Proj_Airbnb_Houseing_Price

October 19, 2016

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import seaborn as sns; sns.set(font_scale=1.7)
import re
from datetime import datetime as dt
%matplotlib inline
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

0.1 Data Cleaning and Exploration

In [3]:
```
## Reads in data
raw = pd.read_csv('Data/London_listings.csv')
raw.shape
```

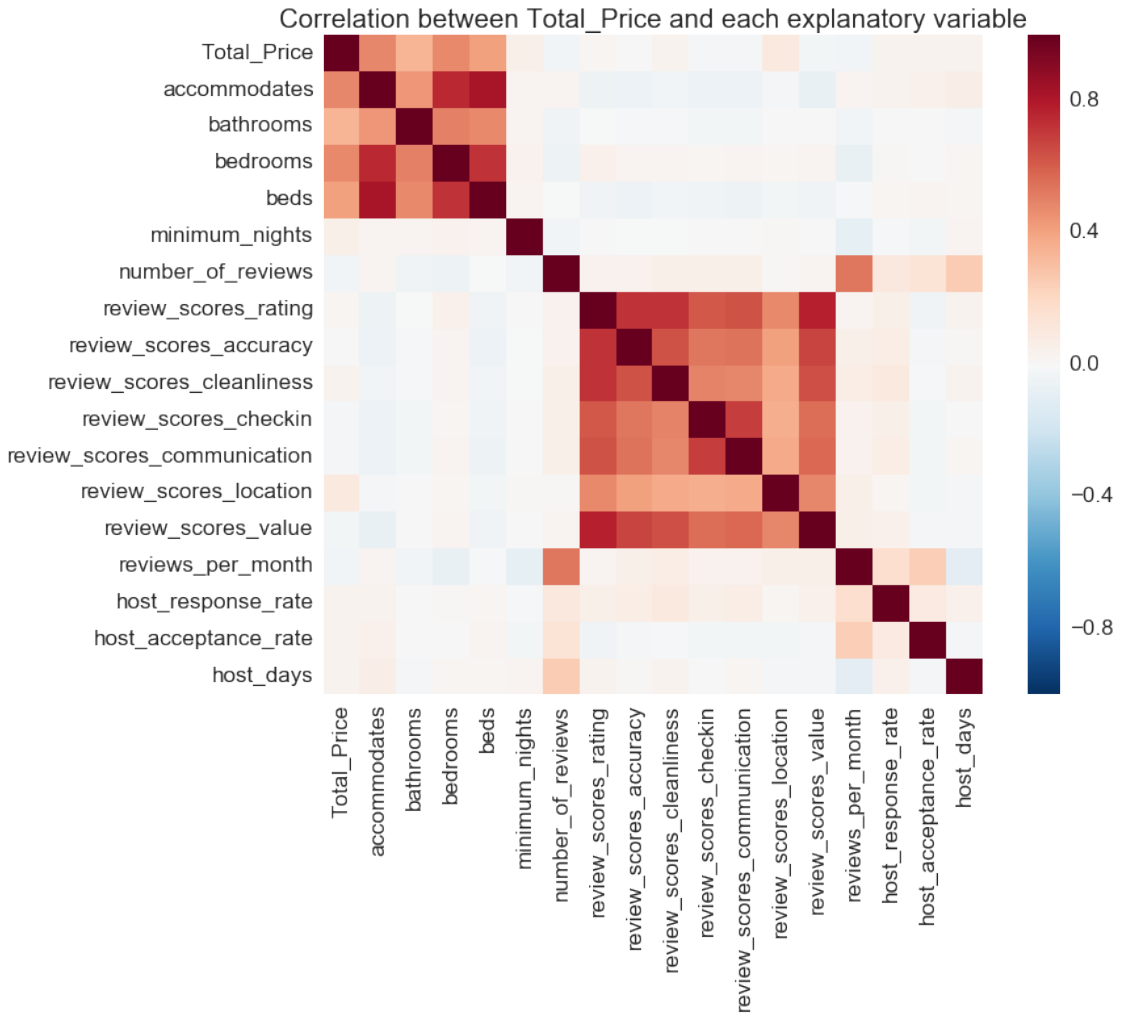
Out[3]:
```
(42646, 95)
```

In [4]:
```
records = raw.shape[0]
features = raw.shape[1]
```

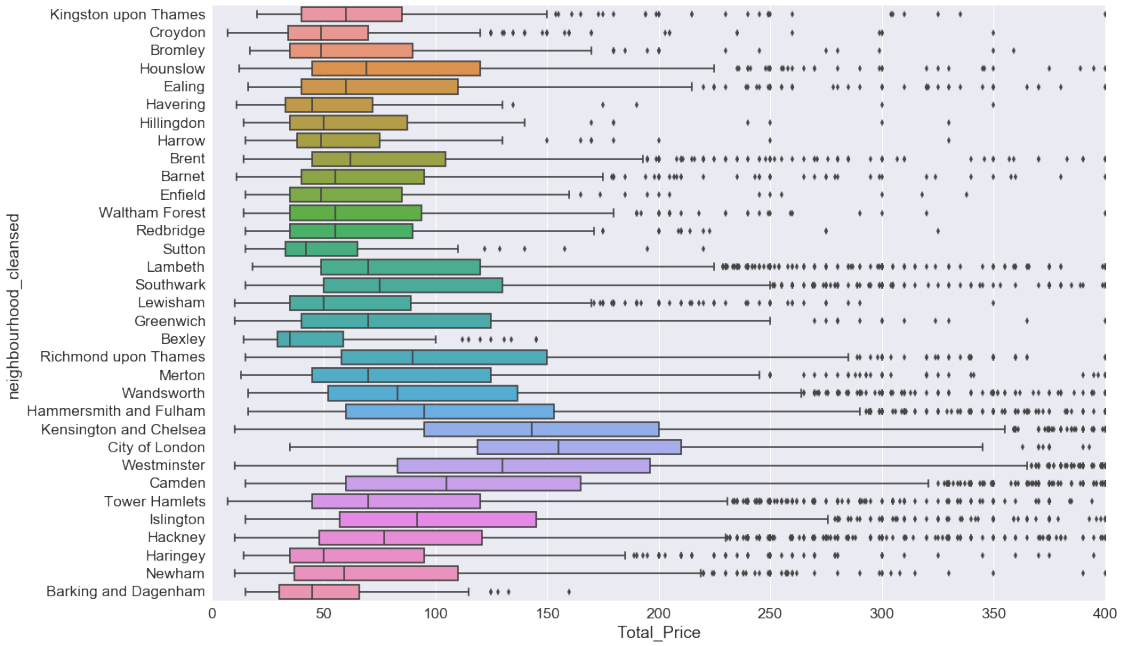
In [5]:
```
## Columns to start with, no need for cleaning

starting_columns = ['accommodates',
                    'bathrooms',
                    'bedrooms',
                    'beds',
                    'minimum_nights',
                    'number_of_reviews',
                    'review_scores_rating',
                    'review_scores_accuracy',
                    'review_scores_cleanliness',
                    'review_scores_checkin',
                    'review_scores_communication',
                    'review_scores_location',
                    'review_scores_value',
                    'reviews_per_month']
```

```
In [6]: ## Use columns listed above, get a copy of raw data for future ML
        data = raw[starting_columns].copy()

In [7]: ## Define a function to convert price related data from string to float
        def convert_prices(col):
            p = raw[col].astype(str).str.strip('$')
            New_Price = np.zeros(records)
            for i in range(records):
                try:
                    New_Price[i] = np.float64(p[i])
                except ValueError:
                    New_Price[i] = np.float64(p[i].replace(',',''))
            return New_Price

In [8]: data['price'] = convert_prices('price')

