# CAPSTONE PROJECT

ALY6020: PREDICTIVE ANALYTICS

AIRBNB DADASET: THE NEXT STEP TO VISITOR JOURNEY

TRUNG PHAM LE MINH

NORTHEASTERN UNIVERSITY – SILLICON VALLEY

## Table of Contents

# AIRBNB: THE NEXT STEP TO VISITOR JOURNEY

## 1.0 INTRODUCTION

### 1.1 Problem Statement

Airbnb, often referred as “the reinvention of hospitality service” in 21st century, is one of the major disruptive technologies in the past decade. Not only did its business model change the way hotel accommodation works, but it also changed the way tourism industry functions. Its business model is growing faster and bigger than ever. Started off as just “air-bed-n-breakfast”, Airbnb now provides millions of listings all over the world, with full package experiences such as experience rentals to restaurant bookings. Its products and services is a big part of its success and all of its success started off at the core of its product: Airbnb.com.

Visitors, travelers from all over the world visits this site daily, to find the right experience for their upcoming trip. What’s so unique about Airbnb is that it provides a very succinct UI and SEO for its customers, that within a few seconds, a list will pop up of all the listings of your desired needs. This “journey” of the customer from the moment they start thinking about a trip to booking a trip on Airbnb can be referred as the “pre-experience” where they haven’t experienced anything yet, but they have already planned out the trip. You might wonder, is the post-experience as important as the pre-experience? Whilst planning and picking the safest places is crucial for its business, Airbnb has a **Reviews** system intact for its travelers once the trip is completed.

The reviews system can be broken down like this:

1. Planning stage: Airbnb customer found a listing they like to book, according to the search optimization where Airbnb filters out customer’s demands into a selective group of listings
2. Booking stage: The listing is booked
3. Experience stage: The host of the listing and the traveler meets (where the service happens)
4. Post-experience stage: When the traveler finishes their stay at the listing
5. Review stage: Both the host and the traveler get to review the place for others to see

The reviews can be a paragraph, text or however the traveler or the host want to talk about. Usually at the end of the stay and the review, a rating score is used to rate the listing and the overall experience. By this time, we can see that the system rewards listings that have good experience and does not reward listings that do not. There will be a discrepancy between lower rating homes and higher rating homes, which is fair. But in an extreme light, a lower rating home will forever not be booked whilst a higher rating home will always be of priority.

### 1.2 Goal

The goal for this project is to **predict and classify reviews into ratings scale of 1 to 5 to see what differences we can make to improve the quality of all listings, and not just higher home ratings.** In other words, we will be more focused on what makes a home listing good and what makes a home listing bad and how we can reduce this discrepancy for the overall experience of Airbnb visitors.

## 1.3 The Data

The Airbnb database consists of data of listings from all over the world, and when we talk about reviews, a listing can have many reviews. Therefore, in picking our population and sample size, we cannot pick the whole population since there is way too many reviews to analyze! Hence, we will be looking at reviews of English-speaking language for listings in Boston, Massachusetts United States. There are two datasets we will be looking at: the listings dataset (detailed listings data for Boston) and reviews dataset (detailed review data for Boston).

The reviews dataset has approximately 199330 observations and consists of 6 variables:

- **Listing ID: Unique identifier for each listing (Foreign Key)**
- **ID: Unique identifier for each comment/review (Primary Key)**
- Date: Date in which the comment/review was written
- Reviewer ID: Unique identifier of the reviewer
- Reviewer name: First name of Reviewer
- Comments: the reviews itself

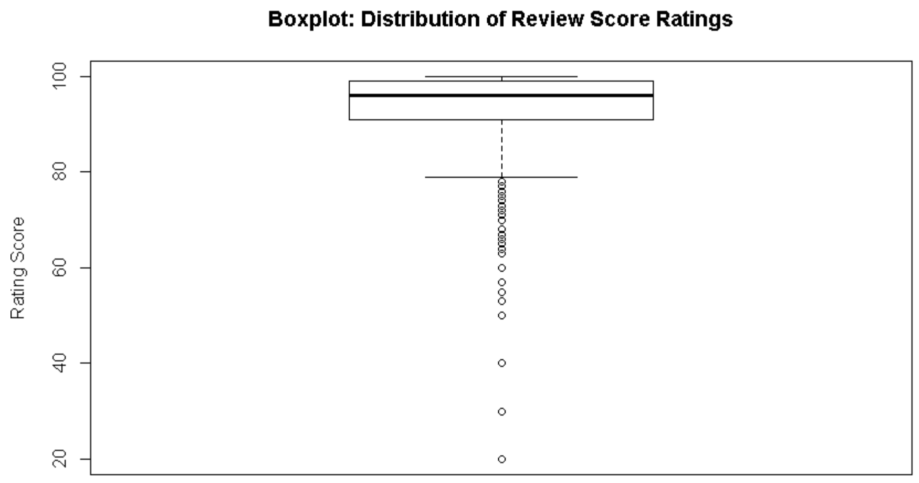
The listings dataset has 6155 observations and consists of 106 variables. This tells us that there are just over 6000 listings in Boston area alone. Since there are a lot of variables in this dataset, we will only be looking at the 2 most important variables where we are concerned about:

- **ID: Unique identifier for each listing (Primary Key and Foreign Key)**
- Review scores rating: Average review scores for listing (0 – 100)

The review scores rating is now on a scale of 0 – 100, however the scale is too wide for the use of our research. We only want a few benchmarks of where the list ratings are low, medium and high. Therefore, we will address this problem in our data cleaning stage.

