AIRBNB NYC

By: Cameron Smith, Sravani Yerramaneni, Austin Nocero

DATASET

AIRBNB = An online marketplace that lets people rent out their properties or spare rooms to guests

Dataset includes the following information on Airbnb Listings:

- **Host**
  - ID and Name
- **Location**
  - Latitude/Longitude
  - Neighbourhood
- **Price**
- **Room Type**
  - Private/Shared/Entire home
- **Availability**
  - days out of the year the listing is up
- **Number of Reviews**
- **Reviews Per Month**
- **Number of Listings Host has**

Airbnb Dataset

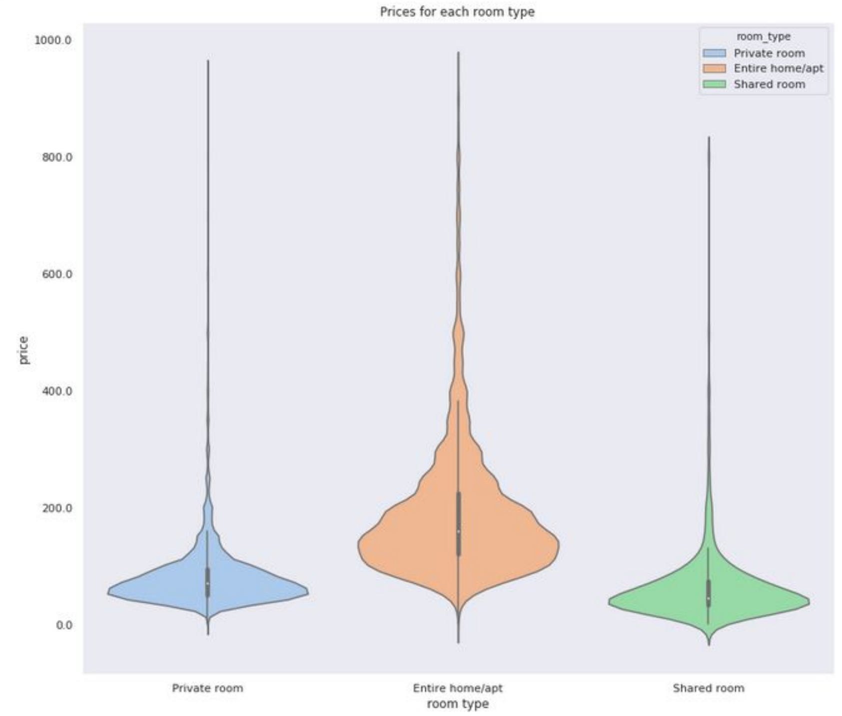
This dataset describes the listing activity and metrics in NYC, NY for 2019.

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	2018-10-19	0.21	6	365
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	0.38	2	355
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	NaN	1	365
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05	4.64	1	194
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	0.10	1	0

DESCRIPTIVE STATISTICS

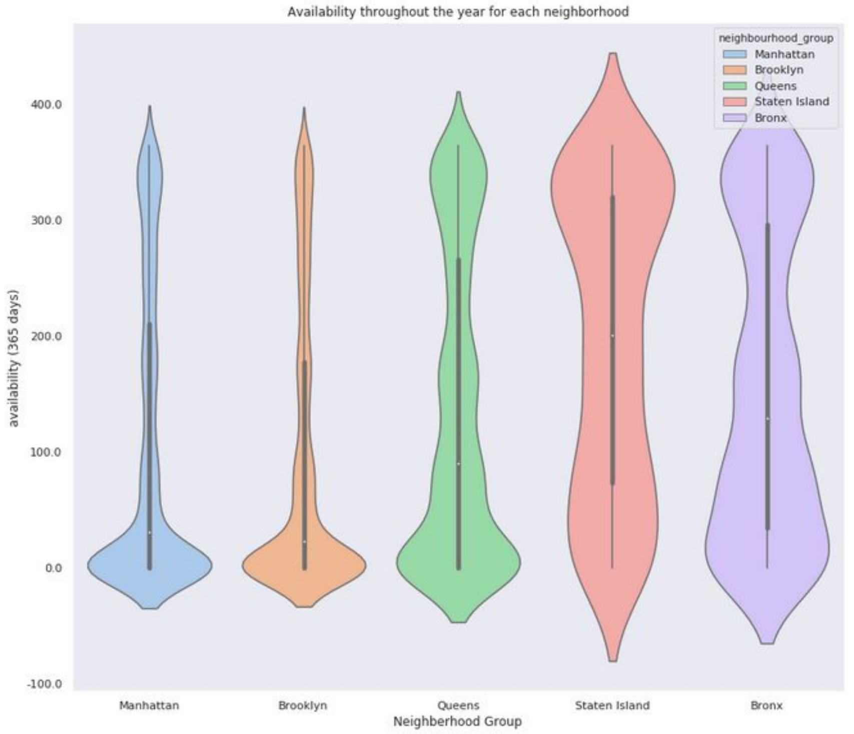
Price Per Room Type

```
sns.set(style="dark")
price = 950

priceType = data[data.price < price]

fig, ax = plt.subplots(figsize=(12, 12))
density_neigh_price_plot = sns.violinplot(ax=ax, x="room_type", y="price",
                                          hue="room_type", data=priceType,
                                          palette="pastel", dodge=False)
density_neigh_price_plot.set(xlabel='room type', ylabel='price',
                             title='Prices for each room type')
ylabels = ['{}'.format(x) for x in density_neigh_price_plot.get_yticks()]
density_neigh_price_plot.set_yticklabels(ylabels)
plt.show()
```



