a
t
a
M
i
n
i
n
g

Airbnb Sentiment Analysis from Boston and Seattle

Dr. Hu

Team Members:

Amjad,

Max Bezahler,

Ben Campos,

Yiran Qiao

June 15, 2019

Abstract

This project looked at data provided by AirBnB on Kaggle for Boston and Seattle and developed models based on a derived sentiment score, based on the comments of renters, called polarity to predict review score and using the same models using the features of price, bed, gender, accommodates to predict review score. Both results were extremely accurate with the more featured model providing slightly more accuracy. We believe this likely due to similarity in the distribution of data rather than the accurate translation of polarity score into review score. The data prepartion for this project was a significiant effort to determine sentiment in the form of polarity, gender and language of the comment and is detailed below.

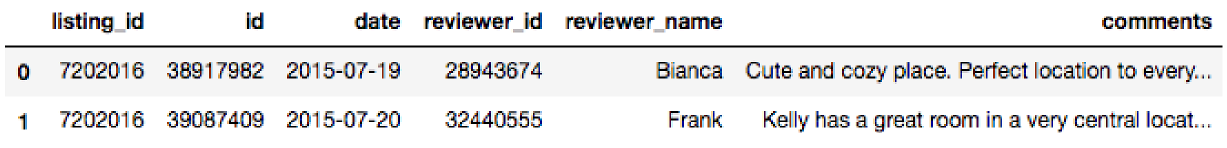
Software

Jupyter notebook 5.4.0 and Python 3.64 were used with compatible pandas and scikit learn packages cited below. Software used include textblob, Numpy, Matplotlib, Seaborn, gender guesser, and langdetect. Details below.

Introduction

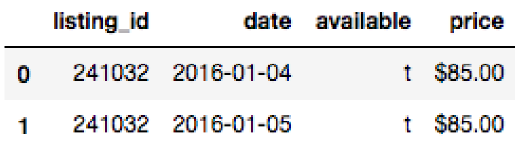
Our group arranged this project involving Airbnb data from Seattle, Washington and Boston, Massachusetts. We noticed that the review score data was overwhelming positive and decided to see if we could analyze the comments left by the reviewers with the score. We also wanted to test whether the sentiment polarity score which we derived from the comment could server as a predictor of the review score. We did this with two sets of data 50K records from Boston and 70K records from Seattle Airbnb reviews and listing from Kaggle. In addition, we tested a model for predicting review scores not based on the comments but based on features such as beds, gender, bathrooms, price, accommodates. This data contained the appropriate variables for a sentiment analysis of the reviewer’s comments. What interested us was the sheer amount of data, there are many attributes and instances that make this analysis plentiful and insightful. Once downloaded, the data contains three datasets – Reviews, Calendar, Listings. Screenshots of the data set (*Seattle, it's the same for Boston as well) are shown respectively below.

The Reviews data set contains Reviewers names, comments, dates they stayed and appropriate id's to coordinate with the other data sets.



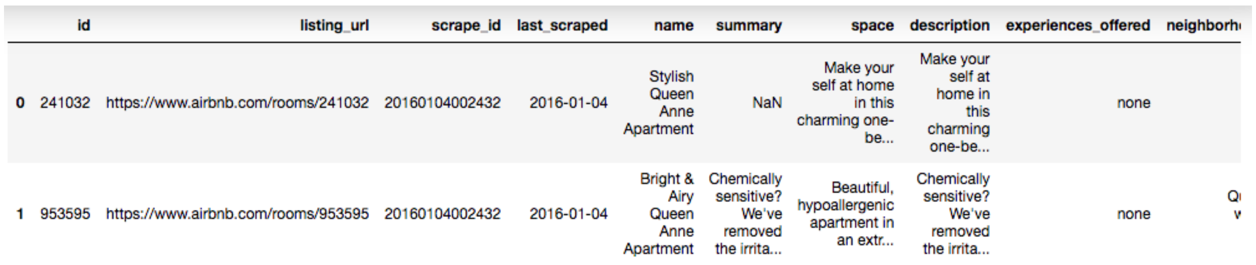
	listing_id	id	date	reviewer_id	reviewer_name	comments
0	7202016	38917982	2015-07-19	28943674	Bianca	Cute and cozy place. Perfect location to every...
1	7202016	39087409	2015-07-20	32440555	Frank	Kelly has a great room in a very central locat...

The Calendar data set contains dates, availability for Airbnb, and price along with the listing id.



	listing_id	date	available	price
0	241032	2016-01-04	t	$85.00
1	241032	2016-01-05	t	$85.00

The listings data set contains many attributes that describe specifications of the Airbnb. Some attributes were used in the analysis but not all.



	id	listing_url	scrape_id	last_scraped	name	summary	space	description	experiences_offered	neighborh
0	241032	https://www.airbnb.com/rooms/241032	20160104002432	2016-01-04	Stylish Queen Anne Apartment	NaN	Make your self at home in this charming one-be...	Make your self at home in this charming one-be...	none	
1	953595	https://www.airbnb.com/rooms/953595	20160104002432	2016-01-04	Bright & Airy Queen Anne Apartment	Chemically sensitive? We've removed the irrita...	Beautiful, hypoallergenic apartment in an extr...	Chemically sensitive? We've removed the irrita...	none	Q v

The data was acquired from Kaggle.com the links are provided.

https://www.kaggle.com/airbnb/boston/kernels https://www.kaggle.com/airbnb/boston/kernels

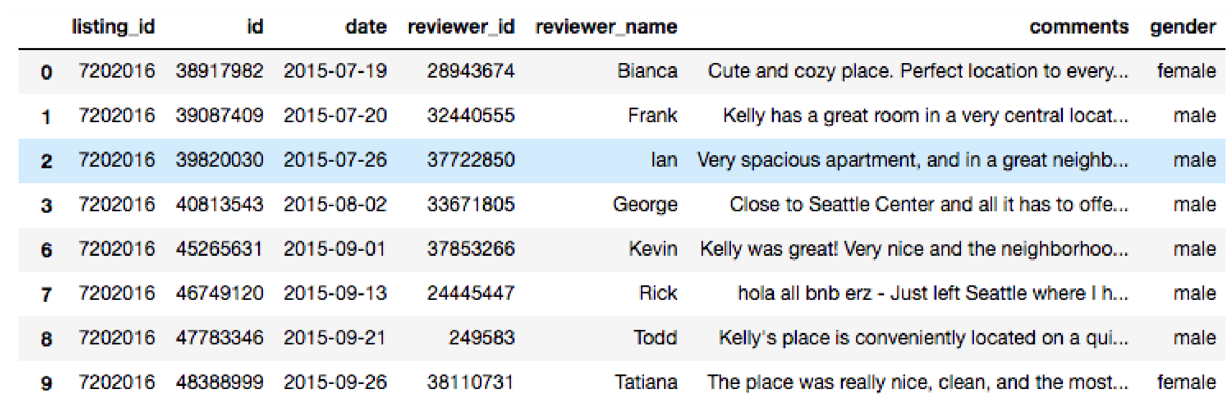
Data Description

Viewing this data at first, we saw some promising attributes within the (Reviews) dataset such as Comments, Reviewer_name, and Date, to name a few. These attributes stood out to us because comments are provided by the reviewer that stayed at a specific location that can be analyzed for sentiment value. With these initial attributes we decided to find out more about who exactly is our audience in regard to gender.

Gender Data Manipulation

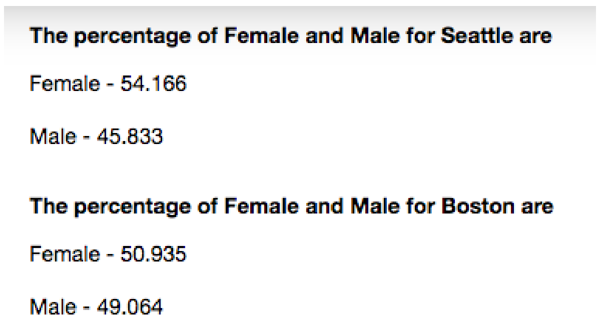
Using the library gender_guesser.detector by Jorg Michael. The program “gender.c” uses the dictionary file “name_dict.txt” as a data source. This file contains a list of more than 40,000 first names and gender, plus some 600 pairs of “equivalent” names. This list covers the majority of first names in all European countries and in some overseas countries (e.g. China, India, Japan, U.S.A.) as well. More information about this module is found in this website https://autohotkey.com/board/topic/20260-gender-verification-by-forename-cmd-line-tool-db/

Once the module was applied to the comments column, a new column was created that showcased the appropriate gender.



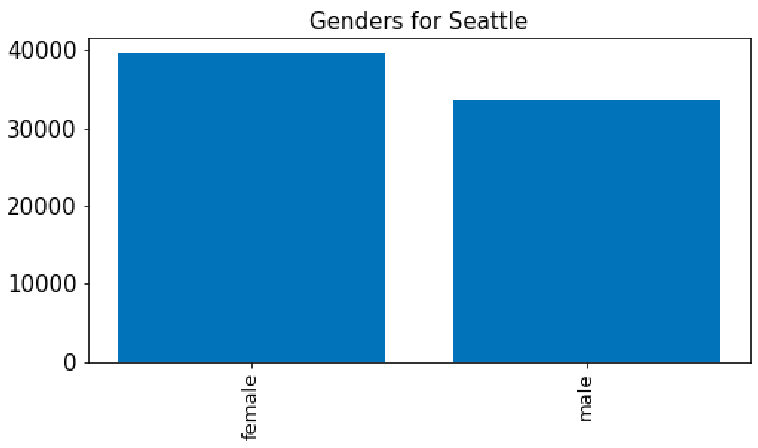
	listing_id	id	date	reviewer_id	reviewer_name	comments	gender
0	7202016	38917982	2015-07-19	28943674	Bianca	Cute and cozy place. Perfect location to every...	female
1	7202016	39087409	2015-07-20	32440555	Frank	Kelly has a great room in a very central locat...	male
2	7202016	39820030	2015-07-26	37722850	Ian	Very spacious apartment, and in a great neighb...	male
3	7202016	40813543	2015-08-02	33671805	George	Close to Seattle Center and all it has to offe...	male
6	7202016	45265631	2015-09-01	37853266	Kevin	Kelly was great! Very nice and the neighborhoo...	male
7	7202016	46749120	2015-09-13	24445447	Rick	hola all bnb erz - Just left Seattle where I h...	male
8	7202016	47783346	2015-09-21	249583	Todd	Kelly's place is conveniently located on a qui...	male
9	7202016	48388999	2015-09-26	38110731	Tatiana	The place was really nice, clean, and the most...	female

