Portfolio

Ruiwei Zhang

NO. 4869644301

Syracuse University, School of Information Studies
M.S. Applied Data Science

Content

INTRODUCTION

The Applied Data Science program at Syracuse University's School of Information Studies provides students the opportunity to collect, manage, analyze, and develop insights using data from a multitude of domains using various tools and techniques. In courses such as Database Administration (IST 659, 2019), Introduction to Data Science (IST 687, 2019), Natural Language Processing (IST 664, 2020), and Big Data Analytics (IST 718, 2018), reports and presentations were developed to deliver insights using Microsoft Access, SQL Server Management Studio, Python, R and Excel. The skills developed at the School of Information Studies furnish data scientists focused in the field of marketing analytics with the ability to generate value within their organizations and produce actionable recommendations.

The Applied Data Science Program has seven learning objectives which were exemplified by the applications in this portfolio:

1. Describe a broad overview of the major practice areas in data science.
2. Collect and organize data.
3. Identify patterns in data via visualization, statistical analysis, and data mining.
4. Develop alternative strategies based on the data.
5. Develop a plan of action to implement the business decisions derived from the analyses.
6. Demonstrate communication skills regarding data and its analysis for relevant professionals in their organization.
7. Synthesize the ethical dimensions of data science practice.

IST-659: IBRANCH NETWORK MANAGEMENT DATABASE

Project Summary

The Project mainly focus on designing a database system for iBranch, which is a student organization in the School of Information Studies at Syracuse university. iBranch aims to provide academic and career support and advice to Chinese graduate students, and to help Chinese students adapt to diversified cultural environment in Syracuse university. iBranch operated by students who are interviewed and have been selected into organization. Student members mainly participate in organizing weekly events, academic, and professional information on social media, and provide face to face Q&A for students who have questions about academic, career and life.

The Event's organizer is in charge of scheduling site, records attendant students, which sometimes nearly 50+ people, and also needs to records speaker's information. Those processes are usually done by hand, which can take hours to record and organize. Organizer usually need to check emails to record speakers and alumni's schedule and information. those also takes a lot of effort for summary past work in every internal work summary.

Our proposed system is a method to automate the process of scheduling and reduce the manual work effort for recording attendance. Especially every events site has maximum accommodation, so that it is troublesome for keep calculate current attendant students and reminds how many seats we left. In our system, we can automatically provide current attendant people to organizer, and send a message to potential students to participate in activities, which remind them to sign up to participate form as soon as possible. Also, it is way easier for generate summary report for organization manager, such as total number of attendants this semester, and average attendance

rate every month, rather than calculate one by one by hand.

Entity and attributes

1. Student

The information of current students in ischool

Entity: Student	Attributes	Field Type	Null able	Foreign Key Constraints	Domain Constraints	Description
Primary key	StudentID	VARCHAR(20)	Not Null			The unique ID for every student.
	SFName	VARCHAR(20)	Not Null			The student's first name.
	SLName	VARCHAR(20)	Not Null			The student's last name.
	SExpected Grad	DATETIME	Not Null			The student's expected graduation date.
	SPhone	VARCHAR(20)	Not Null			The student's phone number
	SisiBranch	VARCHAR(5)	Not Null		'Yes' or 'No'	A tag on whether the student is a member in iBranch

2. Alumni

The information of an alumni

Entity: Alumni	Attributes	Field Type	Null able	Foreign Key Constraints	Domain Constraints	Description
Primary key	AlumniID	VARCHAR(20)	Not Null			The unique ID for every alumni.
	AFName	VARCHAR(20)	Not Null			The alumni's first name
	ALName	VARCHAR(20)	Not Null			The alumni's last name
	AGrad	DATETIME	Not			The alumni's

			Null			graduation date
	ACompany	VARCHAR(20)				The current working company of the alumni. Not required

3. Activity

The activity information

Entity: Activity	Attributes	Field Type	Nullable	Foreign Key Constraints	Domain Constraints	Description
Primary key	ActivityID	VARCHAR(20)	Not Null			The unique ID for every activity.
	AcName	VARCHAR(20)	Not Null			The activity's name
	AcDate	DATETIME	Not Null			The planned date for the activity
	AcType	VARCHAR(20)	Not Null		'academic', 'career' or 'life'	The activity's type
	AcCapacity	INT	Not Null			The capacity of the activity
	AcSemester	VARCHAR(20)	Not Null			The activity semester.

4. Registration

The registration information between a student and an activity

Entity: Registration	Attributes	Field Type	Nullable	Foreign Key Constraints	Domain Constraints	Description
Primary key	RegistrationID	VARCHAR(20)	Not Null			The unique ID for every registration.
Foreign key	ActivityID	VARCHAR(20)	Not Null	REFERENCES Activity(Activity ID)		The corresponding activity ID

Foreign key	StudentID	VARCHAR(20)	Not Null	REFERENCES Student(StudentID)		The corresponding student ID
	RDate	DATETIME	Not Null			The date for a student's registration on an activity
	RSemester	VARCHAR(20)	Not Null			The registration semester.

5. Organization

The relationship between a student organizer and an activity

Entity: Organization	Attributes	Field Type	Null able	Foreign Key Constraints	Domain Constraints	Description
Primary key Foreign key	ActivityID	VARCHAR(20)	Not Null	REFERENCES Activity(ActivityID); ON DELETE CASCADE		The corresponding activity ID
Primary key Foreign key	StudentID	VARCHAR(20)	Not Null	REFERENCES Student(StudentID); ON DELETE CASCADE		The corresponding student ID

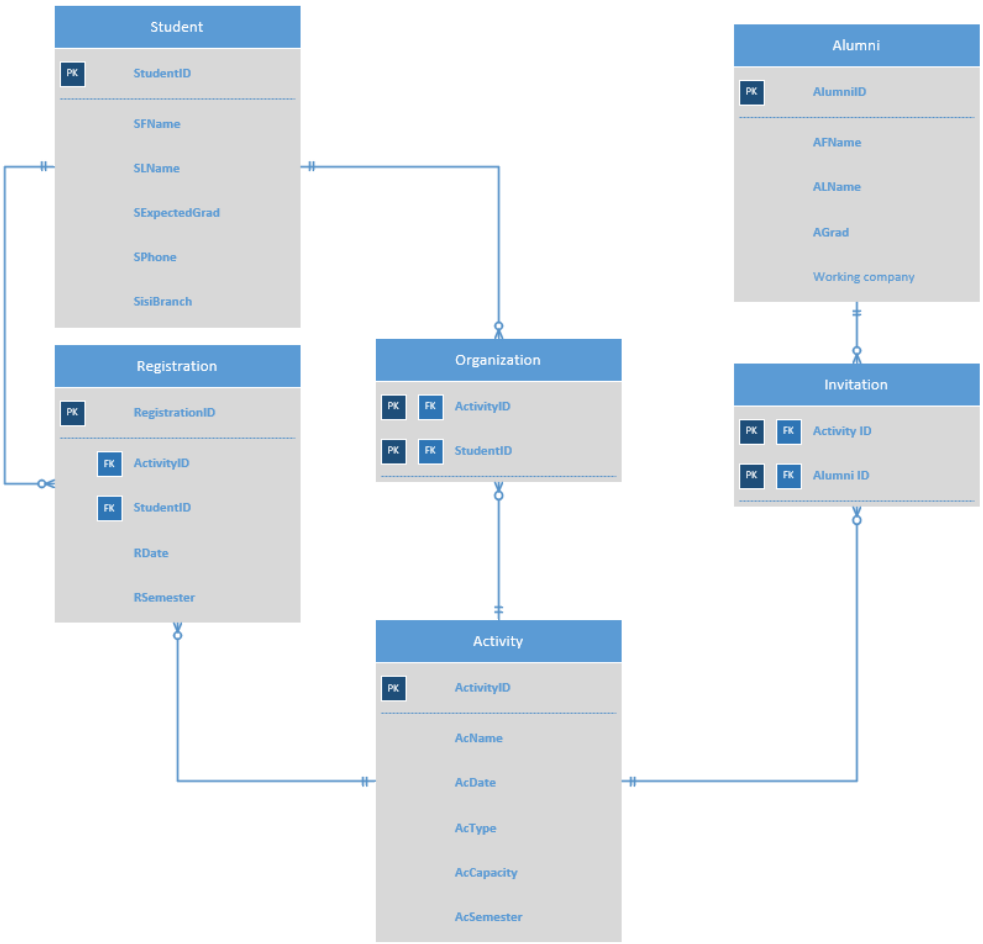
6. Invitation

The relationship between an alumni and an activity

Entity: Invitation	Attributes	Field Type	Null able	Foreign Key Constraints	Domain Constraints	Description
Primary key Foreign key	ActivityID	VARCHAR(20)	Not Null	REFERENCES Activity(ActivityID); ON DELETE CASCADE		The corresponding activity ID
Primary key Foreign key	AlumniID	VARCHAR(20)	Not Null	REFERENCES Alumni(AlumniID);		The corresponding alumni ID

				ON DELETE CASCADE		

Entity Relationship Diagram



Database System Architecture

We used the following tools to create and implement this project:

1. MS Visio: We created entity relationship diagram using MS Visio. It shows the complete structure and the relationship of our database.

2. SQL Server: We used SQL Server as the database that stored all the tables and their data. We use several queries to create, insert and select the correlated data set and data.

3. MS Access: We used MS Access to create the interface for the system. Using Access, we link our tables that were created in SQL Server. Once, the tables were linked, we created forms that could take user input or display the necessary information to the users. Based on the data, we used MS Access to generate reports for the users of the system.

IST-687 PROJECT

Introduction and descriptive statistics

How do we get our Customer back? Southeast Airline

Important takeaways

· We are losing our customers.

· Current Loyalty Program does not help us retain our customers

· Bad partner airlines - Northwest & FlyFast – considering to stop partnering with them or try to find some way to deal with the low NPS scores from customers when flying with them.

· Low NPS for the flight from the north – Reasonable since our data period is from January to March, but we should still find some way to improve it such as provide more help to the customer when their flights are canceled.

Current Problems: Customers are leaving us. Are we using the wrong way to retain our customers? What can we do to keep them?

What we are currently doing: Southeast believes that the best way to keep their customer is the loyalty programs, which is pretty common in the airline industry. Basically, customers get reward points when they take flight with Southeast. Southeast use Net Promoter Score (NPS), on a scale 1-10, to see how good they are doing by asking a single and simple question: "How likely is it that you will recommend our airline to a friend or colleague?".

Problem? We are still losing customers. Customer churn is a lagging indicator. Customer churning is happening so we need to take action as soon as possible to keep our customers. We need to find the right leading indicators to help us to provide the right services that the customers need in order to keep them flying with Southeast.

Data and Method: More than 10,000 survey data were collected from our customers. The time frame of the data is from January to March. After cleaning all the data, we show the descriptive statistic of the data, providing a rough idea of data structure. Next, we use three different models to find the relation between variables. We provided some graphs to help readers to understand some major

factors that affect the NPS score. The results should provide some insights for the management to make the right decision to reduce the attrition rate.

What we find:

1. Loyalty program does not work: The results from our models suggest that NPS is not significantly associated with a loyalty program, suggesting that what southeast is doing is not really helpful to retain our customers.

2. Poor NPS Score related to partner airlines: Customers tend to give a lower NPS score when flying with FlyFast and Northwest, two of our partner airlines. Our linear models show that when customers flying with them, the NPS score is lower and the result is significant statistically.

3. Poor NPS Score related to the geographic area: NPS scores tend to be lower for the flight from northern states. This result is reasonable since our sample period is from January to March.

4. Our results from the Apriori model suggest that Southeast is doing good when the passenger holds Silver airline status, is a male passenger, not price-sensitive, flying with the flight that is on-time and not cancelled, and flying with partner DL. This result is pretty similar to the result we got from the linear regression model.

Suggestions:

Based on our analysis, we provide the following suggestions:

1. Less emphasis on Loyalty Program: our result shows that loyalty is not significantly related to customer satisfaction. Southeast should spend less effort on the loyalty program.

2. Stop partnering with FlyFast and Northwest: Partnering with FlyFast and Northwest seems to make our customers unhappy. Southwest should consider stop partnering with them or at least find some ways to deal with the low satisfaction with those partner airlines.

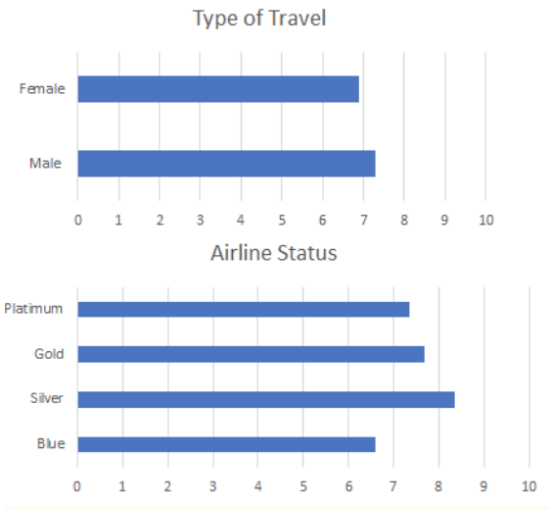
3. Provide more help in the northern region to the customers – Customers are not satisfied with our service in the northern region in our sample. Although it is reasonable since our sample period is in the winter, Southeast can provide some help to the unsatisfied customers, such as free meals or transportation when the flight is delayed or canceled.

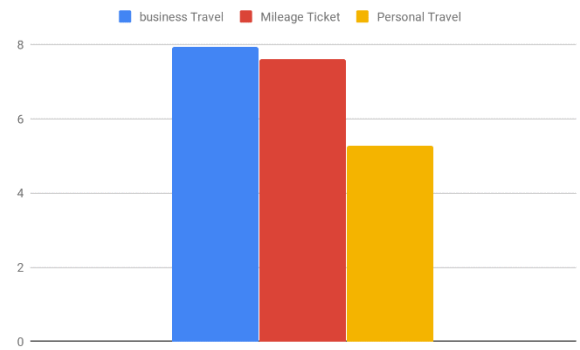
Descriptive statistics:

We got our sample from the survey from our customer. The sample period is from January to March. We have 10,282 observations in our sample. The following figures are some important statistics in our sample.