## 2.0 DATA CLEANING AND PREPARATION

First and foremost, in order to use only a few benchmarks for the ratings of our concern, we must look at the representation of the ratings in all listings:



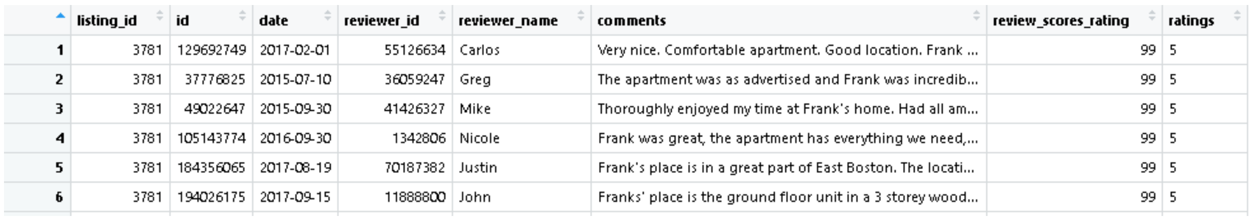
Since we want to use a 5-star rating system, we need to use 4 cut-off thresholds for listings in Boston. From the boxplot, we can see that the distribution of listings range (IQR) from 90 to 96, which is high. Hence, there will be two thresholds at this distribution. Next, we see that there a bunch of outliers at a mark below 80, we will use this as our next threshold. Finally, the few extremes we see can be categorized below 50. With that in mind, we can create a new reviews column with 5-star rating system:

```
levels <- c(-Inf, 50, 80, 90, 95, Inf)
labels <- c("1", "2", "3", "4", "5")
lists_cleaned <- lists %>%
                        select(id, review_scores_rating) %>%
                        mutate(ratings = cut(review_scores_rating, levels, labels = labels))
```

```
> summary(lists_cleaned$ratings)
   1    2    3    4    5 
  29   79 1098 1197 2453 
```

The outcome can be seen above, as we can see the distribution of ratings. There is a problem here in our case, however that there is a clear class bias in our dataset. There are a lot more ratings of 3 and above than ratings of 2 and below. Since we are in no position to collect more data for 1 and 2 star rated listings, we will address this issue in the following sections.

Using the merge function on R, we will be able to join both datasets together using their foreign key (listing ID on ID). The head results of the data frame can be seen below:



| | listing_id | id | date | reviewer_id | reviewer_name | comments | review_scores_rating | ratings |
|---|---|---|---|---|---|---|---|---|
| 1 | 3781 | 129692749 | 2017-02-01 | 55126634 | Carlos | Very nice. Comfortable apartment. Good location. Frank ... | 99 | 5 |
| 2 | 3781 | 37776825 | 2015-07-10 | 36059247 | Greg | The apartment was as advertised and Frank was incredib... | 99 | 5 |
| 3 | 3781 | 49022647 | 2015-09-30 | 41426327 | Mike | Thoroughly enjoyed my time at Frank's home. Had all am... | 99 | 5 |
| 4 | 3781 | 105143774 | 2016-09-30 | 1342806 | Nicole | Frank was great, the apartment has everything we need,... | 99 | 5 |
| 5 | 3781 | 184356065 | 2017-08-19 | 70187382 | Justin | Frank's place is in a great part of East Boston. The locati... | 99 | 5 |
| 6 | 3781 | 194026175 | 2017-09-15 | 11888800 | John | Franks' place is the ground floor unit in a 3 storey wood... | 99 | 5 |

We should now check for any NA values that exists in the dataset:

```
In [120]: print('Number of missing comments in comment text:')
     ...: reviews['comments'].isnull().sum()
Number of missing comments in comment text:
Out[120]: 91
```

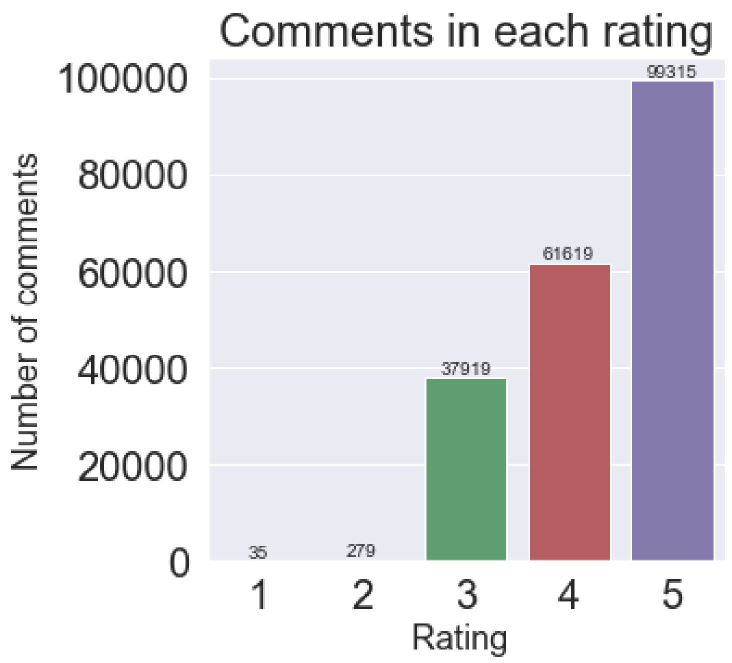
Since there are only 91 missing comments for the dataset, we will be only using complete cases as a result for our dataset. The resulting data frame now has 199,167 observations which is a plenty to analyze.

**3.0 EDA (EXPLORATORY DATA ANALYSIS)**

Before data-preprocessing, we will be looking at Exploratory Data Analysis, which is a crucial part to data analysis where we gain some insights of our data before we do some testing.

<u>3.1 Distribution of number of comments per rating</u>

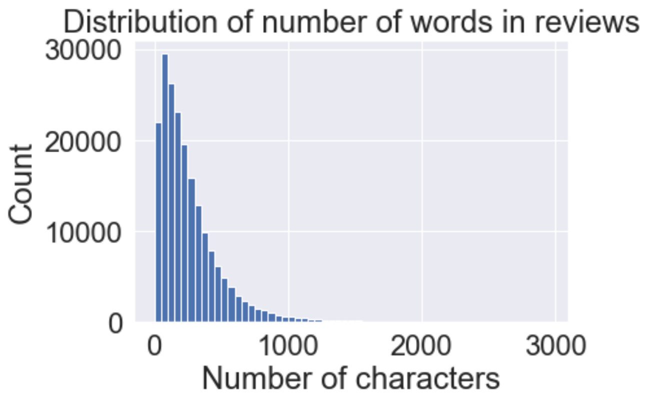
First, we will be looking at the number of comments in each rating category:



This is not surprising as our assumption from the beginning tells us that good listings will be booked more frequently, and as a result have more reviews. Since we also would like to look at lower ratting listings, text-mining and classification processes will help us identify the issues with lower-rating homes.