        data['cleaning_fee'] = convert_prices('cleaning_fee')

In [9]: ## Define a function to convert rate related data from string to float
        def convert_rate(col):
            r = raw[col].astype(str).str.strip('%')
            New_Rate = np.zeros(records)
            for i in range(records):
                try:
                    New_Rate[i] = np.float64(r[i])/ 100
                except ValueError:
                    pass
            return New_Rate

In [10]: data['host_response_rate'] = convert_rate('host_response_rate')

         data['host_acceptance_rate'] = convert_rate('host_acceptance_rate')

In [11]: data['cleaning_fee'] = data['cleaning_fee'].fillna(0)
         data['Total_Price'] = data['price'] + data['cleaning_fee']

In [12]: plt.figure(figsize = (15, 9))
         sns.distplot(data.ix[data['Total_Price'] <400, 'Total_Price'],bins = 40, c
         plt.title('Total Price Distribution')

Out[12]: <matplotlib.text.Text at 0x12a24bd30>
```


```
In [13]: col = data.columns.tolist()
         col_new = col[-1:]+ col[:-1]
         data = data[col_new]

In [14]: ## Convert time related data from string to standard datetime format
         host_since_dt = []
         for t in raw['host_since']:
             try:
                 host_since_dt.append(dt.strptime(t, '%Y-%m-%d'))
             except TypeError:
                 host_since_dt.append(dt(2016,6,2))

In [15]: scraped_date = dt(2016, 6, 2)
         data['host_days'] = [ (scraped_date - time).days for time in  host_since_d

In [16]: corr = data.drop(['price', 'cleaning_fee'], axis = 1).select_dtypes(includ
         plt.figure(figsize=(12, 9))
         sns.heatmap(corr, vmax=1, square=True)
         ax = plt.axes()
         ax.set_title('Correlation between Total_Price and each explanatory variabl

Out[16]: <matplotlib.text.Text at 0x129bb17b8>
```