	Min	Q1	Median	Mean	Q3	Max
Age	15	33	45	46.32	59	85
Price Sensitivity	0	1	1	1.279	2	4
Loyalty	-0.9762	-0.7037	-0.4410	-0.2766	0.0588	1
Total Freq Flyer Accts	0	0	0	0.8928	2	10
Likelihood to Recommend	1	6	8	7.073	9	10

From the above figure, we find that the mean age of our customers is 46.32. The average price sensitivity is 1.279, which suggests that our customers are not price-sensitive. The mean loyalty is negative, providing some insight that customers may not be as loyal as we think. Customers only hold 0.8928 frequent flyer accounts, suggesting that most of the customers do not really care about the loyalty program. Lastly, the average likelihood to recommend our service is 7.073, which seems to be not too bad. The following bar charts give us more details about our data. We are doing good when providing services to male customers. Interestingly, customers with Silver status are pretty satisfied with our service, while Blue status customers are not satisfied. The graph on the right shows that people who fly for their personal travel is pretty angry with our service. We should try to see whether that is really an important factor in our modeling analysis.





Data cleanse

a. NA cleaning

In the data cleaning part, the NAs should be repaired first.

It is found that the NAs only in the following columns: arrive delay, depart delay, flight time and the free text.

Free text has 10000 NAs while the other three have about 200 NAs. The NAs in arrive delay, depart delay and flight time are all caused by the flight cancelled. Hence, it could be obvious that if the flight is cancelled, NAs should be created in these three columns. After discovering this feature and considering the requirement of our models, these NAs are modified with the 0 value.

For the free text attribute, since it is useful only for the text mining, it is taken apart and will be used for the text mining separately.

b. Data split

To ensure the accuracy of our models,the whole dataset is split into two parts. The ratio between train dataset and test dataset is 2:1. This ratio is a priori number and it is widely accepted.

c. Data transform

For different models, the requirement for data type is distinct. Hence, we need to transform the data type according to the model requirement. For example, in linear model, the data type should be numeric while in apriori model, it can be character.

Modeling technique

a. Linear model

The first step is to decide which parameters are required for the linear model. The likelihood to recommend should be the result of our linear model. After some analysis, the following parameters are selected as our factors of the model.

Destination city
Origin City
Airline Status
Type of Travel
Partner code
Gender
Class
Flight cancelled
Age
Price sensitivity

Year of first flight
Flight per year
Loyalty
Total Freq Flyer Accts
Shopping Amounts at airport
Eating and drinking at airports
Departure delay in minutes
Arrival delay in minutes
Flight time

For those numeric parameters, no changes need to be applied, while for those character parameters,they should be transformed into dummy parameters, which means add a column to decide whether the observation has this parameter or not.

In addition to the original data type, we want to dig deeper in the dataset. Hence, we consider season depending on the month column and the daytime depending on the scheduled departure hour. It is found that all the cases are in winter. Hence, it may have some missing

After turning all the data into numeric, the linear model could be applied to find some features of the flights. Those features can be useful to check our linear model fitness.

The result is shown below.

```
Residuals:
    Min      1Q  Median      3Q     Max
-7.2406 -1.0196  0.2127  1.1897  5.3170

Coefficients: (15 not defined because of singularities)
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                       -1.503e+01  1.431e+01  -1.050 0.293822
Airline.Status1                    1.366e+00  5.419e-02  25.210  < 2e-16 ***
Airline.Status2                    7.184e-01  7.751e-02   9.268  < 2e-16 ***
Airline.Status3                    3.021e-02  1.171e-01   0.258 0.796490
Age                               -6.442e-03  1.482e-03  -4.348 1.40e-05 ***
Gender1                           -1.525e-01  4.383e-02  -3.479 0.000507 ***
Price.Sensitivity                 -1.110e-01  3.920e-02  -2.832 0.004637 **
Year.of.First.Flight               1.168e-02  7.082e-03   1.650 0.099038 .
Flights.Per.Year                  -8.156e-03  2.216e-03  -3.681 0.000234 ***
Loyalty                           -2.799e-02  5.953e-02  -0.470 0.638262
Type.of.Travel1                   -1.486e-01  8.084e-02  -1.838 0.066050 .
Type.of.Travel2                   -2.374e+00  5.272e-02 -45.025  < 2e-16 ***
Total.Freq.Flyer.Accts            -2.323e-02  2.139e-02  -1.086 0.277347
Shopping.Amount.at.Airport         1.479e-03  3.945e-04   3.749 0.000179 ***
Eating.and.Drinking.at.Airport     2.705e-03  4.035e-04   6.705 2.18e-11 ***
Class1                             7.598e-03  6.924e-02   0.110 0.912624
Class2                             2.243e-01  7.876e-02   2.848 0.004409 **
Partner.CodeAS                    -7.194e-02  1.841e-01  -0.391 0.696044
Partner.CodeB6                    -2.435e-01  1.505e-01  -1.618 0.105785
Partner.CodeDL                    -2.504e-01  1.156e-01  -2.165 0.030385 *
Partner.CodeEV                    -7.209e-01  1.287e-01  -5.600 2.23e-08 ***
Partner.CodeF9                    -2.156e-01  1.989e-01  -1.084 0.278558
Partner.CodeFL                    -1.812e-01  1.716e-01  -1.056 0.290968
Partner.CodeHA                     3.534e-01  7.419e-01   0.476 0.633776
Partner.CodeMQ                    -2.271e-01  1.431e-01  -1.587 0.112579
Partner.CodeOO                    -2.057e+00  1.277e-01 -16.106  < 2e-16 ***
Partner.CodeOU                    -1.495e-01  1.207e-01  -1.238 0.215631
Partner.CodeUS                    -2.703e-01  1.294e-01  -2.089 0.036750 *
Partner.CodeVX                     4.556e-02  2.201e-01   0.207 0.836009
Partner.CodeWN                    -2.159e-01  1.133e-01  -1.906 0.056743 .
Departure.Delay.in.Minutes         3.017e-03  1.948e-03   1.549 0.121413
Arrival.Delay.in.Minutes          -9.157e-03  1.921e-03  -4.767 1.91e-06 ***
Flight.cancelled                  -5.260e-01  1.902e-01  -2.766 0.005697 **
Flight.time.in.minutes            -3.433e-03  1.265e-03  -2.714 0.006674 **
Flight.Distance                    3.599e-04  1.534e-04   2.346 0.019026 *
Day                                2.662e-02  5.156e-02   0.516 0.605655
`Origin.City_Sacramento, CA`      -7.726e-02  1.735e+00  -0.045 0.964475
`Origin.City_Salt Lake City, UT`   4.023e-01  1.721e+00   0.234 0.815146
`Origin.City_Seattle, WA`          2.366e-01  1.722e+00   0.137 0.890759
`Origin.City_New York, NY`         2.600e-01  1.720e+00   0.151 0.879864
`Origin.City_Boston, MA`           5.786e-01  1.723e+00   0.336 0.737041
`Origin.City_Atlanta, GA`          3.201e-01  1.719e+00   0.186 0.852252
`Origin.City_Crescent City, CA`   -1.621e+00  2.100e+00  -0.772 0.440019
`Origin.City_Cleveland, OH`        2.897e-02  1.733e+00   0.017 0.986661
`Origin.City_San Diego, CA`        2.553e-01  1.723e+00   0.148 0.882238
`Origin.City_Washington, DC`       4.859e-01  1.721e+00   0.282 0.777659
`Origin.City_Detroit, MI`          3.810e-01  1.722e+00   0.221 0.824925
`Origin.City_San Francisco, CA`    2.587e-01  1.720e+00   0.150 0.880432
`Origin.City_Denver, CO`           1.177e-01  1.719e+00   0.068 0.945392
`Origin.City_Charlotte, NC`        5.725e-01  1.723e+00   0.332 0.739771
`Origin.City_Phoenix, AZ`          2.856e-01  1.720e+00   0.166 0.868070
```

```
`Origin.City_Huntsville, AL`           4.966e-01  1.822e+00   0.273 0.785151
`Origin.City_Monroe, LA`               2.133e+00  1.982e+00   1.076 0.281960
`Origin.City_Bozeman, MT`             -1.315e+00  1.852e+00  -0.710 0.477833
`Origin.City_Lafayette, LA`            5.295e-01  1.822e+00   0.291 0.771383
`Origin.City_Spokane, WA`             -5.063e-01  1.766e+00  -0.287 0.774417
`Origin.City_Savannah, GA`             5.106e-01  1.810e+00   0.282 0.777902
`Origin.City_Nome, AK`                 8.398e-01  1.926e+00   0.436 0.662859
`Origin.City_La Crosse, WI`            4.044e-01  1.982e+00   0.204 0.838374
`Origin.City_Billings, MT`            -1.339e-01  1.879e+00  -0.071 0.943191
`Origin.City_Valdosta, GA`             3.476e+00  2.427e+00   1.432 0.152203
`Origin.City_Aguadilla, PR`           -3.085e-01  2.430e+00  -0.127 0.898983
`Origin.City_Trenton, NJ`              1.062e+00  1.926e+00   0.551 0.581538
`Origin.City_Fayetteville, AR`         4.593e-02  1.788e+00   0.026 0.979505
`Origin.City_Dickinson, ND`           -2.170e-01  2.424e+00  -0.090 0.928672
`Origin.City_Appleton, WI`            -3.276e-01  1.854e+00  -0.177 0.859725
`Origin.City_Casper, WY`              -1.403e+00  1.834e+00  -0.765 0.444128
`Origin.City_Modesto, CA`              1.712e+00  1.981e+00   0.864 0.387432
`Origin.City_Muskegon, MI`             1.971e+00  2.424e+00   0.813 0.416323
`Origin.City_Montgomery, AL`          -7.815e-01  2.102e+00  -0.372 0.710111
`Origin.City_Laredo, TX`               1.102e+00  2.102e+00   0.524 0.600050
`Origin.City_Missoula, MT`             8.426e-01  2.100e+00   0.401 0.688277
`Origin.City_Evansville, IN`          -1.269e-01  1.985e+00  -0.064 0.949016
`Origin.City_Pocatello, ID`           -8.867e-01  2.425e+00  -0.366 0.714605
`Origin.City_Rapid City, SD`           2.426e-01  1.918e+00   0.127 0.899338
`Origin.City_Bellingham, WA`           1.634e+00  1.988e+00   0.822 0.411121
`Origin.City_Barrow, AK`              -2.502e-01  2.106e+00  -0.119 0.905432
`Origin.City_Tallahassee, FL`          6.523e-01  1.821e+00   0.358 0.720165
`Origin.City_Santa Maria, CA`          2.167e+00  2.100e+00   1.032 0.302245
`Origin.City_State College, PA`        9.050e-01  2.432e+00   0.372 0.709832
`Origin.City_Aberdeen, SD`            -2.532e-01  1.981e+00  -0.128 0.898292
`Origin.City_Shreveport, LA`           1.023e-01  1.855e+00   0.055 0.956045
`Origin.City_South Bend, IN`           1.253e+00  1.919e+00   0.653 0.513899
`Origin.City_Portland, ME`             5.941e-01  1.920e+00   0.309 0.757038
`Origin.City_Kalamazoo, MI`           -2.891e-01  1.986e+00  -0.146 0.884236
`Origin.City_Twin Falls, ID`           2.204e+00  2.426e+00   0.908 0.363847
`Origin.City_Toledo, OH`               5.842e-01  2.107e+00   0.277 0.781599
`Origin.City_Burlington, VT`           6.553e-01  2.106e+00   0.311 0.755685
`Origin.City_Chico, CA`               -1.175e+00  2.426e+00  -0.484 0.628114
`Origin.City_Duluth, MN`              -4.607e+00  2.424e+00  -1.900 0.057438 .
`Origin.City_Traverse City, MI`        1.143e+00  2.104e+00   0.543 0.586917
`Origin.City_Amarillo, TX`             1.602e-01  1.881e+00   0.085 0.932134
`Origin.City_Hibbing, MN`              2.013e+00  2.425e+00   0.830 0.406506
`Origin.City_International Falls, MN`  2.423e-01  1.980e+00   0.122 0.902606
`Origin.City_Cody, WY`                -1.404e+00  2.424e+00  -0.579 0.562540
`Origin.City_Myrtle Beach, SC`         6.545e-01  2.101e+00   0.311 0.755450
`Origin.City_Minot, ND`               -1.218e-01  1.980e+00  -0.062 0.950943
`Origin.City_Paducah, KY`              1.499e+00  2.424e+00   0.618 0.536426
 [ reached getOption("max.print") -- omitted 45 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.713 on 6626 degrees of freedom
Multiple R-squared:  0.4693,    Adjusted R-squared:  0.4509
F-statistic: 25.58 on 229 and 6626 DF,  p-value: < 2.2e-16
```

In the practical modeling, we find that the after throwing the destination city, our model fits much better. This means that customers are not so sensitive to the destination compared with other attributes.

From the graph, we can see that the r-square value is about 0.45, which is a reasonable value for so many attributes. The p-value is almost zero. It can be a good sign for our model fitness.

The most significant attributes are airline status, age, gender, flights per year, type of travel, shopping amount at airports, eating and drinking at airports, arrive delay and two partners.

These factors are the most important ones that have an actual effect on the likelihood to recommend. However, the Loyalty factor is not significant at all. Hence, the company should modify their

grading standard.