3.2 Distribution of number of characters in reviews

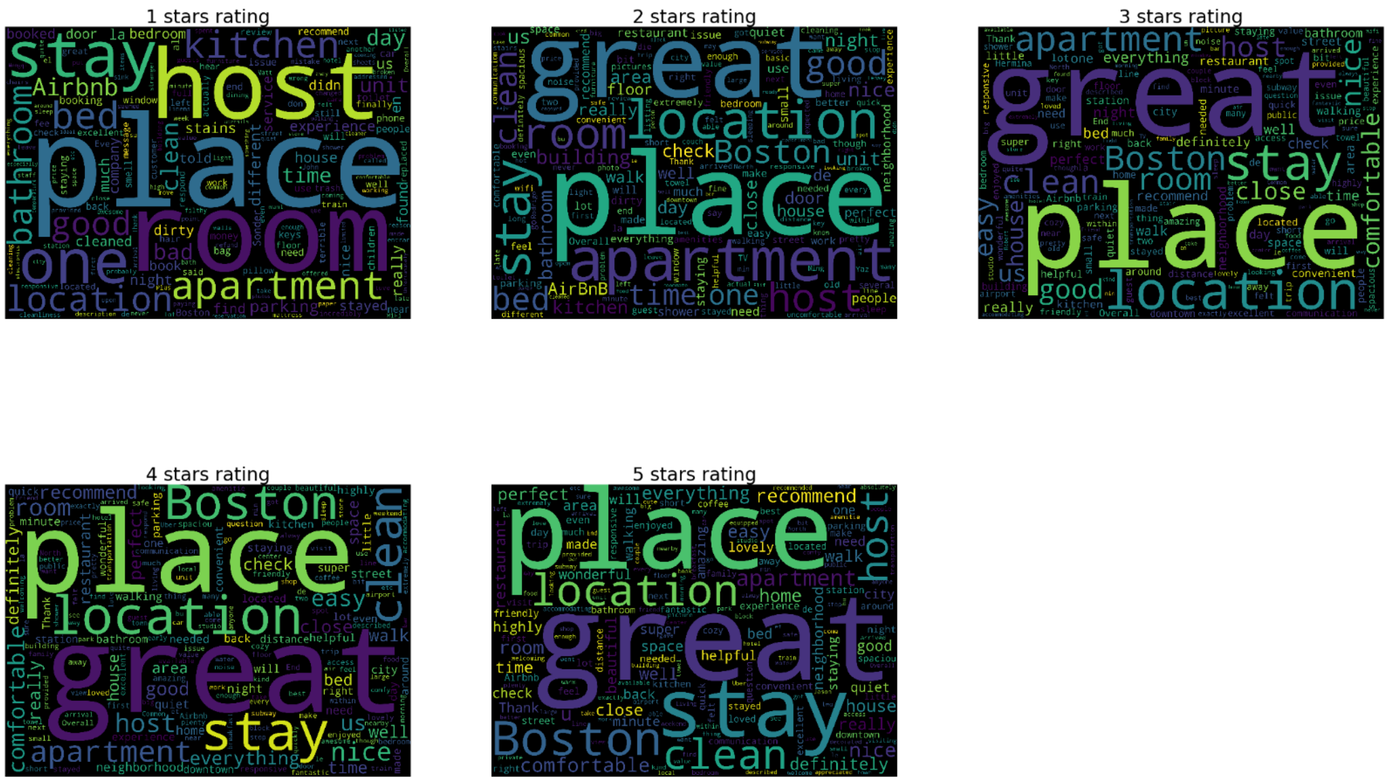
Next, we will be looking at the distribution of number of words in reviews, where we can see typically how many characters reviewers leave for the host and its listing:



We can see that the distribution is positively skewed, since there are a few extremes of guests that writes very long reviews. Typically, the number of characters range from 0 – 1000 characters for reviews as we can see from the graph above.

3.3 Word Cloud

Finally, we will be using a special package from Python: "WordCloud" in order to see the relative frequency of words use for each category of rating homes. The result can be seen below:



The resulting WordCloud tells us a lot about higher rating homes as oppose to lower rating homes. For instance, 4- and 5-star rating homes have a lot of good words use such as “great”, “clean”, “place”, “Boston” whilst 1-star rating home reviews have a lot of words use such as “place”, “room”, “stay”, “host”. Although the resulting word-cloud for lower star rating homes does not give us a good context to the situation, we have some idea that the reviewers are talking about the place, the kitchen, the bed and the room instead of talking about how “great” it is. We will address this issue later on as we use uni-gram, bi-gram and tri-gram in order to get more context from our reviews.

## 4.0 DATA PRE-PROCESSING

There are three major steps in data pre-processing for text mining purposes:

Stop Words**.** The first step is to eliminate stop-words in our reviews. Stop words are words which are filtered out before or after the process of natural learning language data. They are a set of commonly use words in any language, and not just English. The reason behind removing stop words in comments is because if we remove the commonly used words in a given language, we can focus on more important words instead.

Stemming**.** Next, we have stemming. Stemming is the process of reducing inflected (or sometimes derived) words to their word-stem, base or root-form. For example, a word like “dog” can be used in many cases such as dogs or doggy and all these words will be transformed to its root which is “dog”.

```
stemmer = SnowballStemmer('english')
words = stopwords.words("english")

reviews['cleaned'] = reviews['comments'].apply(lambda x: " " .join([stemmer.stem(i) for
       i in re.sub("[^a-zA-Z]"," ", x).split() if i not in words]).lower())
```

The code above gives us the new cleaned comments without stop words and stemmers.

