

# Boston Airbnb Customer Review Sentiment Analysis Project

DongminZheng

JiangyiFang

MengyuanWang

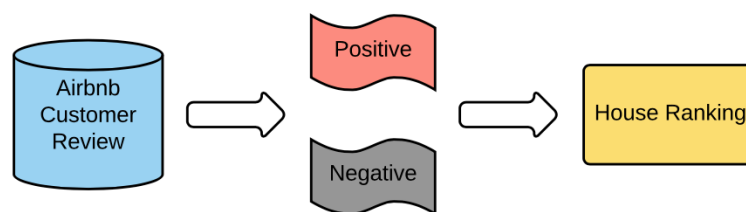
Mentor: Prof. KunpengZhang

# Background

## Research Description

The beauty of big data lies in understanding the customer behavior. Many organizations are harnessing the power of big data through behavioral analytics to deliver big value to businesses. Without exception, Airbnb has set up AIRDNA where they use sophisticated technology to pick up and analyze intricate data points on every Airbnb listing in the world.

Since Airbnb leaves a review place and an overall rating out of 5 stars for their customers, which enables them to immediately response to any positive or negative comments accordingly. And what we do in our project is to analyze customer sentiments and rank the places based on the reviews collection of each Airbnb houses. Listed below are the research steps.



After we get scores for each Airbnb, we will cluster the Airbnb house. Since in our data there are summaries and descriptions of each Airbnb, we cluster them based on the description and summary. In the future, if user want to go to Boston again, we can recommend the Airbnb in the cluster that the Airbnb s/he already lived in and give the good recommend or the cluster that is further to the Airbnb s/he has lived in and give the negative recommendation. The positive or negative of an Airbnb is done by ranking.

We also want to develop a model that can predict how well an Airbnb is based on some features, so that if there is a new Airbnb, we can rank and recommend it without any reviews. The model we use is Linear model, random forest, regression tree. We want to use these three methods to build models that can predict the score of a new Airbnb. After that, we will compare each model to find the best fitted model to implement in our recommendation system.

## Data Source

Our main data set of Airbnb consists of three excel files: calendar, listings and reviews, including housing information, reservation information and customer reviews.

Calendar, including listing id and the price and availability for that day.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	listing_id	date	available	price															
2	5506	9/5/17	t	\$145.00															
3	5506	9/4/17	t	\$145.00															
4	5506	9/3/17	t	\$145.00															
5	5506	9/2/17	t	\$145.00															

Listings, including full descriptions of the Airbnb, like summary, neighbor, house rule, host about and average review score.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	
1	id	listing_url	scrape_id	last_scraped	name	summary	space	description	experiences_offered	neighborhood	overview	notes	transit	access	interaction	house_rules	thumbnail_url	medium_url	picture_url	xl_picture_url	host_id	host_url	host_name	host_since	host_location	host_about
2	12147973	https://www.	2.02e+13	9/7/16	Sunny Bung	Cozy, sunny	The house ?	Cozy, sunny	none	Rosindale is quiet, convenient and	The bus stop	You will have access to 2	Clear up an	https://a2.n	https://a2.n	https://a2.n	https://a2.n	https://a2.n	https://a2.n	https://a2.n	31303940	https://www.Virginia	4/15/15	Boston, Ma	We are coui	N/A
3	3075044	https://www.	2.02e+13	9/7/16	Charming ri	Charming ai	Small but cc	Charming ai	none	The room is in Rosindale	If you don't	Plenty of so	Apt has one if i	am at ho	Pet friendly	https://a1.n	https://a1.n	https://a1.n	https://a1.n	https://a1.n	2572247	https://www.Andrea	6/7/12	Boston, Ma	I live in Bost	withi
4	6976	https://www.	2.02e+13	9/7/16	Mexican Poi	Come stay v	Come stay v	Come stay v	none	The LOCATION: Rosind	I am in a scc	PUBLIC TRA	I am living i	ABOUT ME	encourage	https://a2.n	https://a2.n	https://a2.n	https://a2.n	https://a2.n	36701	https://www.Phil	5/11/09	Boston, Ma	I am a	withi