For the two partners, we did a more detailed analysis.

```
Call:
lm(formula = Likelihood.to.recommend ~ ., data = b)

Residuals:
    Min      1Q  Median      3Q     Max
-6.6628 -0.8494  0.1205  1.0871  4.4133

Coefficients: (109 not defined because of singularities)
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                        2.4792852 47.5642930   0.052  0.95845
Airline.Status1                    1.3377477  0.1986327   6.735  3.8e-11 ***
Airline.Status2                    0.6893904  0.2720545   2.534  0.01153 *
Airline.Status3                    0.4713120  0.3815717   1.235  0.21724
Age                               -0.0028066  0.0050312  -0.558  0.57715
Gender1                           -0.4682815  0.1539326  -3.042  0.00245 **
Price.Sensitivity                 -0.1207766  0.1390293  -0.869  0.38535
Year.of.First.Flight               0.0031691  0.0236851   0.134  0.89361
Flights.Per.Year                   0.0033205  0.0081116   0.409  0.68242
Loyalty                            0.3820898  0.2069868   1.846  0.06538 .
Type.of.Travel1                   -0.0879685  0.2672898  -0.329  0.74218
Type.of.Travel2                   -2.5616339  0.1822961 -14.052  < 2e-16 ***
Total.Freq.Flyer.Accts             0.0438709  0.0756645   0.580  0.56226
Shopping.Amount.at.Airport         0.0016926  0.0014703   1.151  0.25011
Eating.and.Drinking.at.Airport     0.0030223  0.0012635   2.392  0.01706 *
Class1                             0.1530833  0.2206109   0.694  0.48801
Class2                             0.1549554  0.2868846   0.540  0.58930
Departure.Delay.in.Minutes         0.0010186  0.0073125   0.139  0.88927
Arrival.Delay.in.Minutes          -0.0080208  0.0075528  -1.062  0.28867
Flight.cancelled                  -0.4646908  0.5233151  -0.888  0.37490
Flight.time.in.minutes            -0.0061211  0.0063082  -0.970  0.33227
Flight.Distance                    0.0006692  0.0008122   0.824  0.41026
Day                                0.0114835  0.1758220   0.065  0.94795
`Origin.City_Sacramento, CA`              NA         NA      NA       NA
`Origin.City_Salt Lake City, UT`          NA         NA      NA       NA
`Origin.City_Seattle, WA`                 NA         NA      NA       NA
`Origin.City_New York, NY`        -1.0660492  1.8816324  -0.567  0.57122
`Origin.City_Boston, MA`                  NA         NA      NA       NA
`Origin.City_Atlanta, GA`         -0.8247434  1.8307285  -0.451  0.65251
`Origin.City_Crescent City, CA`           NA         NA      NA       NA
`Origin.City_Cleveland, OH`       -1.1704411  1.8652909  -0.627  0.53058
`Origin.City_San Diego, CA`               NA         NA      NA       NA
`Origin.City_Washington, DC`      -0.9879449  1.8509696  -0.534  0.59371
`Origin.City_Detroit, MI`         -0.2498478  1.8558929  -0.135  0.89295
`Origin.City_San Francisco, CA`           NA         NA      NA       NA
`Origin.City_Denver, CO`          -1.1817443  1.8677223  -0.633  0.52715
`Origin.City_Charlotte, NC`        0.1724489  2.0285742   0.085  0.93228
`Origin.City_Phoenix, AZ`                 NA         NA      NA       NA
```

```
`Origin.City_Evansville, IN`                  NA         NA      NA       NA
`Origin.City_Pocatello, ID`                   NA         NA      NA       NA
`Origin.City_Rapid City, SD`          -0.5136618  2.5778576  -0.199  0.84213
`Origin.City_Bellingham, WA`                  NA         NA      NA       NA
`Origin.City_Barrow, AK`                      NA         NA      NA       NA
`Origin.City_Tallahassee, FL`         -0.9635752  2.0972165  -0.459  0.64607
`Origin.City_Santa Maria, CA`                 NA         NA      NA       NA
`Origin.City_State College, PA`       -0.1267687  2.6091588  -0.049  0.96127
`Origin.City_Aberdeen, SD`                    NA         NA      NA       NA
`Origin.City_Shreveport, LA`          -0.9654183  1.9677173  -0.491  0.62387
`Origin.City_South Bend, IN`                  NA         NA      NA       NA
`Origin.City_Portland, ME`                    NA         NA      NA       NA
`Origin.City_Kalamazoo, MI`                   NA         NA      NA       NA
`Origin.City_Twin Falls, ID`                  NA         NA      NA       NA
`Origin.City_Toledo, OH`                      NA         NA      NA       NA
`Origin.City_Burlington, VT`          -0.4063465  2.2628997  -0.180  0.85755
`Origin.City_Chico, CA`                       NA         NA      NA       NA
`Origin.City_Duluth, MN`                      NA         NA      NA       NA
`Origin.City_Traverse City, MI`       -0.2242363  2.2229398  -0.101  0.91968
`Origin.City_Amarillo, TX`            -0.1153958  2.2338974  -0.052  0.95882
`Origin.City_Hibbing, MN`                     NA         NA      NA       NA
`Origin.City_International Falls, MN`         NA         NA      NA       NA
`Origin.City_Cody, WY`                        NA         NA      NA       NA
`Origin.City_Myrtle Beach, SC`        -1.5674870  2.5733805  -0.609  0.54267
`Origin.City_Minot, ND`                       NA         NA      NA       NA
`Origin.City_Paducah, KY`                     NA         NA      NA       NA
`Origin.City_Flagstaff, AZ`                   NA         NA      NA       NA
`Origin.City_Watertown, NY`                   NA         NA      NA       NA
`Origin.City_Charlottesville, VA`      0.9445646  2.5646701   0.368  0.71278
`Origin.City_Sitka, AK`                       NA         NA      NA       NA
`Origin.City_Jackson, WY`                     NA         NA      NA       NA
`Origin.City_Roanoke, VA`             -1.7027511  2.1146086  -0.805  0.42100
`Origin.City_Corpus Christi, TX`              NA         NA      NA       NA
`Origin.City_Dothan, AL`              -1.6675495  2.0951545  -0.796  0.42640
`Origin.City_Redding, CA`                     NA         NA      NA       NA
`Origin.City_Kotzebue, AK`                    NA         NA      NA       NA
`Origin.City_Williston, ND`                   NA         NA      NA       NA
`Origin.City_Kodiak, AK`                      NA         NA      NA       NA
`Origin.City_Dubuque, IA`                     NA         NA      NA       NA
 [ reached getOption("max.print") -- omitted 32 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.789 on 609 degrees of freedom
Multiple R-squared:  0.5157,    Adjusted R-squared:  0.4186
F-statistic: 5.315 on 122 and 609 DF,  p-value: < 2.2e-16
```

For the FlyFast Airways Inc, we applied linear model once more to see which attributes are important to final scores.

The result shows that the FlyFast Airways Inc is bad in the personal travel. Moreover, the average effects of the original city are negative, while most of them are also negative.

```
Residuals:
    Min      1Q  Median      3Q     Max
-5.6011 -0.8830  0.0575  1.0319  4.7124

Coefficients: (110 not defined because of singularities)
                                      Estimate Std. Error t value Pr(>|t|)
(Intercept)                         -4.310e+01  4.256e+01  -1.013 0.311445
Airline.Status1                      1.263e+00  1.595e-01   7.918 9.04e-15 ***
Airline.Status2                      8.369e-01  2.378e-01   3.519 0.000459 ***
Airline.Status3                     -7.279e-01  3.416e-01  -2.131 0.033445 *
Age                                 -6.609e-03  4.366e-03  -1.514 0.130522
Gender1                             -1.401e-01  1.315e-01  -1.066 0.286978
Price.Sensitivity                   -2.865e-01  1.129e-01  -2.537 0.011401 *
Year.of.First.Flight                 2.458e-02  2.119e-02   1.160 0.246542
Flights.Per.Year                     5.365e-03  6.391e-03   0.839 0.401505
Loyalty                              1.061e-01  1.755e-01   0.605 0.545495
Type.of.Travel1                     -2.143e-01  2.513e-01  -0.853 0.394027
Type.of.Travel2                     -2.493e+00  1.571e-01 -15.875  < 2e-16 ***
Total.Freq.Flyer.Accts               1.357e-02  6.340e-02   0.214 0.830593
Shopping.Amount.at.Airport           2.465e-03  1.166e-03   2.114 0.034823 *
Eating.and.Drinking.at.Airport       1.676e-03  1.297e-03   1.292 0.196662
Class1                              -3.144e-01  2.001e-01  -1.571 0.116657
Class2                               1.280e-01  2.409e-01   0.531 0.595483
Departure.Delay.in.Minutes          -3.706e-03  6.712e-03  -0.552 0.581063
Arrival.Delay.in.Minutes            -1.099e-03  6.603e-03  -0.167 0.867808
Flight.cancelled                     2.070e-01  5.773e-01   0.359 0.720029
Flight.time.in.minutes              -7.944e-04  4.386e-03  -0.181 0.856328
Flight.Distance                      2.270e-04  5.378e-04   0.422 0.673071
Day                                  5.020e-02  1.500e-01   0.335 0.738019
`Origin.City_Sacramento, CA`         5.358e-01  1.799e+00   0.298 0.765989
`Origin.City_Salt Lake City, UT`     3.783e-01  1.721e+00   0.220 0.826086
`Origin.City_Seattle, WA`            1.977e-01  1.760e+00   0.112 0.910586
`Origin.City_New York, NY`                  NA         NA      NA       NA
`Origin.City_Boston, MA`             1.783e+00  2.428e+00   0.734 0.462891
`Origin.City_Atlanta, GA`           -6.577e-01  1.918e+00  -0.343 0.731695
`Origin.City_Crescent City, CA`     -1.551e+00  2.089e+00  -0.742 0.458097
`Origin.City_Cleveland, OH`          1.081e+00  1.983e+00   0.545 0.585703
`Origin.City_San Diego, CA`         -2.344e-02  1.764e+00  -0.013 0.989404
`Origin.City_Washington, DC`        -6.211e-02  1.831e+00  -0.034 0.972952
`Origin.City_Detroit, MI`            8.015e-01  1.799e+00   0.446 0.656019
`Origin.City_San Francisco, CA`      3.749e-01  1.723e+00   0.218 0.827836
`Origin.City_Denver, CO`            -1.984e-01  1.723e+00  -0.115 0.908371
`Origin.City_Charlotte, NC`                 NA         NA      NA       NA
`Origin.City_Phoenix, AZ`            5.456e-01  1.730e+00   0.315 0.752488
`Origin.City_Charleston, SC`                NA         NA      NA       NA
```

```
`Origin.City_Rapid City, SD`           1.889e-01  1.980e+00   0.095 0.924023
`Origin.City_Bellingham, WA`                  NA         NA      NA       NA
`Origin.City_Barrow, AK`                      NA         NA      NA       NA
`Origin.City_Tallahassee, FL`                 NA         NA      NA       NA
`Origin.City_Santa Maria, CA`          2.065e+00  2.093e+00   0.986 0.324268
`Origin.City_State College, PA`               NA         NA      NA       NA
`Origin.City_Aberdeen, SD`            -2.650e-01  1.979e+00  -0.134 0.893522
`Origin.City_Shreveport, LA`                  NA         NA      NA       NA
`Origin.City_South Bend, IN`           9.257e-01  1.934e+00   0.479 0.632258
`Origin.City_Portland, ME`                    NA         NA      NA       NA
`Origin.City_Kalamazoo, MI`                   NA         NA      NA       NA
`Origin.City_Twin Falls, ID`           2.916e+00  2.435e+00   1.197 0.231620
`Origin.City_Toledo, OH`                      NA         NA      NA       NA
`Origin.City_Burlington, VT`                  NA         NA      NA       NA
`Origin.City_Chico, CA`               -9.143e-01  2.422e+00  -0.377 0.705941
`Origin.City_Duluth, MN`              -4.686e+00  2.409e+00  -1.945 0.052102 .
`Origin.City_Traverse City, MI`               NA         NA      NA       NA
`Origin.City_Amarillo, TX`                    NA         NA      NA       NA
`Origin.City_Hibbing, MN`              2.243e+00  2.415e+00   0.929 0.353337
`Origin.City_International Falls, MN`  2.510e-01  1.972e+00   0.127 0.898748
`Origin.City_Cody, WY`                -1.388e+00  2.405e+00  -0.577 0.564072
`Origin.City_Myrtle Beach, SC`                NA         NA      NA       NA
`Origin.City_Minot, ND`                1.038e+00  2.089e+00   0.497 0.619536
`Origin.City_Paducah, KY`              1.809e+00  2.411e+00   0.750 0.453252
`Origin.City_Flagstaff, AZ`           -5.480e-01  2.409e+00  -0.227 0.820117
`Origin.City_Watertown, NY`                   NA         NA      NA       NA
`Origin.City_Charlottesville, VA`             NA         NA      NA       NA
`Origin.City_Sitka, AK`                       NA         NA      NA       NA
`Origin.City_Jackson, WY`             -1.519e+00  2.090e+00  -0.727 0.467467
`Origin.City_Roanoke, VA`                     NA         NA      NA       NA
`Origin.City_Corpus Christi, TX`              NA         NA      NA       NA
`Origin.City_Dothan, AL`                      NA         NA      NA       NA
`Origin.City_Redding, CA`             -1.188e+00  2.405e+00  -0.494 0.621658
`Origin.City_Kotzebue, AK`                    NA         NA      NA       NA
`Origin.City_Williston, ND`           -7.236e-01  2.427e+00  -0.298 0.765716
`Origin.City_Kodiak, AK`                      NA         NA      NA       NA
`Origin.City_Dubuque, IA`                     NA         NA      NA       NA
 [ reached getOption("max.print") -- omitted 32 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.692 on 725 degrees of freedom
Multiple R-squared:  0.4851,    Adjusted R-squared:  0.3992
F-statistic: 5.645 on 121 and 725 DF,  p-value: < 2.2e-16
```

For the Northwest Business Airlines Inc, the situation is different. This partner also works bad in the personal travel. However, even if the average effect of the original city is negative, most of the effect is positive. This means in some specific cities, this partner's service is really bad so that customers grade it incredibly low.

b. Apriori model

We want to make sure that the data are organized in the right way. First we repair the NAs in the data set and we continue our analysis based on the dataset without NAs. As the original dataset is in JSON format, converting it into a data frame makes it easier to do various manipulations, and at the same time the vectors are all converted to factors. We discuss and agree that the columns of Origin.State, Destination.State, Flight.Distance, Day.of.Month, Partner.Name, olong, olat, dlong, dlat are all useless or redundant information, hence we delete these columns. At this time, the structure of the data set is amenable to Apriori algorithm. However, the levels of some columns are too many. For example, in the Age column, one age is a level and it has 67 levels totally. I divide the columns into intervals of different categories based on data distribution and some common practice. Still in the Age column, as 18 is the typical age of attaining legal adulthood and 66 is the Social Security Retirement Age and they both affect one's income, they are set to cut points.