TF-IDF**.** After splitting the data into train and test datasets, we want to summarize our comments and convert them into numerical vectors. One technique is to pick out the most frequently used terms (high term frequency, hence **TF)**. However, using term frequency alone is not as useful as words like “this”, “a” occur very frequently across all comments. Hence, we will also measure how unique a word is or how infrequent the words occur amongst our comments (**IDF-** inverse document frequency). So the sum of TF and IDF will give us a word that gives a product of how frequent this word is in the document multiplied by how unique the word is. For the purpose of this project, we will be using TfidfVectorizer package in Python.

*All of these data-pre-processing techniques are captured in detail on Python code provided.* Now, we are ready for the next step, which is to use multi-label classification techniques and get some insight to our data.

## 5.0 METHODOLOGY

### 5.1 Evaluation Metrics

Since we are dealing with multi-label classification problem, there are a few different metrics use to evaluate our model. Some methods which comes to mind are micro-averaging & macro-averaging or Hamming-Loss, however, our main concern for this project is to look at the differences in higher rating homes and lower rating homes, we will be using Exact Match Ratio (Subset accuracy) and F1 score as our major metrics.

Exact Match Ratio is one of the stricter metrics, indicating the percentage of samples that have all their labels classified correctly.

$$ExactMatchRatio, MR = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i)$$

One of the disadvantages of this metric is that multi-class classification problems have a chance of being partially correct, but here we ignore that. We will be using a function from scikitlearn which implements this metric called **accuracy_score.**

F1 score, also known as balanced F-score or F-measure can be interpreted as weighted average of precision and recall, where F1 score reaches its best value at 1 and worst at 0. It is extremely useful in the case of multi-class and multi-label classification, and the formula can be referred to as:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

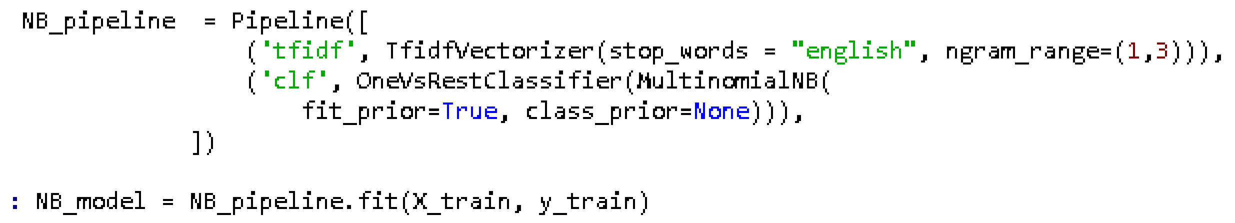
We will be using a function from scikitlearn which implements this metric called **f1_score**. Since there is a label imbalance in our case, we will be tuning the “average” parameter inside f1_score to be “weighted”, meaning it will calculate metrics for each label, and find their average by support (the number of true instances for each label).

### 5.2 Multi-label Classification Techniques

We will be running in total of 3 different models, with a 70/30 train-test split. For all models run, we will use **Pipeline,** which is provided by scikit-learn which is a pipeline utility to help automate machine learning workflows. Pipelines are very common in machine learning systems since there is a lot of data to manipulate and many data transformation to apply. For two of the three models, we will use **OneVsRest** multi-label strategy, which is the multi-label algorithm that accepts binary mask over multiple labels. In this strategy, one could build multiple independent classifiers, and for unseen instance, choose the class for which the confidence is maximized. The main assumption here is that the labels are mutually exclusive, meaning we do not consider any underlying correlation between classes in this method. We will be using Naïve-Bayes and Logistic Regression Classifier for this strategy since both supports multi-class, and both will be wrapped in the OneVsRestClassifier. The third model we will be using will be the LinearSVC model. We will compare each of these models using our stated evaluation metric: Exact Match Ratio, and we will be looking at the VfidfVectorizer to see for each rating, what are the related words associated.

## 6.0 MODEL APPLICATION

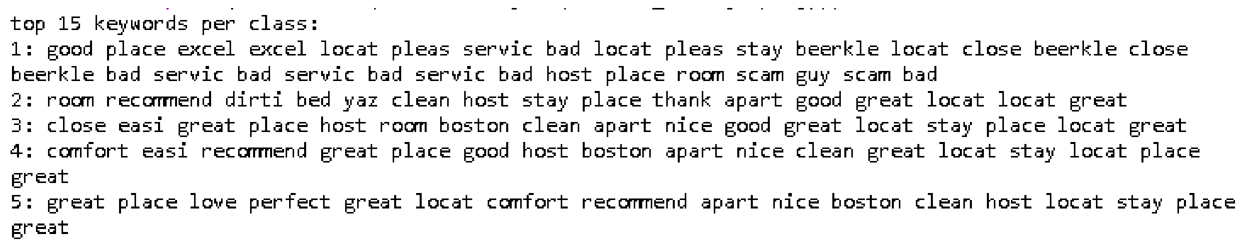
### 6.1 Naïve Bayes Model



```
NB_pipeline  = Pipeline([
                ('tfidf', TfidfVectorizer(stop_words = "english", ngram_range=(1,3))),
                ('clf', OneVsRestClassifier(MultinomialNB(
                    fit_prior=True, class_prior=None))),
            ])

: NB_model = NB_pipeline.fit(X_train, y_train)
```

To make sense of our Pipeline code, we first put in the tfidf argument using the TfidfVectorizer package. Inside this function, we use only English stop-words and ngram_range from 1 to 3. Setting ngram range from 1 to 3 basically means that our vectorizer will look at instances of uni-gram, bi-gram and tri-gram. Essentially what this does is looking at one-word instances, two-word instances and three-word instances. For example, a bi-gram of "not-good" can make a big difference since if we were to use uni-gram, only the word "good" will appear and we want to include the context behind that word. Next, we pass our classifier function, which in this case is the Naïve-Bayes classifier wrapped on the OneVsRest multi-label strategy. We then fit our pipeline and create our model based on our training dataset.