Reviews, including listing id, reviewer unique id, date and comments for the house.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	listing_id	id	date	reviewer_id	reviewer_name	comments													
2	1178162	4724140	5/21/13	4298113	Olivier	My stay at Islam's place was really cool! Good location, 5min away from subway, then 10min from downtown. The room was nice, all place was clean. Islam managed pretty well our ar													
3	1178162	4869189	5/29/13	6452964	Charlotte	Great location for both airport and city - great amenities in the house: Plus Islam was always very helpful even though he was away													
4	1178162	5003196	6/6/13	6449554	Sebastian	We really enjoyed our stay at Islams house. From the outside the house didn't look so inviting but the inside was very nice! Even though Islam himself was not there everything was pre													

Scores, including dependent variables: scores, and independent variables: host\_identity\_verified, property\_type, room\_type, accommodates, bathrooms, bedrooms, beds, price, guests\_included, extra\_people, minimum\_nights, num of reviews, review score accuracy, cancellation, reviews per monthly, which reflect Airbnb house characters and can be used for score prediction and regression analysis as well.

	host_identity_verified	property_type	room_type	accommodates	bathrooms	bedrooms	beds	price	guests_included	extra_people	minimum_nights	number_of_reviews	review_scores_accuracy	cancellation_policy	reviews_per_month	review_scores_rating
2	2	2	2	2	2	1.0	1	65	0	0	2	36	10	2	1.30	94
3	2	2	2	2	2	1.0	1	65	1	20	3	41	10	2	0.47	98
4	1	10	2	4	1.0	1	2	75	2	25	1	1	10	2	1.00	100
5	2	10	2	2	1.5	1	2	79	1	0	2	29	10	1	2.25	99
6	2	6	2	2	1.0	1	1	75	1	0	2	8	10	1	1.70	100
7	2	2	1	3	1.0	1	2	100	1	25	1	57	10	3	4.00	90
8	2	10	2	2	2.0	1	1	75	1	15	1	67	10	2	2.38	96
9	2	6	2	2	1.0	1	2	58	2	0	2	65	10	2	5.36	96
10	2	2	1	5	1.0	2	2	229	4	25	4	33	10	3	1.01	94

## Methodology

### Sentiment Analysis

Basically, we adopt sentiment analysis to determine the attitude of an Airbnb customer to the house he has lived, whether it is positive or negative.

The target we want to analyze is the customer review of Boston Airbnb. But rather than giving an overall polarity, we extract the word from all reviews as corpus, then classify words as “positive” or “negative”, give an associated number to the sentiment word from -15 to 15 scale, reviews can be given a positive and negative sentiment strength score.

### TF IDF

TF-IDF is used to reflect the importance of a term to a document in the corpus.  $tf-idf(t,d,D) = tf(t,d) \cdot idf(t,D)$ . Term frequency  $tf(t,d)$  is the number of times that term  $t$  appears in document  $d$ , which reflects high marks if a house with large amount reviews however it's unfair for the house with less reviews. Therefore,  $tf$  Frequently occurring words present in all files of corpus irrespective of the sentiment, which will be treated equally like other distinguishing words in the document. And  $tf-idf$  weighting factor can eliminate the limitations while calculating the scores.

## **Kmeans**

Kmeans is a way to cluster the data based on the attributes it has. We prepare the data of summary/description regarding to each Airbnb ID. Then we clean the data and do the TF-IDF of each word in the summary/description. Then, each word in a single document become a vector and each document is described by multiple vectors. Each document is related to a specific Airbnb ID. After the data preparation, we run the kmeans to get the distance between two Airbnb and cluster them based on the distance of Airbnb to each other. The shorter the distance is means they are more similar to each other and the longer the distance is means that they are more different to each other.

## **Linear Regression**

We choose to use identity, property, room, accommodates, bathrooms, bedrooms, beds, price, guests, people, minnights, numreviews, accuracy, cancellation, reviewspermonth as the independent variables to predict the score of a new Airbnb. We divide the data into 70% training and 30% testing to measure the performance of each model. After data preparation, we did the linear model first. The linear model used a straight line to predict the score of Airbnb based on the independent variables.