Moreover, the histograms of the columns are plotted and adjusted the cut points to make sure that not too much or little data falls in an interval. There is often a lot of back and forth through the process. At last, the data frame is converted into a transactions dataset.

Step 2 is developing the Apriori model to scan the data set for appropriate rules. We know that support for a rule refers to the frequency of occurrence of both members of the pair, i.e., LHS and RHS together. The confidence of a rule refers to the proportion of the time that LHS and RHS occur together versus the total number of appearances of LHS. What we are interested in is which parts of people are likely to recommend Southeast Airlines or not, so the confidence is set to a relatively large number. As for support, because some categories of people account for a large proportion, for example, Flight.cancelled=No accounts for 97.94%, Class=Eco accounts for 81.11%, even if the percentage of promoters or detractors in these groups is low, the overall percentage will not be low. Thus, support is less important than confidence in analysis. At the same time, the support can be too small, otherwise people of the rule will have a low percentage. Changing their attitude won't have a significant effect on the Net Promoter Score (NPS). What's more, sometimes too many rules were generated to examine. Then, the support or confidence are analyzed and operated to get a manageable number of rules to examine one by one.

It is found that people who are likely to recommend had following typical characteristics.

1: Airline Status=Silver

2: Age=35-49, Type of Travel=Business travel

3. Type of Travel=Business travel, Eating and Drinking at Airport=50-99

4. Gender=Male, Price Sensitivity=1, Type of Travel=Business travel

5. Price Sensitivity=1, Type of Travel=Business travel, Arrival Delay in Hours=0

6. Price Sensitivity =1, Flight Age=5-9, Type of Travel=Business travel, Flight cancelled=No

```
     lhs                                          rhs                                    support confidence     lift count
[1]  {Airline.Status=Silver}                   => {Likelihood.to.recommend=promoters} 0.1058160  0.5248432 1.582533  1088
[2]  {Airline.Status=Silver,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1046489  0.5274510 1.590396  1076
[3]  {Age=35-49,
      Type.of.Travel=Business travel}          => {Likelihood.to.recommend=promoters} 0.1251702  0.5183246 1.562878  1287
[4]  {Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99}    => {Likelihood.to.recommend=promoters} 0.1385917  0.5825838 1.756635  1425
[5]  {Age=35-49,
      Type.of.Travel=Business travel,
      Class=Eco}                               => {Likelihood.to.recommend=promoters} 0.1026065  0.5161448 1.556305  1055
[6]  {Age=35-49,
      Type.of.Travel=Business travel,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1239059  0.5200000 1.567930  1274
[7]  {Price.Sensitivity=1,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99}    => {Likelihood.to.recommend=promoters} 0.1037736  0.5904815 1.780449  1067
[8]  {Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Class=Eco}                               => {Likelihood.to.recommend=promoters} 0.1116514  0.5800910 1.749119  1148
[9]  {Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1379109  0.5859504 1.766787  1418
[10] {Gender=Male,
      Price.Sensitivity=1,
      Type.of.Travel=Business travel}          => {Likelihood.to.recommend=promoters} 0.1076639  0.5187441 1.564143  1107
[11] {Price.Sensitivity=1,
      Type.of.Travel=Business travel,
      Arrival.Delay.in.Hours=0}                => {Likelihood.to.recommend=promoters} 0.1240031  0.5124598 1.545194  1275
[12] {Age=35-49,
      Type.of.Travel=Business travel,
      Class=Eco,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1014394  0.5181321 1.562298  1043
[13] {Price.Sensitivity=1,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1031900  0.5930688 1.788250  1061
[14] {Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Class=Eco,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1109706  0.5839304 1.760696  1141
[15] {Gender=Male,
      Price.Sensitivity=1,
      Type.of.Travel=Business travel,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1068858  0.5223384 1.574980  1099
```

Figure 1: Promoters, support=0.1,confidence=0.5

```
[16] {Price.Sensitivity=1,
      Flight.Age=5-9,
      Type.of.Travel=Business travel,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1007586  0.5041363 1.520096  1036
[17] {Price.Sensitivity=1,
      Type.of.Travel=Business travel,
      Class=Eco,
      Arrival.Delay.in.Hours=0}                => {Likelihood.to.recommend=promoters} 0.1004668  0.5157264 1.555044  1033
[18] {Price.Sensitivity=1,
      Type.of.Travel=Business travel,
      Arrival.Delay.in.Hours=0,
      Flight.cancelled=No}                     => {Likelihood.to.recommend=promoters} 0.1225443  0.5191595 1.565395  1260
```

Figure 2:Promoters, support=0.1,confidence=0.5

As mentioned above, the percentage of some groups of people is high with the characteristics of Flight cancelled=No, Class=Eco, Airline Status=Blue, Price Sensitivity=1, Type of Travel=Business travel. Some rules are similar just adding one of the characteristics to another.

It is also discovered that people who are not quite likely to recommend had following typical characteristics.

1. Loyalty=-1~-0.5, Type of Travel=Personal Travel
2. Airline Status=Blue, Type of Travel=Personal Travel

3. Loyalty=-1~-0.5, Type of Travel=Personal Travel, Total Freq Flyer Accts=0
4. Airline Status=Blue, Gender=Female, Type of Travel=Personal Travel

```
     lhs                                  rhs                                    support confidence     lift count
[1]  {Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel}  => {Likelihood.to.recommend=detractors} 0.1343124  0.7010152 2.106939  1381
[2]  {Airline.Status=Blue,
      Type.of.Travel=Personal Travel}  => {Likelihood.to.recommend=detractors} 0.1794398  0.7445521 2.237791  1845
[3]  {Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel,
      Total.Freq.Flyer.Accts=0}        => {Likelihood.to.recommend=detractors} 0.1128185  0.7064555 2.123290  1160
[4]  {Airline.Status=Blue,
      Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel}  => {Likelihood.to.recommend=detractors} 0.1202101  0.7610837 2.287478  1236
[5]  {Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel,
      Class=Eco}                       => {Likelihood.to.recommend=detractors} 0.1120405  0.7045872 2.117675  1152
[6]  {Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel,
      Flight.cancelled=No}             => {Likelihood.to.recommend=detractors} 0.1292550  0.7024313 2.111195  1329
[7]  {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Total.Freq.Flyer.Accts=0}        => {Likelihood.to.recommend=detractors} 0.1241004  0.7604291 2.285511  1276
[8]  {Airline.Status=Blue,
      Gender=Female,
      Type.of.Travel=Personal Travel}  => {Likelihood.to.recommend=detractors} 0.1177786  0.7507750 2.256495  1211
[9]  {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Shopping.Amount.at.Airport=0}    => {Likelihood.to.recommend=detractors} 0.1154445  0.7460717 2.242359  1187
[10] {Airline.Status=Blue,
      Price.Sensitivity=1,
      Type.of.Travel=Personal Travel}  => {Likelihood.to.recommend=detractors} 0.1065940  0.7471029 2.245458  1096
[11] {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Class=Eco}                       => {Likelihood.to.recommend=detractors} 0.1473449  0.7511155 2.257518  1515
[12] {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Flight.cancelled=No}             => {Likelihood.to.recommend=detractors} 0.1730208  0.7477932 2.247533  1779
[13] {Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel,
      Total.Freq.Flyer.Accts=0,
      Flight.cancelled=No}             => {Likelihood.to.recommend=detractors} 0.1080529  0.7067430 2.124154  1111
[14] {Airline.Status=Blue,
      Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel,
      Class=Eco}                       => {Likelihood.to.recommend=detractors} 0.1005641  0.7659259 2.302032  1034
```

Figure 3: Detractors, support=0.1,confidence=0.7

```
[15] {Airline.Status=Blue,
      Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel,
      Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.1154445  0.7643271 2.297226  1187
[16] {Loyalty=-1~-0.5,
      Type.of.Travel=Personal Travel,
      Class=Eco,
      Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.1078584  0.7050223 2.118982  1109
[17] {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Total.Freq.Flyer.Accts=0,
      Class=Eco}                     => {Likelihood.to.recommend=detractors} 0.1024120  0.7708638 2.316873  1053
[18] {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Total.Freq.Flyer.Accts=0,
      Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.1191402  0.7641921 2.296821  1225
[19] {Airline.Status=Blue,
      Gender=Female,
      Type.of.Travel=Personal Travel,
      Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.1134021  0.7551813 2.269738  1166
[20] {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Shopping.Amount.at.Airport=0,
      Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.1111651  0.7509855 2.257128  1143
[21] {Airline.Status=Blue,
      Price.Sensitivity=1,
      Type.of.Travel=Personal Travel,
      Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.1025092  0.7491116 2.251495  1054
[22] {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Class=Eco,
      Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.1418012  0.7530992 2.263480  1458
```

Figure 4:Detractors, support=0.1,confidence=0.7

Then, we try to reduce the support and raise the confidence to find some factors with low percentages in the whole dataset. One promoter is found: Partner Code=DL.

```
     lhs                                        rhs                                    support confidence     lift count
[1]  {Age=35-49,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Partner.Code=DL}                       => {Likelihood.to.recommend=promoters} 0.01108734  0.8085106 2.437861   114
[2]  {Age=35-49,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Partner.Code=DL,
      Flight.cancelled=No}                   => {Likelihood.to.recommend=promoters} 0.01108734  0.8085106 2.437861   114
[3]  {Gender=Male,
      Price.Sensitivity=1,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Partner.Code=DL}                       => {Likelihood.to.recommend=promoters} 0.01118459  0.8156028 2.459246   115
[4]  {Gender=Male,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Class=Eco,
      Partner.Code=DL}                       => {Likelihood.to.recommend=promoters} 0.01137911  0.8068966 2.432994   117
[5]  {Flight.Age=5-9,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Class=Eco,
      Partner.Code=DL}                       => {Likelihood.to.recommend=promoters} 0.01225443  0.8025478 2.419882   126
[6]  {Gender=Male,
      Price.Sensitivity=1,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Partner.Code=DL,
      Flight.cancelled=No}                   => {Likelihood.to.recommend=promoters} 0.01118459  0.8156028 2.459246   115
[7]  {Gender=Male,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Class=Eco,
      Partner.Code=DL,
      Flight.cancelled=No}                   => {Likelihood.to.recommend=promoters} 0.01137911  0.8068966 2.432994   117
[8]  {Flight.Age=5-9,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Class=Eco,
      Partner.Code=DL,
      Flight.cancelled=No}                   => {Likelihood.to.recommend=promoters} 0.01225443  0.8025478 2.419882   126
```

Figure 5: Promoters,support=0.01,confidence=0.8

```
[9]  {Airline.Status=Silver,
      Price.Sensitivity=1,
      Flight.Age=5-9,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Arrival.Delay.in.Hours=0}              => {Likelihood.to.recommend=promoters} 0.01011476  0.8125000 2.449890   104
[10] {Airline.Status=Silver,
      Price.Sensitivity=1,
      Flight.Age=5-9,
      Type.of.Travel=Business travel,
      Eating.and.Drinking.at.Airport=50-99,
      Arrival.Delay.in.Hours=0,
      Flight.cancelled=No}                   => {Likelihood.to.recommend=promoters} 0.01001751  0.8174603 2.464847   103
```

Figure 6:Promoters, support=0.01,confidence=0.8

```
    lhs                                rhs                                      support confidence     lift count
[1] {Gender=Female,
     Type.of.Travel=Personal Travel,
     Partner.Code=00}               => {Likelihood.to.recommend=detractors} 0.02129936  0.9521739 2.861810   219
[2] {Airline.Status=Blue,
     Type.of.Travel=Personal Travel,
     Partner.Code=00}               => {Likelihood.to.recommend=detractors} 0.02752383  0.9593220 2.883294   283
[3] {Gender=Female,
     Type.of.Travel=Personal Travel,
     Partner.Code=00,
     Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.02071581  0.9551570 2.870776   213
[4] {Airline.Status=Blue,
     Type.of.Travel=Personal Travel,
     Class=Eco,
     Partner.Code=00}               => {Likelihood.to.recommend=detractors} 0.02373079  0.9531250 2.864669   244
[5] {Airline.Status=Blue,
     Type.of.Travel=Personal Travel,
     Partner.Code=00,
     Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.02664851  0.9614035 2.889550   274
[6] {Airline.Status=Blue,
     Type.of.Travel=Personal Travel,
     Class=Eco,
     Partner.Code=00,
     Flight.cancelled=No}           => {Likelihood.to.recommend=detractors} 0.02295273  0.9554656 2.871703   236
```

Figure 7:Detractors, support=0.02,confidence=0.95

In conclusion, the common factors of promoters are,

Airline Status=Silver, Age=35-49, Type of Travel=Business travel, Eating and Drinking at Airport=50-99, Gender=Male, Price Sensitivity=1, Arrival Delay in Hours=0, Flight Age=5-9, Flight cancelled=No, Partner Code=DL.

The common factors of detractors are,

Loyalty=-1~-0.5, Type of Travel=Personal Travel, Airline Status=Blue, Total Freq Flyer Accts=0, Gender=Female, Code=OO.

c. SVM

The SVM is used for the data prediction. The data preparation step is almost the same with linear model. After the data type transformation, a SVM model is applied depending on the training set. The cost parameter, the `C'-constant of the regularization term in the Lagrange formulation is 5. This means that our demand on this model is high. The result of comparing predictions with the test set is shown below.

```
> svmPred <- predict(svmOutput, testData)
> #validate the SVM model
> svmPred1 <- round(svmPred)
> correctRate <- sum(svmPred1[,1] == testData$Likelihood.to.recommend)/dim(testData)[1]
> correctRate
[1] 0.7619603
```

From the graph, the accuracy is 76% which means the prediction can be viewed as valid.

