RentalListingVisualizations

April 4, 2017

1 Initializing Dataset

We must initialize the dataset first, loading it in as a pandas object from a JSON object.

```
In [43]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns

         colors = sns.color_palette()

         trainData = pd.read_json("train.json")
         testData = pd.read_json("test.json")
         trainData.head()
```

```
Out[43]:         bathrooms  bedrooms                       building_id  \
         10            1.5         3  53a5b119ba8f7b61d4e010512e0dfc85
         10000         1.0         2  c5c8a357cba207596b04d1afd1e4f130
         100004        1.0         1  c3ba40552e2120b0acfc3cb5730bb2aa
         100007        1.0         1  28d9ad350afeaab8027513a3e52ac8d5
         100013        1.0         4                                 0

                             created  \
         10      2016-06-24 07:54:24
         10000   2016-06-12 12:19:27
         100004  2016-04-17 03:26:41
         100007  2016-04-18 02:22:02
         100013  2016-04-28 01:32:41

                                                      description  \
         10      A Brand New 3 Bedroom 1.5 bath ApartmentEnjoy ...
         10000
         100004  Top Top West Village location, beautiful Pre-w...
         100007  Building Amenities - Garage - Garden - fitness...
         100013  Beautifully renovated 3 bedroom flex 4 bedroom...

                     display_address  \
```

```
10        Metropolitan Avenue
10000         Columbus Avenue
100004            W 13 Street
100007       East 49th Street
100013      West 143rd Street

                                                features interest_level
10                                                    []         medium
10000   [Doorman, Elevator, Fitness Center, Cats Allow...           low
100004  [Laundry In Building, Dishwasher, Hardwood Flo...          high
100007                        [Hardwood Floors, No Fee]             low
100013                                        [Pre-War]             low

        latitude  listing_id  longitude                        manager_id
10       40.7145     7211212   -73.9425  5ba989232d0489da1b5f2c45f6688adc
10000    40.7947     7150865   -73.9667  7533621a882f71e25173b27e3139d83d
100004   40.7388     6887163   -74.0018  d9039c43983f6e564b1482b273bd7b01
100007   40.7539     6888711   -73.9677  1067e078446a7897d2da493d2f741316
100013   40.8241     6934781   -73.9493  98e13ad4b495b9613cef886d79a6291f

                                                   photos  price  \
10      [https://photos.renthop.com/2/7211212_1ed4542e...   3000
10000   [https://photos.renthop.com/2/7150865_be3306c5...   5465
100004  [https://photos.renthop.com/2/6887163_de85c427...   2850
100007  [https://photos.renthop.com/2/6888711_6e660cee...   3275
100013  [https://photos.renthop.com/2/6934781_1fa4b41a...   3350

                 street_address
10      792 Metropolitan Avenue
10000       808 Columbus Avenue
100004          241 W 13 Street
100007     333 East 49th Street
100013    500 West 143rd Street
```

In [44]: `testData.head()`

```
Out[44]:         bathrooms  bedrooms                       building_id  \
         0             1.0         1  79780be1514f645d7e6be99a3de696c5
         1             1.0         2                                 0
         100           1.0         1  3dbbb69fd52e0d25131aa1cd459c87eb
         1000          1.0         2  783d21d013a7e655bddc4ed0d461cc5e
         100000        2.0         2  6134e7c4dd1a98d9aee36623c9872b49

                             created  \
         0       2016-06-11 05:29:41
         1       2016-06-24 06:36:34
         100     2016-06-03 04:29:40
         1000    2016-06-11 06:17:35
```

```
100000  2016-04-12 05:24:17

                                                  description  \
0       Large with awesome terrace--accessible via bed...
1       Prime Soho - between Bleecker and Houston - Ne...
100             New York chic has reached a new level ...
1000    Step into this fantastic new Construction in t...
100000  ~Take a stroll in Central Park, enjoy the ente...

              display_address  \
0               Suffolk Street
1              Thompson Street
100        101 East 10th Street
1000       South Third Street\r
100000  Midtown West, 8th Ave

                                                     features  latitude  \
0       [Elevator, Laundry in Building, Laundry in Uni...   40.7185
1                    [Pre-War, Dogs Allowed, Cats Allowed]   40.7278
100                            [Doorman, Elevator, No Fee]   40.7306
1000    [Roof Deck, Balcony, Elevator, Laundry in Buil...   40.7109
100000  [Common Outdoor Space, Cats Allowed, Dogs Allo...   40.7650

        listing_id  longitude                        manager_id  \
0          7142618   -73.9865  b1b1852c416d78d7765d746cb1b8921f
1          7210040   -74.0000  d0b5648017832b2427eeb9956d966a14
100        7103890   -73.9890  9ca6f3baa475c37a3b3521a394d65467
1000       7143442   -73.9571  0b9d5db96db8472d7aeb67c67338c4d2
100000     6860601   -73.9845  b5eda0eb31b042ce2124fd9e9fcfce2f

                                                   photos  price  \
0       [https://photos.renthop.com/2/7142618_1c45a2c8...   2950
1       [https://photos.renthop.com/2/7210040_d824cc71...   2850
100     [https://photos.renthop.com/2/7103890_85b33077...   3758
1000    [https://photos.renthop.com/2/7143442_0879e9e0...   3300
100000  [https://photos.renthop.com/2/6860601_c96164d8...   4900

                  street_address
0                99 Suffolk Street
1              176 Thompson Street
100           101 East 10th Street
1000    251  South Third Street\r
100000        260 West 54th Street
```