## **Tree Regression**

After we prepare the data, the second model we run is the regression tree. Regression Tree make all the possible binary divides on all variables and find the best split value of it. The variables that used in the earlier splits means they are more important to predict the result. The final nodes are the average score of each hotel.

## **Random Forest Regression**

After we get our data prepared, the last model we run is the Random Forest. Random

Forest is the method that randomly choose subset of each variables to make the prediction. After all the prediction is done, it uses average of all random subset models to get the final model. It is good method to lower the bias as well as the variance. In addition, it can avoid the influence of outliers. It may be a good method to give a precise prediction.

## Analysis Process

Here we use Java to do the data preparation, review token, TFIDF calculation based on MapReduce algorithm.

### First Stage: data preparation

The first stage in our project is to clean data: Airbnb customer review. Listed below is one example of review, which contains punctuation, emoticon, mixed case and even wrong spelling.

My stay at islam's place was really cool! Good location, 5min away from subway, then 10min from downtown. The room was nice, all place was clean. Islam managed pretty well our arrival, even if it was last minute ;) i do recommand this place to any airbnb user :)

In our project, we use `line.replaceAll()` to remove and convert them as space. Then `line.toLowerCase()` to preserve for word in all lower cases

```
while((line = bufferedReader.readLine()) != null) {  
    line = line.replaceAll("[^\\w\\s-]", " ");  
    line = line.toLowerCase();  
    bufferedWriter.write(line);  
    bufferedWriter.newLine();  
}
```

### Second Stage: tokenization & sentiment word extraction

The second stage depends on MapReduce. In first job, we use WordCount to do review segmentation by each word and count their frequency. Explore what drives consumer sentiment, then we sorted the word from high to low by number of occurrences, and manually create two categories. One is “Positive – enjoy, clean”, the other is “Negative – dirty, noise”.

#### Job1: require word segment from customer reviews

Mapper:

Each mapper processes one line of the review.txt

Input key: filename review.txt

Input value: the content of the line

Map ()

1. Take the review content of a line and convert it into a list of 1-grams

```
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
}
```

Output key: 1-grams

Output value: 1

Reducer:

Input key: 1-grams

Input value: 1

Reduce()

1. Aggregate all values for each 1-grams

```
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

Output key: 1-gram

Output value: the number of 1-gram

Then we list a table containing 70 sentiment word and 6 of them are negative. Here we use all words but nearly 90% of them are adjective. One importance is how do we deal with negation, in the first stage, `don't, can't` such word are changed into `don t, can t`. Therefore, once it occurs `t`, we consider it as a negative word.

After making this distinction, we could build a sentiment scoring system, taking into consideration the following sentiment ratings:

- (++) emotional positive
- (+) rational positive

- neutral
- (-) rational negative
- (--) emotional negative

The result is the sentiment.txt.

### Third Stage: calculate ranking

Job1: aggregate customer reviews for each Airbnb house

Mapper:

Each mapper processes one line of the review.txt

Input key: byte offset

Input value: content of the line

Map ()

1. Take the value and parse the content to obtain listingID and customer review

```
public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
    String text = value.toString();
    String regexp = "\\d\\d\\d\\d\\d+";
    String reviews[] = text.split(regexp);
    Pattern p = Pattern.compile(regexp);
    Matcher m = p.matcher(text);
    int start[] = new int[reviews.length - 1];
    int end[] = new int[reviews.length - 1];
    int i = 0;
    while (m.find()) {
        start[i] = m.start();
        end[i++] = m.end();
    }
    Text rev = new Text();
    for (i = 0; i < reviews.length - 1; i++) {
        String id = text.substring(start[i], end[i]);
        word.set(id);
        rev.set(reviews[i + 1]);
        context.write(word, rev);
    }
}
```

Output key: listingID

Output value: customer review

Reducer:

Input key: listingID

Input value: customer review

Reduce ()

1. Concatenate all values for each Airbnb house by listingID

```

public static class Reduce extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {
        StringBuffer sb = new StringBuffer();
        for (Text val : values) {
            sb.append(val.toString());
        }
        context.write(key, new Text(sb.toString()));
    }
}

```

Output key: listingID

Output value: the aggregate of customer reviews, named as aggreview.txt

Then we filtered out the unrelated words in aggreview.txt and only left the sentiment word for each house review based on sentiment.txt. The outcome is also named as aggreview.txt.