Visualization

a. US map

Based on the results from our modelling technique, we can now affirm that the customers are mostly affected by the Origin State of travel. Hence, using the ggplot function we plot a map to gauge the likelihood of customers to recommend depending on their origin state of travel. On plotting the map, one of the insights that can be regarded of high importance is the very low satisfaction of customers travelling from the Northern region.

In order to plot a US map following steps were followed:

Step 1:

Since our dataset has a column named Origin.State we map it with the pre-existing dataset within RStudio called "state", we use the function map_state for extracting and storing the map data and stored the mapped values in a new data frame(dfNew1).

Step 2:

We now provide the new created data frame values to summarize the records of the customers depending on each state. To sum up the values of each state we use the group_by function.

Step 3:

The summarize() provides us with two details, first is the mean recommendation within each state using the mean() on all the recommendations while total recommendation gives the sum of all the recommendations received by the particular state.

Step 4:

We now calculate the "customer satisfaction" for each state depending on the output in the Step 3. Hence, we divide the total recommendation by the mean recommendation of each state.

Note: Customer Satisfaction is the basis of our mapping within the states in the ggplot which scales from 200-800 because of the above calculations made.

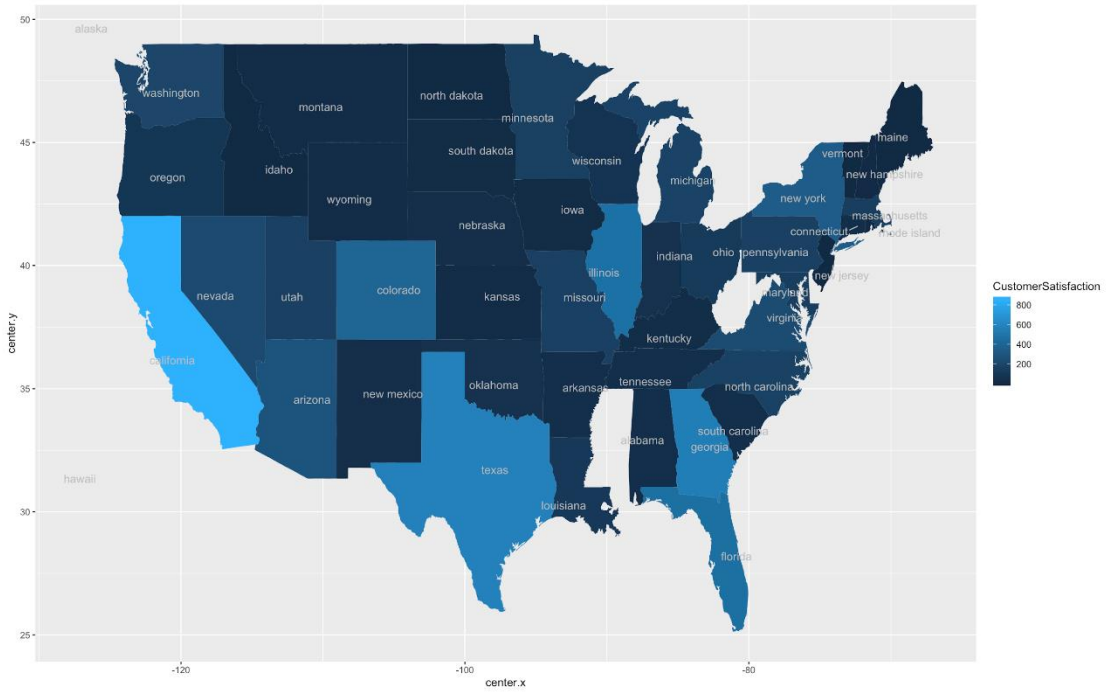
Step 5:

We mapped a ggplot using the stateName as the map_id within the ggplot(). Also, the data frame provided is fetched from Step 3 and Step 4 together mapped with the US state map by "stateName". Within ggplot() we have used the labels as "stateName".

Step 6: The output of Step 5 is provided to geom_map which actually sketches the US map and fill the customer satisfaction scale depending on the calculation made in Step 4.(Please refer the note).

Step 7: In this step we take the map and sketch it around the center.x axis and center.y axis using the us dataset values.

Step 8: We provide the latitudes and longitudes of the us map to expand_limits function and produce the map below.



From the above map we can clearly notice that there is a severe concern in the satisfaction of customers travelling from the North side of the country. While, customer's in the Southern region are satisfied with their origin state of travel. Most likely, showing us that since the data given to us is from January to March maybe weather can be the reason for the dissatisfaction.

Since, Northern States of America face severe winters they may have to cancel flights due to weather conditions.

b. Word cloud

Now, throwing some light on the customer experience using the word cloud text mining concept we have showed

1)What is important to the customer?

2)To gain customer satisfaction and more promoters where do we need to focus.

Thus, the airlines should focus more on service, seat, time and delay at highest priority become it shows that majority of customers are looking forward to them and affect the promoting factors.

Insights

Based on analysis from our three models, we suggest Southeast Airline to put less emphasis on Current Loyalty Program and pay more attention to the factors that affect the customers' satisfaction the most. One of the methods to improve customers' satisfaction toward our airline is to avoid partnership with Northwest Business Airlines and FlyFast Airline. Based on the linear model and the apriori model analysis, we observed that these two airlines have been receiving NPS(Net Promoter Score) below average, which have a negative effect on total NPS.

The statistics from linear model show that partnership with FlyFast Airways has decreased total NPS by approximately 0.72points. That shows dissatisfaction of customers significantly affects overall experience in Southeast airline especially when our airline has partnership with FlyFast. In more details, we observed NPS of customers who fled with FlyFast Partnership individually. We found out that most of the coefficients of partnership with FlyFast are negative values in between 0

and 3. It indicates that most of the customers with FlyFast provided NPS that negatively contributes to overall NPS.

Similarly, partnership with Northwest Business Airlines has a negative effect on overall NPS of Southeast Airline. From observation of linear model statistics, the factor of partnership with Northwest Business Airlines has a coefficient of approximately -2 points toward dependent variable customer satisfaction, which shows that partnership with Northwest Business Airlines decreased the score of recommendation by approximately 2 points. After taking a closer look to individual customer response, we found out that the NPSs that are provided by customers with Northwest partnership shows extremely unstability. The score of recommendation varies significantly among individuals, including negative and positive coefficients in linear model. We found out that customer experience is highly unpredictable, which leads to an overall negative effect on NPS. The inconsistency of Northwest Business Airlines' services might be responsible for the precariousness of customers' ratings. Thus we get the conclusion of avoiding partnership with these two airline companies.

By contrast, more partnership with the Sigma Airlines Inc should be useful for the overall NPS of Southeast Airline, since the Sigma Airlines Ins is a promoter.

Another suggestion is to improve service quality in northern regions, especially during the winter. Since the weather influence can cause dissatisfaction of customers even further when their flights are delayed or cancelled, we suggest airline company to provide services such as taxis and hotels for those customers who need to stay overnight due to flight cancellation. Moreover, Airline company should offer compensation for customers whose flights are cancelled or delayed based on tickets purchase and time consumption. We also suggest Southeast Airline to provide better training for flight attendants and workers in northern states to improve overall quality of customer service.

IST-664: BOSTON AIRBNB FEEDBACK ANALYSIS

Abstract

Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present more unique, personalized way of experiencing the world. The Airbnb has been trying to improve their service. In the past few years, applying Natural Language Process (NLP) to analyze users' satisfaction becomes a trend for service improvement. Hence, we find a dataset of the guests' comments on the Airbnb services.

This dataset describes the listing activity and metrics in Boston. We are focusing on the following Airbnb activity included in this Boston dataset. Based on the comments of the reviewers, we want to dig deeper on the general feedbacks of each Boston neighborhood and give specific suggestions on the Airbnb services.

The sentiment analysis will be done on both the paragraph and the sentence level. What users pay attention to is the target of the analysis.

Methods

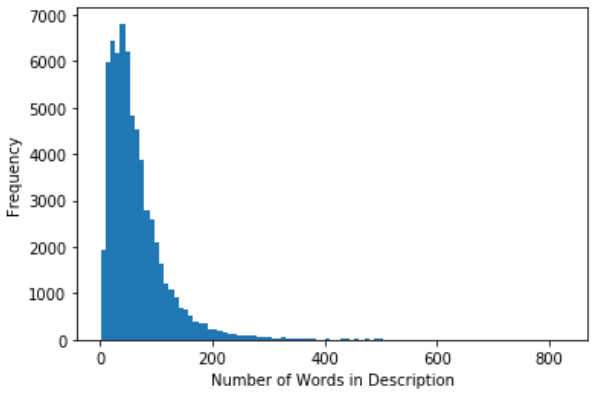
Our sentimental analysis will be applied on both sentence and comment level.
The k-means model will be applied to discover the similar patterns. This will classify the words in the comments into different categories.
The sentiment polarity of comments is also in our consideration. In this step, words will be judged whether they are positive or negative.
After this step, it could be seen clearly that what kinds of words are the guests' focus. In addition, the Latent Dirichlet allocation (LDA) will be used for topic analysis. It can separate the topics that users pay most attention on. In this way, specific suggestions can be given to the Airbnb for future improvement.

preprocessing

Before building a K-means model, we did some exploring and preprocessing to the data. Firstly, we explored the distribution of number of words in the customers review.

```
In [31]: df = scored_reviews
         df['word_count'] = df['review'].apply(lambda x: len(str(x).split(" ")))
         print("Word Count Median: " + str(df['word_count'].median()))
         print(df['word_count'].describe())
         x = df['word_count']
         n_bins = 95
         plt.hist(x, bins=n_bins)
         plt.xlabel('Number of Words in Description')
         plt.ylabel('Frequency')
         plt.show()
```

```
Word Count Median: 51.0
count    63992.000000
mean        64.917958
std         54.255704
min          1.000000
25%         30.000000
50%         51.000000
75%         83.000000
max        827.000000
Name: word_count, dtype: float64
```



From the summary information of the distribution and the histogram, we could see that the distribution of number of words was a right skewed one but with a wide range of distribution (from 1 to 827). This could be beneficial to the k-means model, since the model using tf-idf score as features and we could get high variance tf-idf scores from the data set.

Lowercase all the words

```
review = df['review'][w].lower()
```

NA removement

```
reviews_df=reviews.dropna(subset=["comments"])
```

Tokenization

```
split_text = word_tokenize(review)
```

Language detection

```
def get_language_likelihood(input_text):
    input_text = input_text.lower()
    input_words = wordpunct_tokenize(input_text)

    language_likelihood = {}
    total_matches = 0
    for language in stopwords._fileids:
        language_likelihood[language] = len(set(input_words) &
                set(stopwords.words(language)))

    return language_likelihood

def get_language(input_text):

    likelihoods = get_language_likelihood(input_text)
    return sorted(likelihoods, key=likelihoods.get, reverse=True)[0]

reviews_f = [r for r in negreview.values if pd.notnull(r) and get_language(r) == 'english']
```

```
La maison située
dans East Boston
était sympathique,
propre et non loin
du métro (arrêt
"airport" de la blue
line). La cuisine
est bien équipée
avec lave vaisselle,
micro ondes,
cafetière et
autres!! ...
```

As for the preprocessing part, we removed the stop words and negation words, did the lemmatization and used the tf-idf vectorizer to transform the original data.

In order to improve the performance of our clustering algorithm, we chose to add some stop words to the original nltk stop words. Because the negation words only reflected the customers' attitudes and we cared more about the topic words, so we removed them.

```
#create a list of stop words
nltkstopwords = stopwords.words("english")
morestopwords = ['could','would','might','must','need','sha','wo','y',"'s","'d","'ll","'t","'m","'re","'ve", "n't",'!',
stopwords = nltkstopwords + morestopwords

#remove negationwords
negationwords = ['no', 'not', 'never', 'none', 'nowhere', 'nothing','noone', 'rather',
                 'hardly', 'scarcely', 'rarely', 'seldom', 'neither', 'nor']
negationwords.extend(['ain', 'aren', 'couldn', 'didn', 'doesn', 'hadn', 'hasn', 'haven',
                      'isn', 'ma', 'mightn', 'mustn', 'needn', 'shan', 'shouldn', 'wasn', 'weren', 'won', 'wouldn'])
newstopwords = [word for word in stopwords if word not in negationwords]
```

Remove all the numbers, the characters that are not alphabetical, the stop words, words whose lengths are less than three

```
In [298]:
# remove unwanted characters, numbers and symbols
# remove unwanted characters, numbers and symbols
most_negative = most_negative.str.replace("[^a-zA-Z#]", " ")

In [299]:

from nltk.corpus import stopwords
stop_words = stopwords.words('english')

In [300]:
# function to remove stopwords
def remove_stopwords(rev):
    rev_new = " ".join([i for i in rev if i not in stop_words])
    return rev_new

# remove short words (length < 3)
most_negative = most_negative.apply(lambda x: ' '.join([w for w in x.split() if len(w)>2]))

# remove stopwords from the text
neg_reviews = [remove_stopwords(r.split()) for r in most_negative]

# make entire text lowercase
neg_reviews = [r.lower() for r in neg_reviews]
```

Then we did the lemmatization to the words that not appeared in the stop words and have a length more than 2.

```
#Lemmatization
lem = WordNetLemmatizer()
split_text = [lem.lemmatize(word) for word in split_text if not word in newstopwords and len(word) >2]
split_text = " ".join(split_text)
clean_review.append(split_text)
```

Modeling

K-means Clustering

Since our Airbnb data set does not have a target variable such as score or attitude. It would be suitable to use unsupervised learning to do some analysis.

K-means is a simple but powerful unsupervised learning algorithm (meaning there are no target labels) that allows us to identify similar groups or clusters of data points within data. Using this algorithm, we could cluster our customer reviews into some specific number of groups to figure out what words the customers care the most and we could gain some insights about the service quality in the Boston area.