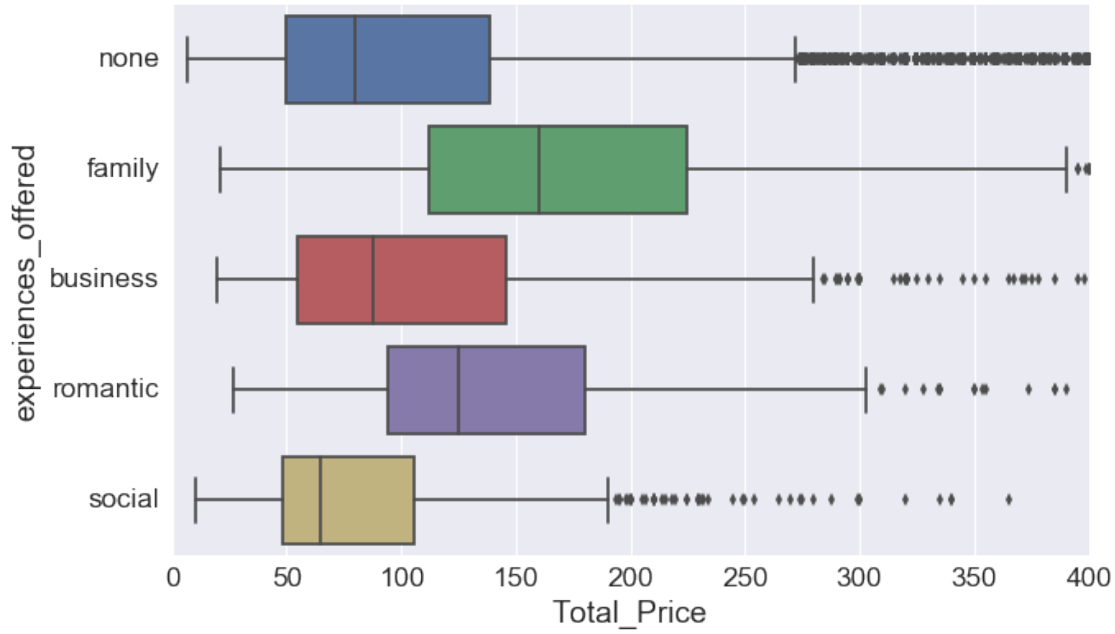



```
In [17]: ## List all the nominal features
         dummies =['experiences_offered', 'neighbourhood_cleansed', 'room_type', 'h

In [18]: data = pd.concat((data, raw[dummies]), axis = 1)

In [19]: data_new = data[data['Total_Price'] <=400]

In [20]: plt.figure(figsize = (18, 12))
         sns.boxplot(y = dummies[1], x = 'Total_Price',  data = data_new, )

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x129736710>
```