```

while((line = bufferedReader.readLine()) != null) {
    String tmp[] = line.split("\\s");
    for (int i = 0; i < tmp.length; i++) {
        if (tmp[i].matches("\\d\\d\\d\\d\\d+") || kw.contains(tmp[i])) {
            bufferedWriter.write(tmp[i] + " ");
        }
    }
}

```

## Job2: calculate IDF

Mapper:

Each mapper processes one line of the aggreview.txt

Input key: byte offset

Input value: content of the line

Map ()

1. Take each of the word in one house review, once it occurs in sentiment set, we count 1.

```

HashSet<String> s;
for (int i = 0; i < review.length; i++) {
    if (!review[i].equals(" ") && !review[i].equals("")) {
        String keys[] = review[i].trim().split("\\s");
        s = new HashSet<String>();
        for (int j = 0; j < keys.length; j++) {
            s.add(keys[j]);
        }
        for (String kw : s) {
            word.set(kw);
            context.write(word, one);
        }
    }
}
}

```

Output key: sentiment word



Output value: 1

Reducer:

Input key: sentiment word

Input value: 1

Reduce ()

1. Concatenate all values by sentiment word
2. Calculate  $\log \frac{N}{|\{d \in D: t \in d\}|}$  for each sentiment word

```
for (DoubleWritable val : values) {
    sum += val.get();
}
double idf = Math.log10(total / sum);
context.write(key, new DoubleWritable(idf));
```

Output key: sentiment word

Output value: idf value (counts of each sentiment word)

**Job3: calculate TF**

Mapper:

Each mapper processes one line of the aggreview.txt

Input key: byte offset

Input value: the content of the line

Map ()

1. Take each of the word in one house review, we count 1.

```
public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
    String text = value.toString();
    String regexp = "\\d\\d\\d\\d\\d+";
    String reviews[] = text.split(regexp);
    Pattern p = Pattern.compile(regexp);
    Matcher m = p.matcher(text);
    int i = 1;
    while (m.find()) {
        if (!reviews[i].equals(" ") && !reviews[i].equals("")) {
            String keys[] = reviews[i].trim().split("\\s");
            for (int j = 0; j < keys.length; j++) {
                word.set(text.substring(m.start(), m.end()) + "\t" + keys[j]);
                context.write(word, one);
            }
        }
        i++;
    }
}
```

Output key: listingId, sentiment word

Output value: 1

Reducer:

Input key: listingId, sentiment word

Input value: 1

Reduce ()

1. Concatenate all values by sentiment word of each house

```
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {  
    public void reduce(Text key, Iterable<IntWritable> values, Context context)  
        throws IOException, InterruptedException {  
        int sum = 0;  
        for (IntWritable val : values) {  
            sum += val.get();  
        }  
        context.write(key, new IntWritable(sum));  
    }  
}
```

Output key: listingId, sentiment word

Output value: tf value (counts of the sentiment word of each house)

Take the output from Job2 and Job3, calculate  $TF * IDF * \text{sentiment score}$ . The output is  $\langle (\text{listingId}, \text{sentiment word}), \text{tf*idf*sentiment score} \rangle$ . After that, sum the score for each house.

## Forth Stage: Kmeans Clustering

### Data Cleaning

Input data is the listing Id and the paragraph of summary and description.

We read into the spark as a data frame. First column is the ID and second column is the description. After we clean the data to have only the word and remove the stop words such as "A", "the" etc. to only left meaningful words.

### Data Preparation

Then, we run the TFIDF to get the TFIDF of each word regarding to each Airbnb ID. We normalized the TFIDF score to prevent some extreme score influence the measure of distance.

### Run Kmeans

At last we run the Kmeans regarding to each ID which has vectors as normalized TFIDF score of all the words. We divided the data into 10 clusters.