After those preprocessing, we used the *TfidfVectorizer* class to do the vectorization on the cleaned reviews.

```
#TF-IDF vectorizer
tfv = TfidfVectorizer(stop_words = newstopwords, ngram_range = (1,1))
#transform
vec_text = tfv.fit_transform(clean_review)
#returns a list of words.
words = tfv.get_feature_names()
```

Then we built a 10-clusters K-means clustering model using the sklearn package in python. We set the max number of iterations as 200 to decrease the execution time and set the relative tolerance as 0.01 to improve the performance.

```
#setup kmeans clustering
kmeans = KMeans(n_clusters = 10, n_init = 17, n_jobs = -1, tol = 0.01, max_iter = 200)
#fit the data
kmeans.fit(vec_text)
#this loop transforms the numbers back into words
common_words = kmeans.cluster_centers_.argsort()[:,-1:-11:-1]
for num, centroid in enumerate(common_words):
    print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))
```

The clusters we got were as follows.

```
0 : canceled, automated, posting, reservation, arrival, day, host, 103, 100, 163
1 : house, room, stay, nice, boston, clean, great, really, host, comfortable
2 : place, stay, room, boston, great, clean, host, not, location, comfortable
3 : great, location, place, stay, host, clean, boston, definitely, everything, easy
4 : highly, recommend, place, great, recommended, boston, stay, host, location, apartment
5 : apartment, great, boston, location, stay, clean, everything, perfect, well, easy
6 : home, feel, made, stay, boston, welcome, great, felt, host, comfortable
7 : check, easy, great, apartment, stay, location, place, clean, time, boston
8 : nice, place, really, room, clean, stay, great, host, location, apartment
9 : good, location, place, nice, room, clean, host, stay, great, experience
```

From the visualization we could see that the first several clusters were pretty clear about certain topics. For example, the cluster 0 included ‘canceled’, ‘day’, ‘host’ and some numbers, which could be parts of the time topic. The cluster 1 included ‘house’, ‘room’, ‘stay’, we could consider it as the topic mainly about the living environment. What is more, there were some words that showed in

nearly every cluster such as ‘clean’ and ‘comfortable’. Those words could reflect what customers care the most and are valuable for us to dig deeper.

To summarize, we used K-means algorithm as our first unsupervised learning method. We built a 10 clusters K-means model and found that time, comfortability, location would be the most important features of service quality. And ‘clean’, ‘comfortable’ would be the evaluation factors that nearly every single customer would consider.

Sentiment Polarity for Reviews—Bigram

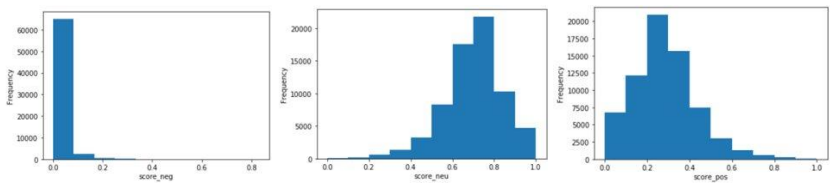
Comment level

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

This package labels comments based on positive, negative, and neutral words in the comments, and then come up a compound score to identify the polarity.

```
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 :)
compound: 0.9626, neg: 0.0, neu: 0.648, pos: 0.352,
```

These three charts illustrate the distribution of positive, negative, and neutral score of all comments. We can learn that most words are positive and neutral.



To find out more informative negative comments, we label the comments 1 when its compound score above 0, others label to 0. Exact all the comments with label 0 to do bigram analysis.

```
label_df['label'] = label_df['compound'].apply(lambda x: 1 if x >= 0 else 0)
label_df.head(5)
```

1) Bigram

```
[('room', 'exactly'),
 ('exactly', 'pictured'),
 ('pictured', 'frills'),
 ('frills', 'yet'),
 ('yet', 'adequate'),
 ('adequate', 'needs'),
 ('needs', 'street'),
 ('street', 'parking'),
 ('parking', 'nearly'),
 ('nearly', 'impossible'),
 ('impossible', 'find'),
 ('find', 'expect'),
 ('expect', 'much'),
 ('much', 'better'),
 ('better', 'boston'),
 ('boston', 'recommend'),
 ('recommend', 'room'),
 ('room', 'others'),
 ('others', 'izzy'),
 ('izzy', 'great')]
```

'Impossible find 'is the only one clear positive comment we find, and the other most frequent bigrams are more about positive comments.

2) bigram_measures.raw_freq

```
[(('great', 'location'), 0.00283409284874639 7),
 (('walking', 'distance'), 0.0018196164312974026),
 (('great', 'host'), 0.0017874108307434663),
 (('place', 'stay'), 0.0016102800276968164),
 (('highly', 'recommend'), 0.0015780744271428802),
 (('definitely', 'stay'), 0.0014331492246501666),
 (('definitely', 'recommend'), 0.0013204296227113895),
 (('everything', 'needed'), 0.0010305792177259625),
 (('great', 'place'), 0.0010144764174489945),
 (('stay', 'boston'), 0.0009822708168950 58),
 (('recommend', 'staying'), 0.0009661680166180899),
 (('apartment', 'clean'), 0.0009500652163411217),
 (('clean', 'comfortable'), 0.0009339624160641536),
 (('location', 'great'), 0.0009178596157871854),
 (('room', 'clean'), 0.0009017568155102173),
 (('enjoyed', 'stay'), 0.0008856540152332491),
 (('downtown', 'boston'), 0.0008534484146793127),
 (('easy', 'get'), 0.0008373456144023446),
 (('great', 'stay'), 0.0008373456144023446),
 (('great', 'time'), 0.0008373456144023446)]
```

Almost all bigrams here are positive, which means the way we label comment based on compound score are not accurate and suitable to our analysis

3) bigram_measures.pmi

```
[(('orient', 'heights'), 13.922328879612357),
 (('stone', 'throw'), 13.922328879612357),
 (('pots', 'pans'), 13.600400784724997),
 (('adams', 'brewery'), 13.3373663788912),
 (('ear', 'plugs'), 13.3373663788912),
 (('sam', 'adams'), 13.114973957554753),
 (('exceeded', 'expectations'), 12.922328879612358),
 (('dunkin', 'donuts'), 12.86343519055879),
 (('commuter', 'rail'), 12.793045862667391),
 (('longwood', 'medical'), 12.752403878170043),
 (('sight', 'seeing'), 12.699936458275909),
 (('francine', 'patrik'), 12.600400784724993),
 (('faneuil', 'hall'), 12.462897260975058),
 (('jon', 'margrit'), 12.462897260975058),
 (('newly', 'renovated'), 12.3373663788912),
 (('francine', 'patrick'), 12.185363285446 15),
 (('years', 'ago'), 12.185363285446 15),
 (('forest', 'hills'), 12.114973957554753),
 (('friday', 'saturday'), 12.114973957554753),
 (('closet', 'hangers'), 12.047859761696216)]
```

'stone throw' is the negative comment, and here are 5 fields--

'Ear plugs', 'Commuter rail', 'Newly renovated', 'Closet hangers'—that we can pay more attention to improve the service quality.

Sentence level

We extract comments that were labeled 0 to do some sentence level analysis.

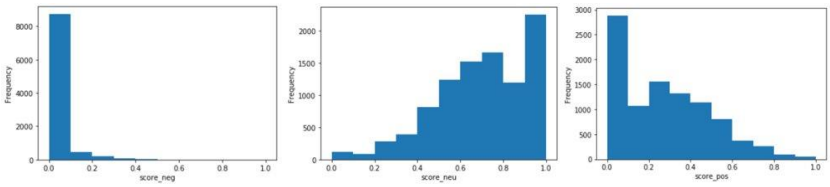
```
from nltk.tokenize import sent_tokenize
neg_review_sent=[]
for r in reviews_f:
    review_token=nltk.sent_tokenize(r)
    for s in review_token:
        neg_review_sent.append(s)
```

```
len(neg_review_sent)
```

```
9554
```

```
the room was exactly as pictured, no frills, yet adequate for my needs.
compound: -0.0772, neg: 0.146, neu: 0.728, pos: 0.126,
the street parking was nearly impossible to find,  but you can't expect much better in boston.
compound: -0.4782, neg: 0.172, neu: 0.828, pos: 0.0,
```

Similar with score distribution in comment level, most words are positive and neutral.



To find out more informative negative comments, we label the sentence 1 when its compound score above 0, others label to 0. Exact all the sentence with label 0 to do bigram analysis.

```
label_sent['label'] = label_sent['compound'].apply(lambda x: 1 if x >= 0 else 0)
negreview_sent=label_sent[label_sent["label"]==0].sentence
```

1) Bigram

```
[('room', 'exactly'),
 ('exactly', 'pictured'),
 ('pictured', 'frills'),
 ('frills', 'yet'),
 ('yet', 'adequate'),
 ('adequate', 'needs'),
 ('needs', 'street'),
 ('street', 'parking'),
 ('parking', 'nearly'),
 ('nearly', 'impossible'),
 ('impossible', 'find'),
 ('find', 'expect'),
 ('expect', 'much'),
 ('much', 'better'),
 ('better', 'boston'),
 ('boston', 'izzy'),
 ('izzy', 'great'),
 ('great', 'clear'),
 ('clear', 'instructions'),
 ('instructions', 'problems')]
```

'impossible find', 'instruction problems' are the two negative comments we find.

2) bigram_measures.raw_freq

```
[(('blocks', 'away'), 0.0015812776723592662),
 (('air', 'conditioning'), 0.0014231499051233396),
 (('bus', 'stop'), 0.0014231499051233396),
 (('let', 'us'), 0.0012650221387887413),
 (('block', 'away'), 0.0009487666034155598),
 (('even', 'though'), 0.0009487666034155598),
 (('late', 'night'), 0.0009487666034155598),
 (('walking', 'distance'), 0.0009487666034155598),
 (('street', 'parking'), 0.0007906388361796331),
 (('us', 'drop'), 0.0007906388361796331),
 (('apartment', 'clean'), 0.0006325110689437065),
 (('check', 'time'), 0.0006325110689437065),
 (('conveniently', 'located'), 0.0006325110689437065),
 (('dirty', 'dishes'), 0.0006325110689437065),
 (('first', 'night'), 0.0006325110689437065),
 (('grocery', 'store'), 0.0006325110689437065),
 (('half', 'block'), 0.0006325110689437065),
 (('minute', 'walk'), 0.0006325110689437065),
 (('much', 'time'), 0.0006325110689437065),
 (('next', 'day'), 0.0006325110689437065)]
```

'dirty dishes' –clear negative comment

'Air conditioning', 'Bus stop', 'Street parking', 'Apartment clean', 'Check time', 'Grocery store'—where we can pay more attention to improve service quality

3) bigram_measures.pmi

```
[(('orient', 'heights'), 13.922328879612357),
 (('stone', 'throw'), 13.922328879612357),
 (('pots', 'pans'), 13.600400784724997),
 (('adams', 'brewery'), 13.337366378891 2),
 (('ear', 'plugs'), 13.3373663788912),
 (('sam', 'adams'), 13.114973957554753),
 (('exceeded', 'expectations'), 12.922328879612358),
 (('dunkin', 'donuts'), 12.86343519055879),
 (('commuter', 'rail'), 12.793045862667391),
 (('longwood', 'medical'), 12.752403878170043),
 (('sight', 'seeing'), 12.699936458275909),
 (('francine', 'patrik'), 12.600400784724993),
 (('faneuil', 'hall'), 12.462897260975058),
 (('jon', 'margrit'), 12.462897260975058),
 (('newly', 'renovated'), 12.3373663788912),
 (('francine', 'patrick'), 12.18536328544615),
 (('years', 'ago'), 12.18536328544615),
 (('forest', 'hills'), 12.114973957554753),
 (('friday', 'saturday'), 12.114973957554753),
 (('closet', 'hangers'), 12.047859761696216)]
```

'stone throw'—clear negative comment

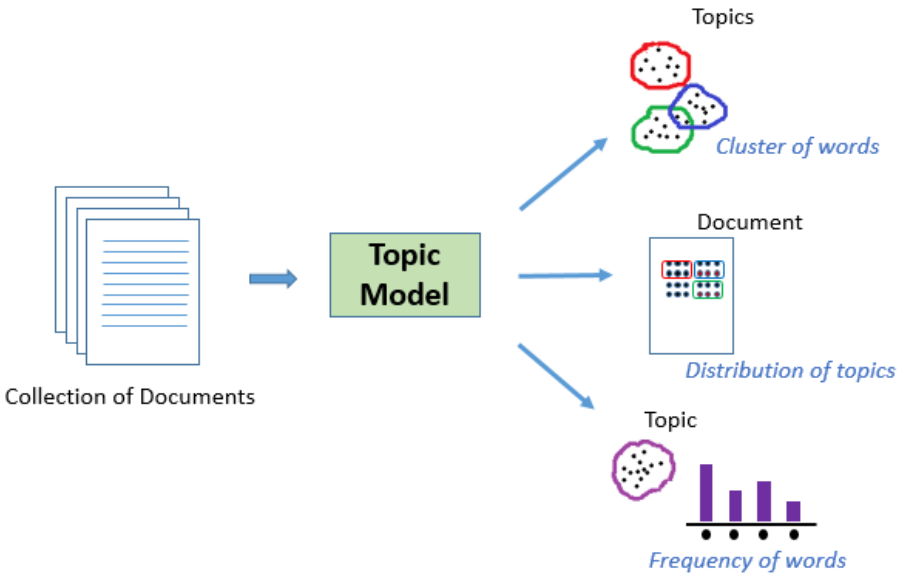
'Ear plugs', 'Commuter rail', 'Newly renovated', 'Closet hangers'—where we can pay more attention to improve service quality

Latent Dirichlet allocation (LDA)

Topic Modeling is a process to automatically identify topics present in a text object and to derive hidden patterns exhibited by a text corpus. Topic Models are very useful for multiple purposes, including:

- Document clustering
- Organizing large blocks of textual data
- Information retrieval from unstructured text
- Feature selection

A good topic model, when trained on some text about the stock market, should result in topics like “bid”, “trading”, “dividend”, “exchange”, etc. The below image illustrates how a typical topic model works:



In our project, we have extreme Airbnb reviews. Our aim here is to extract a certain number of groups of important words from the reviews. This review can help us find the topic which they really care about. So we used Latent Dirichlet Allocation as our topic modeling technique.