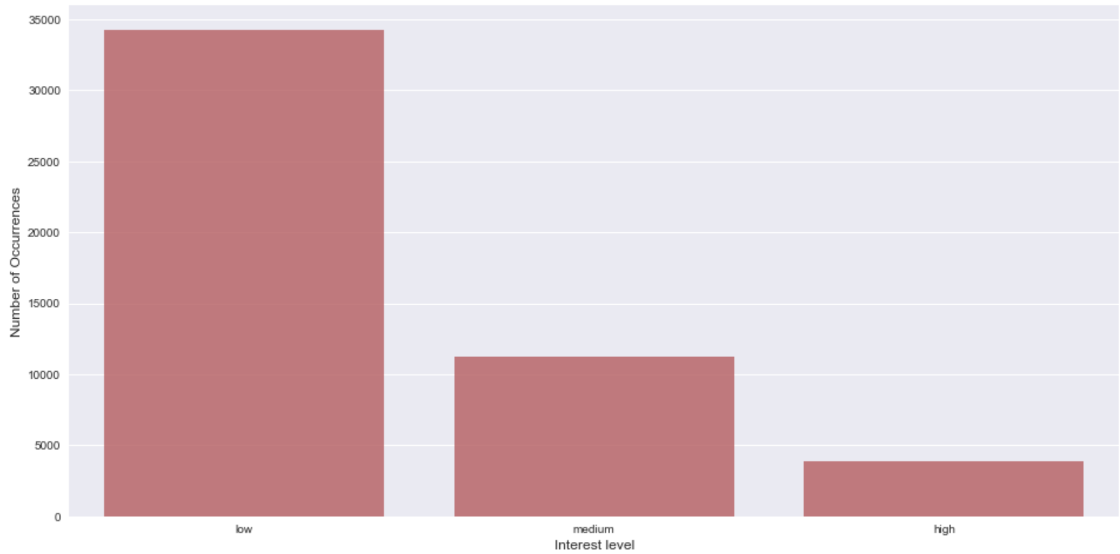
Notice that the training set and the test set have the same factors besides interest_level. That's because interest_level is our target variable, and we'll use it for comparison and analysis in the following below.

2 Visualizing Data

We want to start taking a look at what variables have most influence over the target variable. So, we want to compare the distributions of the varying factors.

2.0.1 Interest Levels

```
In [45]: interestLevelCounts = trainData['interest_level'].value_counts()

         plt.figure(figsize=(16,8))
         sns.barplot(interestLevelCounts.index, interestLevelCounts.values, alpha=0
         plt.ylabel('Number of Occurrences', fontsize=12)
         plt.xlabel('Interest level', fontsize=12)
         plt.show()
```

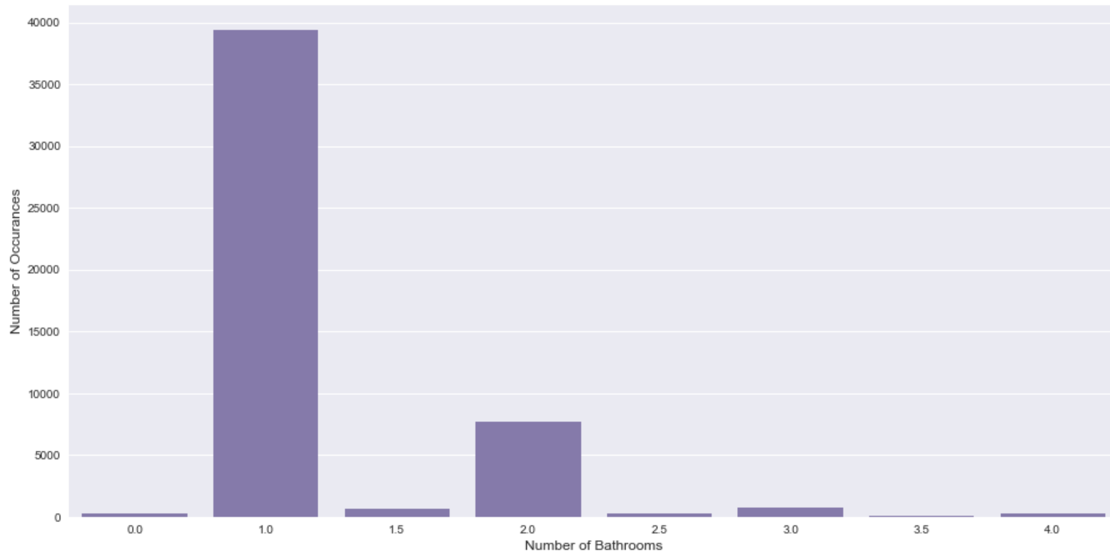


The amount of 'low' interest level houses outnumbers the amount of 'medium' and 'high' occurances. This means the distribution is a bit unbalanced, and we need to be careful when analyzing direct numbers of the factors. We may want to consider using percentages when comparing variables.

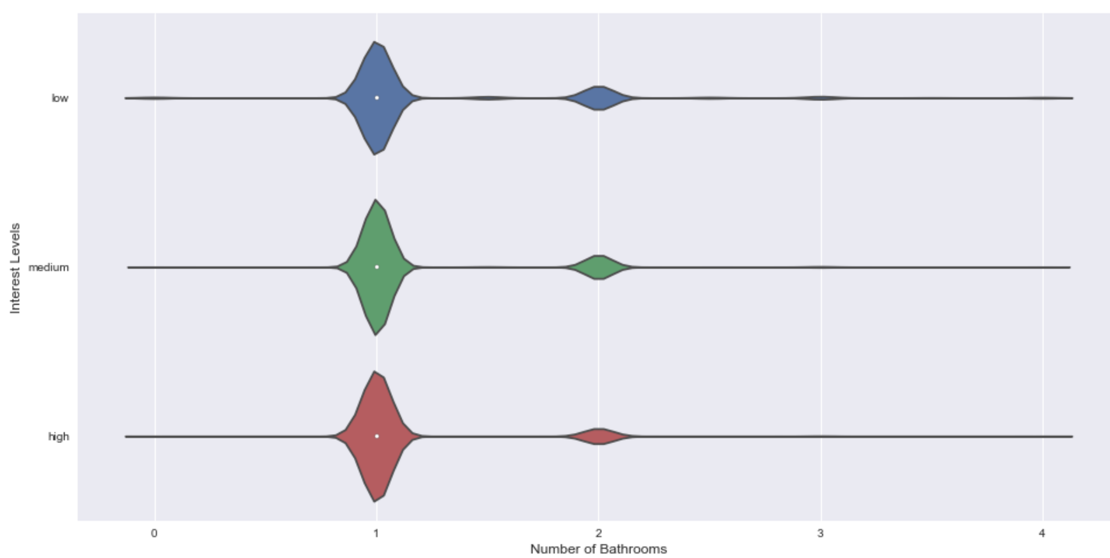
2.0.2 Bathrooms

```
In [46]: # Normalizing the outliers
         outlierBathroomRows = trainData['bathrooms'] > 4
         trainData.loc[outlierBathroomRows, 'bathrooms'] = 4

         bathroomCount = trainData['bathrooms'].value_counts()

         plt.figure(figsize=(16,8))
         sns.barplot(bathroomCount.index, bathroomCount.values, color=colors[3])
```

```
plt.ylabel('Number of Occurances', fontsize=12)
plt.xlabel('Number of Bathrooms', fontsize=12)
plt.show()
```