```
wangmengyuans-MacBook-Pro:rr wangmengyuan$ spark-submit --master local[*] --packages com.databricks:spark-csv_2.10:1.2.0 cluster.py
Ivy Default Cache set to: /Users/wangmengyuan/.ivy2/cache
The jars for the packages stored in: /Users/wangmengyuan/.ivy2/jars
:: loading settings :: url = jar:file:/Users/wangmengyuan/spark/lib/spark-assembly-1.6.1-hadoop1.2.1-jar!/org/apache/ivy/core/settings/ivysettings.xml
com.databricks#spark-csv_2.10 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent;1.0
  confs: [default]
  found com.databricks#spark-csv_2.10:1.2.0 in central
  found org.apache.commons#commons-csv;1.1 in central
  found com.univocity#univocity-parsers;1.5.1 in central
:: resolution report :: resolve 180ms :: artifacts dl 6ms
  :: modules in use:
    com.databricks#spark-csv_2.10:1.2.0 from central in [default]
    com.univocity#univocity-parsers;1.5.1 from central in [default]
    org.apache.commons#commons-csv;1.1 from central in [default]
  -----
  | conf | number | search|dwnlded|evicted| | number|dwnlded|
  -----+-----+-----+-----+-----+
  | default | 3 | 0 | 0 | 0 | | 3 | 0 |
  -----
:: retrieving :: org.apache.spark#spark-submit-parent
  confs: [default]
  0 artifacts copied, 3 already retrieved (0KB/6ms)

+-----+-----+
| houseid|prediction|
+-----+-----+
| 12147973| 6|
| 3075044| 8|
| 6976| 8|
| 1436513| 8|
| 7651065| 8|
| 12386020| 8|
| 5706985| 8|
| 2843445| 8|
| 753446| 8|
| 849408| 4|
| 12023024| 1|
| 1668313| 8|
| 2684840| 1|
| 13547301| 9|
| 5434353| 8|
| 225979| 8|
| 3420384| 8|
| 13512930| 1|
| 7482195| 1|
| 7252607| 1|
+-----+-----+
only showing top 20 rows
```

## Fifth Stage: score regression

Here we run three regression methods, 70% training data set and 30% testing data set. First, we use training data set to run three model separately. Then we use testing data set to check the performance based on RMSE and also make the prediction.

Label is the actual review score given by Airbnb customer. Features is the 15 independent variables we use. Prediction is the predicted result of score given the already known selected features.

### Method1: Linear Regression