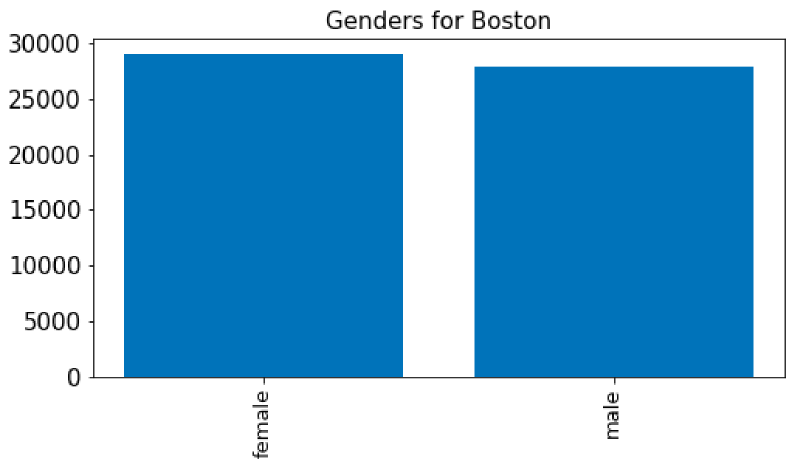
After the module was applied to both data sets, we used a simple value counts method and then divided by the total to find percentages of genders for both Seattle and Washington.



```
The percentage of Female and Male for Seattle are

Female - 54.166

Male - 45.833

The percentage of Female and Male for Boston are

Female - 50.935

Male - 49.064
```

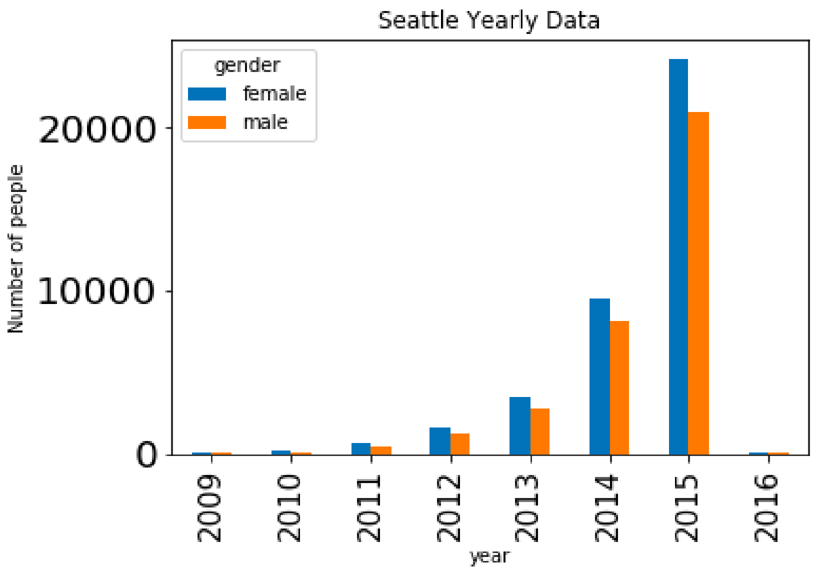
This finding told us something about the data, and that is the majority of genders that stay at these locations are mostly Females. Here is a bar plot showing the difference in genders for both locations

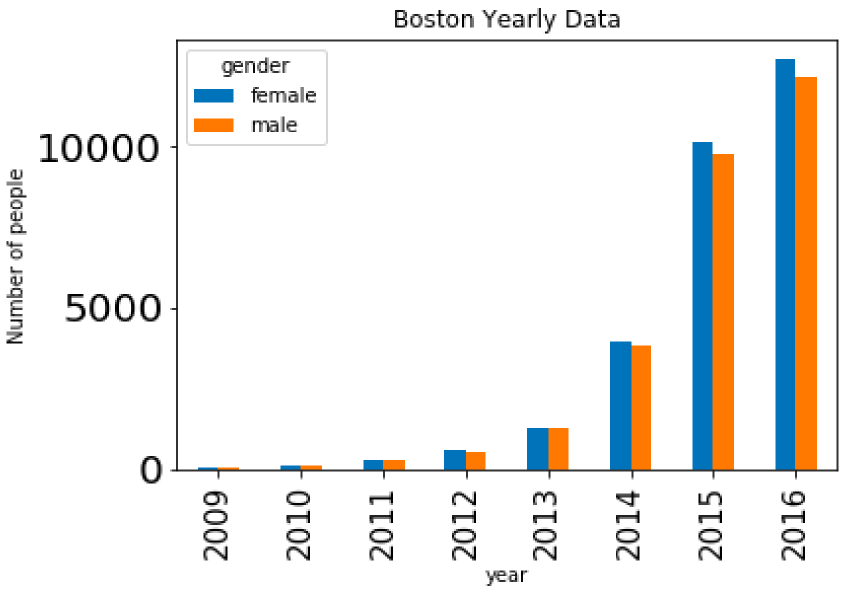




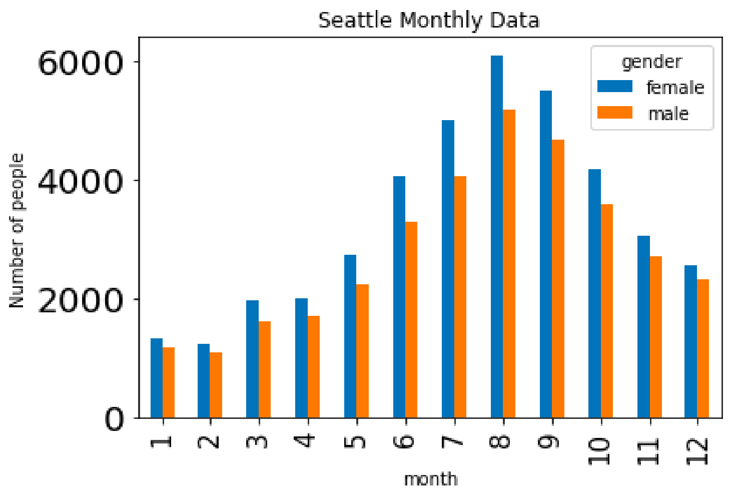
Dates

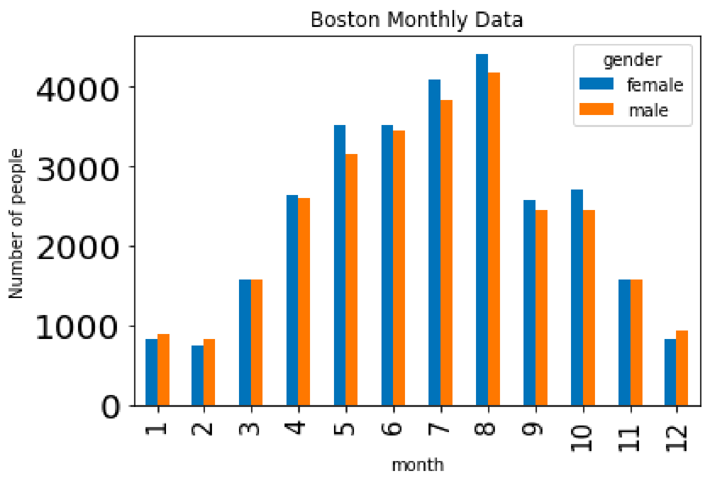
In the dates column we noticed a slight issue in how the format is. For every instance there is a specific day, month, and year. This posed an issue to our analysis because when we would plot the data it would show a noisy and gigantic plot showing many dates. We decided to split the year and month into two separate columns for ease of access. Provided below are the years of data which are from 2009-2016 for both locations. It showcases the genders frequency.





We can see the majority of data was primary 2014 – 2016. We believe the reason for this increase of stays is because Airbnb started back in August 2008. Maybe at the beginning not too many people were familiar with this service. Now what we wanted to find out was which months are the most popular? This is answered by the graphs below



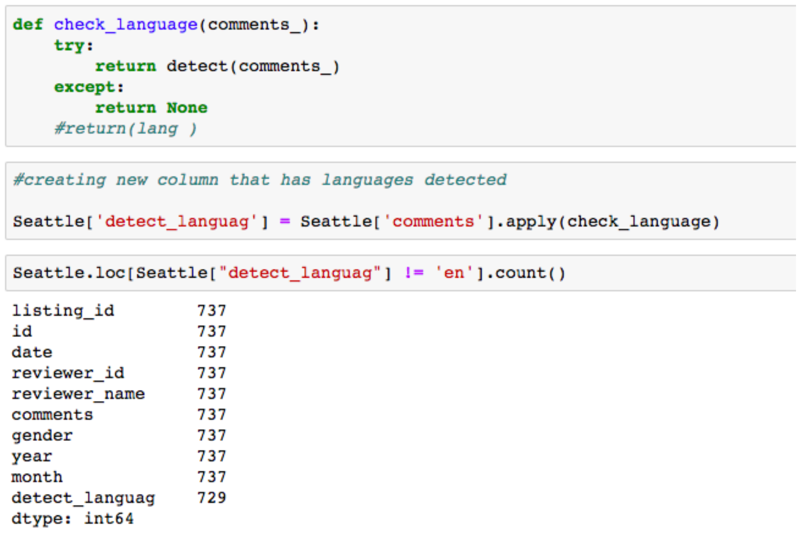


Not surprisingly, the month with the highest peak of Airbnb stays was August. Summer time is always a great time for traveling and staying at Airbnb's.

Pre-Processing Data

The most important and rather tedious part of a proper analysis is cleaning the data. Making sure there are no null values or incorrect/weird looking values. We glanced at the data quickly and noticed some weird things that were going on with the sentiment analysis that we did not expect.

The first issue we encountered was foreign languages, languages that were not in English could not be processed and valued properly by textblob, so we removed these rows that contained foreign languages. The way we checked to see if they were foreign or not was by using the module *langdetect*.



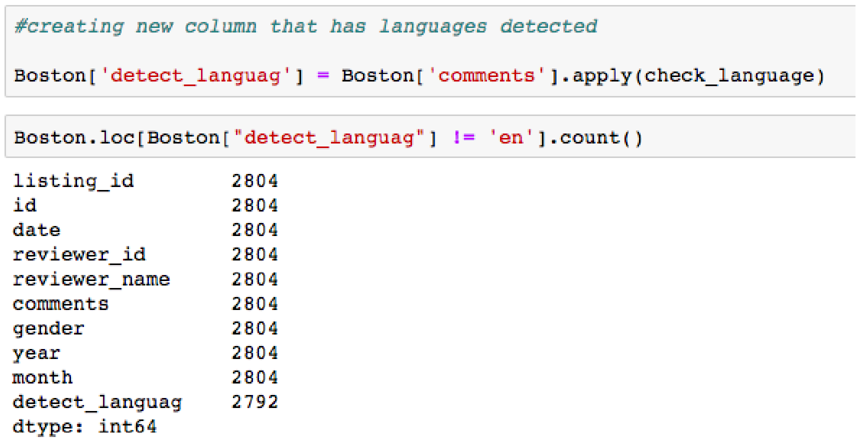
```
def check_language(comments_):
    try:
        return detect(comments_)
    except:
        return None
    #return(lang )
```

```
#creating new column that has languages detected

Seattle['detect_languag'] = Seattle['comments'].apply(check_language)
```

```
Seattle.loc[Seattle["detect_languag"] != 'en'].count()
```

```
listing_id        737
id                737
date              737
reviewer_id       737
reviewer_name     737
comments          737
gender            737
year              737
month             737
detect_languag    729
dtype: int64
```

There were exactly 737 instances of a comment that was not in English within Seattle.