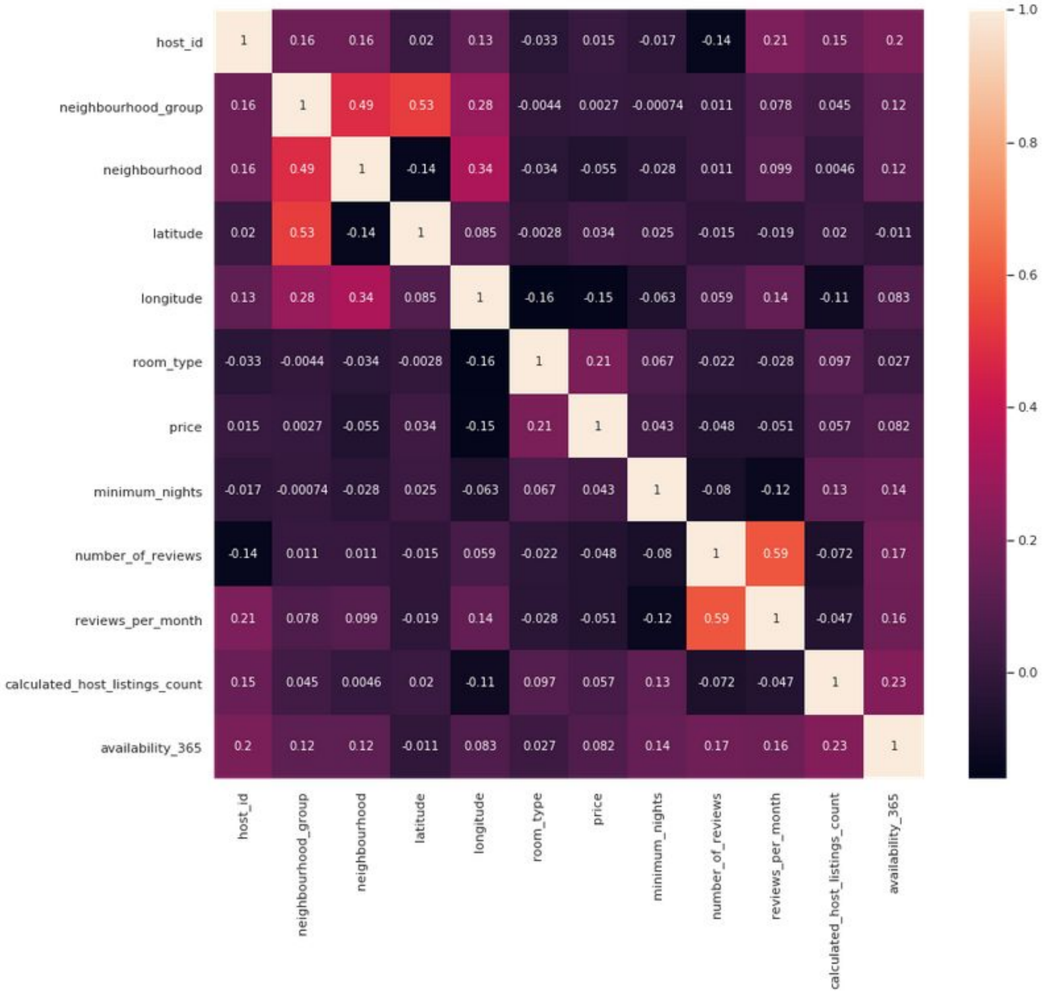
Availability Year Round in Each Neighbourhood Group

```
sns.set(style="dark")
days = 365

avl = data[data.availability_365 < days]

fig, ax = plt.subplots(figsize=(12, 12))
density_neigh_price_plot = sns.violinplot(ax=ax, x="neighbourhood_group", y="availability_365",
                                          hue="neighbourhood_group", data=avl,
                                          palette="pastel", dodge=False)
density_neigh_price_plot.set(xlabel='Neighberhood Group', ylabel='availability (365 days)',
                             title='Availability throughout the year for each neighborhood')
ylabels = ['{}'.format(x) for x in density_neigh_price_plot.get_yticks()]
density_neigh_price_plot.set_yticklabels(ylabels)
plt.show()
```



