exploratory-data-analysis

February 8, 2020

1 Exploratory analysis

By Sina Khalili, Logan Militzer, and Phong Le

- First some general comments about the shape of the data
- Then some charts about the various attributes
- Some outlier and missing values analysis

2 Link to code:

On github
Or use the link here: `https://github.com/SinaKhalili/kaggle-data-analysis/`

2.1 For running the notebook

- The requirements are in `requirements.txt`
- Install with `pip install -r requirements.txt`
- The dataset is the two sigman connect rental
- Install it with the kaggle api
- `kaggle competitions download -c two-sigma-connect-rental-listing-inquiries` at the root of the project
- `unzip two-sigma-connect-rental-listing-inquiries.zip`
- `unzip train.json.zip`

In [7]:
```
df.shape
```

Out[7]:
```
(49352, 15)
```

```
df.head(1) # Quite a a verbose output
```

```
   bathrooms  bedrooms                       building_id              created  \
4        1.0         1  8579a0b0d54db803821a35a4a615e97a  2016-06-16 05:55:27

                                         description      display_address  \
4  Spacious 1 Bedroom 1 Bathroom in Williamsburg!…  145 Borinquen Place

                                            features  latitude  listing_id  \
4  [Dining Room, Pre-War, Laundry in Building, Di…   40.7108     7170325
```

```
   longitude                        manager_id  \
4   -73.9539  a10db4590843d78c784171a107bdacb4

                                              photos  price  \
4  [https://photos.renthop.com/2/7170325_3bb5ac84…   2400

        street_address interest_level
4  145 Borinquen Place         medium
```

```
low       34284
medium    11229
high       3839
Name: interest_level, dtype: int64
```

```
df['price'].max(), df['price'].min() # 43 dollars!
```

```
(4490000, 43)
```

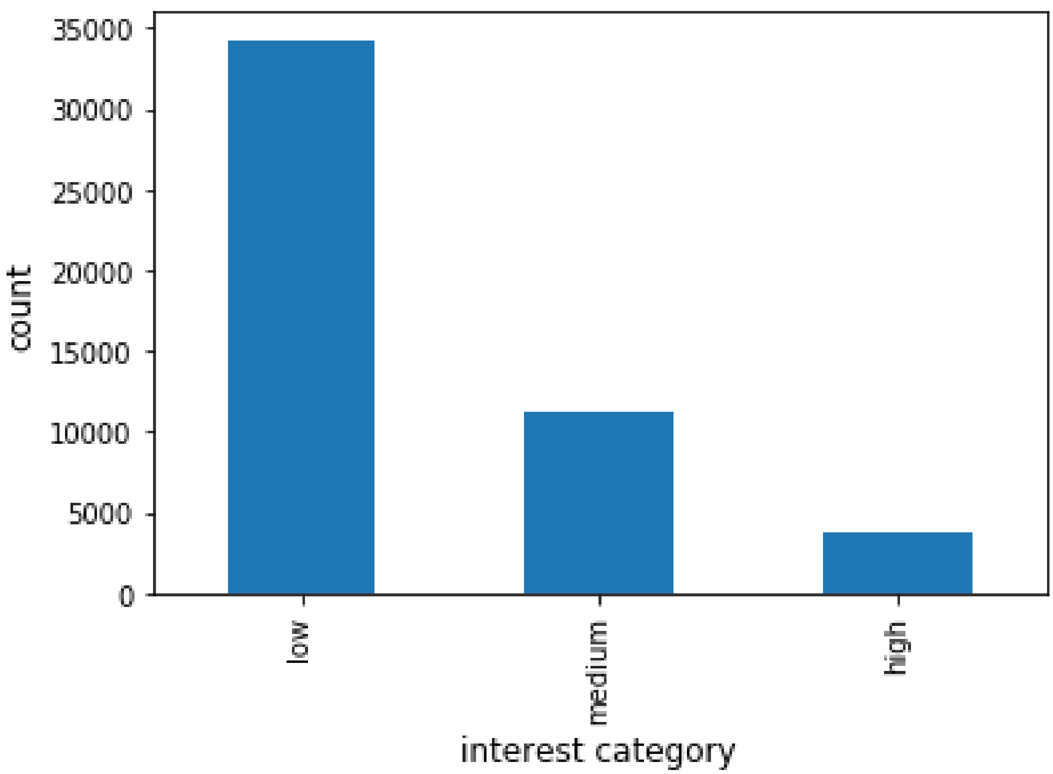
We begin by reading in the data, and reading some basic information about it.

We learn that the data: * Has ~49,000 rows with 15 columns * 14 features, and the class `interest_level` that we must predict * The `interest_level` class has three possible values: * `low`, `medium`, and `high` * Maximum and minimum prices are 4.9 million dollars and 43 dollars respectively * As much as a broke student such as myself would love to believe in the 43 dollar listing, this is probably an error

Let's visualize those target variables :

In [11]:

```
Out[11]: low       0.694683
         medium    0.227529
         high      0.077788
         Name: interest_level, dtype: float64
```



We can see clearly there are many low-interest listings, and it slowly tapers off with very few high interest listings. Proportion percentages are shown on top.

We will now convert the data types in order to parse them correctly (such as dates)

In [12]:

We can now do some analysis on the properties of this dataframe.

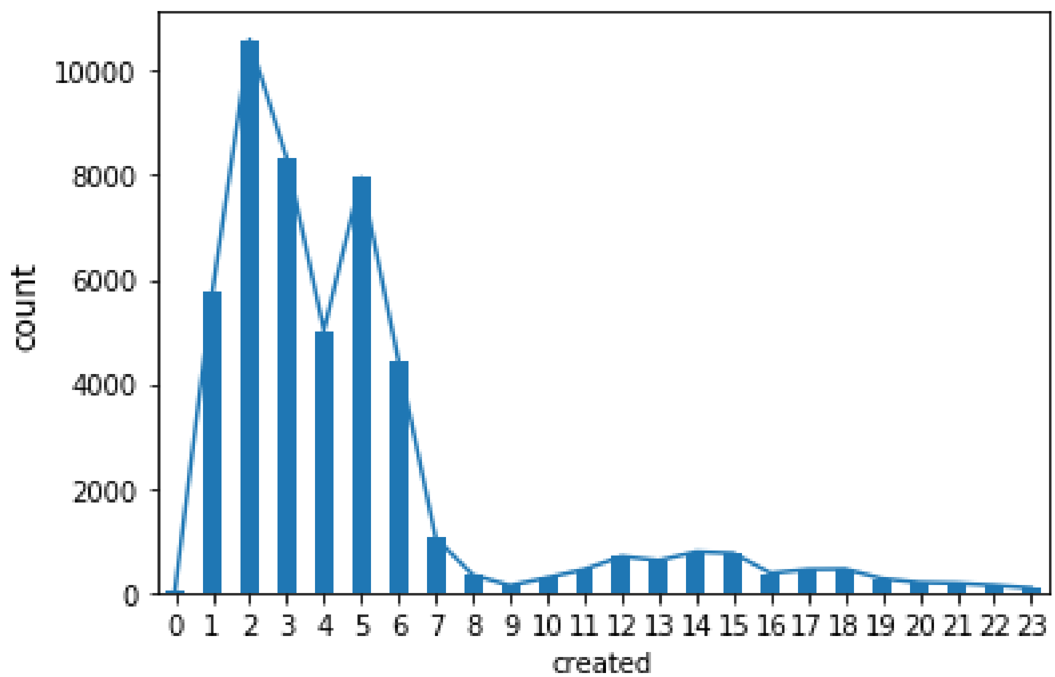
For example, we will plot the

2.1.1 Hour-wise listing counts:

Note this is an *interactive* function and will not render on github. Run the notebook locally instead. Use the dropdown to view the different types of plots.