4.3.1 prepare

What we interested are attitude of comments. To transform it into review sentiments, I used the Vader sentiment model based on NLTK.

What are the positive and negative reviews?

In [342]:
```
reviews_with_listing['language'] = reviews_with_listing['comments'].apply(language_detection)
reviews_with_listing = reviews_with_listing[(reviews_with_listing['language']=='en')]

test_sentence_1 = "the room was exactly as pictured, yet adequate for my need"
# calculates compound sentiment polarity of the sentence
vader_polarity_compound = lambda s: (SentimentIntensityAnalyzer().polarity_scores(s))['compound']
vader_polarity_compound(test_sentence_1)
```

Out[342]: 0.2263

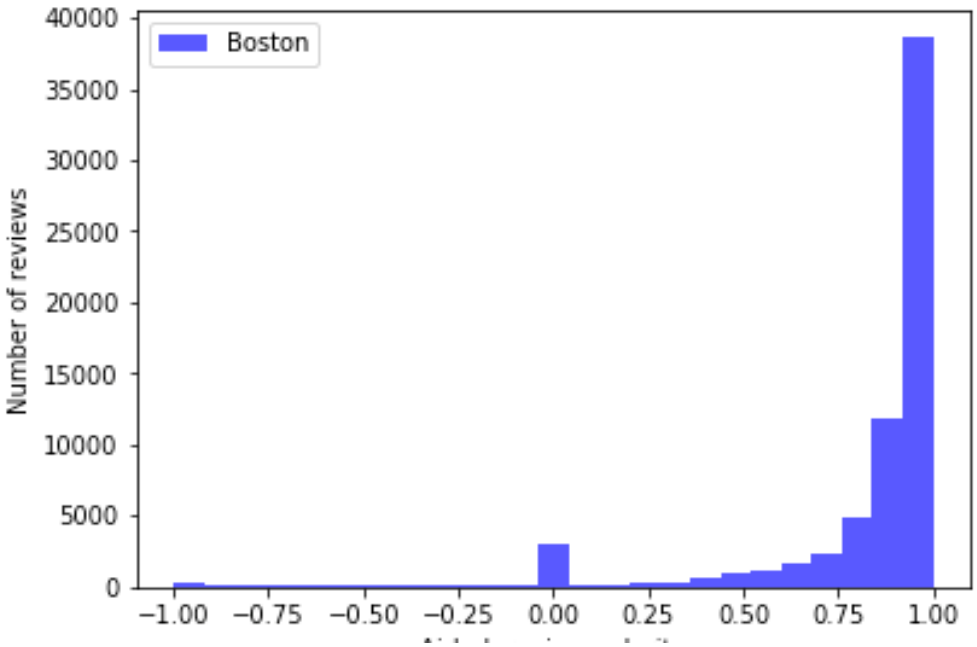
First, we used SentimentIntensityAnalyzer to calculates compound sentiment polarity of the comments.

In [155]:
```
boston_reviews['polarity'] = boston_reviews.comments.map(vader_polarity_compound)
reviews_with_listing['polarity'] = reviews_with_listing.comments.map(vader_polarity_compound)
boston_reviews.to_hdf('boston_reviews.h5', key='boston_reviews', mode='w')
reviews_with_listing.to_hdf('reviews_with_listing.h5', key='reviews_with_listing', mode='w')
```

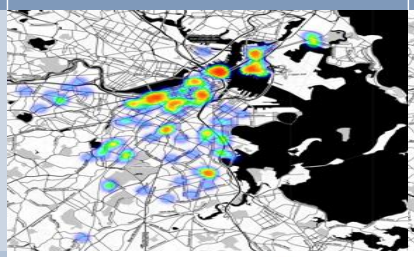
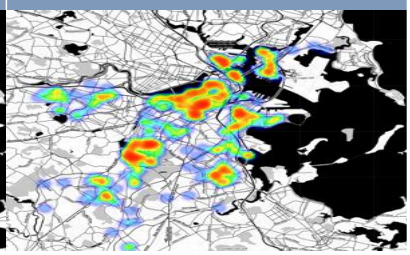
Then, we analyze the distrition of the comments attitude. Then

#We extracted the highest score (> 0.995) and the lowest score (<-0.8). respectively as extremely positive dataset and extrmely negative dataset.

```
In [353]:
1 most_negative=reviews_with_listing[reviews_with_listing.polarity < -0.8].comments
2 most_positive=reviews_with_listing[boston_reviews.polarity > 0.995].comments
```



Then we get two datasets. We used WordCloud and Folium to visualize data according word frequency and geographic distribution. In addition, we preprocessed the comment data for LDA model.

	Very negative reviews	Very positive reviews
Location		
Word		

4.3.2 Negative Topic

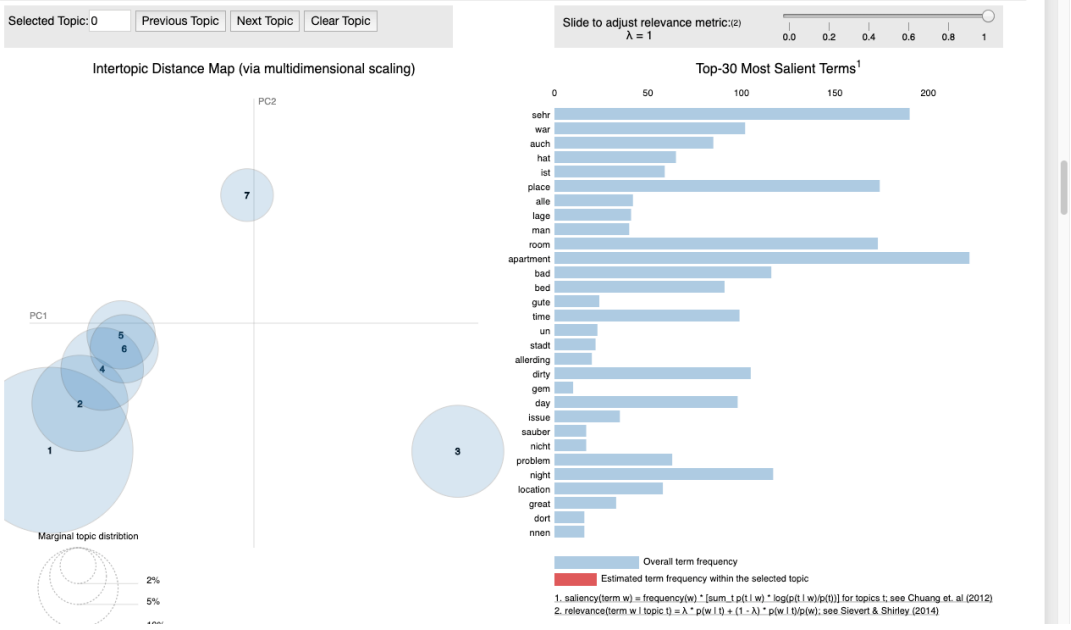
We will start by creating the term dictionary of our negative corpus. Then we will convert the list of reviews (neg_reviews_2) into a Document Term Matrix using the dictionary prepared above.

```
In [307]:
dictionary = corpora.Dictionary(neg_reviews_2)

In [308]:
doc_term_matrix = [dictionary.doc2bow(rev) for rev in neg_reviews_2]

In [309]:
# Creating the object for LDA model using gensim library
LDA = gensim.models.ldamodel.LdaModel

# Build LDA model
lda_model = LDA(corpus=doc_term_matrix, id2word=dictionary, num_topics=7, random_state=100,
                chunksize=1000, passes=50)

In [310]:
lda_model.print_topics()
```

Then we print result and visualization it.

```
In [310]:
lda_model.print_topics()

Out[310]:
[(0,
  '0.019*"room" + 0.017*"apartment" + 0.017*"place" + 0.015*"bathroom" + 0.015*"night" + 0.011*"dirty" + 0.010*"time" + 0.010*"day" + 0.010*"door" +
d"'),
 (1,
  '0.020*"apartment" + 0.010*"bad" + 0.009*"room" + 0.008*"bathroom" + 0.008*"day" + 0.007*"host" + 0.006*"window" + 0.006*"floor" + 0.005*"door" +
l"'),
 (2,
  '0.104*"sehr" + 0.056*"war" + 0.047*"auch" + 0.036*"hat" + 0.031*"ist" + 0.023*"alle" + 0.023*"lage" + 0.021*"man" + 0.016*"bad" + 0.013*"gute"'),
 (3,
  '0.020*"place" + 0.015*"apartment" + 0.007*"location" + 0.007*"day" + 0.007*"review" + 0.006*"time" + 0.006*"difficult" + 0.006*"old" + 0.006*"iss
*"arrival"'),
 (4,
  '0.016*"gem" + 0.011*"bad" + 0.008*"weiterempfehlen" + 0.008*"nlich" + 0.007*"angenehm" + 0.006*"kommunikation" + 0.005*"tage" + 0.005*"tlich" + 0
+ 0.005*"flat"'),
 (5,
  '0.027*"apartment" + 0.022*"place" + 0.020*"room" + 0.014*"bed" + 0.014*"dirty" + 0.010*"night" + 0.008*"bathroom" + 0.007*"time" + 0.007*"day" +
d"'),
 (6,
  '0.012*"time" + 0.009*"room" + 0.008*"great" + 0.007*"place" + 0.006*"host" + 0.006*"location" + 0.006*"door" + 0.006*"poor" + 0.005*"house" + 0.0
k"')]

In [311]:
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, doc_term_matrix, dictionary)
vis
```

Out[311]:



The Topic 0, Topic 1, and Topic 5 has terms like ‘bathroom, ‘clean’, ‘dirty’, indicating that the topic is very much related to bathroom. Similarly, Topic 3 and

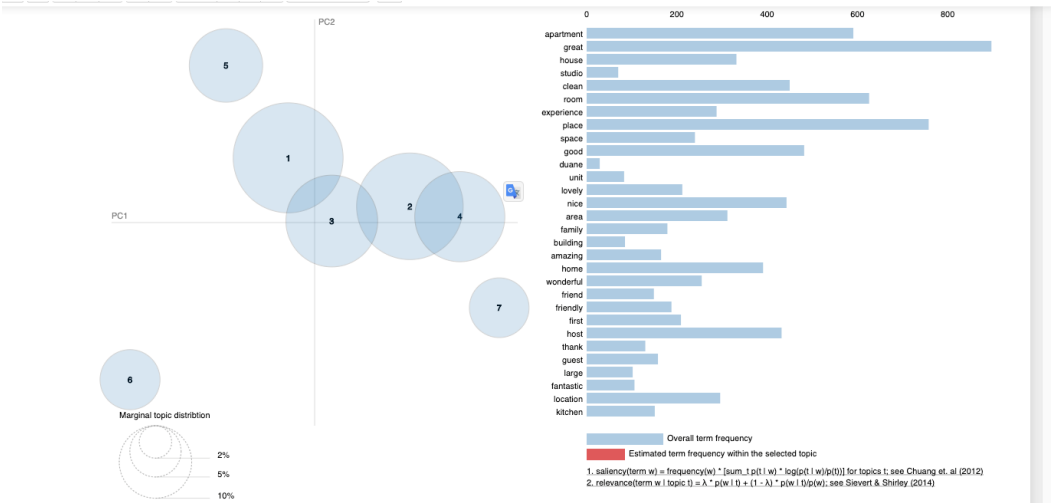
Topic 6 seems to be about the location of the apartment as it has terms like ‘place, and ‘location’.

4.3.3 Positive Topic

Besides, we do same thing with our very positive reviews.

In [317]:

```
# remove unwanted characters, numbers and symbols
most_positive = most_positive.str.replace("[^a-zA-Z#]", " ")
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
# function to remove stopwords
def remove_stopwords(rev):
    rev_new = " ".join([i for i in rev if i not in stop_words])
    return rev_new

# remove short words (length < 3)
most_positive = most_positive.apply(lambda x: ' '.join([w for w in x.split() if len(w)>2]))

# remove stopwords from the text
pos_reviews = [remove_stopwords(r.split()) for r in most_positive]

# make entire text lowercase
pos_reviews = [r.lower() for r in pos_reviews]

nlp = spacy.load('en', disable=['parser', 'ner'])

def lemmatization(texts, tags=['NOUN', 'ADJ']): # filter noun and adjective
       output = []
       for sent in texts:
             doc = nlp(" ".join(sent))
             output.append([token.lemma_ for token in doc if token.pos_ in tags])
       return output

tokenized_pos_reviews = pd.Series(pos_reviews).apply(lambda x: x.split())

pos_reviews_2 = lemmatization(tokenized_pos_reviews)

pos_reviews_3 = []
for i in range(len(pos_reviews_2)):
    pos_reviews_3.append(' '.join(pos_reviews_2[i]))
doc_term_matrix = [dictionary.doc2bow(rev) for rev in pos_reviews_2]
dictionary = corpora.Dictionary(pos_reviews_2)
# Creating the object for LDA model using gensim library
LDA = gensim.models.ldamodel.LdaModel

# Build LDA model
lda_model = LDA(corpus=doc_term_matrix, id2word=dictionary, num_topics=7, random_state=100,
                chunksize=1000, passes=50)
lda_model.print_topics()
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, doc_term_matrix, dictionary)
vis
```




```
In [318]: 1 lda_model.print_topics()

Out[318]: [(0,
  '0.019*"place" + 0.018*"room" + 0.016*"great" + 0.014*"host" + 0.013*"comfortable" + 0.013*"home" + 0.013*"time" + 0.011*"neighborhood" + 0.011*"h
10*"good"'),
 (1,
  '0.026*"place" + 0.020*"great" + 0.018*"good" + 0.017*"home" + 0.015*"room" + 0.015*"time" + 0.014*"house" + 0.014*"clean" + 0.013*"host" + 0.012*
e"'),
 (2,
  '0.024*"great" + 0.014*"apartment" + 0.012*"studio" + 0.012*"space" + 0.011*"good" + 0.011*"area" + 0.010*"room" + 0.009*"duane" + 0.009*"location
*"nice"'),
 (3,
  '0.025*"apartment" + 0.016*"great" + 0.012*"room" + 0.011*"place" + 0.011*"bed" + 0.011*"nice" + 0.010*"comfortable" + 0.009*"clean" + 0.009*"easy
*"neighborhood"'),
 (4,
  '0.038*"apartment" + 0.015*"place" + 0.015*"great" + 0.014*"nice" + 0.013*"clean" + 0.013*"comfortable" + 0.011*"time" + 0.010*"day" + 0.009*"loca
08*"area"'),
 (5,
  '0.021*"room" + 0.018*"great" + 0.017*"place" + 0.016*"clean" + 0.015*"experience" + 0.012*"nice" + 0.011*"good" + 0.010*"host" + 0.010*"wonderful
*"first"'),
 (6,
  '0.030*"great" + 0.017*"place" + 0.013*"comfortable" + 0.012*"room" + 0.011*"house" + 0.011*"good" + 0.010*"easy" + 0.010*"time" + 0.009*"nice" +
t"')]
```

The Topic 0, topic 2, and Topic 4 mentioned 'neighborhood', 'location' which is relevant with 'location' of the apartment. In addition, the Topic 0, Topic 1, Topic 3, Topic 4, Topic 5, and topic 6 mention about 'comfortable' and 'clean' of 'living environment' Topic.