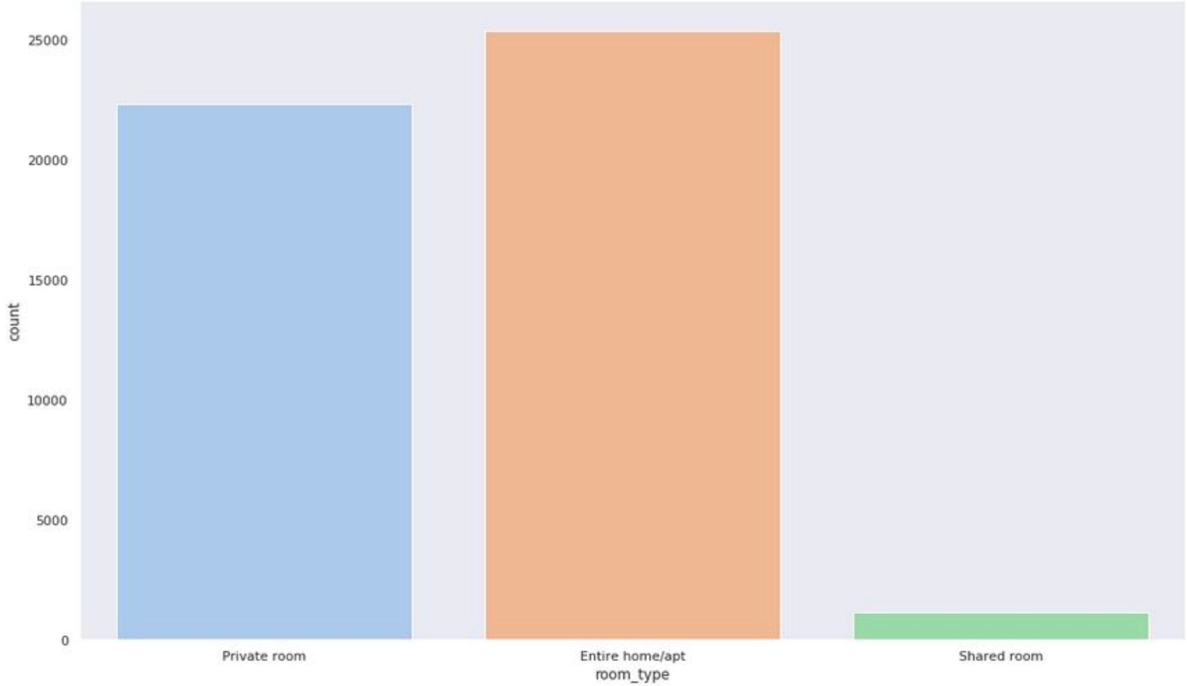
Correlation

```
plt.figure(figsize=(12,12))
ax = sns.heatmap(df.corr(), annot=True)
```



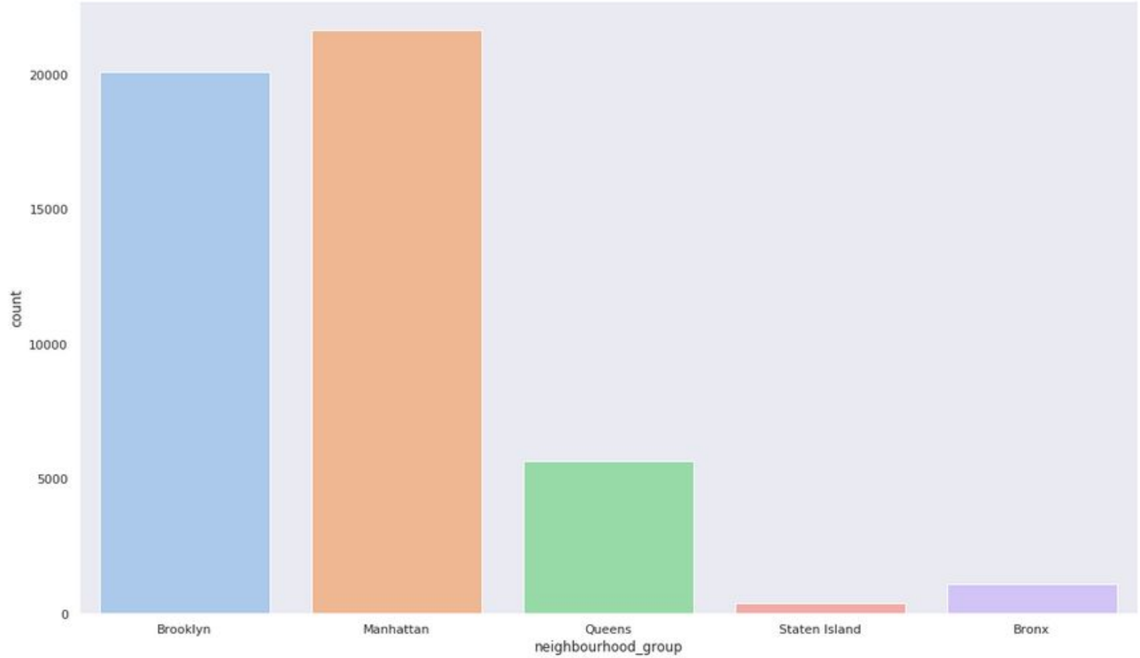
Total Room Types

```
f,ax = plt.subplots(figsize=(15,10))
ax = sns.countplot(data.room_type,palette="pastel")
plt.show()
```