In [47]:
```
#trainData['bathrooms'].ix[trainData['bathrooms'] > 4] = 4

plt.figure(figsize=(16,8))
sns.violinplot(trainData['bathrooms'], trainData['interest_level'], order=
plt.xlabel('Number of Bathrooms', fontsize=12)
plt.ylabel('Interest Levels', fontsize=12)
plt.show()
```



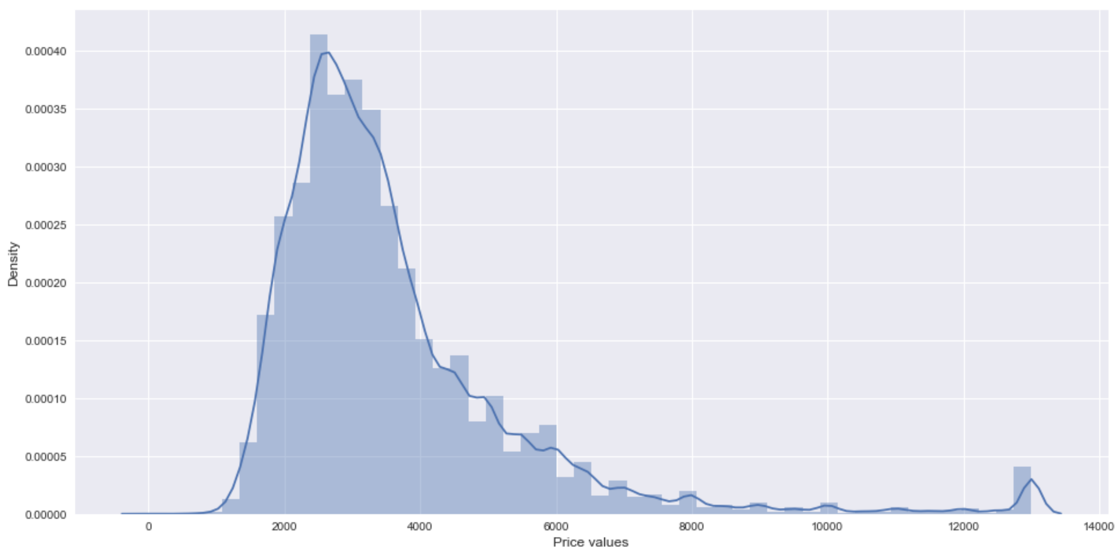
The number of bathrooms appears to be pretty consistent across interest levels. So bathrooms will not be a very important factor in analysis for the target variable, at least at first glance.

2.0.3 Prices

Initially after graphing the prices there were a couple of outliers throwing off the visualization, so we left that out in favor of a more useful approach.

```
In [48]: # Normalizing the outliers
         percentile_99_Price = np.percentile(trainData['price'], 99)
         outlierPriceRows = trainData['price'] > percentile_99_Price
         trainData.loc[outlierPriceRows, 'price'] = percentile_99_Price

         plt.figure(figsize=(16,8))
         sns.distplot(trainData['price'])
         plt.xlabel('Price values', fontsize=12)
         plt.ylabel('Density', fontsize=12)
         plt.show()
```

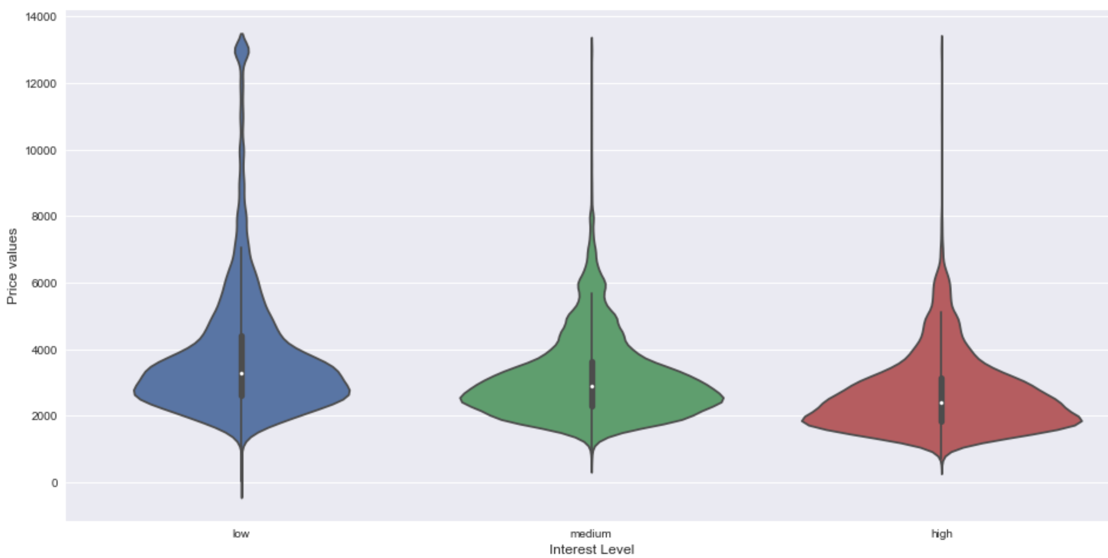
```
/Users/JustinonTG/anaconda/lib/python3.6/site-packages/statsmodels/nonparametric/kc
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```



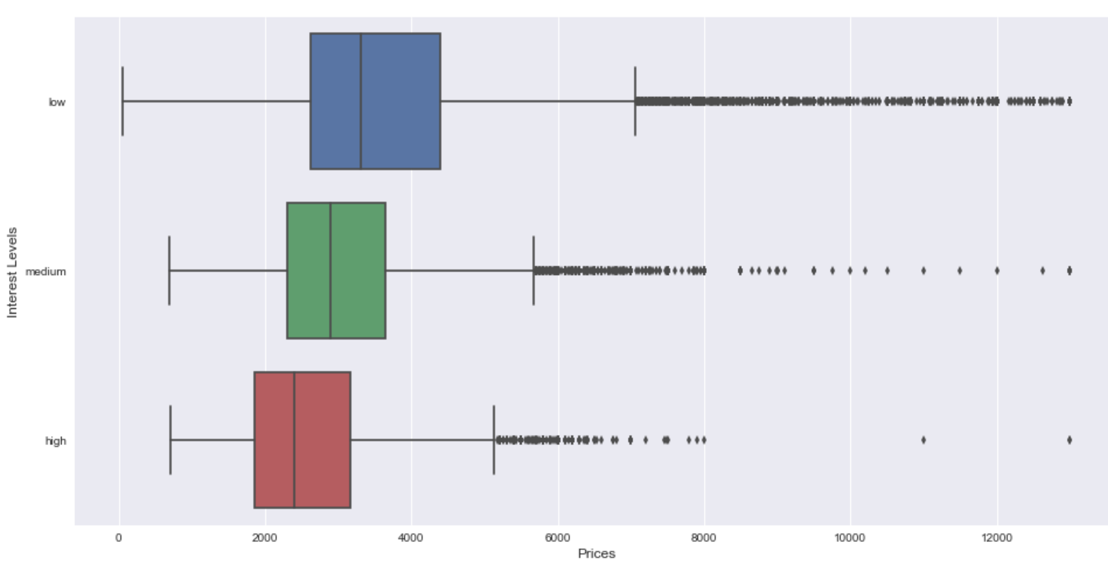
That overall distribution looks quite nice, with a skew right. It should be interesting to visualize the prices against the interest levels.