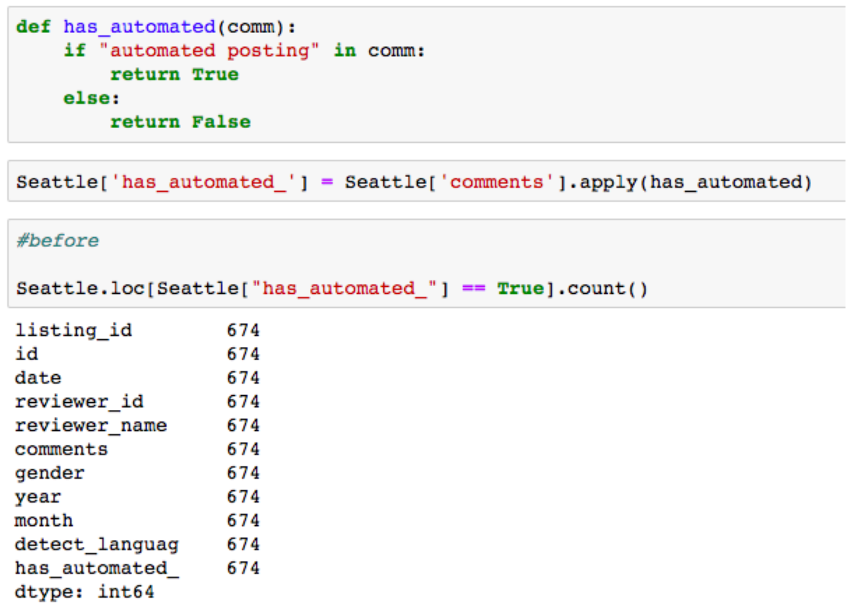
```
#creating new column that has languages detected

Boston['detect_languag'] = Boston['comments'].apply(check_language)
```

```
Boston.loc[Boston["detect_languag"] != 'en'].count()
```

```
listing_id        2804
id                2804
date              2804
reviewer_id       2804
reviewer_name     2804
comments          2804
gender            2804
year              2804
month             2804
detect_languag    2792
dtype: int64
```

For Boston there was 2804 instances of a foreign language detected in the comments.

The second issue that we noticed once we viewed our initial sentimental analysis was that people that cancelled before their reservation day somehow automatically posted a comment that said, “Reservation was cancelled X days before the stay. This is an automated posting.” This posting caused some issue with our textblob function in that it labeled these comments as neutral. We decided to remove these comments since they were not from the reviewer at all.



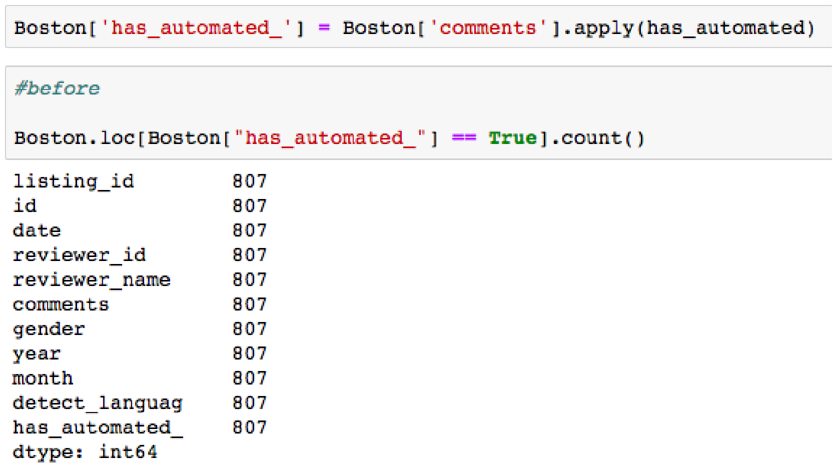
```
def has_automated(comm):
    if "automated posting" in comm:
        return True
    else:
        return False
```

```
Seattle['has_automated_'] = Seattle['comments'].apply(has_automated)
```

```
#before

Seattle.loc[Seattle["has_automated_"] == True].count()
```

```
listing_id         674
id                 674
date               674
reviewer_id        674
reviewer_name      674
comments           674
gender             674
year               674
month              674
detect_languag     674
has_automated_     674
dtype: int64
```

Seattle had 674 automated postings, they were dropped.



```
Boston['has_automated_'] = Boston['comments'].apply(has_automated)
```

```
#before

Boston.loc[Boston["has_automated_"] == True].count()
```

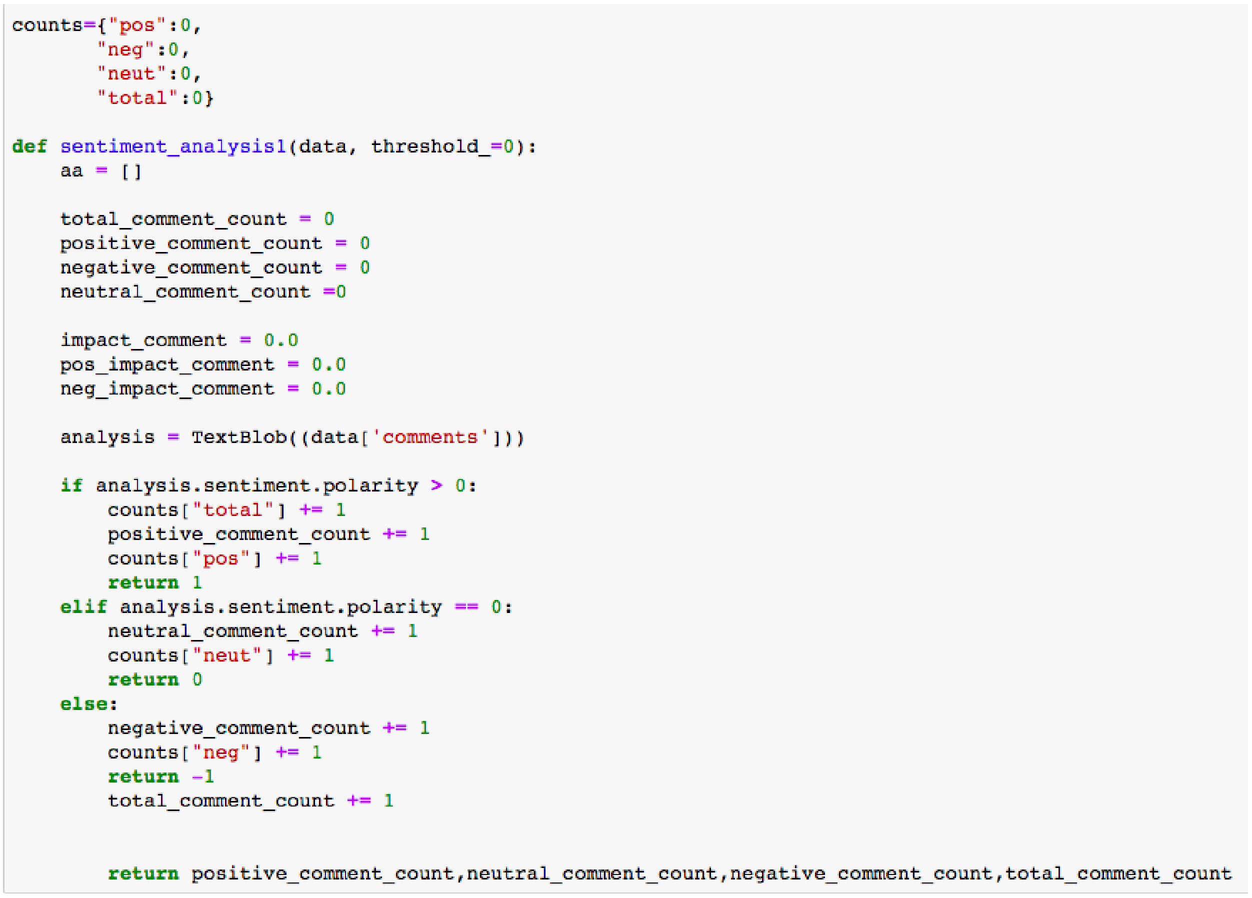
```
listing_id         807
id                 807
date               807
reviewer_id        807
reviewer_name      807
comments           807
gender             807
year               807
month              807
detect_languag     807
has_automated_     807
dtype: int64
```

Boston contained 807 instances of automated postings, they were also dropped.

The third issue we encountered was comments that were too short in length for an appropriate value of sentiment. Comments with 3 characters or less were dropped for both data sets.

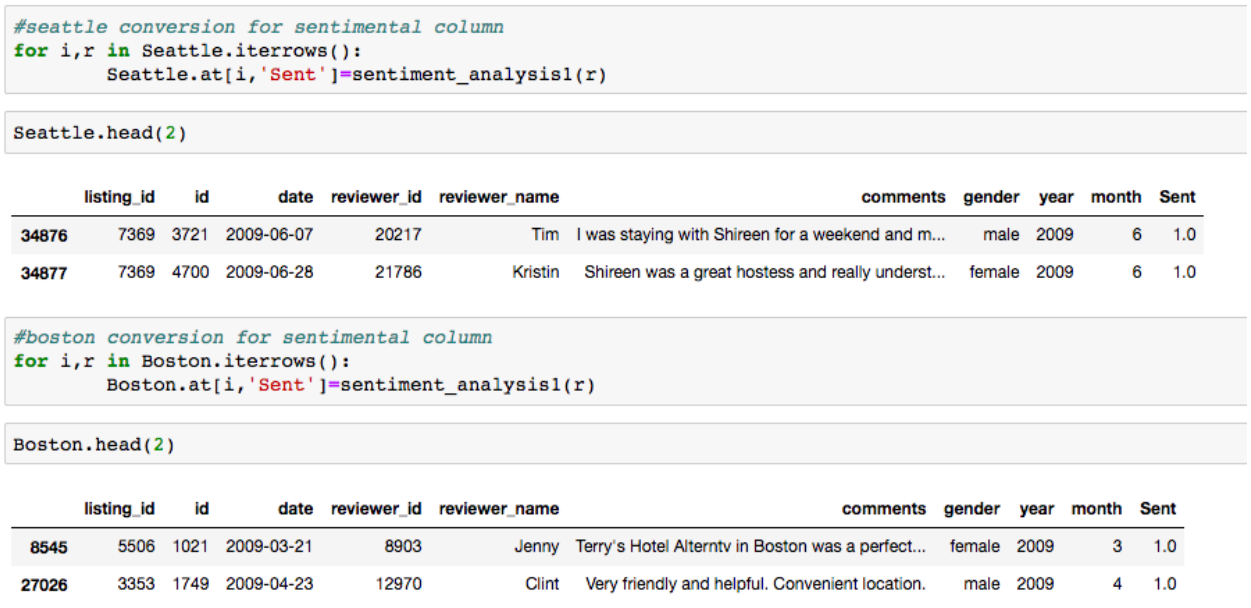
Sentiment Analysis

Now that our data was clean and ready to go, we performed the sentiment analysis for both Boston and Seattle. Below is the code that rates the comments as positive (1), negative (-1), or neutral (0).



```
counts={"pos":0,
       "neg":0,
       "neut":0,
       "total":0}

def sentiment_analysis1(data, threshold_=0):
    aa = []

    total_comment_count = 0
    positive_comment_count = 0
    negative_comment_count = 0
    neutral_comment_count =0

    impact_comment = 0.0
    pos_impact_comment = 0.0
    neg_impact_comment = 0.0

    analysis = TextBlob((data['comments']))

    if analysis.sentiment.polarity > 0:
        counts["total"] += 1
        positive_comment_count += 1
        counts["pos"] += 1
        return 1
    elif analysis.sentiment.polarity == 0:
        neutral_comment_count += 1
        counts["neut"] += 1
        return 0
    else:
        negative_comment_count += 1
        counts["neg"] += 1
        return -1
        total_comment_count += 1


        return positive_comment_count,neutral_comment_count,negative_comment_count,total_comment_count
```