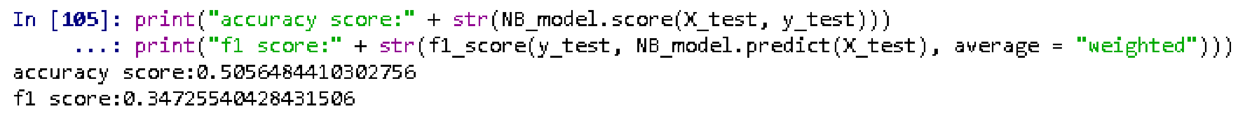
**Top 15 keywords per class**



```
top 15 keywords per class:
1: good place excel excel locat pleas servic bad locat pleas stay beerkle locat close beerkle close
beerkle bad servic bad servic bad servic bad host place room scam guy scam bad
2: room recommend dirti bed yaz clean host stay place thank apart good great locat locat great
3: close easi great place host room boston clean apart nice good great locat stay place locat great
4: comfort easi recommend great place good host boston apart nice clean great locat stay locat place
great
5: great place love perfect great locat comfort recommend apart nice boston clean host locat stay place
great
```

After running our Naïve-Bayes model, we look at the top 15 keywords per class. As we can see here, 1- and 2-star rating class have more negative words such as bad service, terrible, dirty bed, scam, bad, bad host whilst the higher star rating reviews have words such as great place, comfort, recommend, nice, easy etc. This shows us that our model performs really well in training dataset when looking at lower and higher star ratings.

**Accuracy Score and F1_score**

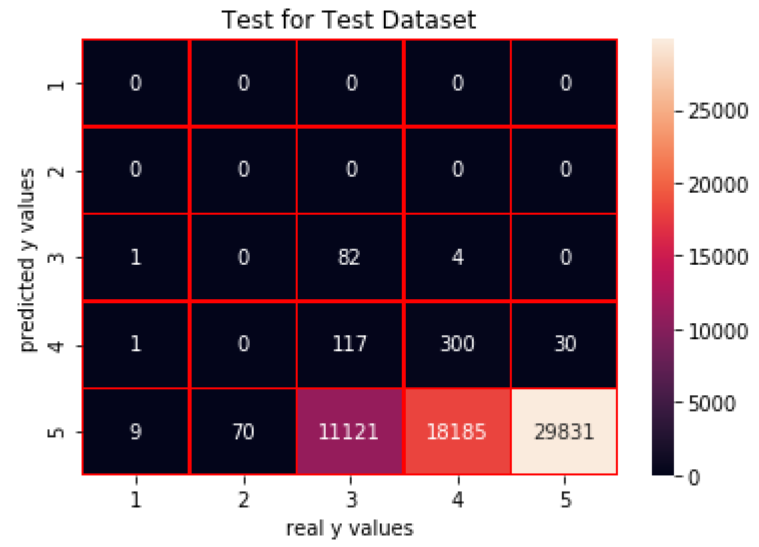


```
In [105]: print("accuracy score:" + str(NB_model.score(X_test, y_test)))
     ...: print("f1 score:" + str(f1_score(y_test, NB_model.predict(X_test), average = "weighted")))
accuracy score:0.5056484410302756
f1 score:0.34725540428431506
```

Looking at the accuracy score for our model against test however, we see quite a low accuracy score of only 50.5%. Our F1 score ranges from 0 to 1, and the reported F1 score for this model is 0.34 which not that good.

**Classification Matrix**

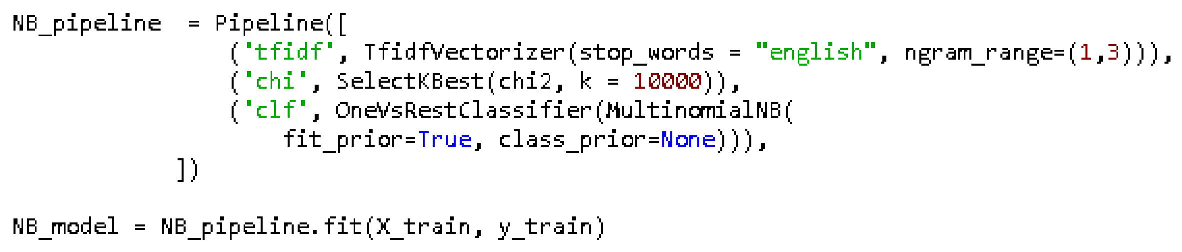
To look closer, we can see the classification matrix which will paint us a better picture of what's going on:



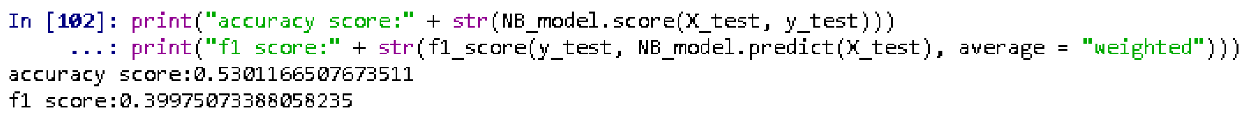
We could see that the model did not perform well in predicting 1- and 2-star ratings as the accuracy rate for these ratings is 0% for both. From our classification matrix, we can see that the model performs not very well in predicting review ratings of 3, 4s , but performs extremely well on predicting 5's. One thing to note here that the model never predicted a value of 1 or 2, this could be the issue of noise level in the data being too high, and the model did not do a good job in identifying the good features for our model.