In [15]:

```
interactive(children=(Dropdown(description='kind', options=('line', 'bar'), value='line'), Output()), _d
```

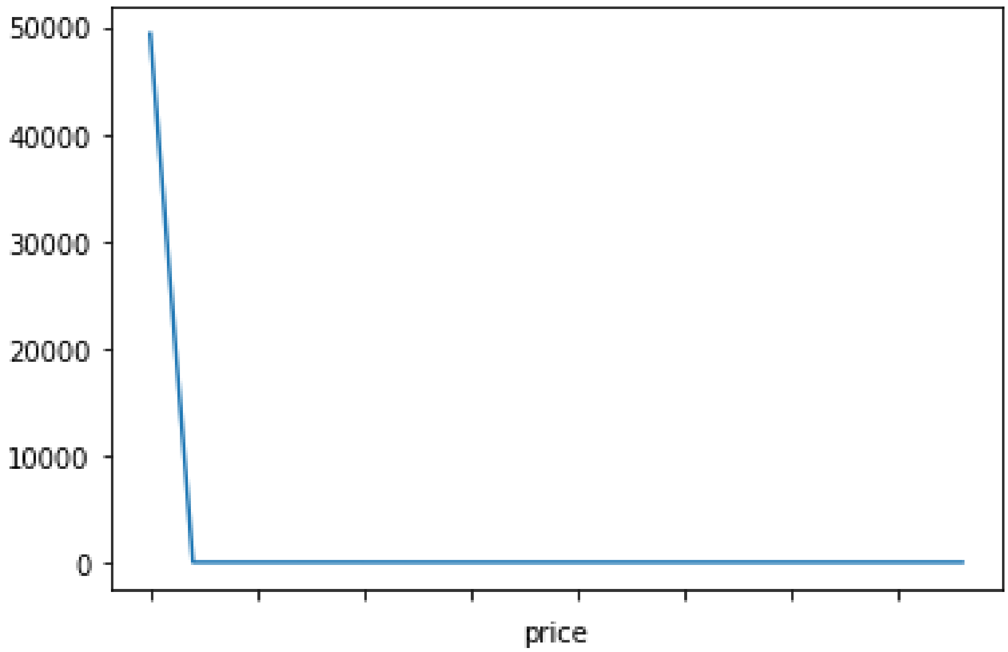



```
In [11]: hours.nlargest(5)
```

```
Out[11]: created
         2    10596
         3     8318
         5     7954
         1     5749
         4     5021
         Name: created, dtype: int64
```

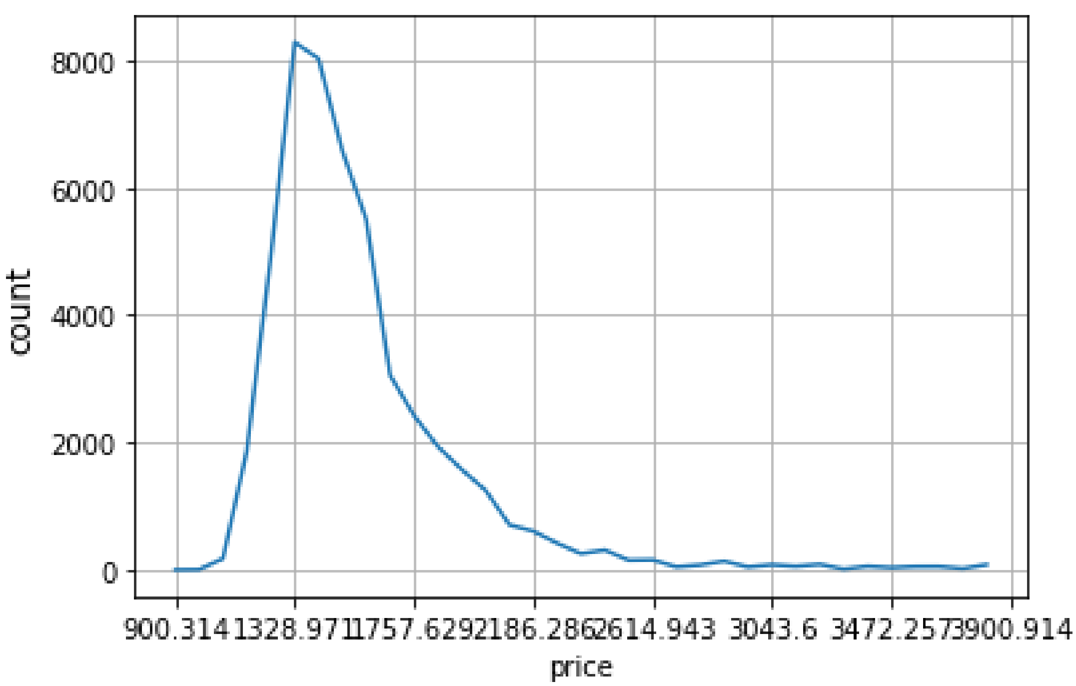
As we can see, the top times are ~1am-7am. Perhaps people are updating then because there is less traffic. Or maybe scripts are running which update at that time. Or perhaps the postings are being made by employees outside of the New York timezone.

Next, we observe the ### Price



We notice that the outliers are intrusive in this case and it would benefit us to remove them. To do this, we will remove the top percentile of some values.

```
interactive(children=(Dropdown(description='kind', options=('line', 'bar', 'barh', 'area'), value='line
```