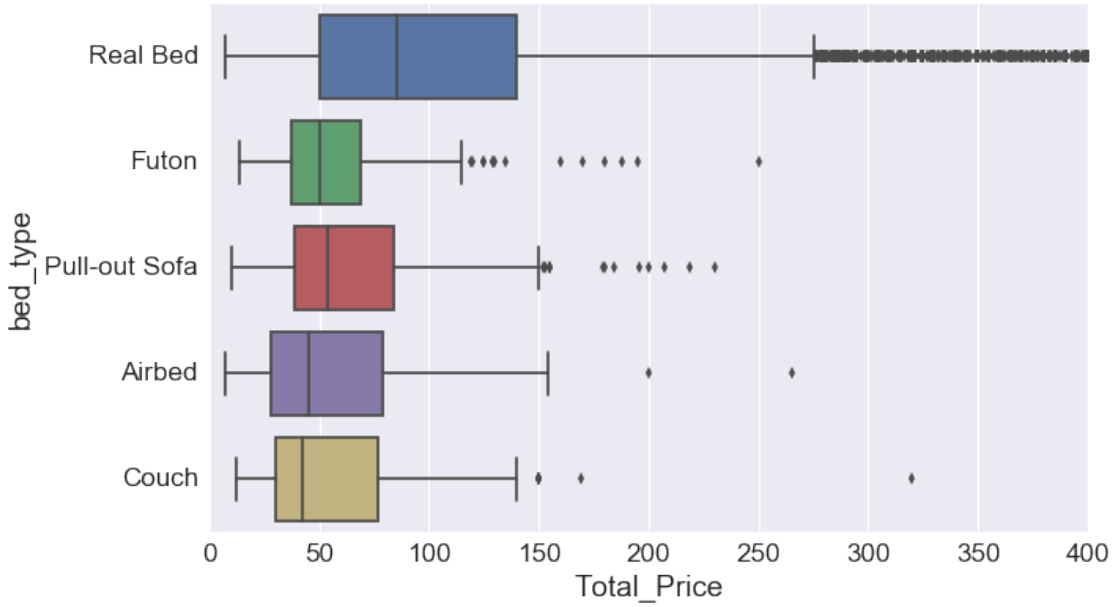

```
In [21]: plt.figure(figsize = (10, 6))
         sns.boxplot(y = dummies[0], x = 'Total_Price',  data = data_new, )
```

```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x129a5fe48>
```



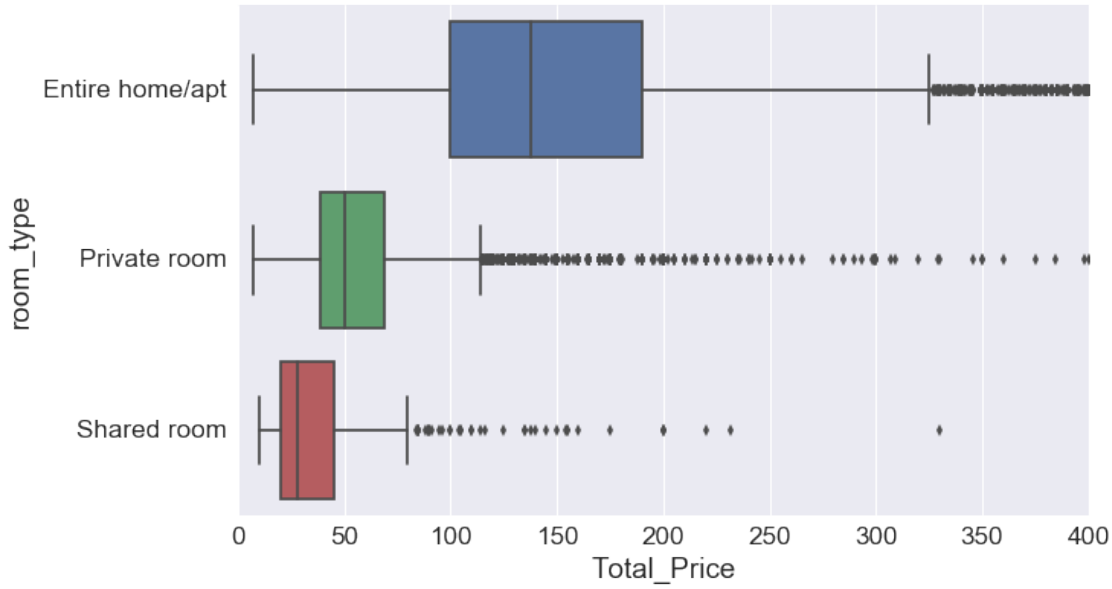
```
In [22]: plt.figure(figsize = (10, 6))
         sns.boxplot(y = dummies[3], x = 'Total_Price',  data = data_new, )
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x129508e48>
```