```
In [49]: plt.figure(figsize=(16,8))
         sns.violinplot(x='interest_level', y='price', data=trainData, order=['low'
```

```
plt.xlabel('Interest Level', fontsize=12)
plt.ylabel('Price values', fontsize=12)
plt.show()
```



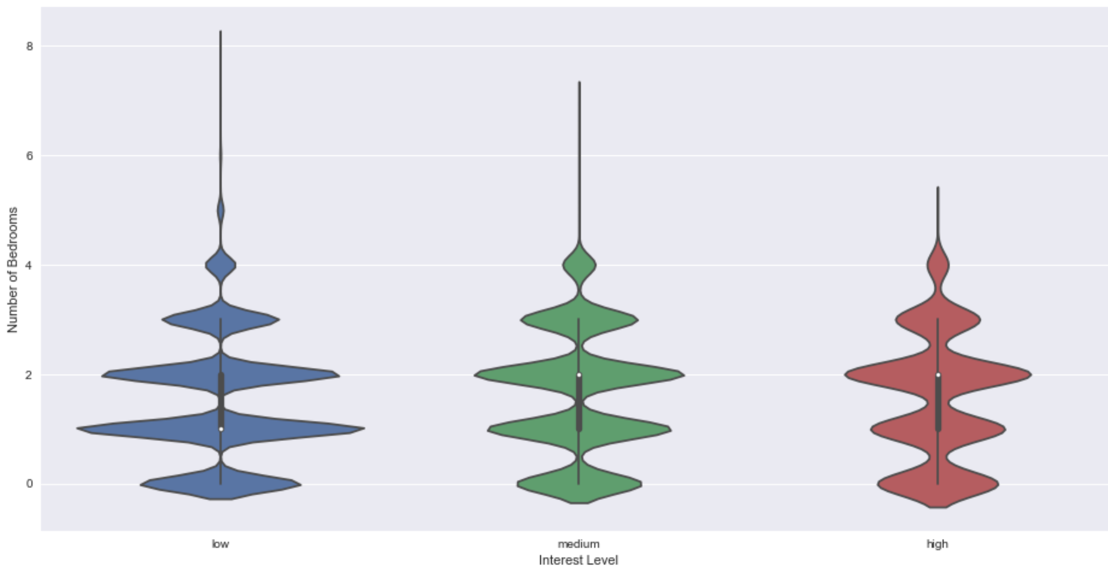

```
In [50]: plt.figure(figsize=(16,8))
         sns.boxplot(x='price', y='interest_level', data=trainData, order=['low','m
         plt.xlabel('Prices', fontsize=12)
         plt.ylabel('Interest Levels', fontsize=12)
         plt.show()
```



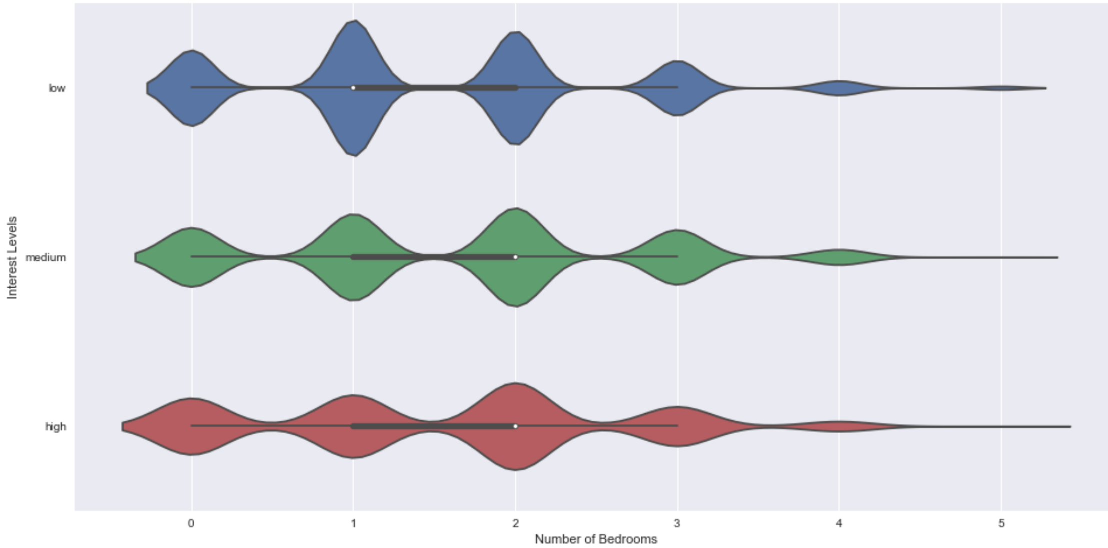
It's easy to notice here that the 'low' interest level flats also are the ones that have the most density in the higher price values. Also, the 'high' interest level flats have a lower average price. This is logical, and helps in our prediction. The price variable therefore should have some more weight than the other factors in predicting the influence level.

2.0.4 Bedrooms

In [51]:
```
plt.figure(figsize=(16,8))
sns.violinplot(x='interest_level', y='bedrooms', data=trainData, order=["l
plt.xlabel('Interest Level')
plt.ylabel('Number of Bedrooms')
plt.show()
```



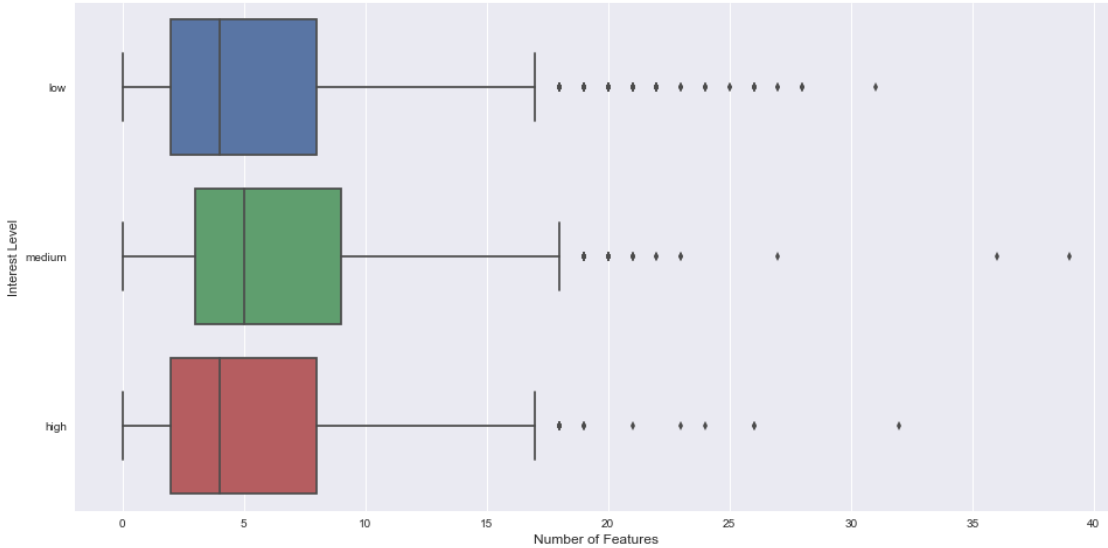
In [62]:
```
# Normalizing the outliers
percentile_995_Bedrooms = np.percentile(trainData['bedrooms'], 99.5)
percentile_005_Bedrooms = np.percentile(trainData['bedrooms'], .5)
outlierHighBedRows = trainData['bedrooms'] > percentile_995_Bedrooms
outlierLowBedRows = trainData['bedrooms'] < percentile_005_Bedrooms
trainData.loc[outlierHighBedRows,'bedrooms'] = percentile_995_Bedrooms
trainData.loc[outlierLowBedRows,'bedrooms'] = percentile_005_Bedrooms