*Parameter Tuning*

In looking at the best features to train our model, we will be using SelectKBest and will be using chi-squared algorithm to get the best 10,000 features for the model from a million features. Our initial K-threshold will be at 10,000. Adding SelectKBest to our Pipeline and the updated accuracy score can be seen below:



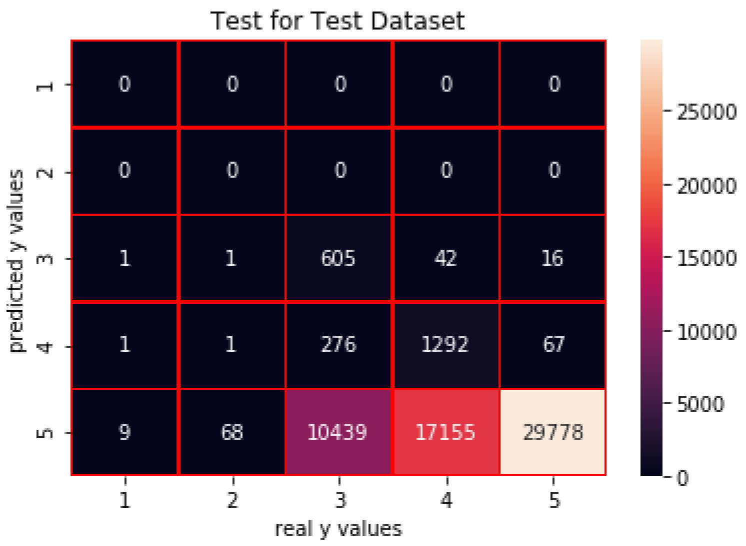
```
NB_pipeline  = Pipeline([
                ('tfidf', TfidfVectorizer(stop_words = "english", ngram_range=(1,3))),
                ('chi', SelectKBest(chi2, k = 10000)),
                ('clf', OneVsRestClassifier(MultinomialNB(
                    fit_prior=True, class_prior=None))),
            ])

NB_model = NB_pipeline.fit(X_train, y_train)
```



```
In [102]: print("accuracy score:" + str(NB_model.score(X_test, y_test)))
     ...: print("f1 score:" + str(f1_score(y_test, NB_model.predict(X_test), average = "weighted")))
accuracy score:0.5301166507673511
f1 score:0.39975073388058235
```

The accuracy score for our model is now higher at 53% from the initial 50.5%. Our F1 score is also higher at 0.399, but still not good.

At this point, we will leave the Naïve Bayes model here for now, since we will be looking at other models to see which one performs better than the others without a lot of parameter tuning, then we will look closer into the model that performs the best. The new classification matrix for our updated model is:



## 6.2 Logistic Regression Classifier

```
LogReg_pipeline = Pipeline([
                ('tfidf', TfidfVectorizer(ngram_range=(1,3), stop_words = "english")),
                ('clf', OneVsRestClassifier(LogisticRegression(solver='sag'), n_jobs=1)),
            ])

LogReg_model = LogReg_pipeline.fit(X_train, y_train)
```

For our logistic regression classifier model, we will be using the same parameters use as our Naïve Bayes model where we will have TfidfVectorizer with ngram_range from 1 to 3 and a OneVsRestClassifier wrapped around the logistic regression model. In addition, since adding SelectKBest is always recommended, we will also use this parameter for our model.

**Top 15 keywords per class**

```
top 15 keywords per class:
1: bad host love room excel good place excel excel locat pleas servic bad locat pleas stay beerkle locat
close beerkle close beerkle bad servic servic bad host bad servic bad scam guy scam bad
2: good locat sleep exact paid great locat roommat yaz great yaz great host build recommend host unit
rodrigo elif ming dirti yaz
3: sean jj jose matthew piter shelley curti paig alan shawn robert steve eddi izzi hermina
4: lucill iren erin adam natalia jordan mikhail anthoni roger kate hasan marcia liza miriam ravi
5: snack muffin gari daniell lori carney bori edward berni mac jason tim jonathan paul lisa
```

Looks like our logistic regression classifier model investigated first names for keywords per class. This is quite interesting since we can see that for 1-star and 2-star ratings, the top keywords are similar to the Naïve Bayes classifier model, where a lot of negative words can be seen. However, we can see that because some host's name has a lot of listings and these listings have high ratings, we can assume that their name is also significant in determining ratings of a listing. Airbnb has a reward system for hosts that have multiple listings and are recognized as good hosts, these hosts are called **Super Hosts** and maybe the keywords from our logistic regression classifier gave us an idea on Super hosts.

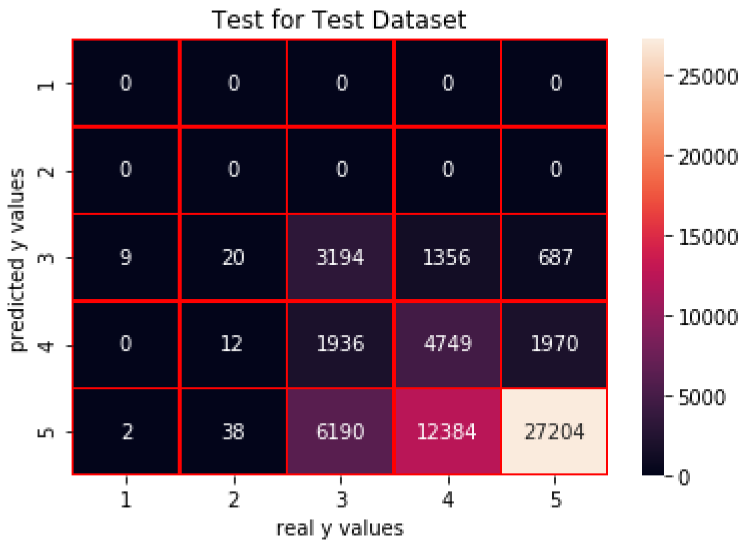
**Accuracy Score and F1 Score**

```
In [108]: print("accuracy score:" + str(LogReg_model.score(X_test, y_test)))
     ...: print("f1 score:" + str(f1_score(y_test, LogReg_model.predict(X_test), average = "weighted")))
accuracy score:0.5882077287409416
f1 score:0.5404613811479441
```

The Logistic model obtained a higher accuracy score than the Naïve Bayes model at 58.8%. Our F1 score is now higher than 0.5, which shows that it there's a good amount of precision and recall in our model.