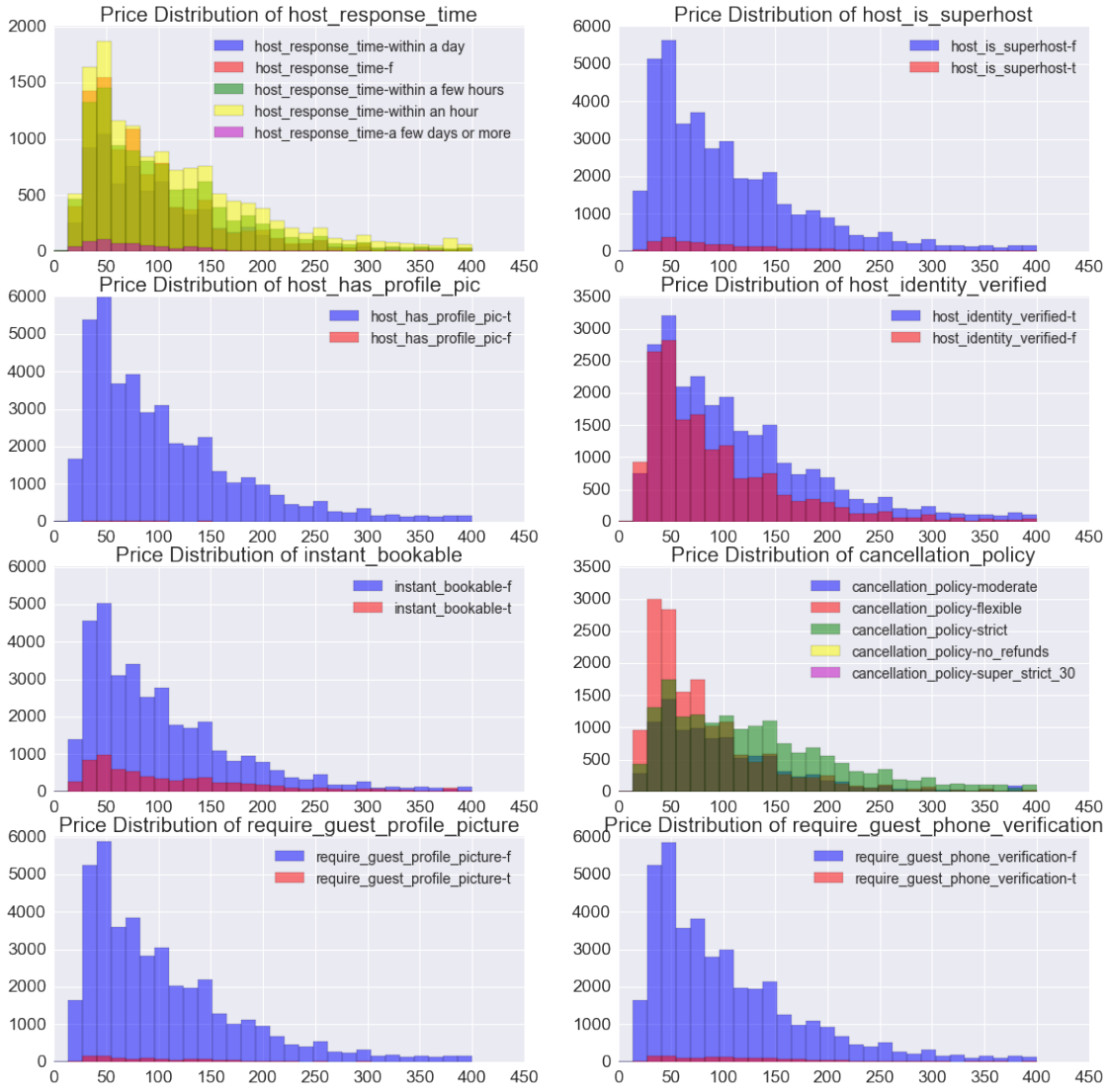



```
In [23]: plt.figure(figsize = (10, 6))
         sns.boxplot(y = dummies[2], x = 'Total_Price',  data = data_new, )
```

```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x129f63748>
```



```
In [24]: ## List all the ordinal features
         labels = ['host_response_time',
                   'host_is_superhost',
                   'host_has_profile_pic',
                   'host_identity_verified',
                   'instant_bookable',
                   'cancellation_policy',
                   'require_guest_profile_picture',
                   'require_guest_phone_verification']

In [25]: data = pd.concat((data, raw[labels]), axis = 1)
         data_new = data[data['Total_Price']<= 400]

In [26]: data_new[labels] = data_new[labels].fillna('f')

In [27]: X = data_new['Total_Price']
         y = data_new[labels[1]]
         colors = ['blue', 'red', 'green', 'yellow', 'm']
         bins = np.linspace(0, 400, 10)
         cat = y.unique()
         n_cat = len(cat)

In [28]: X = data_new['Total_Price']
         colors = ['blue', 'red', 'green', 'yellow', 'm']
         bins = np.linspace(0, 400, 30)

         fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(18,18))
         for ax, cnt in zip(axes.ravel(), range(8)):
             y = data_new[labels[cnt]]
             cat = y.unique()
             n_cat = len(cat)
             # plottling the histograms
             for lab, col in zip(cat, colors[:n_cat]):
                 ax.hist(X[y == lab], color=col, label='%s-%s' % (labels[cnt], lab)