plt.figure(figsize=(16,8))
sns.violinplot(x='bedrooms', y='interest_level', data=trainData, order=["l
plt.xlabel('Number of Bedrooms')
plt.ylabel('Interest Levels')
plt.show()
```



2.0.5 Features

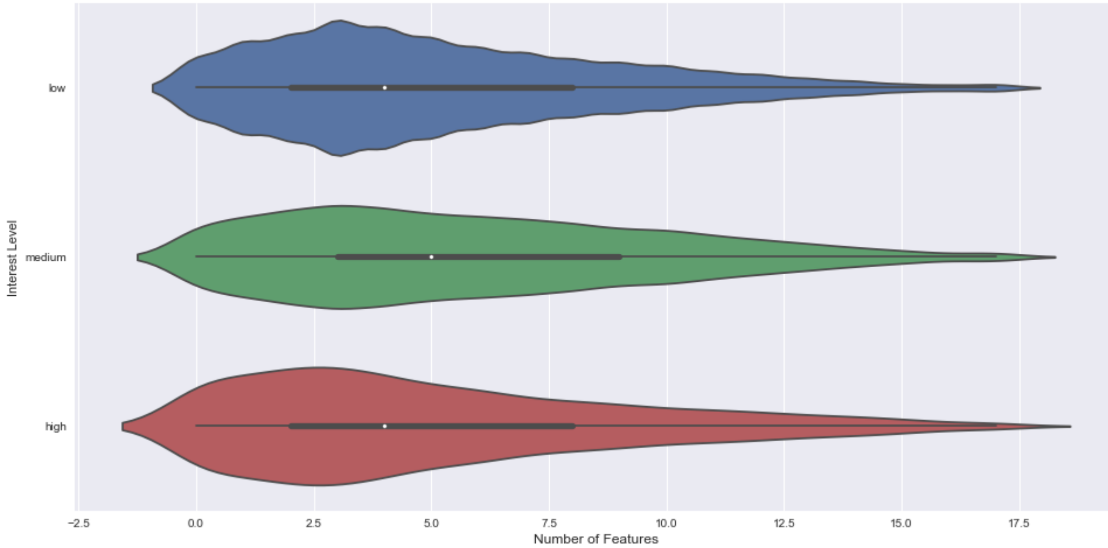
Here we are going to display the various numbers of features and how they're distributed according to the interest level. This will be achieved by counting the number of features provided for each listing and displaying the counts.

```
In [53]: trainData['num_features'] = trainData['features'].apply(len)

         plt.figure(figsize=(16,8))
         listOrder = ['low', 'medium', 'high']
         sns.boxplot(x='num_features', y='interest_level', data=trainData, order=li
         plt.xlabel('Number of Features', fontsize=12)
         plt.ylabel('Interest Level')
         plt.show()
```



Normalizing the outliers yields us the following:

```
In [54]: percentile_995_Features = np.percentile(trainData['num_features'], 99.5)
         percentile_005_Features = np.percentile(trainData['num_features'], .5)
         outlierHighFeatRows = trainData['num_features'] > percentile_995_Features
         outlierLowFeatRows = trainData['num_features'] < percentile_005_Features
         trainData.loc[outlierHighFeatRows,'num_features'] = percentile_995_Feature
         trainData.loc[outlierLowFeatRows,'num_features'] = percentile_005_Features