**Classification Matrix**

To take a closer look, we obtain the classification matrix:



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 9 | 20 | 3194 | 1356 | 687 |
| 4 | 0 | 12 | 1936 | 4749 | 1970 |
| 5 | 2 | 38 | 6190 | 12384 | 27204 |

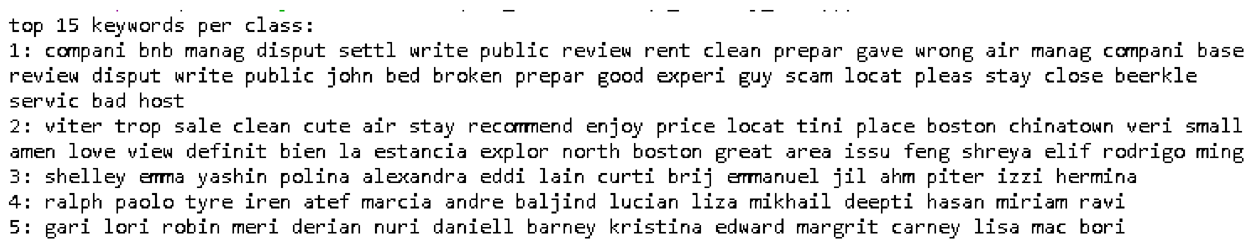
Comparing our first classification matrix under Naïve Bayes, we can say that this model performs worse in predicting ratings of 5 as there are a lot more misclassification rates for rating of 5. However, what this model did better than Naïve Bayes model is in predicting the lower ratings of 3 and 4 where it predicts correctly more instances of 3 and 4 than the Naïve Bayes model. However, this model did not predict well for 1 and 2-star ratings as it obtains a 0% accuracy for low ratings.

## 6.3 Linear SVC model

```
In [46]: pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1,3), stop_words = "english",
sublinear_tf = True, norm ='l2', analyzer ='word')),
    ...:         ('chi', SelectKBest(chi2, k = 10000)),
    ...:         ('clf', LinearSVC(C=1.0, penalty='l1', max_iter = 3000, dual = False))])

In [47]: SVC_model = pipeline.fit(X_train, y_train)
```

Our last model will be the linear SVC model, which proves to work decently better than other models in text classification. The few differences we are making for this linear model, we will be putting the sublinear to be true, meaning that it will use a logarithmic function. Also, we will use the lasso penalty, and according to Coding-Maniac, lasso penalty tends to perform better than ridge algorithm and elasticnet algorithm. The C- parameter for our LinearSVC model for the bias-variance tradeoff, and in fact a regularization factor that controls a high variance. I have had time to play around with this parameter and 1.0 yields the best result.

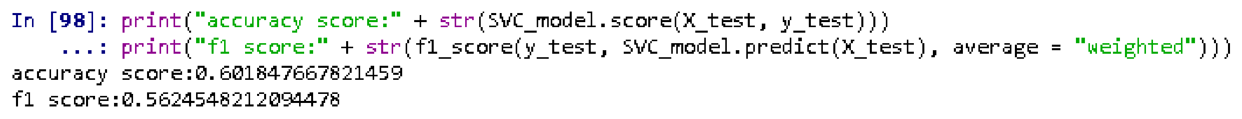
### Top 15 keywords per class



```
top 15 keywords per class:
1: compani bnb manag disput settl write public review rent clean prepar gave wrong air manag compani base
review disput write public john bed broken prepar good experi guy scam locat pleas stay close beerkle
servic bad host
2: viter trop sale clean cute air stay recommend enjoy price locat tini place boston chinatown veri small
amen love view definit bien la estancia explor north boston great area issu feng shreya elif rodrigo ming
3: shelley emma yashin polina alexandra eddi lain curti brij emmanuel jil ahm piter izzi hermina
4: ralph paolo tyre iren atef marcia andre baljind lucian liza mikhail deepti hasan miriam ravi
5: gari lori robin meri derian nuri daniell barney kristina edward margrit carney lisa mac bori
```

As we can see here, the top 15 keywords per class for this model is a bit different than our first two models. The keywords for lower-rating listings are more diverse. We now see "dispute, review, broken, bad service, management" which gives us a better insight than the other two models where only vague negative words are used. Again, we can see here that the higher rating listings have first names as their keywords which gives context to the fact that some hosts have a lot more listings and gets more reviews.

### Accuracy Score and F1 Score

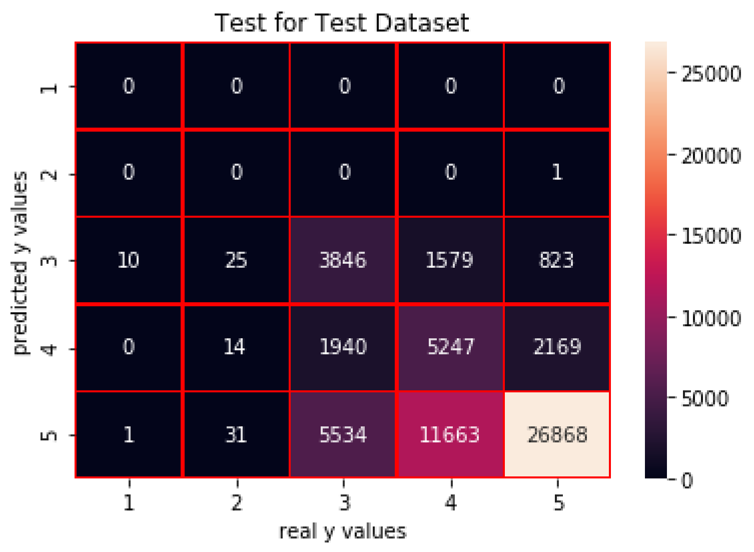


```
In [98]: print("accuracy score:" + str(SVC_model.score(X_test, y_test)))
    ...: print("f1 score:" + str(f1_score(y_test, SVC_model.predict(X_test), average = "weighted")))
accuracy score:0.601847667821459
f1 score:0.5624548212094478
```

The accuracy score for our Linear SVC model is the highest out of the 3 models. We have a 60% accuracy rate and a f1 score of 0.56, which is higher than our previous models.