With this function in place, we still had to tell the code to run through each and every row and rate the comment. The code is displayed below along with a new column “Sent” Sentimental Analysis for short.



```python
#seattle conversion for sentimental column
for i,r in Seattle.iterrows():
        Seattle.at[i,'Sent']=sentiment_analysis1(r)
```

```python
Seattle.head(2)
```

	listing_id	id	date	reviewer_id	reviewer_name	comments	gender	year	month	Sent
34876	7369	3721	2009-06-07	20217	Tim	I was staying with Shireen for a weekend and m...	male	2009	6	1.0
34877	7369	4700	2009-06-28	21786	Kristin	Shireen was a great hostess and really underst...	female	2009	6	1.0

```python
#boston conversion for sentimental column
for i,r in Boston.iterrows():
        Boston.at[i,'Sent']=sentiment_analysis1(r)
```

```python
Boston.head(2)
```

	listing_id	id	date	reviewer_id	reviewer_name	comments	gender	year	month	Sent
8545	5506	1021	2009-03-21	8903	Jenny	Terry's Hotel Alterntv in Boston was a perfect...	female	2009	3	1.0
27026	3353	1749	2009-04-23	12970	Clint	Very friendly and helpful. Convenient location.	male	2009	4	1.0

All comments were given a score. But how accurate is that score? We read some of the comments and realized some scores are not too accurate. This is when we decided to add polarity to our comments. Polarity shows just how positive, and negative a comment really is. The code below is shown for polarity.

```
Seattle['pol_sub'] = Seattle['comments'].apply(sentiment_func)

Seattle['pol_sub'][0][0]

Seattle['Polarity'] = Seattle['pol_sub'].apply(lambda x: x[0])
Seattle['Subjecetivity'] = Seattle['pol_sub'].apply(lambda x: x[1])
```

```
Seattle.head(2)
```

	listing_id	id	date	reviewer_id	reviewer_name	comments	gender	year	month	Sent	pol_sub	Polarity	Subjecetivity
34876	7369	3721	2009-06-07	20217	Tim	I was staying with Shireen for a weekend and m...	male	2009	6	1.0	(0.27187500000000003, 0.725)	0.271875	0.725
34877	7369	4700	2009-06-28	21786	Kristin	Shireen was a great hostess and really underst...	female	2009	6	1.0	(0.6222222222222223, 0.7000000000000001)	0.622222	0.700

```
Boston['pol_sub'] = Boston['comments'].apply(sentiment_func)

Boston['pol_sub'][0][0]

Boston['Polarity'] = Boston['pol_sub'].apply(lambda x: x[0])
Boston['Subjecetivity'] = Boston['pol_sub'].apply(lambda x: x[1])
```

```
Boston.head(2)
```

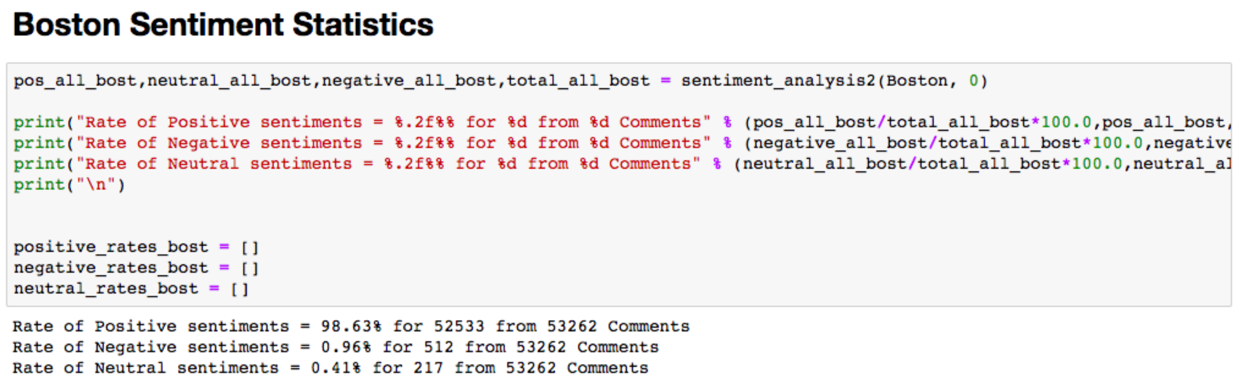
	listing_id	id	date	reviewer_id	reviewer_name	comments	gender	year	month	Sent	pol_sub	Polarity	Subjecetivity
8545	5506	1021	2009-03-21	8903	Jenny	Terry's Hotel Alterntv in Boston was a perfect...	female	2009	3	1.0	(0.75, 0.75)	0.7500	0.75
27026	3353	1749	2009-04-23	12970	Clint	Very friendly and helpful. Convenient location.	male	2009	4	1.0	(0.48750000000000004, 0.65)	0.4875	0.65

Now with the comments with their sentiments we decided to find out how many comments were positive, negative or neutral.

Seattle Sentiment Statistics

```
pos_all,neutral_all,negative_all,total_all = sentiment_analysis2(Seattle, 0)

print("Rate of Positive sentiments = %.2f%% for %d from %d Comments" % (pos_all/total_all*100.0,pos_all, total_all))
print("Rate of Negative sentiments = %.2f%% for %d from %d Comments" % (negative_all/total_all*100.0,negative_all, tota
print("Rate of Neutral sentiments = %.2f%% for %d from %d Comments" % (neutral_all/total_all*100.0,neutral_all, total_a
print("\n")


positive_rates = []
negative_rates = []
neutral_rates = []
```

```
Rate of Positive sentiments = 99.30% for 71168 from 71670 Comments
Rate of Negative sentiments = 0.43% for 306 from 71670 Comments
Rate of Neutral sentiments = 0.27% for 196 from 71670 Comments
```



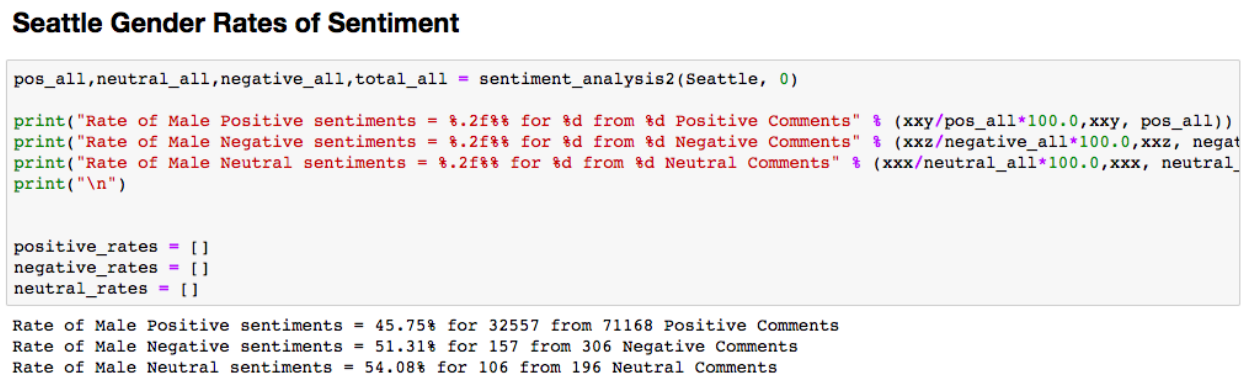
Boston Sentiment Statistics

```
pos_all_bost,neutral_all_bost,negative_all_bost,total_all_bost = sentiment_analysis2(Boston, 0)

print("Rate of Positive sentiments = %.2f%% for %d from %d Comments" % (pos_all_bost/total_all_bost*100.0,pos_all_bost,
print("Rate of Negative sentiments = %.2f%% for %d from %d Comments" % (negative_all_bost/total_all_bost*100.0,negative
print("Rate of Neutral sentiments = %.2f%% for %d from %d Comments" % (neutral_all_bost/total_all_bost*100.0,neutral_al
print("\n")


positive_rates_bost = []
negative_rates_bost = []
neutral_rates_bost = []
```

```
Rate of Positive sentiments = 98.63% for 52533 from 53262 Comments
Rate of Negative sentiments = 0.96% for 512 from 53262 Comments
Rate of Neutral sentiments = 0.41% for 217 from 53262 Comments
```

We were rather surprised that more than the majority of comments were positive. The few comments that were negative or neutral made absolutely no impact in the calculations. This find stumped us and left us wondering what model are we going to implement? Based on our results future predictions are going to be positive!

Another aspect we decide to investigate as well was the sentiment rates for genders.



Seattle Gender Rates of Sentiment

```
pos_all,neutral_all,negative_all,total_all = sentiment_analysis2(Seattle, 0)

print("Rate of Male Positive sentiments = %.2f%% for %d from %d Positive Comments" % (xxy/pos_all*100.0,xxy, pos_all))
print("Rate of Male Negative sentiments = %.2f%% for %d from %d Negative Comments" % (xxz/negative_all*100.0,xxz, negat
print("Rate of Male Neutral sentiments = %.2f%% for %d from %d Neutral Comments" % (xxx/neutral_all*100.0,xxx, neutral_
print("\n")


positive_rates = []
negative_rates = []
neutral_rates = []
```

```
Rate of Male Positive sentiments = 45.75% for 32557 from 71168 Positive Comments
Rate of Male Negative sentiments = 51.31% for 157 from 306 Negative Comments
Rate of Male Neutral sentiments = 54.08% for 106 from 196 Neutral Comments
```

```
pos_all,neutral_all,negative_all,total_all = sentiment_analysis2(Seattle, 0)

print("Rate of Female Positive sentiments = %.2f%% for %d from %d Positive Comments" % (bbb/pos_all*100.0,bbb, pos_all
print("Rate of Female Negative sentiments = %.2f%% for %d from %d Negative Comments" % (bbd/negative_all*100.0,bbd, neg
print("Rate of Female Neutral sentiments = %.2f%% for %d from %d Neutral Comments" % (bbc/neutral_all*100.0,bbc, neutra
print("\n")


positive_rates = []
negative_rates = []
neutral_rates = []
```

```
Rate of Female Positive sentiments = 54.25% for 38611 from 71168 Positive Comments
Rate of Female Negative sentiments = 48.69% for 149 from 306 Negative Comments
Rate of Female Neutral sentiments = 45.92% for 90 from 196 Neutral Comments
```

Boston Gender Rates of Sentiment

```
pos_all_bost,neutral_all_bost,negative_all_bost,total_all_bost = sentiment_analysis2(Boston, 0)

print("Rate of Male Positive sentiments = %.2f%% for %d from %d Positive Comments" % (aaa/pos_all_bost*100.0,aaa, pos_a
print("Rate of Male Negative sentiments = %.2f%% for %d from %d Negative Comments" % (aac/negative_all_bost*100.0,aac,
print("Rate of Male Neutral sentiments = %.2f%% for %d from %d Neutral Comments" % (aab/neutral_all_bost*100.0,aab, neu
print("\n")


positive_rates_bost = []
negative_rates_bost = []
neutral_rates_bost = []
```

```
Rate of Male Positive sentiments = 49.18% for 25836 from 52533 Positive Comments
Rate of Male Negative sentiments = 49.80% for 255 from 512 Negative Comments
Rate of Male Neutral sentiments = 62.67% for 136 from 217 Neutral Comments
```

```
pos_all_bost,neutral_all_bost,negative_all_bost,total_all_bost = sentiment_analysis2(Boston, 0)

print("Rate of Female Positive sentiments = %.2f%% for %d from %d Positive Comments" % (ddd/pos_all_bost*100.0,ddd, pos
print("Rate of Female Negative sentiments = %.2f%% for %d from %d Negative Comments" % (ddf/negative_all_bost*100.0,ddf
print("Rate of Female Neutral sentiments = %.2f%% for %d from %d Neutral Comments" % (dde/neutral_all_bost*100.0,dde,
print("\n")


positive_rates = []
negative_rates = []
neutral_rates = []
```