         plt.figure(figsize=(16,8))
         listOrder = ['low', 'medium', 'high']
         sns.violinplot(x='num_features', y='interest_level', data=trainData, order
         plt.xlabel('Number of Features', fontsize=12)
         plt.ylabel('Interest Level')
         plt.show()
```

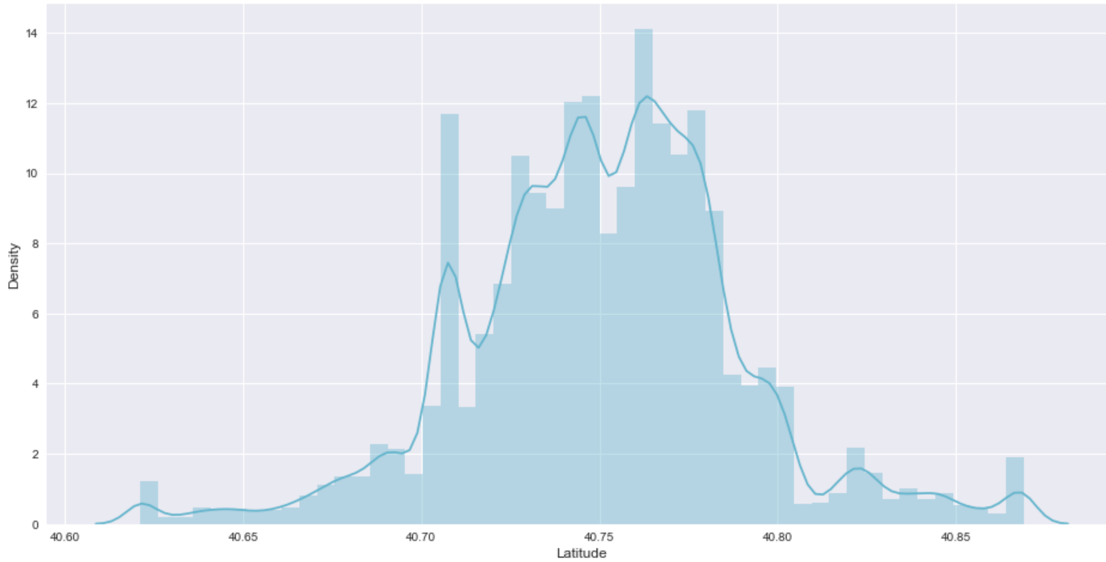


2.1 Geolocation

2.1.1 Latitude

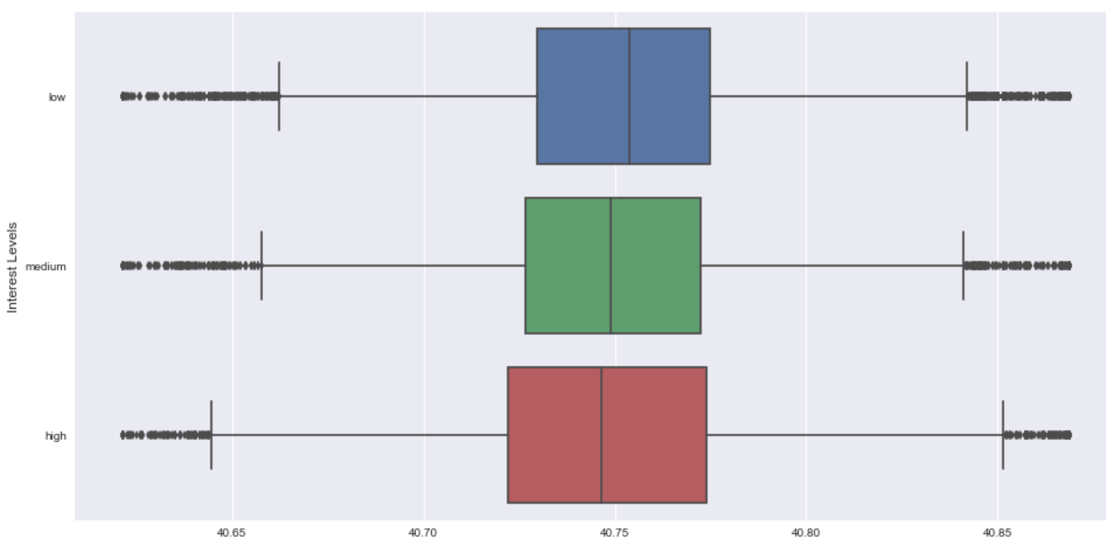
```
In [55]: # Normalizing the outliers
         percentile_995_Latitude = np.percentile(trainData['latitude'], 99.5)
         percentile_005_Latitude = np.percentile(trainData['latitude'], .5)
         outlierUpperLatRows = trainData['latitude'] > percentile_995_Latitude
         outlierLowerLatRows = trainData['latitude'] < percentile_005_Latitude
         trainData.loc[outlierUpperLatRows, 'latitude'] = percentile_995_Latitude
         trainData.loc[outlierLowerLatRows, 'latitude'] = percentile_005_Latitude

         plt.figure(figsize=(16,8))
         sns.distplot(trainData['latitude'], color=colors[5])
         plt.xlabel("Latitude", fontsize=12)
         plt.ylabel("Density", fontsize=12)
         plt.show()
```

```
/Users/JustinonTG/anaconda/lib/python3.6/site-packages/statsmodels/nonparametric/kc
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```



Interestingly, the latitude appears to follow a Gaussian distribution. Further bootstrap testing and Q-Q plotting should be conducted here to compare to the normal.

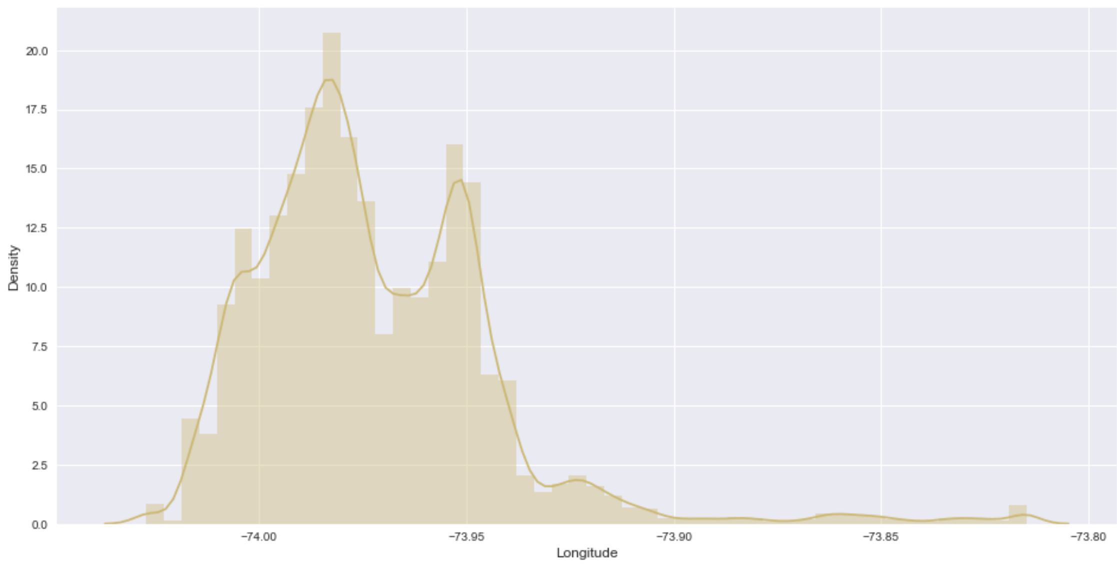
```
In [56]: plt.figure(figsize=(16,8))
         sns.boxplot(x='latitude', y='interest_level', data=trainData, order=["low'
         plt.xlabel("Latitude", fontsize=12)
         plt.ylabel("Interest Levels", fontsize=12)
         plt.show()
```