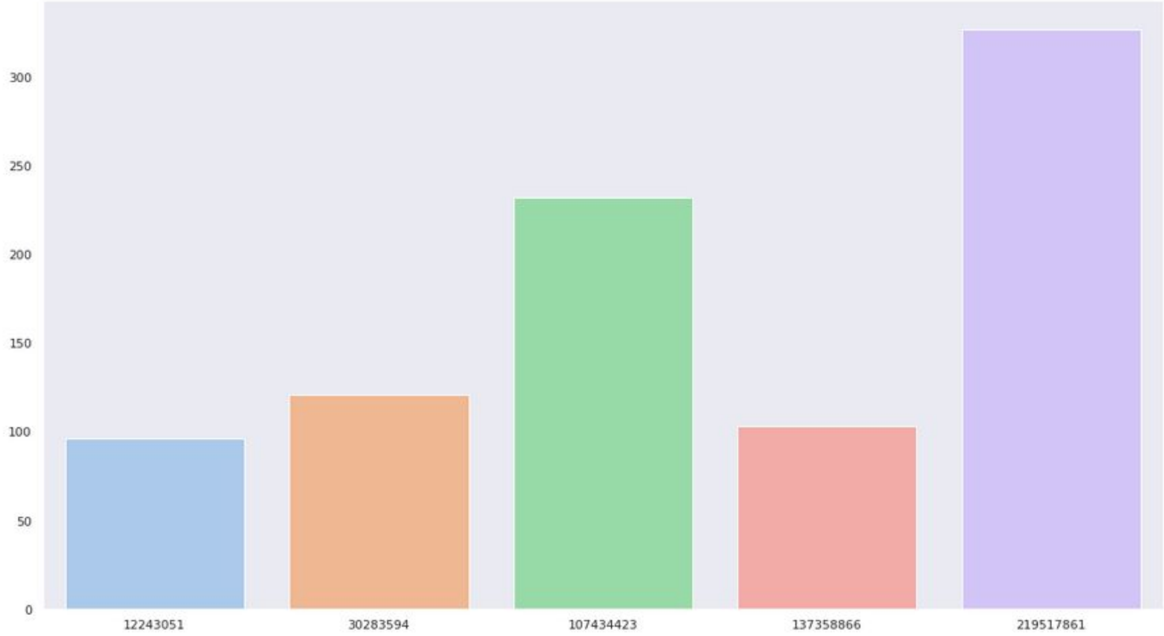
Total Neighbourhood Group Listings

```
f,ax = plt.subplots(figsize=(15,10))
ax = sns.countplot(data.neighbourhood_group,palette="pastel")
plt.show()
```



User ID’s With Most Listings

```
df1 = data.host_id.value_counts()[:5]
f,ax = plt.subplots(figsize=(16,10))
ax = sns.barplot(x = df1.index,y=df1.values,palette="pastel")
plt.show()
```



PREPROCESSING

FEATURES

- **Listing ID**
- **Title of Listing**
- **Host ID**
- **Host Name**
- **Neighbourhood Group**
  - E.g. Manhattan, Brooklyn, etc.
- **Neighbourhood**
- **Latitude**
- **Longitude**
- **Room Type**
- **Price**
- **Minimum Nights**
- **Number of Reviews**
- **Date of Last Review**
- **Reviews Per Month**
- **Number of Listings Host has**
- **Availability**
  - Days out of the year that the listing is on the site