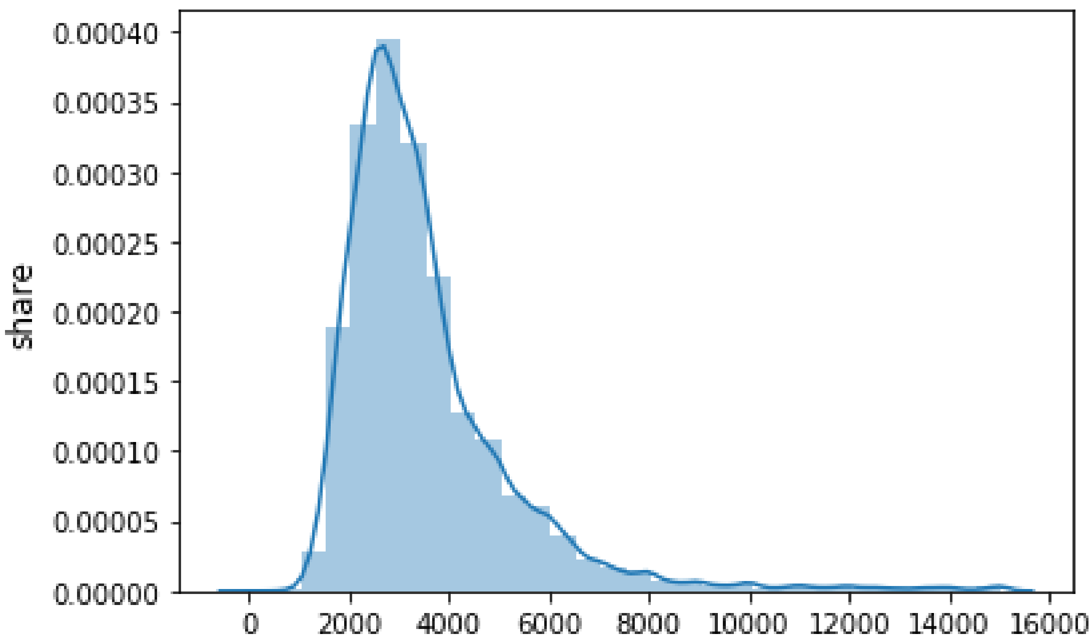
Notice as we drag the percentiles higher (without hitting 100) the graph becomes more and more left-skewed. This shows us almost all the values are in the 2000-6000 dollar range. We find that 20 bins on the bar graph will achieve a good visualization.

Some might find the distribution histogram plot more engaging:

In [15]:

```
interactive(children=(Dropdown(description='kind', options=('line', 'bar', 'barh', 'area'), value='line
```

In [16]:



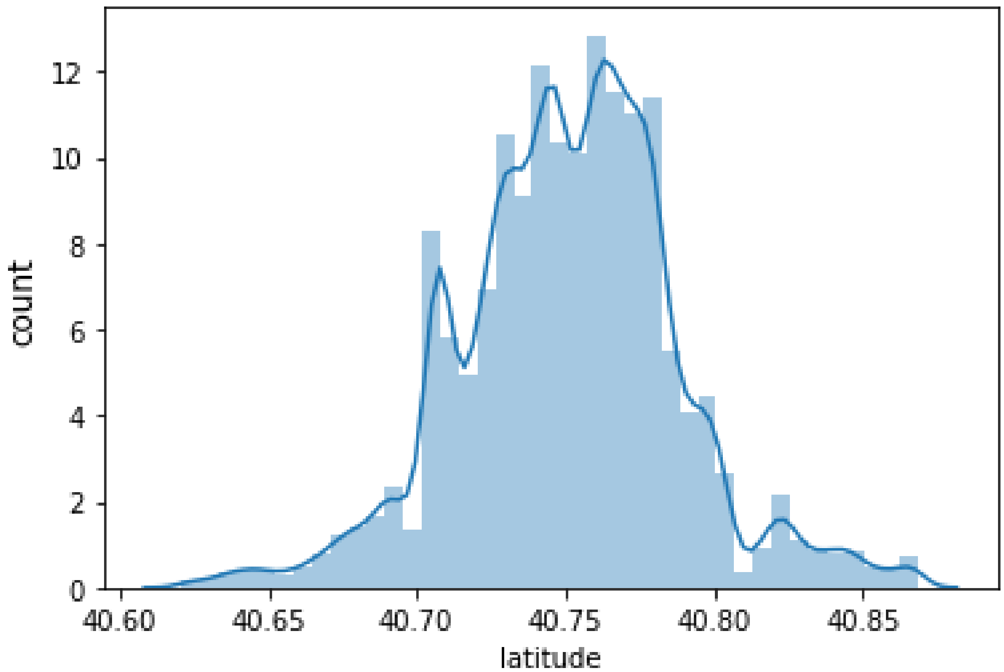
3 Longitude and Lattitude

We will give these columns a similar treatment, allowing the choice between types of graph. Use the dropdowns for your options. We have provided the graphs as well for after for the pdf version of this report.

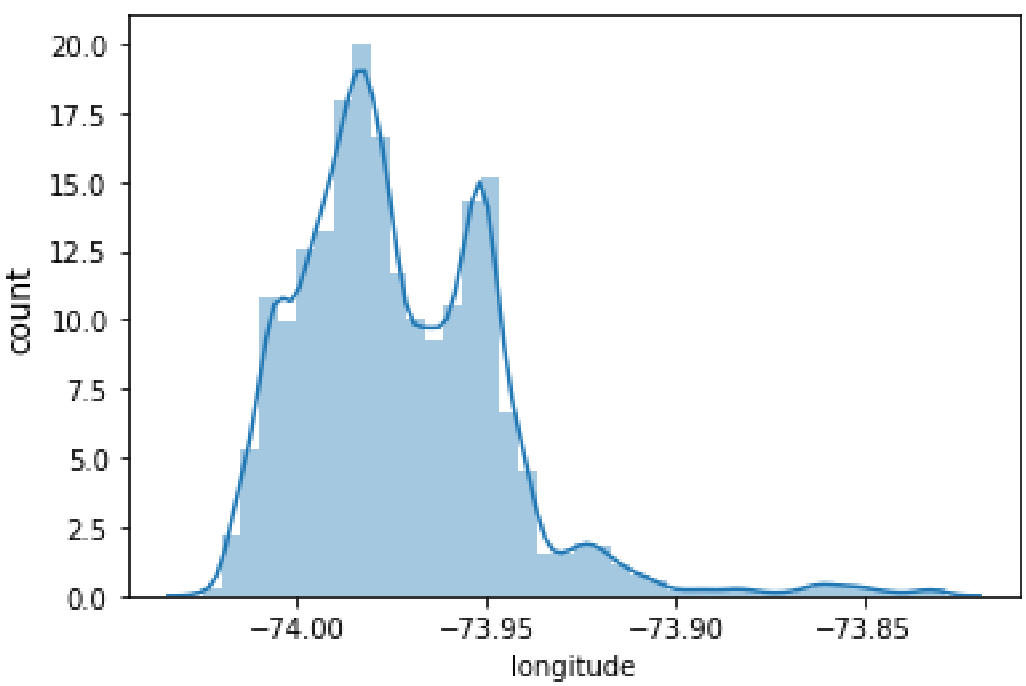
In [17]:

```
interactive(children=(Dropdown(description='axis', options=('longitude', 'latitude'), value='longitude')
```

In [18]:



Notice that the values are mostly in the 40-41 range



4 Conclusion of latitude and longitude

The values are in the -73 to -74 range, and the 40-41 range.

After some searching we realize these values belong to New York City.

Also of note when playing with the interactive plots we created, we realized that the graph varies drastically when the percentile of included data reaches 100 compared to 99.6. This gives us insight into the fact of the amount of outliers within this dataset. We analyze these outliers more later in this report but in essence after viewing the graph skew, we've decided to remove the top 0.3% off the top and 0.2% off the bottom. This loses less than 1 percent of the data but perserves the essence of the graph (i.e. the same general shape of the graph it would look like with, say 25% off the top and bottom).