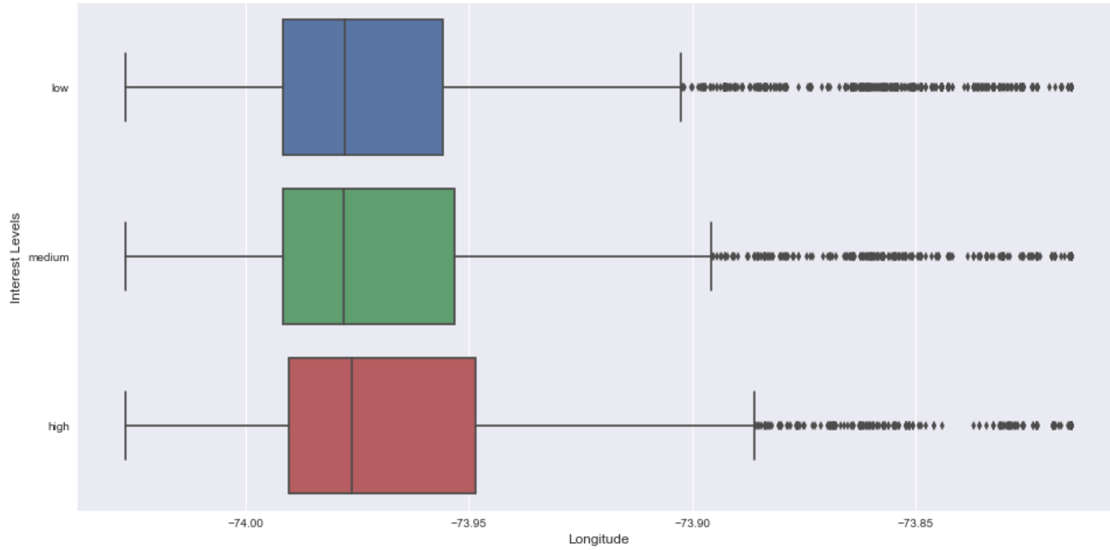
2.1.2 Longitude

```
In [57]: # Normalizing the outliers
         percentile_9975_Longitude = np.percentile(trainData['longitude'], 99.75)
         percentile_0025_Longitude = np.percentile(trainData['longitude'], .25)
         outlierUpperLongRows = trainData['longitude'] > percentile_9975_Longitude
         outlierLowerLongRows = trainData['longitude'] < percentile_0025_Longitude
         trainData.loc[outlierUpperLongRows, 'longitude'] = percentile_9975_Longitu
         trainData.loc[outlierLowerLongRows, 'longitude'] = percentile_0025_Longitu

         plt.figure(figsize=(16,8))
         sns.distplot(trainData['longitude'], color=colors[4])
         plt.xlabel("Longitude", fontsize=12)
         plt.ylabel("Density", fontsize=12)
         plt.show()
```

```
/Users/JustinonTG/anaconda/lib/python3.6/site-packages/statsmodels/nonparametric/kc
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```




```
In [58]: plt.figure(figsize=(16,8))
         sns.boxplot(x='longitude', y='interest_level', data=trainData, order=["low
         plt.xlabel("Longitude", fontsize=12)
         plt.ylabel("Interest Levels", fontsize=12)
         plt.show()
```

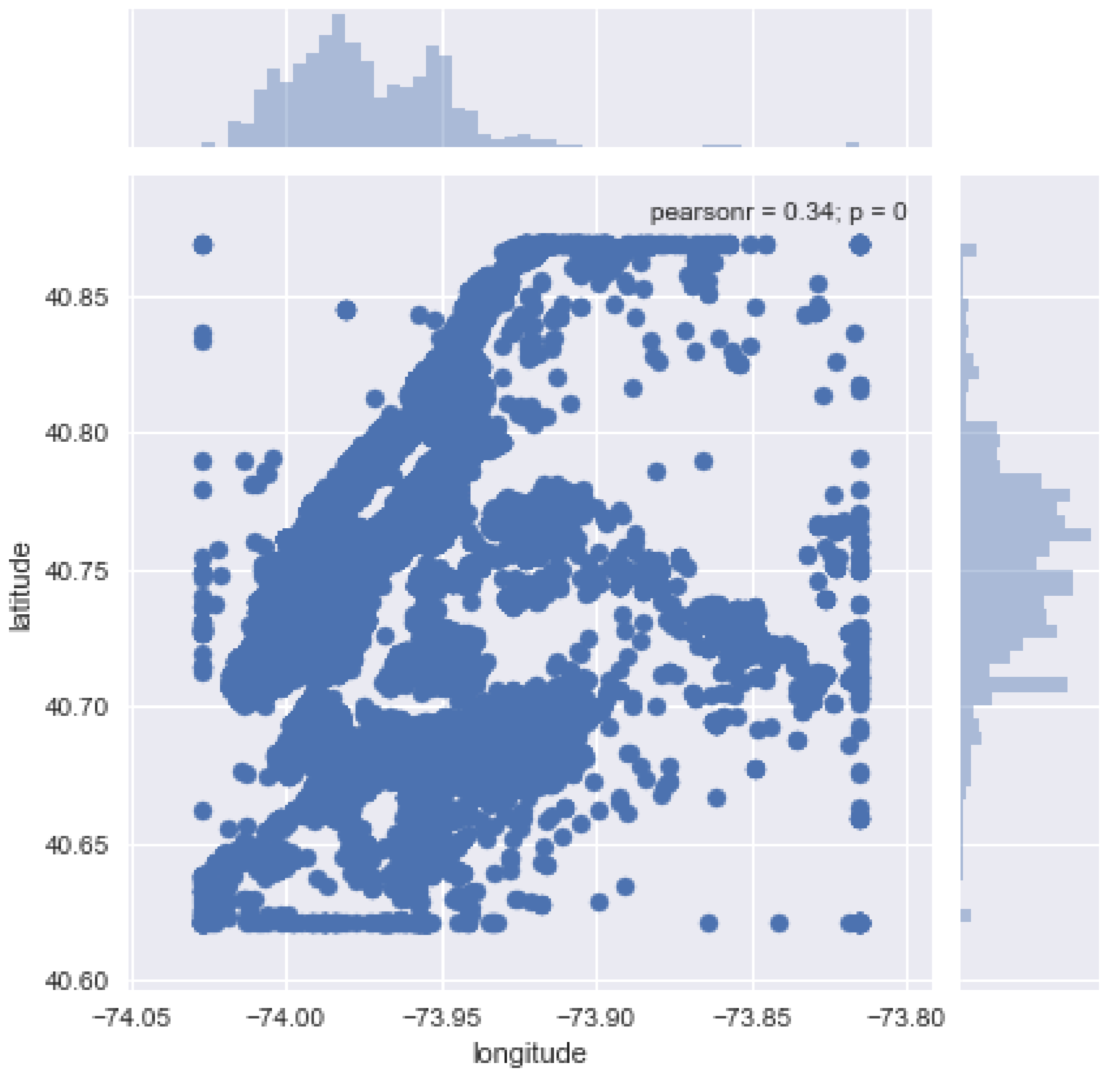


2.1.3 Mapping

First we'll plot the latitude and longitude together to graph the map of the city with which we're dealing.