             ylims = ax.get_ylim()

             # plot annotation
             leg = ax.legend(loc='upper right', fancybox=True, fontsize=14)
             leg.get_frame().set_alpha(0.5)
             ax.set_ylim([0, max(ylims)+2])
             ax.set_title('Price Distribution of %s' % labels[cnt])
```




```
In [29]: data = data.drop(dummies, axis = 1)
```

```
In [30]: ## Define a function to convert nominal features into dummy variables
         def add_dummies(col):
             return pd.get_dummies(raw[col], prefix = col)
```

```
In [31]: ## List all the nominal features
         dummies =['experiences_offered', 'neighbourhood_cleansed', 'room_type', 'h
```

```
In [32]: for col in dummies:
             col_dummies = add_dummies(col)
             data = pd.concat([data, col_dummies], axis =1 )
```

```
In [33]: ## Define a function to do the mapping for ordinal features
         def encode_labels(col, mapping):
             return raw[col].map(mapping)
```

```
In [34]: ## Define mapping rules for ordinal features listed above, in order
         mapping  = [{'within an hour': 0, 'within a few hours': 1, 'within a day':
                     {'f':0, 't':1},
                     {'f':0, 't':1},
                     {'f':0, 't':1},
                     {'f':0, 't':1},
                     {'strict': 1, 'super_strict_60':1, 'super_strict_30':1, 'no_re
                     {'f':0, 't':1},
                     {'f':0, 't':1}]

In [35]: for (col, mapping) in zip(labels, mapping):
             data[col] = encode_labels(col, mapping)

In [39]: data[:10000].to_csv('data_cleansed.csv')
```