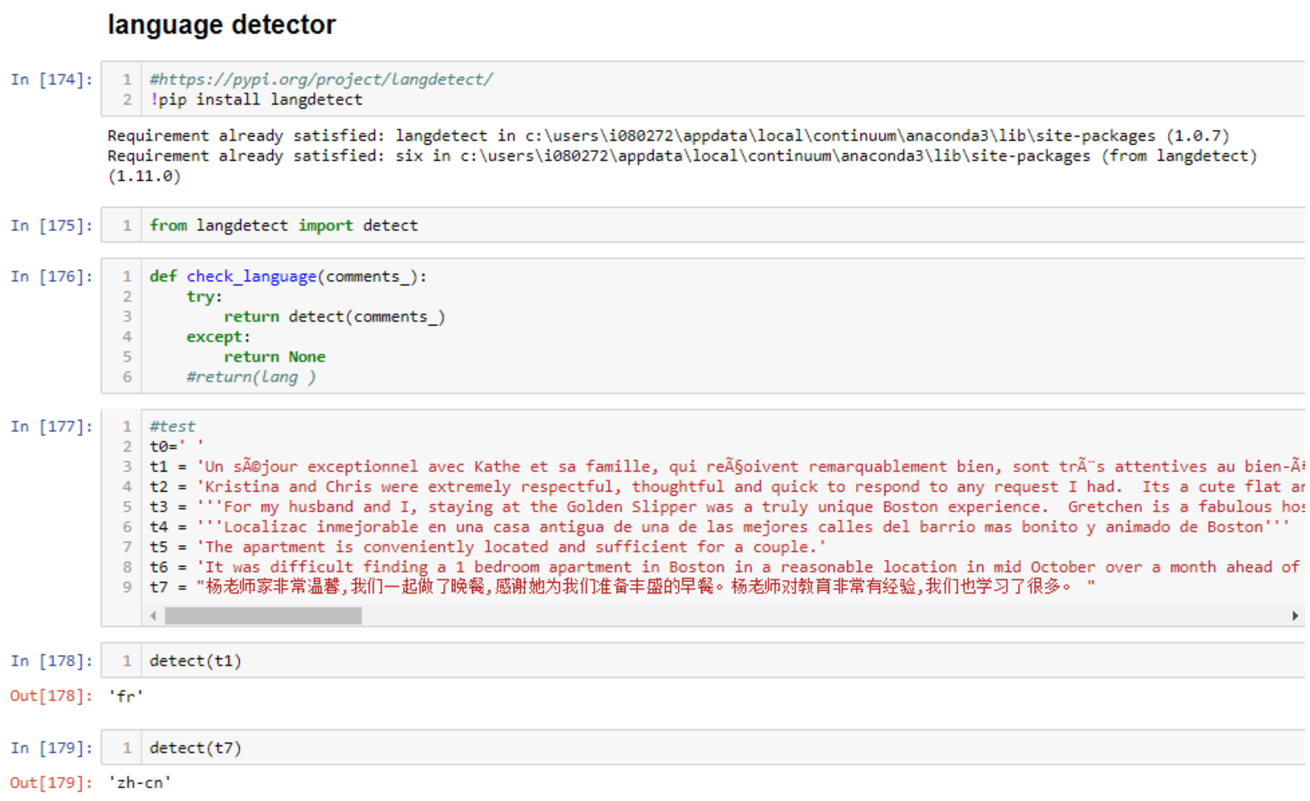
```
Rate of Female Positive sentiments = 50.82% for 26697 from 52533 Positive Comments
Rate of Female Negative sentiments = 50.20% for 257 from 512 Negative Comments
Rate of Female Neutral sentiments = 37.33% for 81 from 217 Neutral Comments
```

Notes on using textblob – Stopwords and Foreign Language

We used textblob (https://pypi.org/project/textblob/) to determine polarity and subjectivity of the sentences that were submitted as comments. We did some quick tests to see if stopwords had an impact on polarity and after testing determined that they did not matter. We did find that using Textblob sentiment scoring on non-English text almost always give a neutral score. This is probably due textblob finding no English negative or positive words and marking it as neutral. To remedy this, we used a language detection module to classify text as non-English and remove from the dataset.

Notes on Language Detect

We found that the language detect functionality in textblob took a very long time to process. We used langdetect (https://pypi.org/project/langdetect/) as an alternative. In the documentation for this tool it states that under 30 characters may result in inaccurate language detection. We found that to be true. For example, the phrase "Nice Condo" would be classified as Portuguese. We decided that if the language was not classified as English then we would drop the comment from our analysis.

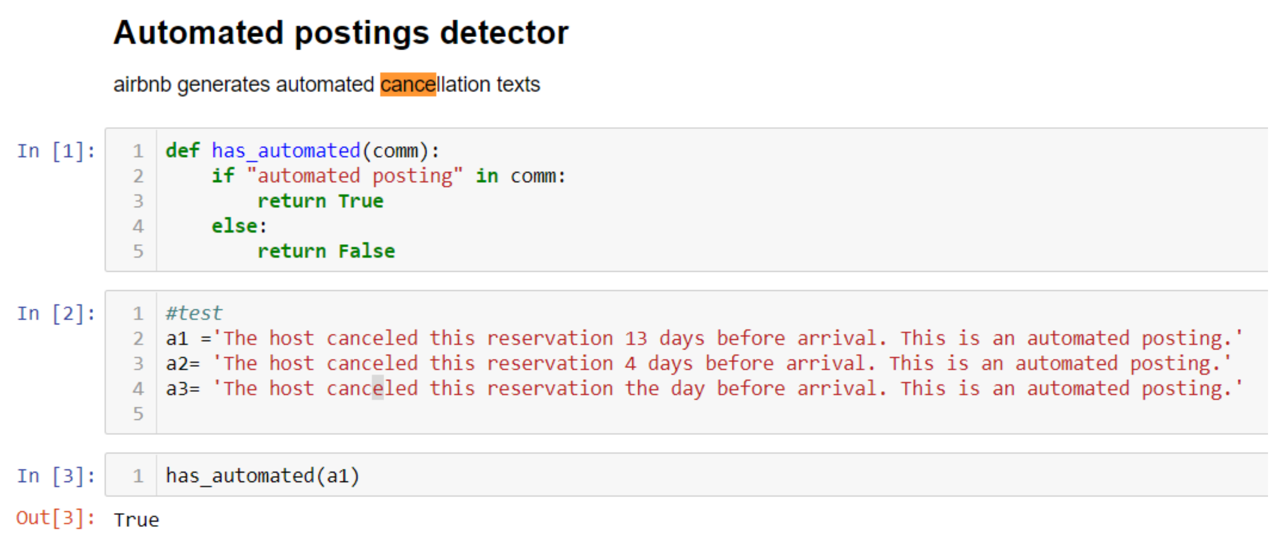


language detector

```
In [174]: #https://pypi.org/project/langdetect/
          !pip install langdetect

Requirement already satisfied: langdetect in c:\users\i080272\appdata\local\continuum\anaconda3\lib\site-packages (1.0.7)
Requirement already satisfied: six in c:\users\i080272\appdata\local\continuum\anaconda3\lib\site-packages (from langdetect)
(1.11.0)

In [175]: from langdetect import detect

In [176]: def check_language(comments_):
              try:
                  return detect(comments_)
              except:
                  return None
              #return(lang )

In [177]: #test
          t0=' '
          t1 = 'Un sÃ©jour exceptionnel avec Kathe et sa famille, qui reÃ§oivent remarquablement bien, sont trÃ¨s attentives au bien-Ã
          t2 = 'Kristina and Chris were extremely respectful, thoughtful and quick to respond to any request I had.  Its a cute flat ar
          t3 = '''For my husband and I, staying at the Golden Slipper was a truly unique Boston experience.  Gretchen is a fabulous hos
          t4 = '''Localizac inmejorable en una casa antigua de una de las mejores calles del barrio mas bonito y animado de Boston'''
          t5 = 'The apartment is conveniently located and sufficient for a couple.'
          t6 = 'It was difficult finding a 1 bedroom apartment in Boston in a reasonable location in mid October over a month ahead of
          t7 = "杨老师家非常温馨,我们一起做了晚餐,感谢她为我们准备丰盛的早餐。杨老师对教育非常有经验,我们也学习了很多。 "

In [178]: detect(t1)
Out[178]: 'fr'

In [179]: detect(t7)
Out[179]: 'zh-cn'
```

Issues with Autogenerated Text

We examined the reviews comments and found that many of them had a stock sentence that seems to be provided by Airbnb in case of cancellation.


Automated postings detector

airbnb generates automated cancellation texts

```
In [1]:
def has_automated(comm):
    if "automated posting" in comm:
        return True
    else:
        return False
```

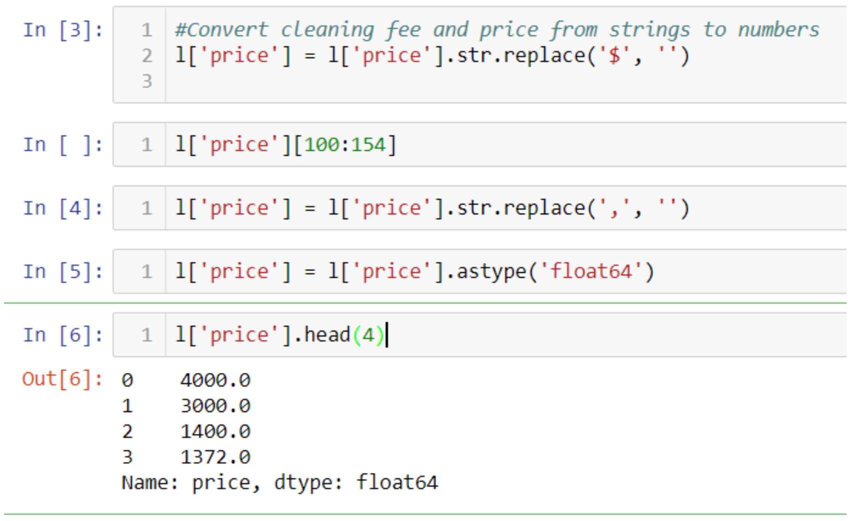
```
In [2]:
#test
a1 ='The host canceled this reservation 13 days before arrival. This is an automated posting.'
a2= 'The host canceled this reservation 4 days before arrival. This is an automated posting.'
a3= 'The host canceled this reservation the day before arrival. This is an automated posting.'
```

```
In [3]:
has_automated(a1)
```

```
Out[3]: True
```