```
+-----+-----+-----+
|label|          features|          prediction|
+-----+-----+-----+
| 48.0|(15,[0,1,2,3,4,5,...| 49.404626951772144|
| 53.0|(15,[0,1,2,3,4,5,...| 54.24482381073747|
| 60.0|(15,[0,1,2,3,4,5,...| 61.02109941328892|
| 60.0|(15,[0,1,2,3,4,5,...| 61.02109941328892|
| 60.0|(15,[0,1,2,3,5,6,...| 61.02109941328892|
+-----+-----+-----+
only showing top 5 rows
```

Model: Root Mean Squared Error = 0.318974033224

### Method2: Tree Regression

label	features	prediction
85.0	[1.0, 1.0, 2.0, 1.0, ...]	90.80681818181819
81.0	[2.0, 1.0, 1.0, 1.0, ...]	89.12260536398468
100.0	[2.0, 1.0, 1.0, 1.0, ...]	96.0817236255572
86.0	[2.0, 1.0, 2.0, 1.0, ...]	90.80681818181819
90.0	[2.0, 1.0, 2.0, 1.0, ...]	89.12260536398468

only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 7.61484

### Method3: Random Forest Regression

label	features	prediction
93.0	[2.0, 1.0, 2.0, 1.0, ...]	95.13889202848998
78.0	[2.0, 1.0, 2.0, 1.0, ...]	79.22125442738424
96.0	[2.0, 1.0, 2.0, 1.0, ...]	95.27642707752254
100.0	[2.0, 1.0, 2.0, 1.0, ...]	95.52483698386428
93.0	[2.0, 1.0, 2.0, 1.0, ...]	94.26189063737284

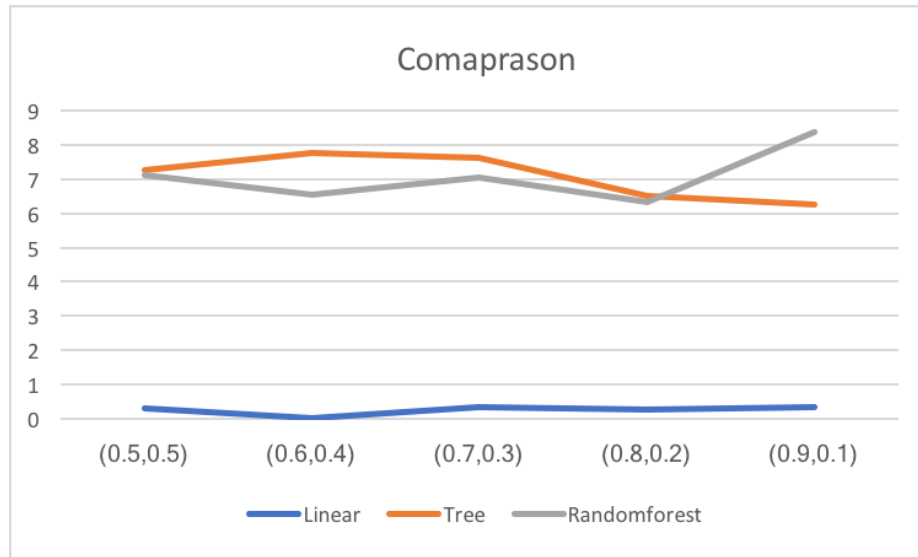
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 7.03731

### Comparison

To compare the performance of each methods. we run three methods under different random splitting. Here we can clearly see Linear regression yields the highest performance, thereby maximizing the accuracy of Airbnb house scores. Actual training sets are not huge. Further, performance speed for prediction is also really fast. Hence, we recommend using linear regression for this problem.

For random forest method, (0.8,0.2) splitting generates the least RMSE for testing data set. For tree, (0.9,0.1) splitting generates the least RMSE for testing data set. However, overfitting can lead to low bias and high variance. Therefore, we can see for random forest method, (0.8,0.2) split has the highest RMSE rather than (0.9,0.1).



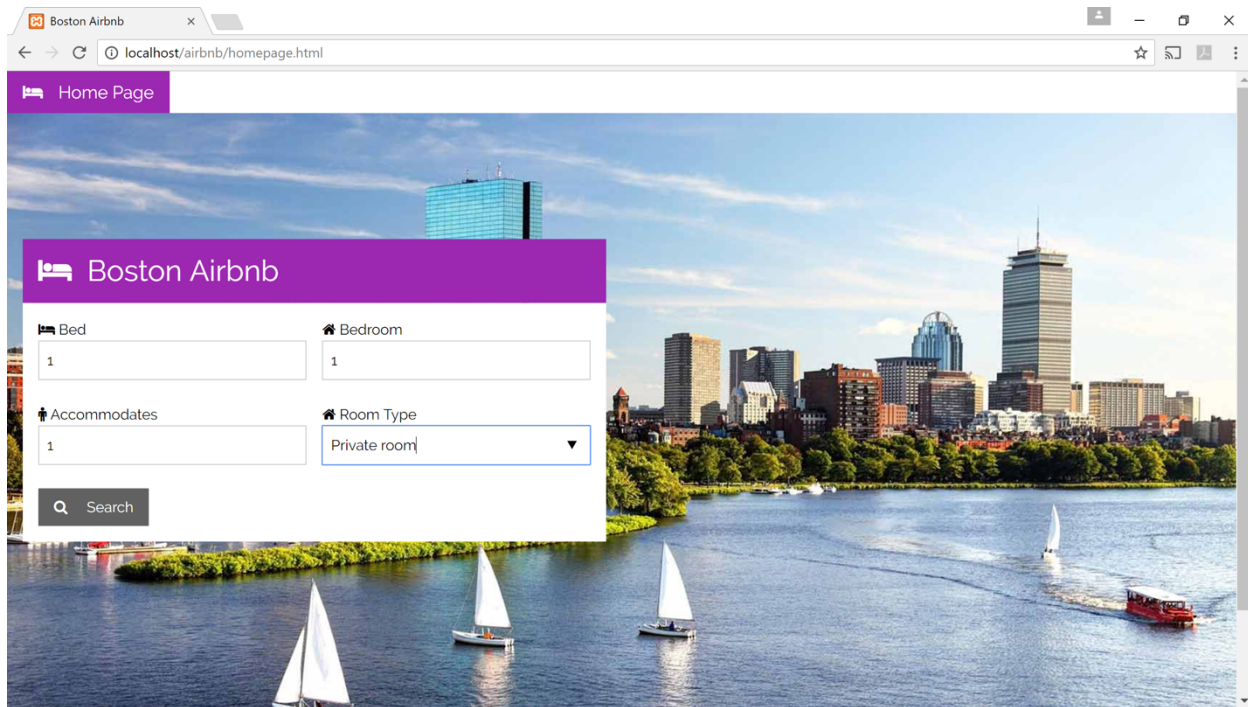
	Linear Method	Tree	Radom Forest
Advantage	Interpretable Easy to implement High Performance	Fast	Fast
Disadvantage	No	Low interpretability (without plot) Lowest Performance ((0.5,0.5) to (0.8,0.2) split)	Low interpretability (without plot) Lower Performance (than linear)