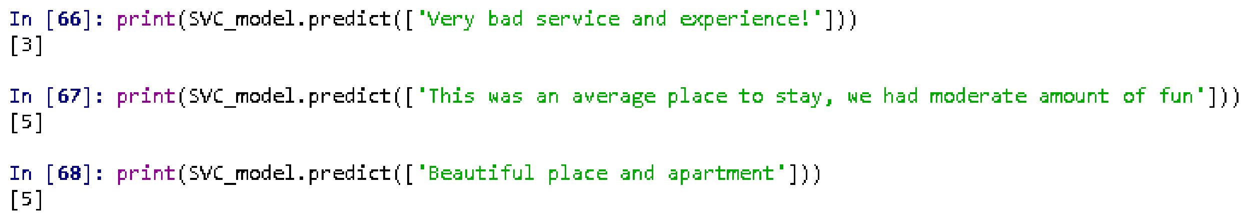
### Classification Matrix

To take a closer look, we will obtain the classification matrix:



Our last model clearly classifies much better instances of lower-review ratings. Out of the 11 review ratings of 1, it classified 10 as review rating of 3, which is a number that is the highest out of the 3 models in predicting a lower-rating review. There are also rate of review ratings for 3, 4 and 5 and overall does a better job than the other two classifying methods.

We can play around our model with 3 different examples of reviews:



```
In [66]: print(SVC_model.predict(['Very bad service and experience!']))
[3]

In [67]: print(SVC_model.predict(['This was an average place to stay, we had moderate amount of fun']))
[5]

In [68]: print(SVC_model.predict(['Beautiful place and apartment']))
[5]
```

We can see that the model isn't perfect since it does not correctly identify a very bad listing, but at least it didn't give a rating of 5 which is better than the previous models. This is largely due to the fact that our dataset had a **class bias** where there's not a lot of 1- or 2-star rating listings.

## 7.0 CONCLUSION

An accuracy table is formed after testing against three different classification models below.

| Model | Class rating accuracy (%) | | | | | Overall accuracy score (%) | F1-Score |
|---|---|---|---|---|---|---|---|
| | 1-star | 2-star | 3-star | 4-star | 5-star | | |
| Naïve Bayes | 0.0 | 0.0 | 5.3 | 7.0 | **99.7** | 53.0 | 0.39 |
| Logistic Regression | 0.0 | 0.0 | 28.2 | 25.7 | 91.1 | 58.8 | 0.54 |
| Linear SVC | 0.0 | 0.0 | **34.0** | **28.4** | 90.0 | **60.2** | **0.56** |

We can derive from our accuracy table the following:

- All 3 models did not perform well in predicting 1-star and 2-star rating homes
- Naïve Bayes model performed the best at predicting 5-star rating homes
- LinearSVC model performed the best at predicting 3-star and 4-star rating homes, and yields the highest overall accuracy rate
- Logistic Regression model performed better than Naïve Bayes at predicting 3-star and 4-star rating homes
- Relatively, both Logistic and LinearSVC predicted lower-rating homes (1-star and 2-star) closer than Naïve Bayes model (refer back to classification matrix)

As our overall question indicates, we are concerned with both low-star rating homes as well as high-star rating homes. Although VfidfVectorizer gave us a clear image of the differences in reviews of each class, our models didn't perform well in predicting lower-star ratings. This is in fact due to the class bias that exists in the dataset. Although the models did not accurately predict 1-star and 2-star ratings, it did however predict those lower-rating homes as 3-stars, which in ordinal terms, is closer than predicting 1-star and 2-star ratings as 5-stars. The LinearSVC model performed best on our training and testing dataset.

From this research, we can say that there exists a clear difference in service level of lower-rating homes and higher-rating homes. Lower-rating homes tend to have less reviews since less people would be likely to go for experiences known to be "bad" or "horrible" meanwhile higher-rating homes will most likely be booked more often. For 1-star and 2-star rating homes, it is suggested that Airbnb should do a quality control check and see what improvements could be made for these types of homes in the future to ensure the quality of all listings on Airbnb in Boston, Massachusetts.

## 8.0 REFERENCES

Airbnb. “Listings.csv.” Airbnb, 9 Feb. 2019. insideairbnb.com/get-the-data.html.

Airbnb. “reviews.csv.” *Inside Airbnb*, 2019, insideairbnb.com/get-the-data.html.

Coding-Maniac, Jo, director. *Machine Learning - Text Classification with Python, Nltk, Scikit & Pandas*. *YouTube*, YouTube, 21 Sept. 2017, www.youtube.com/watch?v=5xDE06RRMFk&t=902s.

Li, Susan. “Multi Label Text Classification with Scikit-Learn.” *Towards Data Science*, Towards Data Science, 21 Apr. 2018, towardsdatascience.com/multi-label-text-classification-with-scikit-learn-30714b7819c5?fbclid=IwAR3388sBxoCNchr1tXvjmM7TsH-m0vYSxq16pbmKO8VGNyp7k0-MnNJowU4.

Nooney, Kartik. “Deep Dive into Multi-Label Classification..! (With Detailed Case Study).” *Towards Data Science*, Towards Data Science, 8 June 2018, towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff?fbclid=IwAR3tZ7F73UFeCfhdBv--9adiFV8LK5jIfb7EXvP9luRlGgyiv40A-ecgK2o.

Prabhakaran, Selva. “Eval(ez_write_tag([[728,90],'r_statistics_co-Box-3','ezslot_3',109,'0']));Logistic Regression.” *Logistic Regression With R*, 2014, r-statistics.co/Logistic-Regression-With-R.html.