Issues with Prices

In reviewing the data, we noticed that price was coming in as an object in Pandas. This indicated that it was being treated as a string “$120.00” rather than a numeric value. We wrote a quick routine to remove the “$” and convert float. We noticed another issue which the use of a comma was to denote thousands, i.e. $4,000.00 this issue was also removed from the data. Many of the reviews were missing pricing data. To compensate for this we sorted the data in list id order and then used forward fill to provide prices for reviews based on the listing property id: (`listing['price'] = l['price'].ffill()`)



```
In [3]:
#Convert cleaning fee and price from strings to numbers
l['price'] = l['price'].str.replace('$', '')
```

```
In [ ]:
l['price'][100:154]
```

```
In [4]:
l['price'] = l['price'].str.replace(',', '')
```

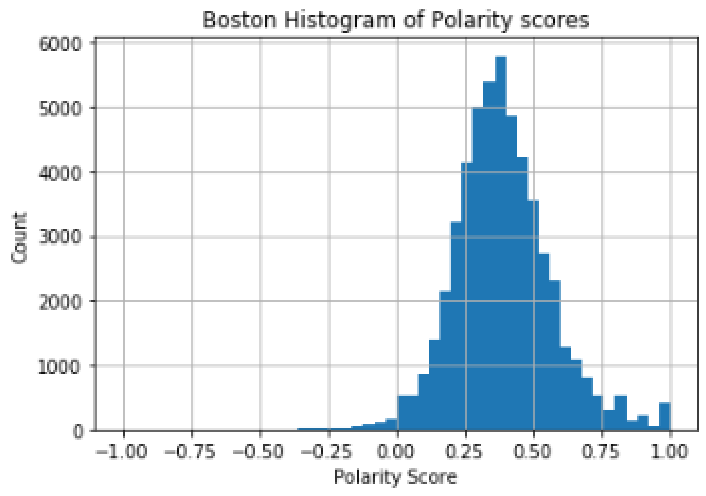
```
In [5]:
l['price'] = l['price'].astype('float64')
```

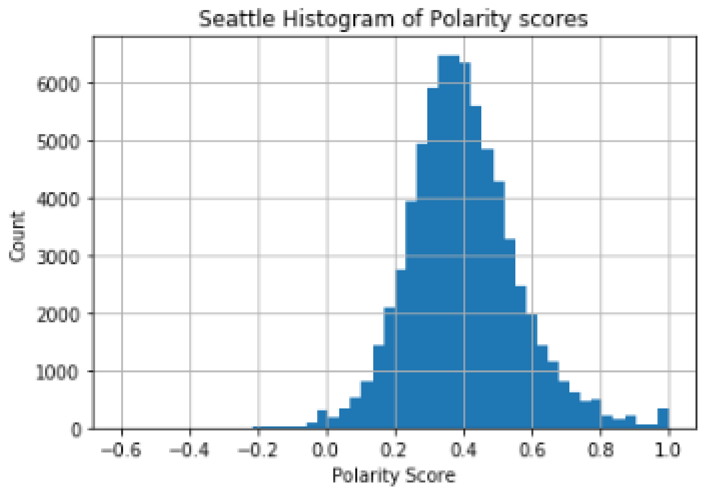
```
In [6]:
l['price'].head(4)
```

```
Out[6]: 0    4000.0
        1    3000.0
        2    1400.0
        3    1372.0
        Name: price, dtype: float64
```

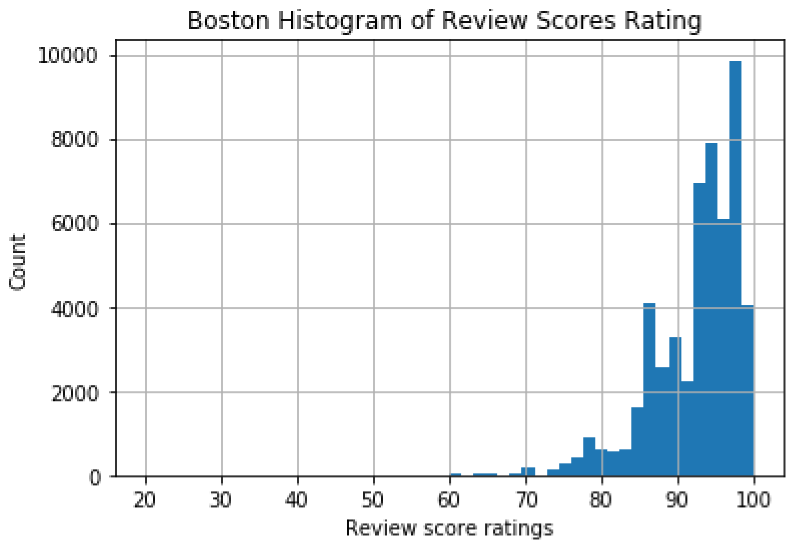
Analysis Section

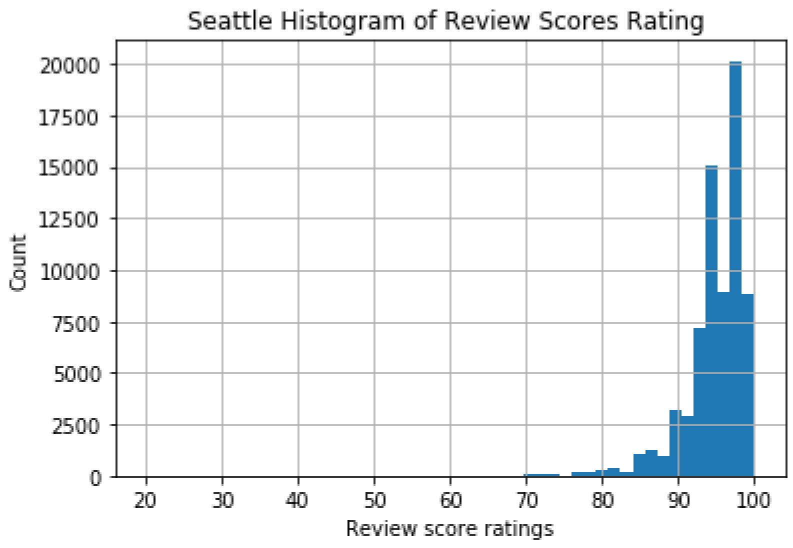
We reviewed the data and looked for interesting patterns. Looking at the distribution of Polarity scores between Boston and Seattle we can see that they are about the same with Seattle being wider probably due to more data points sampled.



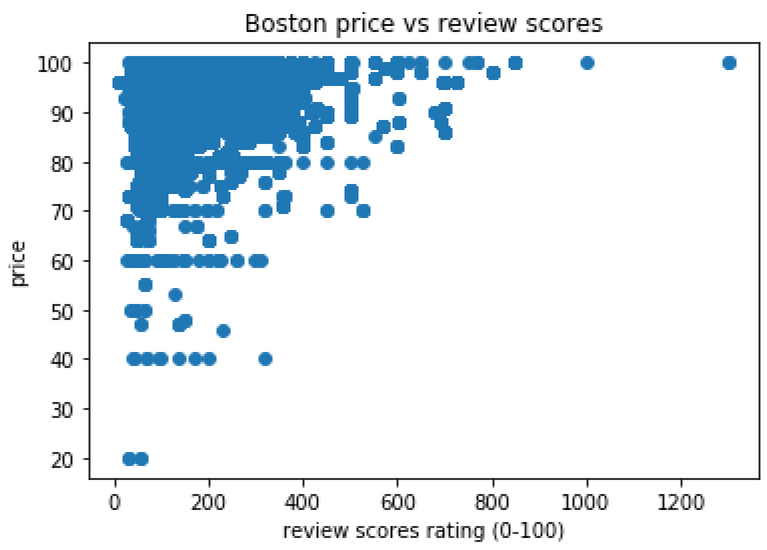


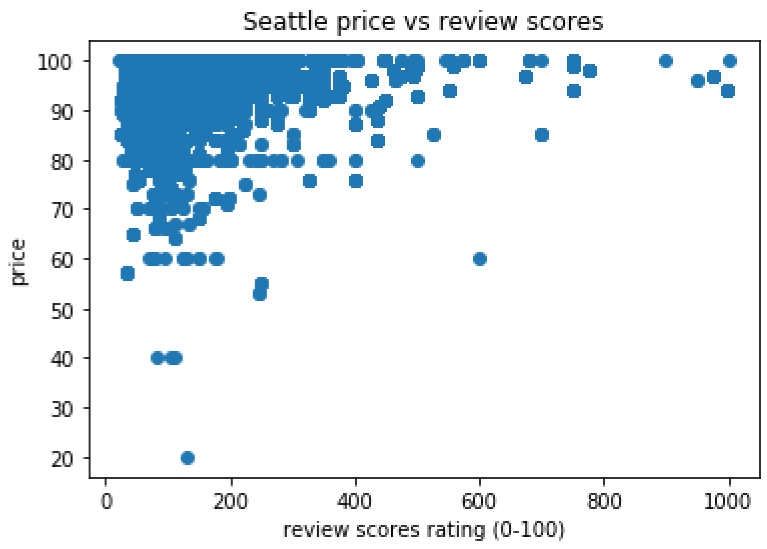
Looking at review scores are also very comparable where both curves are cut off the maximum of 100 and most scores above 70%.





Looking at Price versus review scores both Boston and Seattle are very similar with Seattle having more scores and more outliers as a result.





Gender was discussed above and noted that it is approximately the same distribution for Boston and Seattle.

We created several models that uses polarity score from the comments to predict reviewer_score from both Boston and Seattle. The results were for Boston using polarity to predict review scores was

- KNNeighbors classifier (with neighbors=3) : 99.48%
- Bayesian classifier: 99.98%
- Decision tree classifier: 99.97%

And for Seattle the results were:

- KNNeighbors classifier (with neighbors=3) : 99.50%
- Bayesian classifier: 99.98%
- Decision tree classifier: 99.98%

Using Polarity as a predictor of review score for both Boston and Seattle resulted in highly accurate models. This is not surprising because the distributions of polarity for Boston and Seattle were almost identical as were the distributions of the review scores for both Boston and Seattle.

As an additional test we looked at whether price, gender, beds and the accommodation number could be used to predict review score. Again, we ran the model on both Boston and Seattle using the same sets of classifiers. For Boston

- KNNeighbors classifier (with neighbors=3) : 99.11%
- Bayesian classifier: 99.98%
- Decision tree classifier: 99.98%

And for Seattle the results were:

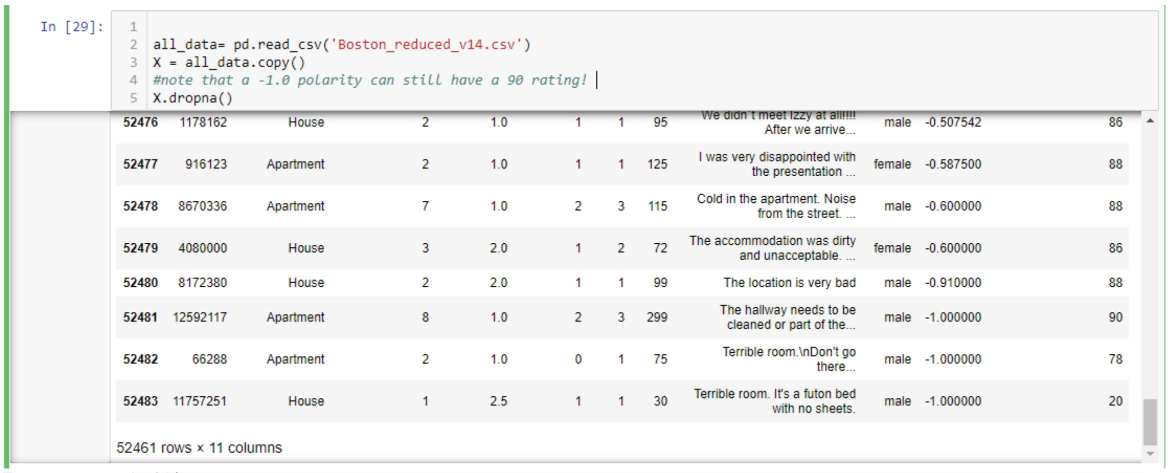
- KNNeighbors classifier (with neighbors=3) : 99.50%
- Bayesian classifier: 99.99%
- Decision tree classifier: 100%

The results were approximately the same with decision tree classifier scoring a 100%. Based on these models having price, gender, beds and accommodation adds enough information to very accurately predict the review score.