More Info On Dataframe

```python
data.shape
```

```
(48895, 16)
```

```python
print(data.describe())
```

```
                 id       host_id      latitude     longitude         price
count  4.889500e+04  4.889500e+04  48895.000000  48895.000000  48895.000000
mean   1.901714e+07  6.762001e+07     40.728949    -73.952170    152.720687
std    1.098311e+07  7.861097e+07      0.054530      0.046157    240.154170
min    2.539000e+03  2.438000e+03     40.499790    -74.244420      0.000000
25%    9.471945e+06  7.822033e+06     40.690100    -73.983070     69.000000
50%    1.967728e+07  3.079382e+07     40.723070    -73.955680    106.000000
75%    2.915218e+07  1.074344e+08     40.763115    -73.936275    175.000000
max    3.648724e+07  2.743213e+08     40.913060    -73.712990  10000.000000

       minimum_nights  number_of_reviews  reviews_per_month  \
count    48895.000000       48895.000000       38843.000000
mean         7.029962          23.274466           1.373221
std         20.510550          44.550582           1.680442
min          1.000000           0.000000           0.010000
25%          1.000000           1.000000           0.190000
50%          3.000000           5.000000           0.720000
75%          5.000000          24.000000           2.020000
max       1250.000000         629.000000          58.500000

       calculated_host_listings_count  availability_365
count                    48895.000000      48895.000000
mean                         7.143982        112.781327
std                         32.952519        131.622289
min                          1.000000          0.000000
25%                          1.000000          0.000000
50%                          1.000000         45.000000
75%                          2.000000        227.000000
max                        327.000000        365.000000
```

Null Values

```
data.isnull().sum()
```

```
id                                    0
name                                 16
host_id                               0
host_name                            21
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                       10052
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

FEATURES TO GET RID OF

- Listing ID
- Title of Listing
- Host ID
- Host Name
- Neighbourhood Group
  - E.g. Manhattan, Brooklyn, etc.
- Neighbourhood
- Latitude
- Longitude
- Room Type
- Price
- Minimum Nights
- Number of Reviews
- Date of Last Review
- Reviews Per Month
- Number of Listings Host has
- Availability
  - Days out of the year that the listing is on the site

FEATURES TO GET RID OF

- **Listing ID**
  - Is unique to each record
  - Doesn't contain any valuable information
- **Title of Listing**
  - Mostly contains type of room and location; Not very relevant
- **Host Name**
- **Date of Last Review**
  - Contains 10,052 null values
  - Around 1/5th of the values are null

FEATURES KEPT

- **Listing ID**
- **Title of Listing**
- **Host ID**
- **Host Name**
- **Neighbourhood Group**
  - E.g. Manhattan, Brooklyn, etc.
- **Neighbourhood**
- **Latitude**
- **Longitude**
- **Room Type**
- **Price**
- **Minimum Nights**
- **Number of Reviews**
- **Date of Last Review**
- **Reviews Per Month**
- **Number of listings host has**
- **Availability**
  - Days out of the year that the listing is on the site

CATEGORICAL VS CONTINUOUS FEATURES

- **Host ID**
- **Neighbourhood Group**
  - E.g. Manhattan, Brooklyn, etc.
- **Neighbourhood**
- **Latitude**
- **Longitude**

- **Room Type**
- **Price**
- **Minimum Nights**
- **Number of Reviews**
- **Reviews Per Month**
- **Host Listings Count**
  - Number of listings host has
- **Availability**
  - Days out of the year that the listing is on the site

Removing Outliers

```
def detect_outlier(df):

    threshold=3
    mean_1 = np.mean(df)
    std_1 =np.std(df)


    for y in df:
        z_score= (y - mean_1)/std_1
        if np.abs(z_score) > threshold:
            df.replace(y, np.nan, inplace=True)
```

```
detect_outlier(df.price)
detect_outlier(df.number_of_reviews)
detect_outlier(df.minimum_nights)
```

```
dropdata = df.dropna()
```

```
dropdata.isnull().sum()
```

```
host_id                           0
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

```
dropdata.shape
```

```
(46981, 12)
```

OBJECTIVES

Predict which neighborhood group a listing belongs to.

- **Random Forest Classifier**

Unsupervised Clustering.

- **K-Means Clustering**

Predict Price.