```
In [59]: plt.figure(figsize=(16,16))
         sns.jointplot(x='longitude',y='latitude', data=trainData)
         plt.show()
```

```
<matplotlib.figure.Figure at 0x13d1772e8>
```

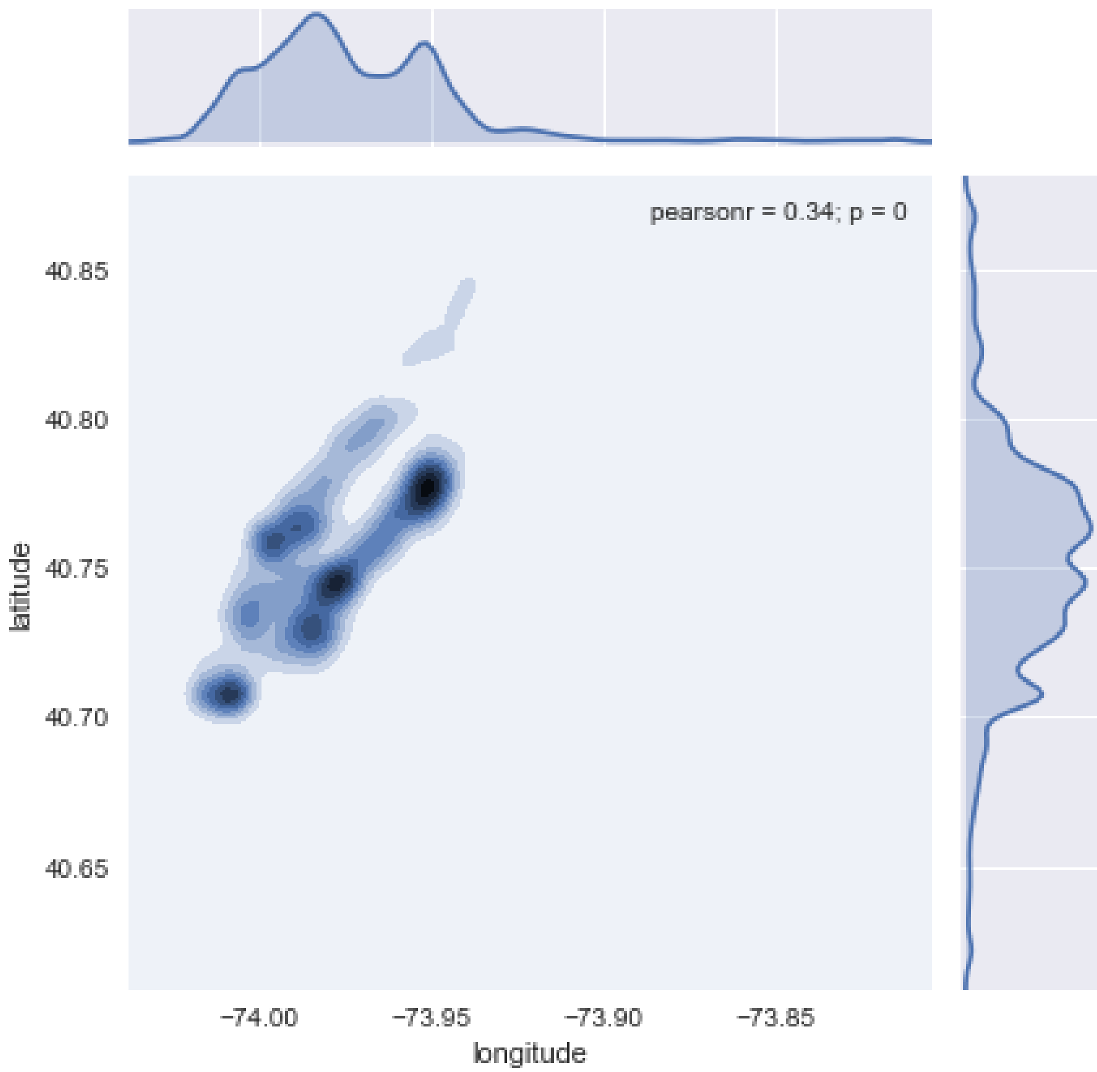


Now to check for the actual densities within the region, we'll use the kernel density estimation joint plot.

```
In [60]: plt.figure(figsize=(16,16))
         sns.jointplot(x='longitude',y='latitude', data=trainData, kind='kde')
         plt.show()
```

```
/Users/JustinonTG/anaconda/lib/python3.6/site-packages/statsmodels/nonparametric/kc
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

```
<matplotlib.figure.Figure at 0x131521630>
```


pearsonr = 0.34; p = 0
40.85
40.80
40.75
40.70
40.65
latitude
−74.00
−73.95
−73.90
−73.85
longitude

```
In [ ]:
```