Discussion

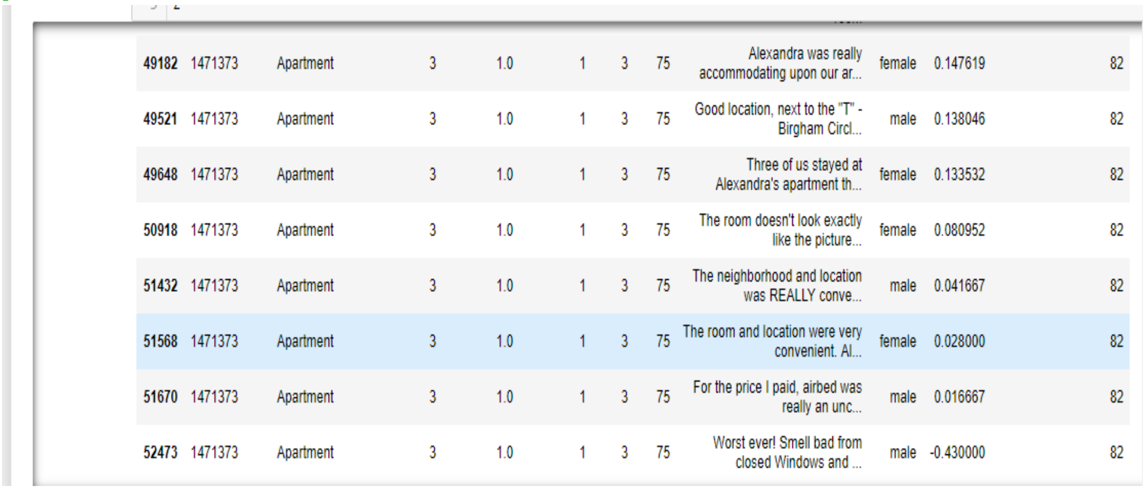
From the data analysis section, we can see that the data for polarity as derived by textblob has the same basic skew and distribution for both Boston and Seattle. In addition, the review scores between both Boston and Seattle appear approximately the same with a large skew to the 100 review score. We believe that this led to the creation of a highly accurate model based on polarity. Looking at the distributions for price, gender, beds and accommodates (number of people in a listed property) we see that again the distribution in price, gender (approximate 50/50), beds and accommodates is again very similar between Boston and Seattle. When this data is used to create a model to predict review score a highly accurate, from a prediction point of view, is created with decision tree classifier scoring 100%.

Why then would extremely negative reviews result in high ratings? Looking at the data we could see that there were very negative comments that still had very good reviewer score.   The comment of “Worst ever! Smells bad…” would result in a polarity of -.430 and yet the review score would be 82. By quantifying the comments by polarity score we can clearly see that the comment does not translate in review score.


In [29]:
```

all_data= pd.read_csv('Boston_reduced_v14.csv')
X = all_data.copy()
#note that a -1.0 polarity can still have a 90 rating!
X.dropna()
```

52476	1178162	House	2	1.0	1	1	95	We didn't meet Izzy at all!!!! After we arrive...	male	-0.507542	86
52477	916123	Apartment	2	1.0	1	1	125	I was very disappointed with the presentation ...	female	-0.587500	88
52478	8670336	Apartment	7	1.0	2	3	115	Cold in the apartment. Noise from the street. ...	male	-0.600000	88
52479	4080000	House	3	2.0	1	2	72	The accommodation was dirty and unacceptable. ...	female	-0.600000	86
52480	8172380	House	2	2.0	1	1	99	The location is very bad	male	-0.910000	88
52481	12592117	Apartment	8	1.0	2	3	299	The hallway needs to be cleaned or part of the...	male	-1.000000	90
52482	66288	Apartment	2	1.0	0	1	75	Terrible room.\nDon't go there...	male	-1.000000	78
52483	11757251	House	1	2.5	1	1	30	Terrible room. It's a futon bed with no sheets.	male	-1.000000	20

52461 rows × 11 columns



49182	1471373	Apartment	3	1.0	1	3	75	Alexandra was really accommodating upon our ar...	female	0.147619	82
49521	1471373	Apartment	3	1.0	1	3	75	Good location, next to the "T" - Birgham Circl...	male	0.138046	82
49648	1471373	Apartment	3	1.0	1	3	75	Three of us stayed at Alexandra's apartment th...	female	0.133532	82
50918	1471373	Apartment	3	1.0	1	3	75	The room doesn't look exactly like the picture...	female	0.080952	82
51432	1471373	Apartment	3	1.0	1	3	75	The neighborhood and location was REALLY conve...	male	0.041667	82
51568	1471373	Apartment	3	1.0	1	3	75	The room and location were very convenient. Al...	female	0.028000	82
51670	1471373	Apartment	3	1.0	1	3	75	For the price I paid, airbed was really an unc...	male	0.016667	82
52473	1471373	Apartment	3	1.0	1	3	75	Worst ever! Smell bad from closed Windows and ...	male	-0.430000	82

We then looked for other explanations and found that at Airbnb a reviewer’s name and photo is linked to their id (2). This meant that their score would be visible to future hosts and this could be a deciding factor in whether a future host would allow the reviewer to stay at their property. If this is the case, then the review score is only loosely correlated with comments. The comments do not drive the review scores but rather separate data points. Another factor that leads to better scores is that a host could get a low score and then rectify the offending issue. This way over several rentals the scores would increase and average out the lower initial scores. An interesting future project would be to devise a review score based on the comments and compare that to published scores.

Conclusion

We created several models to predict review score one based on the polarity of the comments and the other based on features such as price, gender, beds, and accommodates for Airbnb data for Boston and Seattle. The models were highly predictive, but we believe that this is more likely due to the similarity and distribution of the underlying data and not to the semantic score of the comments based on polarity. More work would need to be done to determine if sentiment analysis in the form of polarity scores could be used to derive a more "honest" review score.

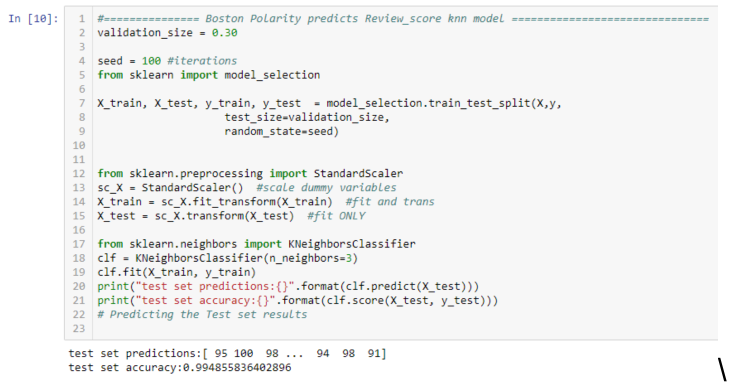
References

1. Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

2. "After a disappointing Airbnb stay, I realized there's a major flaw in the review system", https://www.businessinsider.com/why-airbnb-reviews-are-a-problem-for-the-site-2015-6, accessed June 15, 2019

Appendix 1 Results Using Polarity as Predictor or Review score

Boston Polarity predicts review score KNN=3

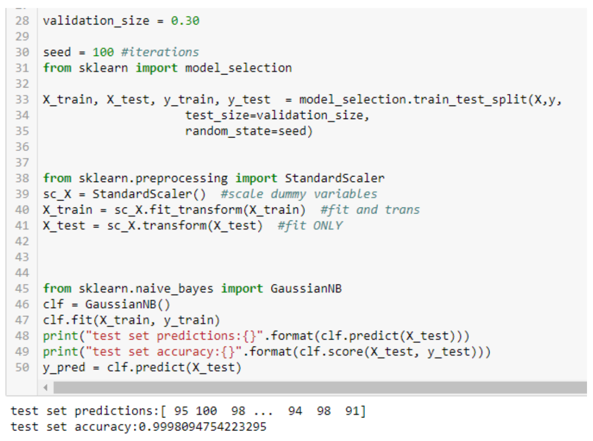


```
In [10]:
#=============== Boston Polarity predicts Review_score knn model ===============================
validation_size = 0.30

seed = 100 #iterations
from sklearn import model_selection

X_train, X_test, y_train, y_test  = model_selection.train_test_split(X,y,
                    test_size=validation_size,
                    random_state=seed)


from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()  #scale dummy variables
X_train = sc_X.fit_transform(X_train)  #fit and trans
X_test = sc_X.transform(X_test)  #fit ONLY

from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
print("test set predictions:{}".format(clf.predict(X_test)))
print("test set accuracy:{}".format(clf.score(X_test, y_test)))
# Predicting the Test set results

```

```
test set predictions:[ 95 100  98 ...  94  98  91]
test set accuracy:0.994855836402896
```

\

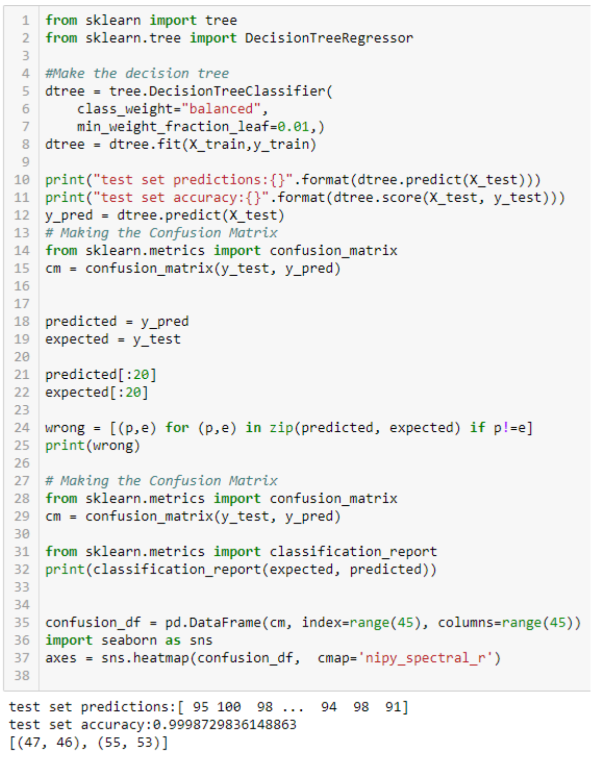
Boston Polarity predicts review score Bayesian



```
validation_size = 0.30

seed = 100 #iterations
from sklearn import model_selection

X_train, X_test, y_train, y_test  = model_selection.train_test_split(X,y,
                    test_size=validation_size,
                    random_state=seed)


from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()  #scale dummy variables
X_train = sc_X.fit_transform(X_train)  #fit and trans
X_test = sc_X.transform(X_test)  #fit ONLY



from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
print("test set predictions:{}".format(clf.predict(X_test)))
print("test set accuracy:{}".format(clf.score(X_test, y_test)))
y_pred = clf.predict(X_test)
```

```
test set predictions:[ 95 100  98 ...  94  98  91]
test set accuracy:0.9998094754223295
```