- **Linear Regression**

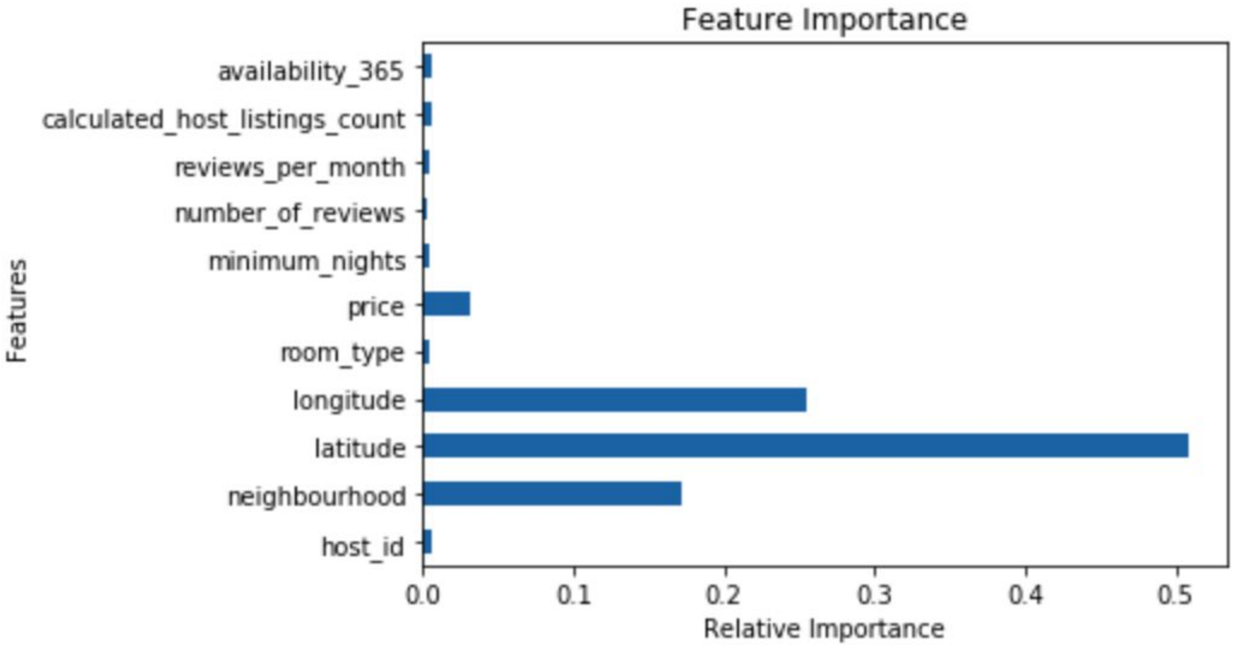
ALGORITHMS

RANDOM FOREST CLASSIFIER

```
clf = RandomForestClassifier(n_estimators=20)
clf.fit(x_train, y_train)
pred_train = clf.predict(x_train)
pred_test = clf.predict(x_test)


from sklearn.metrics import accuracy_score
print("Accuracy: ", accuracy_score(y_test, pred_test))
```

```
Accuracy:  0.9995661874070402
```

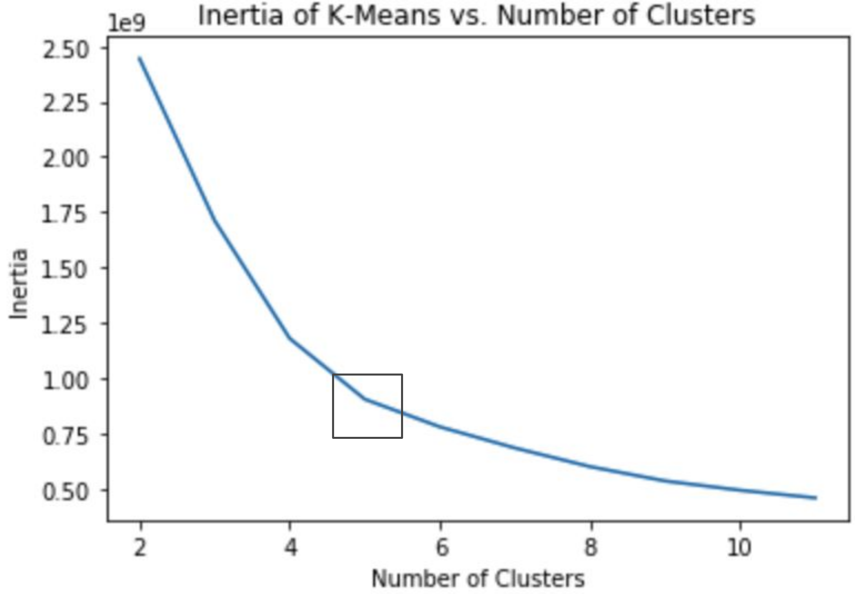
RANDOM FOREST CLASSIFIER



Target:
Neighbourhood Group

```
feat_importances = pd.Series(clf.feature_importances_, index=x.columns)
feat_importances.plot(kind='barh')
plt.title('Feature Importance')
plt.xlabel('Relative Importance')
plt.ylabel('Features')
```

K-MEANS CLUSTERING - Choosing K



Note: Inertia is a measure of how internally coherent clusters are. This is to be minimized.

```
scores = [KMeans(n_clusters=i+2).fit(df).inertia_ for i in range(10)]
sns.lineplot(np.arange(2, 12), scores)
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title("Inertia of K-Means vs. Number of Clusters")
```