We highly encourage readers of this document to use the interactive plots by cloning the repo and running the notebook locally.

This concludes our data exploration phase, though we did look at other things in the dataset such bedroom distribution and other numerical attributes.

4.1 Outliers

We now analyze the outliers more thoroughly. We provide a function to view the number of data points lying outside a certain range of percentile. Again, these numbers were decided by viewing the graphs.

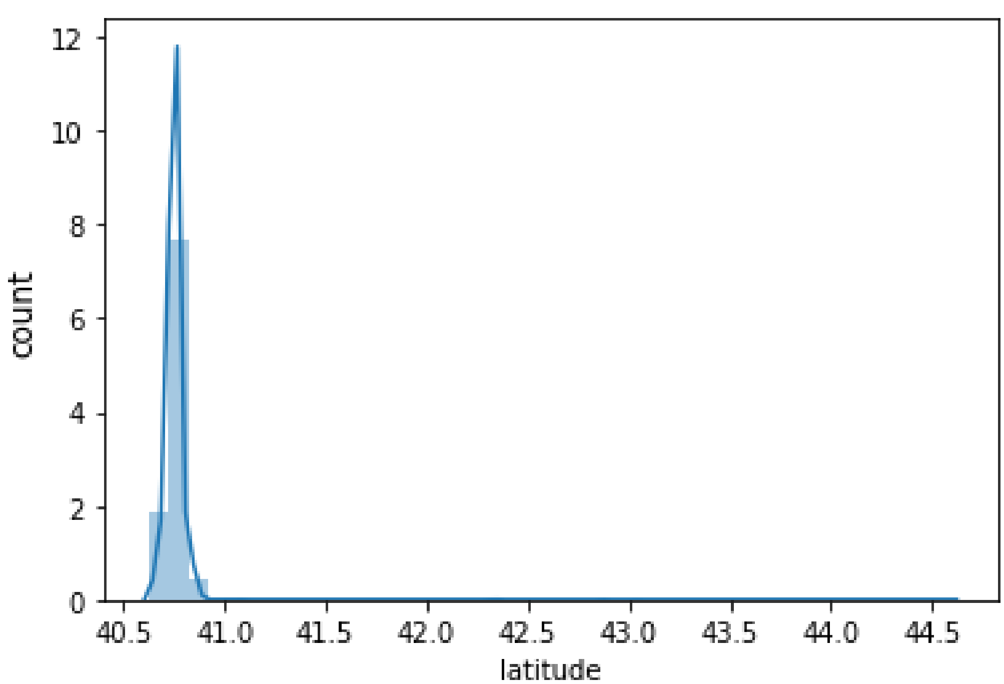
The graphs are then provided after the numbers.

In [30]:

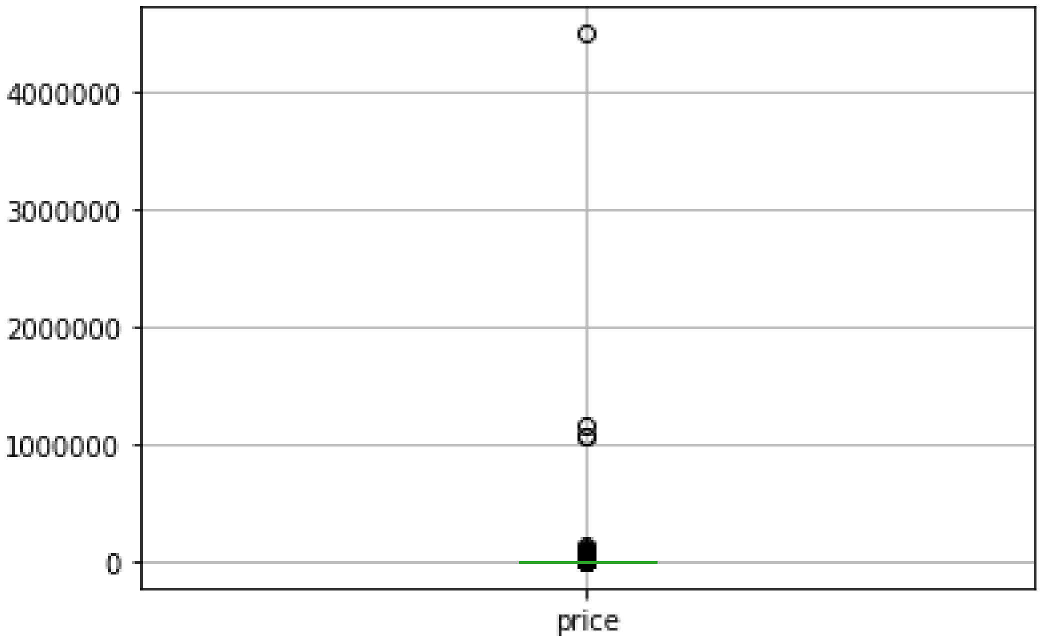
In [31]:

```
Number of items outside the percentiles is 40 in latitude
Number of items outside the percentiles is 40 in longitude
Number of items outside the percentiles is 40 in price
```

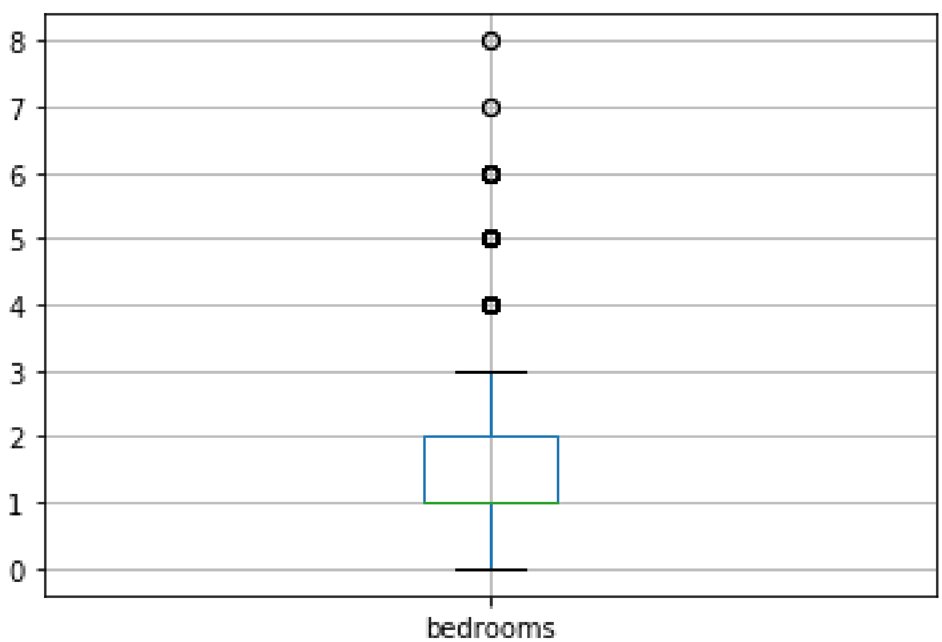
In [22]:



