1-Q2 2016)

Monthly KPI's

Monthly Booking Inquiries for Rio Neighborhoods 2016

	January	February	March	April	May	June	July	August	September	October	November	December
Grand Total	1,365	5,185	2,497	2,385	2,114	1,811	1,807	9,455	350	176	104	638
-unknown-	517	2,186	866	873	747	630	729	4,819	146	59	39	217
Copacabana	375	1,247	674	634	547	470	441	1,775	83	40	18	223
Ipanema	163	607	354	323	271	214	179	706	39	23	17	98
Leblon	53	230	112	100	96	74	55	252	15	9	7	18
Barra da Tijuca	37	72	76	74	85	54	64	490	22	2	4	18
Santa Teresa	35	92	82	57	57	60	54	131	6	6	5	4
Botafogo	37	151	57	55	66	66	55	243	5	9	2	9
Lapa	23	68	47	55	34	46	34	101	7	4		6
Flamengo	23	81	53	42	52	36	42	105	5	12	1	5
Leme	26	79	33	35	25	39	22	105	3	4	3	16

Quick Message Button Setting Mockup