## Results

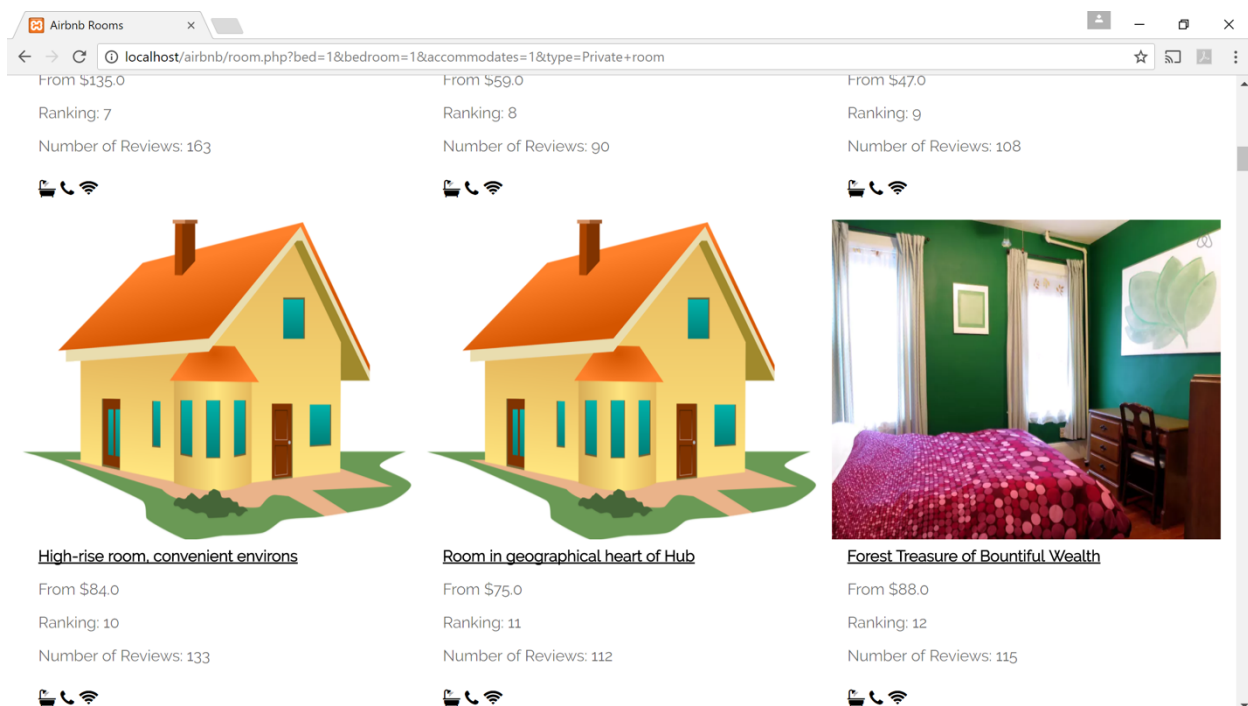
Our Result is shown on the front-end website developed in PHP and back-end database is Mysql.

Database has three tables: listing and ranking. Listing table contains the information about all Airbnb houses. Ranking table is the house id and rating scores. Using join function, third table is the ListRank table based on the common listing id and selected column is house information and its ranking.

Webpage consists of one main page, one listing page and the detailed house page. In the main page, you can choose what kind of Airbnb you are interested by selecting bed, bedroom, accommodates and also room type.



Then you will enter the listing page where shows all filtered results, we have three houses in one line with house name, house price, ranking value and number of reviews.




Then Choose the one you are interested, and more detailed information of the house will be shown, here we also have location map inserted at the lower right corner of the webpage.

Airbnb Rooms


localhost/airbnb/forumdocument.html?rowid=5882411

## Forest Treasure of Bountiful Wealth



Host Name: Derian  
Property Type: Condominium  
Amenities:  
[TV,Internet,"Wireless Internet","Air Conditioning",Kitchen,"Free Parking on Premises",Breakfast,Heating,Washer,Dryer,"Smoke Detector","Carbon Monoxide Detector","First Aid Kit","Fire Extinguisher",Essentials,Shampoo,"24-Hour Check-in",Hangers,"Hair Dryer",Iron,"Laptop Friendly Workspace"]  
From \$88.0  
Address:  
Tower Street, Boston, MA 02130, United States

**Description**  
Urban oasis on a quiet street in ring of Boston's "Emerald Necklace" featuring a sparkly clean bathroom! Just steps to Forest Hills Orange Line Subway T, you can be anywhere in Boston in no time.



Map Satellite

Type here to search

1:33 PM 5/15/2017

We also divided the Boston Airbnb into 10 clusters. Most of the Airbnb fell into 3 clusters which is 1, 8, and 9 with other 7 small clusters have few Airbnb in it. It seems like Airbnb in larger clusters have longer summary/descriptions and the wording is more descriptive such as “cozy”, “beautiful”. While small clusters’ summary/descriptions are more rigid, only says number of amenities Airbnb has without showing any subjective judgement. It is a good result since we have more chances to recommend the Airbnb with longer and more descriptive summary/descriptions, which are what we are normally choose to live.