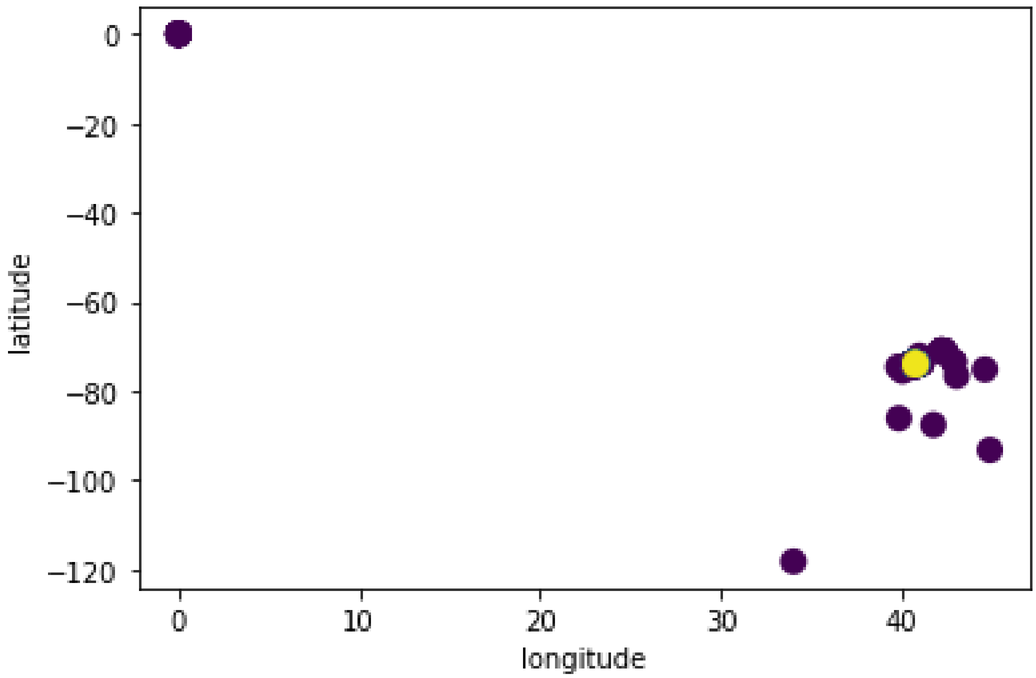
```
In [23]: PriceBoxWOOL = df.boxplot(column = ['price'], showfliers = True); PriceBoxWOOL;
```

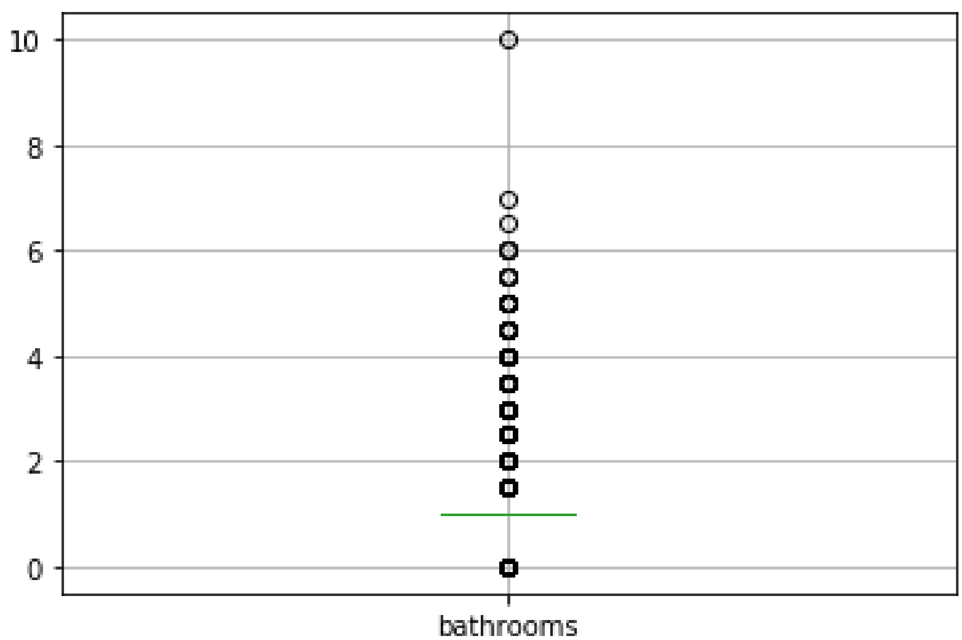



```
In [24]: BedBox = df.boxplot(column = ['bedrooms'], showfliers = True); BedBox;
```




```
In [16]:
```






```
In [30]: df['bathrooms'].max() # Relatively low diff between min and max

Out[30]: 10.0
```

```
In [31]: df['bedrooms'].max() # Relatively low diff between min and max
```

```
Out[31]: 8
```

4.2 Other features

For some of the features, doing outlier analysis is not beneficial. For example, in the strings it is not immediately obvious which metric to base it off. Also in the `id` fields there really is no such thing as an outlier as those fields are deterministically calculated by way of a hashing algorithm.

5 Final thoughts on outliers

We found that that there are relitively few differences between the min and max, we find it is not beneficial for us to remove these rows.

However, for the longitude/latitude as well as price, the difference between outliers is quite drastic and changes the shape of the graph entirely, therefore we chose to remove the top 0.5 and bottom 0.2 percent of the datasets respectively. More concretely:

5.1 Outlier Analysis

For most of the attributes, doing outlier analysis is not beneficial. Attributes that have non-numerical datatypes, such as created, description, display_address, features, photos, street_address, interest_level do not have a metric to clearly define an outlier with, therefore it does not make sense to include them for our analysis.

Attributes with id fields, building_id, listing_id, manager_id, cannot be evaluated as outliers because those fields are deterministically calculated by way of a hashing algorithm.

For bedrooms and bathrooms, we decided against counting these values as outliers. 0-7 and 0-10 for bedrooms and bathrooms respectively was such a tight range of integer values that we had to be conservative about what we considered as an outlier. We considered the rental listing that had 10 bathrooms as an outlier, but with such a small range of possible values, we decided it was not extreme enough to remove the tuple.

For latitude and longitude, we will remove the outliers. Namely, those in the bottom 0.2% and the top 0.3%, which make up for less than 1.0% of the graph. For the histogram above, removal of the outliers makes the graph more readable without compromising the bulk of the data. In addition most of the lattitude/longitute takes place in a very tight range within the New York area, so location vastly outside this range seem unlikely to be beneficial for analysis.

For price, after looking at its boxplot and histogram above, it is evident that we will remove the outliers. Its range of values is too spread out such that outliers heavily skew the graphs. We will consider all prices greater than one million to be outliers because these do not make the 99.5th percentile.