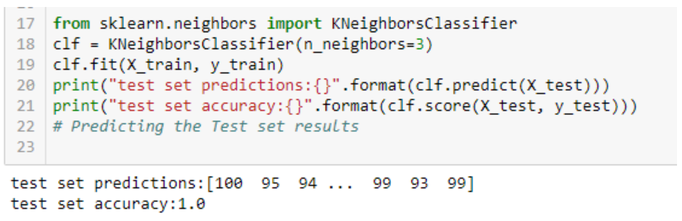
Boston Polarity predicts review score Decision Tree



```python
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor

#Make the decision tree
dtree = tree.DecisionTreeClassifier(
    class_weight="balanced",
    min_weight_fraction_leaf=0.01,)
dtree = dtree.fit(X_train,y_train)

print("test set predictions:{}".format(dtree.predict(X_test)))
print("test set accuracy:{}".format(dtree.score(X_test, y_test)))
y_pred = dtree.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)


predicted = y_pred
expected = y_test

predicted[:20]
expected[:20]

wrong = [(p,e) for (p,e) in zip(predicted, expected) if p!=e]
print(wrong)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

from sklearn.metrics import classification_report
print(classification_report(expected, predicted))


confusion_df = pd.DataFrame(cm, index=range(45), columns=range(45))
import seaborn as sns
axes = sns.heatmap(confusion_df,  cmap='nipy_spectral_r')

```

```
test set predictions:[ 95 100  98 ...  94  98  91]
test set accuracy:0.9998729836148863
[(47, 46), (55, 53)]
```

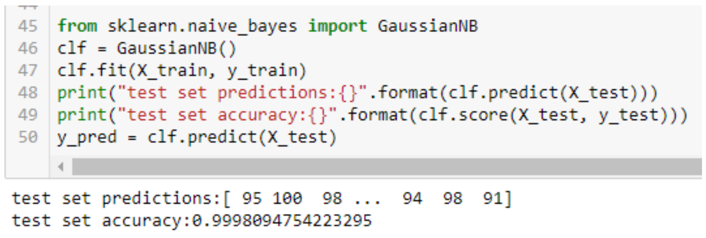
Seattle KNN=3



```python
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
print("test set predictions:{}".format(clf.predict(X_test)))
print("test set accuracy:{}".format(clf.score(X_test, y_test)))
# Predicting the Test set results

```

```
test set predictions:[100  95  94 ...  99  93  99]
test set accuracy:1.0
```

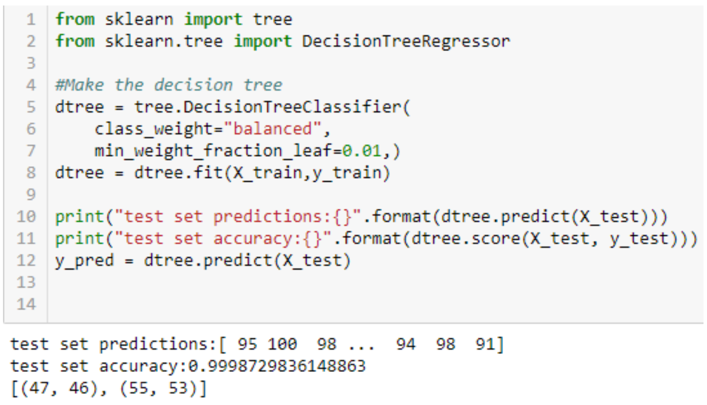
Seattle Naïve Bayes



```python
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
print("test set predictions:{}".format(clf.predict(X_test)))
print("test set accuracy:{}".format(clf.score(X_test, y_test)))
y_pred = clf.predict(X_test)
```

```
test set predictions:[ 95 100  98 ...  94  98  91]
test set accuracy:0.9998094754223295
```

Seattle Decision Tree

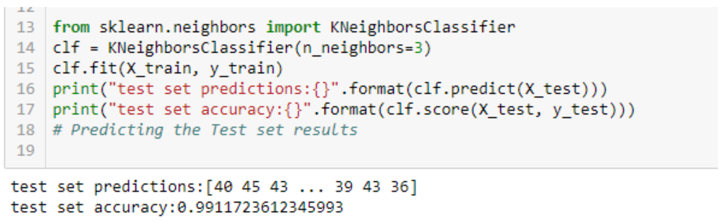


```python
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor

#Make the decision tree
dtree = tree.DecisionTreeClassifier(
    class_weight="balanced",
    min_weight_fraction_leaf=0.01,)
dtree = dtree.fit(X_train,y_train)

print("test set predictions:{}".format(dtree.predict(X_test)))
print("test set accuracy:{}".format(dtree.score(X_test, y_test)))
y_pred = dtree.predict(X_test)
```

```
test set predictions:[ 95 100  98 ...  94  98  91]
test set accuracy:0.9998729836148863
[(47, 46), (55, 53)]
```

Appendix 2 Results Using Accommodates Bed Price and Gender as Predictor or Review score

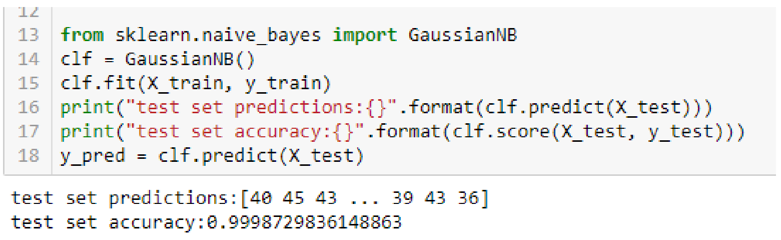
Boston KNN=3



```python
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
print("test set predictions:{}".format(clf.predict(X_test)))
print("test set accuracy:{}".format(clf.score(X_test, y_test)))
# Predicting the Test set results
```

```
test set predictions:[40 45 43 ... 39 43 36]
test set accuracy:0.9911723612345993
```

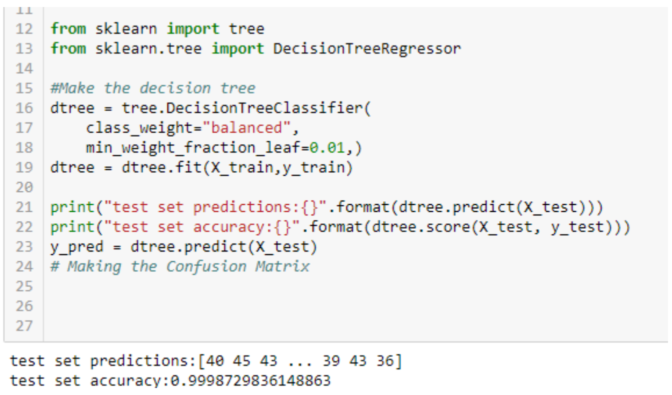
Boston Naïve Bayes



```python
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
print("test set predictions:{}".format(clf.predict(X_test)))
print("test set accuracy:{}".format(clf.score(X_test, y_test)))
y_pred = clf.predict(X_test)
```

```
test set predictions:[40 45 43 ... 39 43 36]
test set accuracy:0.9998729836148863
```

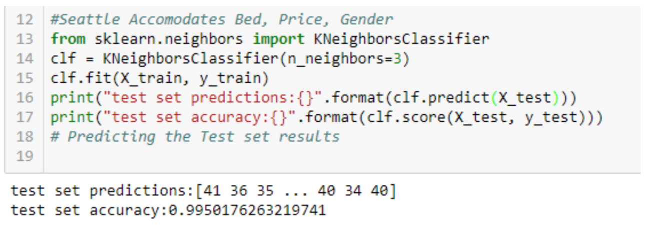
Boston Decision Tree



```python
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor

#Make the decision tree
dtree = tree.DecisionTreeClassifier(
    class_weight="balanced",
    min_weight_fraction_leaf=0.01,)
dtree = dtree.fit(X_train,y_train)

print("test set predictions:{}".format(dtree.predict(X_test)))
print("test set accuracy:{}".format(dtree.score(X_test, y_test)))
y_pred = dtree.predict(X_test)
# Making the Confusion Matrix
```

```
test set predictions:[40 45 43 ... 39 43 36]
test set accuracy:0.9998729836148863
```

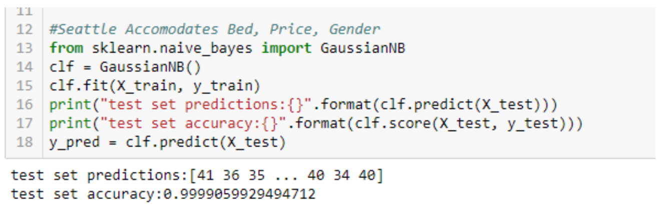
Seattle KNN=3



```
#Seattle Accomodates Bed, Price, Gender
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
print("test set predictions:{}".format(clf.predict(X_test)))
print("test set accuracy:{}".format(clf.score(X_test, y_test)))
# Predicting the Test set results

```

```
test set predictions:[41 36 35 ... 40 34 40]
test set accuracy:0.9950176263219741
```

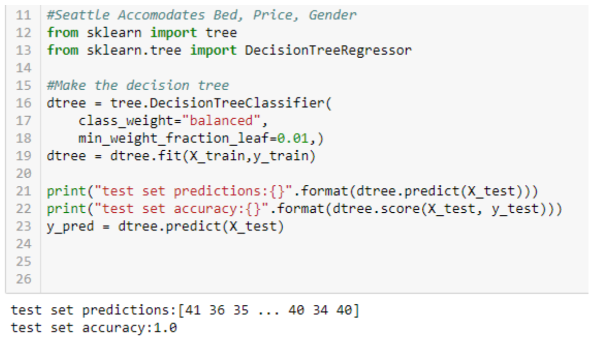
Seattle Naïve Bayes



```
#Seattle Accomodates Bed, Price, Gender
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
print("test set predictions:{}".format(clf.predict(X_test)))
print("test set accuracy:{}".format(clf.score(X_test, y_test)))
y_pred = clf.predict(X_test)
```

```
test set predictions:[41 36 35 ... 40 34 40]
test set accuracy:0.9999059929494712
```

Seattle Decision Tree



```
#Seattle Accomodates Bed, Price, Gender
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor

#Make the decision tree
dtree = tree.DecisionTreeClassifier(
    class_weight="balanced",
    min_weight_fraction_leaf=0.01,)
dtree = dtree.fit(X_train,y_train)

print("test set predictions:{}".format(dtree.predict(X_test)))
print("test set accuracy:{}".format(dtree.score(X_test, y_test)))
y_pred = dtree.predict(X_test)



```

```
test set predictions:[41 36 35 ... 40 34 40]
test set accuracy:1.0
```