## Improvement

### Data Preparation

There are lot of things to do for data preparation. First, we can remove the stop word from reviews and do words stemming to improve the effectiveness of retrieval and text mining. And the sentiment regarding to a particular thing in a review is seldom explicitly positive or negative. Rather people tend to have a mixed opinion. For example, they may compare two house in the review. Also, it is necessary to do the spelling check. Unlike many reviewing sites whose users are professional or well-known results in some grammatical errors in the reviews. In addition, some of the reviews are written by

different languages. Therefore, for more accuracy, we can figure out them in the future.

### **Sentiment Word Corpus**

Now we extract the sentiment words from our existing reviews, which is a good sentiment word corpus. And we can also use the word bags related to houses comment from internet to compare the two results and performance.

### **Dynamic Analysis System**

Now our system is a house ranking website supported by the static database. But in the future, the system can be improved to be capable of detecting and retrieving house reviews on the Airbnb automatically, and extract sentiment words, rank them. In addition to that, we can also do the similar analysis of other regions in USA and compare with each other.

## **Conclusion**

We presented a web based sentiment website for Airbnb reviews and user comments in Boston that supports the individual Airbnb management to monitor based on the published ranking on the web about their houses and also new customer to select the top ranked houses they are interested. As a result, we have shown that, despite some issues, our system and website provides a good UI and performance for the Boston Airbnb sentiment analysis.

In addition, we cluster the Airbnb to have a better recommendation for second-time users based on their previous experience. It also shows some attributes of Airbnb. A better described Airbnb trends to cluster together which has a better chance been recommend to users.