6 Missing values

```
In [27]:
```

```
Out[27]: {'bathrooms': 0,
          'bedrooms': 0,
          'building_id': 8286,
          'created': 0,
          'description': 1446,
          'display_address': 135,
          'features': 0,
          'latitude': 12,
          'listing_id': 0,
          'longitude': 12,
          'manager_id': 0,
```

```
 'photos': 0,
 'price': 0,
 'street_address': 10,
 'interest_level': 0}
```

```
In [28]: missingValues = pd.DataFrame(missing_values,index=['Number of Missing Values'])
         missingValues
```

```
Out[28]:                          bathrooms  bedrooms  building_id  created  \
         Number of Missing Values          0         0         8286        0

                                  description  display_address  features  latitude  \
         Number of Missing Values         1446              135         0        12

                                  listing_id  longitude  manager_id  photos  price  \
         Number of Missing Values           0         12           0       0      0

                                  street_address  interest_level
         Number of Missing Values              10               0
```

6.1 Discussion and plan with regards to the missing values

- We have chosen not to interpret the number `0` as a missing value for the `bedroom` and `bathroom` categories, as it is entirely possible to have a listing with `0` as a value. For instance, no bedroom, but you sleep on the couch.
- For the rest of the columns we look to see if it falls into a few cases of malformed or missing datum. We noticed there were no explicit `None` values.
- Also of note is the `building_id` is the most frequently missing value, however this is inconsequential to our analysis, as it contains no relevant information for this analysis. We can safely drop this whole column, so the missing values don't impact this decision.
- Regarding `description`, we will deal with the missing values by averaging the values from the feature extraction. Compared to the number of total entries, it is a relatively miniscule amount of entries but the text data that it provides is useful for our text data feature extraction, so we will keep them.
- The `display_address` *could also be derived from the `street_address`* (simply by removing the unit number).
- The `latitude` and `longitude` *could be derived from the address*, given that it exists. In the case that **both** the address and the longitude-latitude do not exist, we can safely drop the row, as the number of such cases make up less than 0.01 percent of the dataset, an inconsequential number.
- In the case that the `street_address` is also missing, we will eliminate the row. These cases also align with the cases described above and in the end will lose again less than 0.01 percent of the data, which we consider an inconsequential sum.

7 Feature extraction

We now look at possible features to extract from the non-numeric data such as the text and the images.

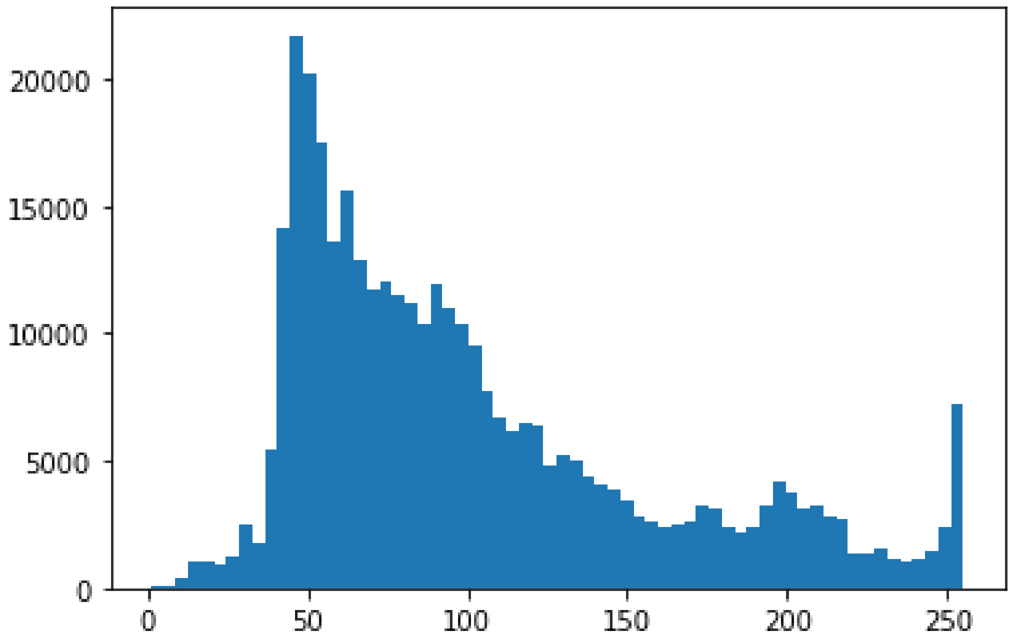
8 Image data

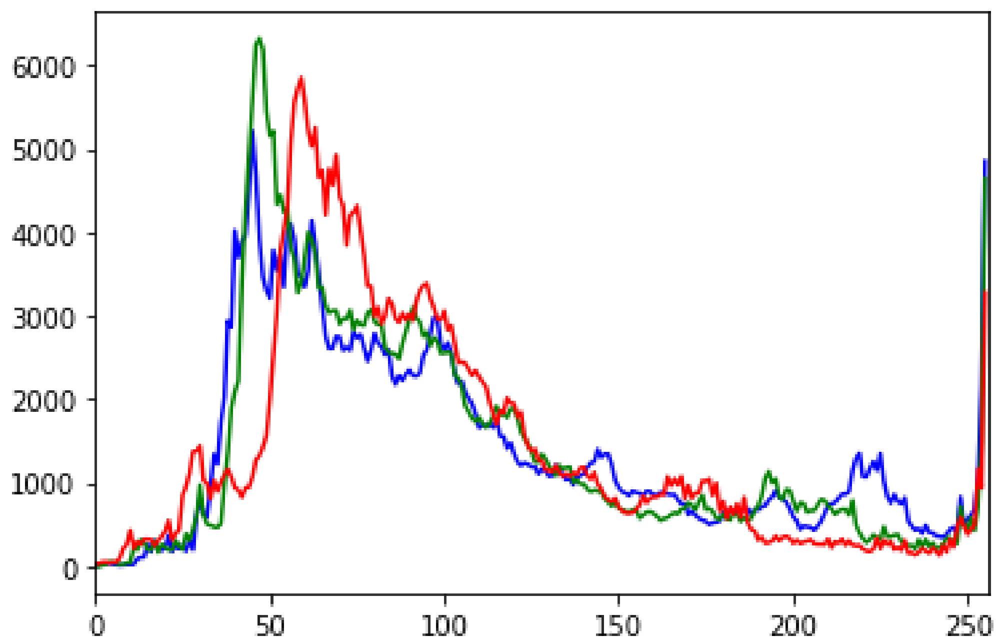
We now show how to extract the features for an image, then we expand this for all images