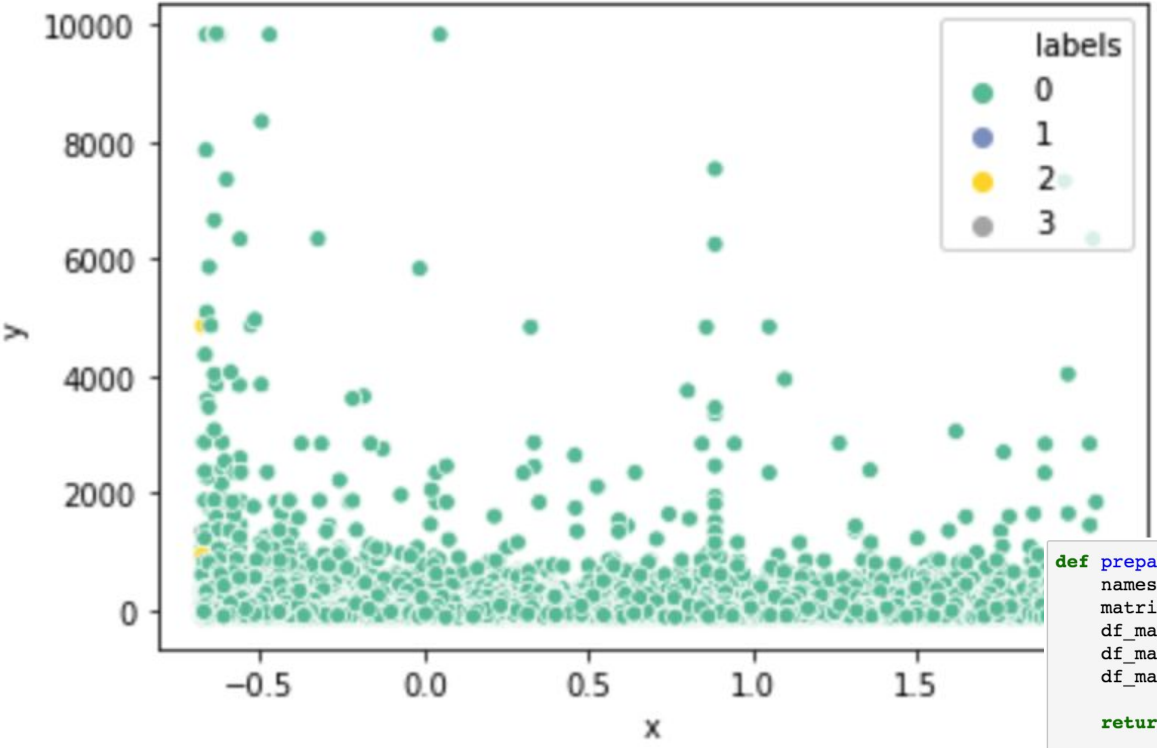
K-MEANS CLUSTERING - Silhouette Score

```
# K-Means
kmeans = KMeans(n_clusters=5).fit(df)

# Cosine K-Means
normalized_vectors = preprocessing.normalize(df)
normalized_kmeans = KMeans(n_clusters=4).fit(normalized_vectors)

# Show associated silhouette score (ranges from -1 to 1; 1 is ideal)
print('kmeans: ', silhouette_score(df, kmeans.labels_, metric='euclidean'))
print('cosine kmeans: ', silhouette_score(normalized_vectors, normalized_k
```

Note: The Silhouette Score is a measure of how similar an object is to its own cluster compared to other clusters. It ranges from -1 to 1 and 1 is ideal.

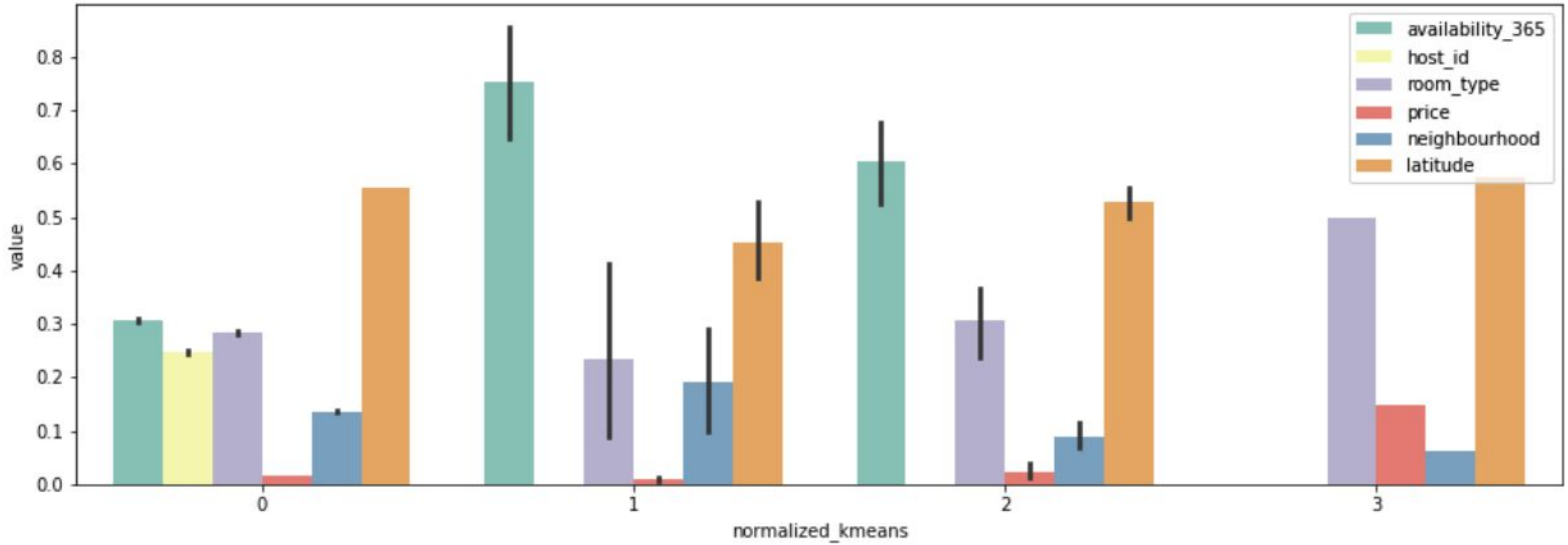
```
kmeans:  0.6325746291550376
cosine kmeans:  0.9975497025477699
```