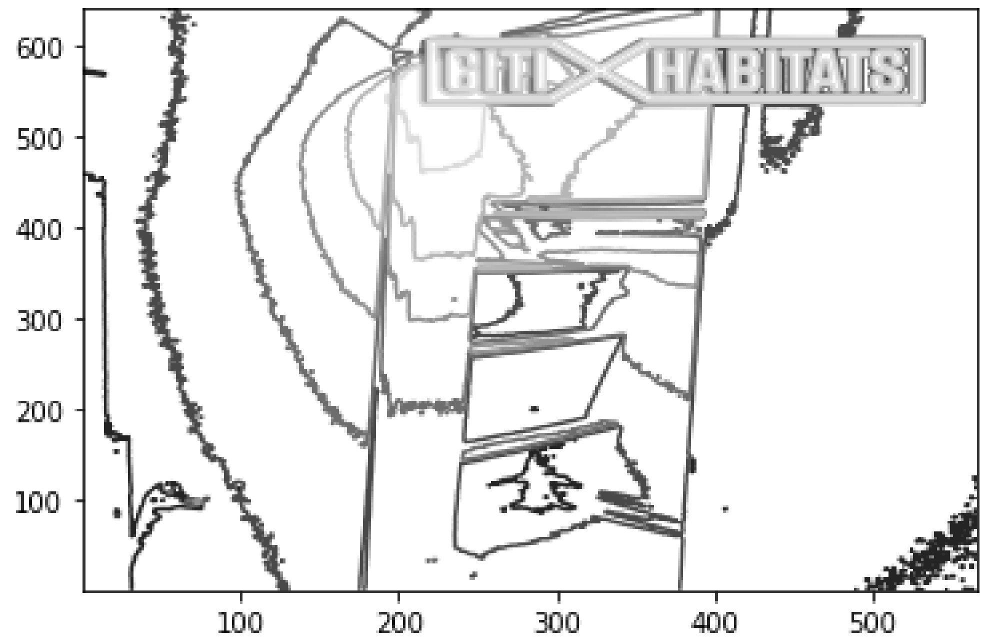
```
import glob
img_filenames = glob.glob('./images_sample/*/*.jpg')
```

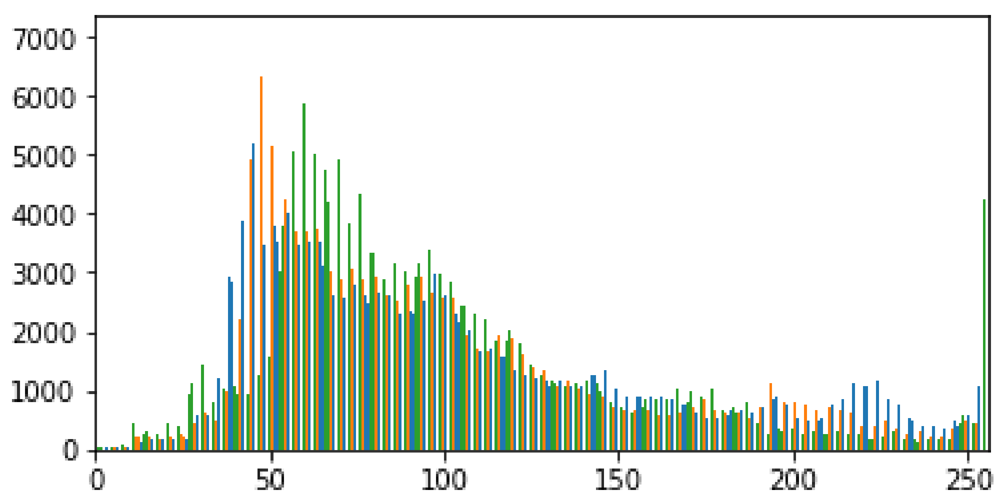
```
interactive(children=(Dropdown(description='FileName', options=('./images_sample/6812061/6812061_95f8fa
```

```
Image Width:  566  pixels
Image Height:  640  pixels
```





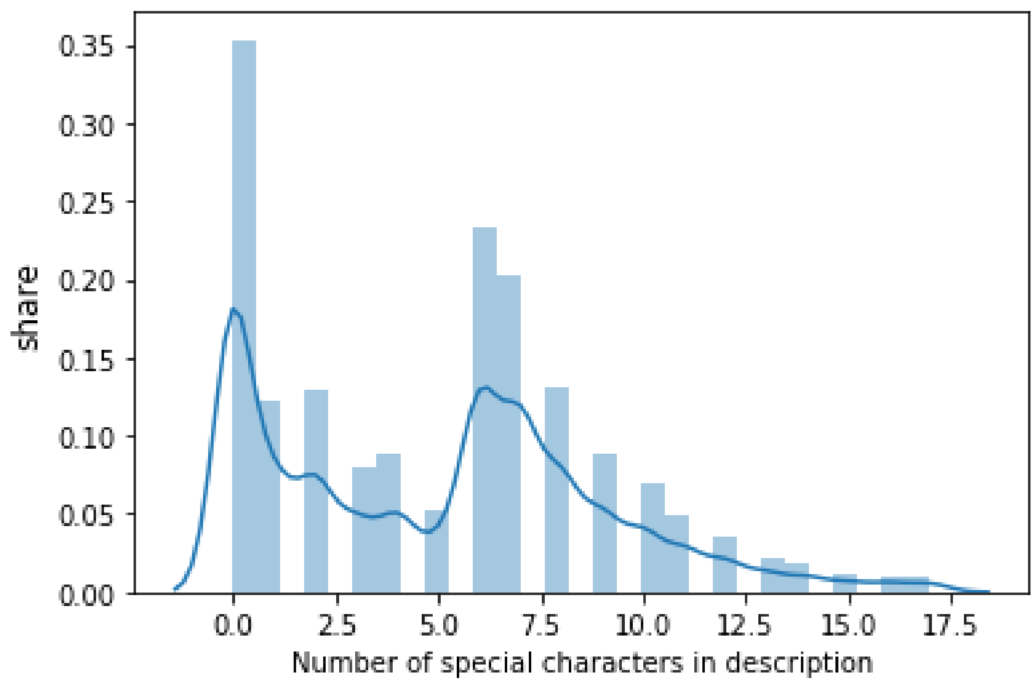




```
import os
import json
import pandas as pd

# path = os.getcwd()
# os.chdir(path)
# print(path)
```

```python
df = pd.read_json('./train.json')

Header1 = 'description'
#Header2 = 'test
'''

'''
## !"#$%@'()*+,-./:;<>=?&[]\^_`~{}|
##Special Character count
def is_special_char(x):
    xs = x.split()
    ans = []
    import re
    special_chars = re.compile("[^\w\s]")
    for xi in xs:
        if special_chars.match(xi) is not None:
            ans.append(xi)
    return len(ans)


# df['Special Characters'] = df[Header1].apply(lambda x: len([x for x in x.split() if '$' in x]))
df['Special Characters'] = df[Header1].apply(is_special_char)

df2 = df[[Header1,'Special Characters']]
chrs = df[[Header1,'Special Characters']]['Special Characters']

@interact
def show_char_plot(
    percentile_limit=(50, 100, 0.5),
    bins=(10,50,1),
):
    lim = np.percentile(chrs, percentile_limit)

    adjusted = df2[df2['Special Characters']<lim]['Special Characters']

    sns.distplot(
        adjusted.values,
        bins=bins,
        kde=True,
    )
    plt.ylabel('share', fontsize=12)
    plt.xlabel('Number of special characters in description')
    plt.show()

show_char_plot(95.5, 29)
```

```
interactive(children=(FloatSlider(value=75.0, description='percentile_limit', min=50.0, step=0.5), IntSl
```



In [177]:

Out[177]:

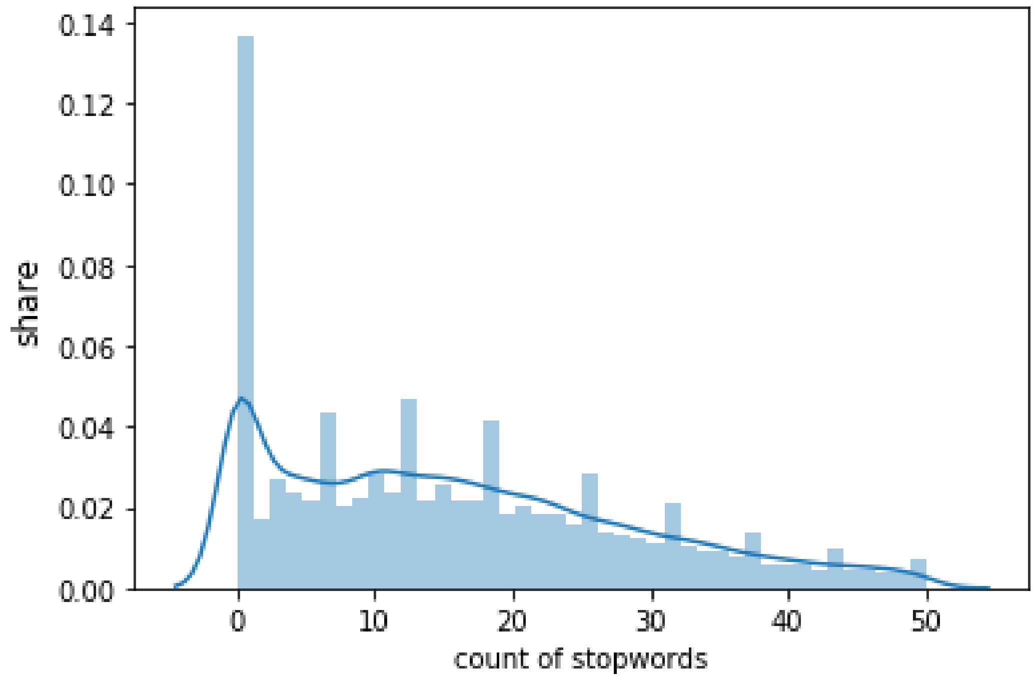
	feature	count
8	Elevator	25915
6	Cats Allowed	23540
4	Hardwood Floors	23527
5	Dogs Allowed	22035
7	Doorman	20898
3	Dishwasher	20426
9	No Fee	18062
2	Laundry in Building	16344
11	Fitness Center	13252
1	Pre-War	9148
10	Laundry in Unit	8738
14	Roof Deck	6542
37	Outdoor Space	5268
0	Dining Room	5136
15	High Speed Internet	4299
38	Balcony	2992
16	Swimming Pool	2730
21	Laundry In Building	2593
32	New Construction	2559
36	Terrace	2283
35	Exclusive	2167
12	Loft	2100
31	Garden/Patio	1943
17	Wheelchair Access	1358
19	Common Outdoor Space	1293

In [26]:

```
and                 135368
/><br               115420
the                  85583
a                    83595
to                   63061
with                 59600
in                   57049
of                   53380
is                   42832
website_redacted     35409
for                  26662
apartment            24377
or                   22454
on                   19408
this                 18843
dtype: int64
Sephora.                      1
STUDIO!<br/><br/><p><a        1
details-                      1
Countertops-great             1
Pool</li><li>Complimentary    1
TRAIN,AND                     1
Slope.<BR>This                1
area.Garage                   1
PETSMUST                      1
it.If                         1
walls.Call,                   1
more).                        1
/>Bathroom:                   1
CONCIERGE***STATE             1
market!!!Spacious             1
dtype: int64
                                          description  Numbers
4   Spacious 1 Bedroom 1 Bathroom in Williamsburg!…        2
6   BRAND NEW GUT RENOVATED TRUE 2 BEDROOMFind you…        2
9   **FLEX 2 BEDROOM WITH FULL PRESSURIZED WALL**L…        2
10  A Brand New 3 Bedroom 1.5 bath ApartmentEnjoy …        1
15  Over-sized Studio w abundant closets. Availabl…        1
```

In [27]:

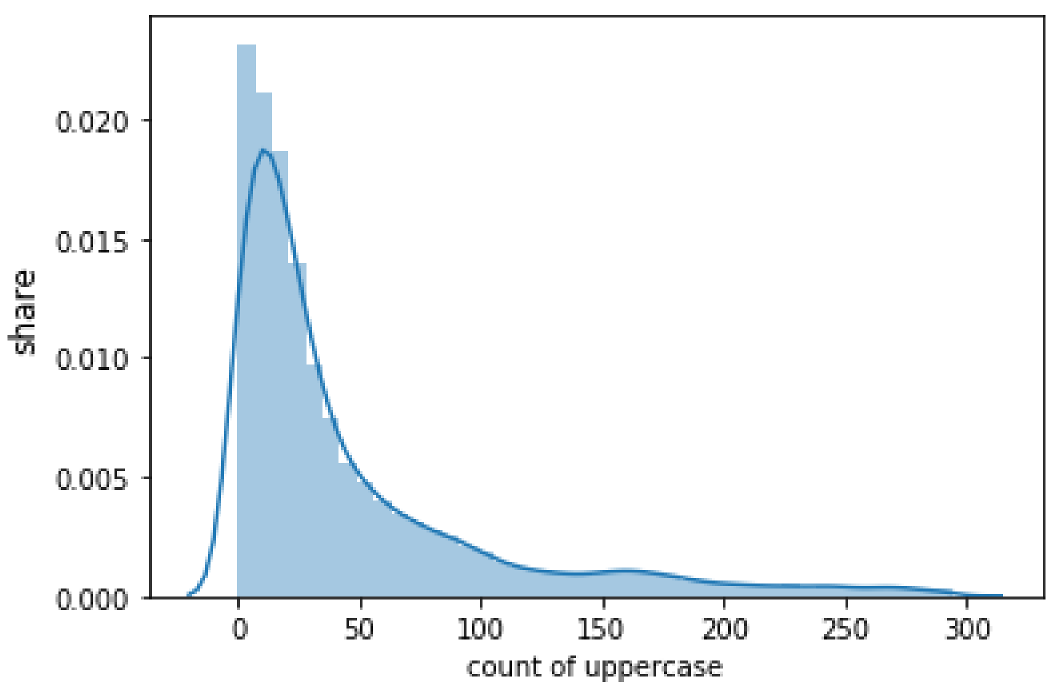
```
interactive(children=(FloatSlider(value=75.0, description='percentile_limit', min=50.0, step=0.5), IntS
```



In this graph, we can see that most rows contain 0 and 20 stopwords. Stopwords are words such as 'and', 'we', 'my', etc. This falls in line with our expectations.

In [24]:

```
interactive(children=(FloatSlider(value=75.0, description='percentile_limit', min=50.0, step=0.5), IntSl
```



This graph is showing count of the number of capitals in each row. As you can see, most rows have between 0 and 50 capitals. This falls in line with our expectations as most descriptions are short and to the point.