K-MEANS CLUSTERING - Visualization with PCA




```python
def prepare_pca(n_components, data, kmeans_labels):
    names = ['x', 'y', 'z']
    matrix = PCA(n_components=n_components).fit_transform(data)
    df_matrix = pd.DataFrame(matrix)
    df_matrix.rename({i:names[i] for i in range(n_components)}, axis=1, in
    df_matrix['labels'] = kmeans_labels

    return df_matrix

pca_df = prepare_pca(3, df, normalized_kmeans.labels_)
sns.scatterplot(x=pca_df.x, y=pca_df.y, hue=pca_df.labels, palette="Set2")
```

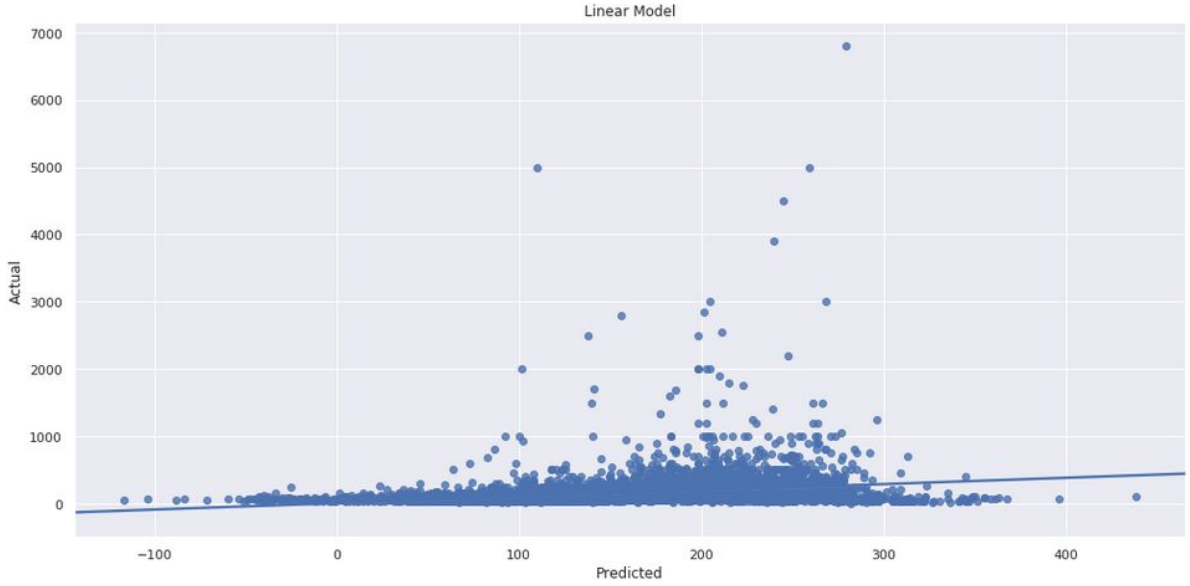
K-MEANS CLUSTERING - Visualization with Bar Plot



LINEAR REGRESSION

```
Mean Squared Error: 181.810009264
R2 Score: 9.81971075263
```

```
plt.figure(figsize=(15,8))
sns.regplot(predicts, y_test)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Linear Model")
plt.grid(True)
plt.show()
```



Linear Regression(Outlier Removal)

```
Mean Squared Error: 97.2393246153
R2 Score: 23.7025898533
```

```
plt.figure(figsize=(15,8))
sns.regplot(predicts, y_test)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Linear Model")
plt.grid(True)
plt.show()
```