Model summary

We used three model to analyze Airbnb reviews. These model have advantages and disadvantages respectively.

First model is K-means. We used K-means model to classify words into different groups, getting general word frequency. However, this model does not tag for reviews and it has no clear classification based on sentiment.

The second model we applied is Vader sentiment model. Now we focus on negative comments and sentences after tagging sentiment polarity and then calculate bigram frequency. But in comments level, the results show that there are lots of positive words misclassified into negative comments. In sentence level, Vader sentiment model extracts noun words without clear sentiment polarity.

The third one is LDA(Latent Dirichlet Allocation) model. After tagging positive and negative words, we use LDA to extract topics and key words from each review. The

disadvantage of this model is that if the data in negative topic and positive topic are not very adequate, the accuracy of topic extraction will be low.

Business question and conclusion

After summarizing the results got from all three models, we can make a conclusion for our business question: what are the major factors that affect customers' satisfaction.

First, cleanliness. Customers really care about if the room, bathroom, the sheet on the bed is clean or not. It is very important factor for them to rating a house or apartment.
Second, facility, like bed is comfortable, the kitchen is well equipped, etc.
Third, location. The house near to public transportation and grocery store are always very popular. Because customers can easy travel to other place for their trip and they can buy necessary supplies nearby.
The last one is about service. Lots of comments mentioned about great host , feel at home. So we think the service provide by host, or the attitude of host may be a significant factor. A nice host will leave great impression to customers.

IST-718: NYC PROPERTY SALES DATA ANALYSIS

Abstract

The coronavirus outbreak might have people feeling uncertain about whether a change in home prices will impact their housing plans. It has already caused many to take a "wait and see" approach towards real state. For property sales companies, what they care most is what kind of property may have the highest preference from the customers. Hence, we use the NYC Property Sales dataset, which contains the house sale prices and relative information about the houses, like address and land square feet, to help predict the property prices.

Our prediction contains the followings:

1. Check if there's a time-related trend in the property price
2. Find out which neighborhood has the highest or property sales prices
3. Predict which neighborhood may have relatively new or old property
4. Predict sale price of different building class.

From the predictions, some inference questions can be answered, and they will be useful as

reference for property sale companies. The inferences are shown below:

1. Find the most important factor on the sale price for all house sales.
2. Distinguish the differences in the most influential factor on house prices in different places
3. Verify if there's a gap between small-size houses and large-size houses in sale price's factors
4. Find the differences in factors on sale prices between residential and commercial units

Check if the building class may affect the sale price factors.

Several models will be applied to the dataset to find the influential factors on housing prices. Linear regression, random forest and gradient boosted trees are applied for our project purpose.

After the data analysis, the main goal of the project is to find ways to calculate property sale price with property features as a reference for buyers and sellers when they are trading property. Companies will be able to make critical decisions related to hiring and investments based on the forecasting.

Data Collection/ Cleaning / Exploration

Data Collection

The NYC Property Sales dataset is a record of every building or building unit (apartment, etc.) sold in New York City over a 12-month period from September 2016 to September 2017. There are 84548 rows and 22 columns in total, containing the location, address, type, sales price and sales date.

Data clean

Remove the duplicate data

We found the duplicate sales data by the columns ('ADDRESS', 'NEIGHBORHOOD','SALE DATE' and 'SALE PRICE'). We found 2523 duplicate data. Then, we drop the duplicate rows

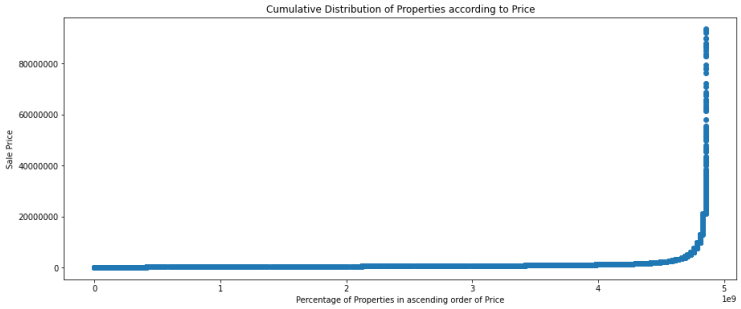
to avoid influence of dulplicate data in future analyze.

Deal with null value

We found there are two forms of the null value existing in our data. The first one is the NA values. In addition, some null value used '-' in our dataset. So, we replace them as '0''. And we remove 0 value in sale price column. Although some zero sales price represent actually transferring of deeds between parties, we still move it to analyze the actual value of the property in NYC.

Deal with outlier

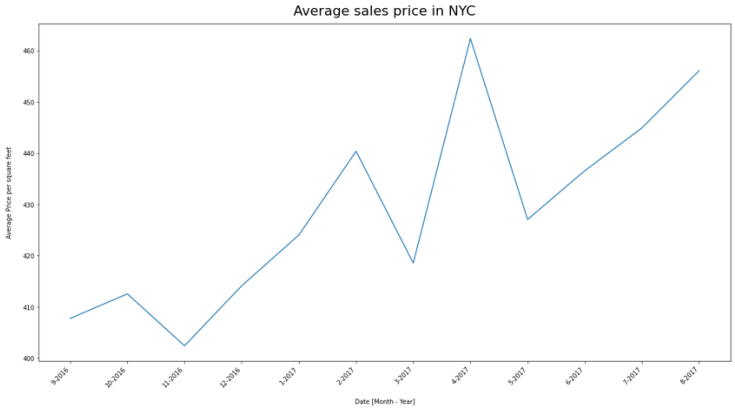
After analyzing the distribution of the sale price in NYC, we can find that part of value is extremely large which may influent our analytics. Therefore, we only take the real estate sales data from $100,000 to $100,000,000. After that the data distributed is more balanced.



Data Exploration

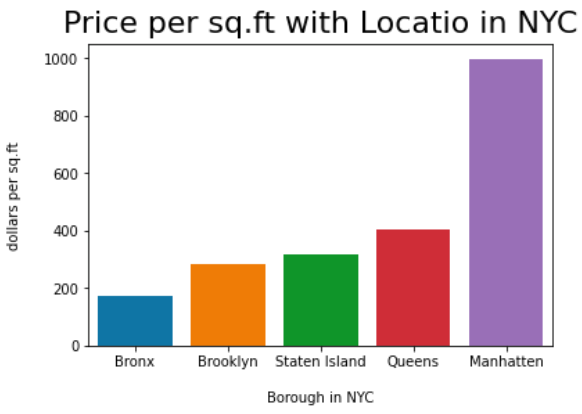
Trends in the Property Market

The price per square feet of property sale in New York City shows a volatile upward trend during this year.



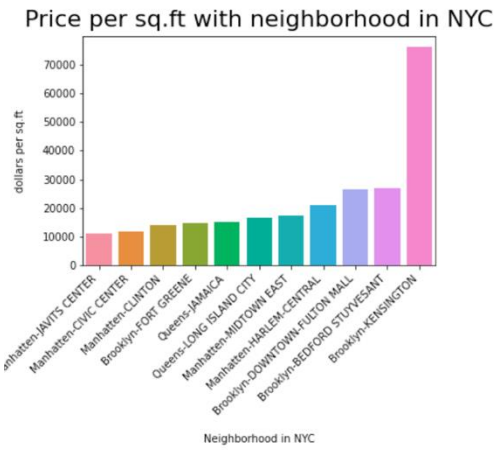
Location vs property sales price

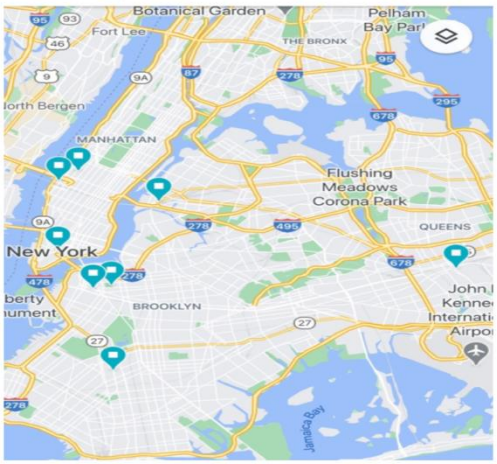
The house price of Manhattan is much higher than other brough in NYC.



Distribution of top property sales price

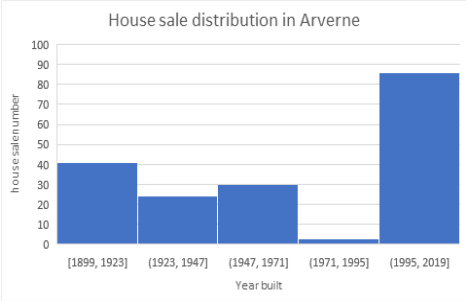
The most expensive house neighbors gather at south part in NYC. Most of them located Distributed along the Hudson River and the East River.

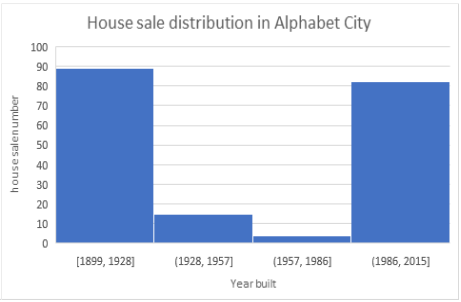




Preference on property age

From the dataset, two random districts are selected from one borough. From the two graphs, one is occupied by the newly built property while the other have an over 50% of old house in the property sales. Hence, people may not have specific preference on house age.





Type of building class vs sales price

There are totally 166 different types of building class, each of which consists of a character and a number representing different usage of the building and number of rooms in the building. [2] For example, R4 stands for condo, residential unit in elevator building. The following picture shows the specific 166 types that appear in our project.

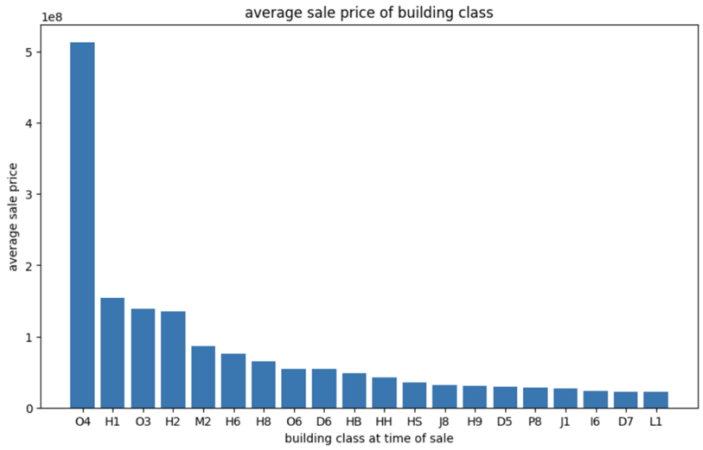
```
166
array(['R4', 'B3', 'A5', 'B1', 'B9', 'D4', 'A1', 'C0', 'R2', 'A2', 'A9',
       'B2', 'C6', 'R9', 'R3', 'S2', 'D7', 'C2', 'RP', 'RB', 'D0', 'C1',
       'G1', 'C7', 'R8', 'G7', 'R1', 'A3', 'V0', 'K2', 'RS', 'R6', 'C3',
       'A0', 'K4', 'RR', 'A4', 'G0', 'RG', 'E1', 'V1', 'S1', 'D1', 'C5',
       'K1', 'O4', 'F5', 'S9', 'G6', 'M4', 'A6', 'F9', 'P9', 'S5', 'H3',
       'S0', 'Z9', 'V2', 'C4', 'RH', 'S4', 'O2', 'N2', 'E2', 'O7', 'E9',
       'G2', 'O5', 'D5', 'F1', 'S3', 'M9', 'G9', 'RK', 'F4', 'D3', 'M1',
       'Z0', 'V3', 'W9', 'E7', 'O6', 'D9', 'L8', 'W1', 'J4', 'W2', 'I7',
       'H2', 'O9', 'W8', 'H8', 'K7', 'O1', 'D6', 'Q9', 'K9', 'G8', 'K5',
       'G4', 'O8', 'RA', 'RW', 'L1', 'H4', 'W4', 'K3', 'I5', 'R5', 'J9',
       'C9', 'P5', 'I1', 'K6', 'P8', 'I6', 'V9', 'I4', 'M3', 'Q1', 'W3',
       'G5', 'D2', 'I9', 'C8', 'R0', 'GU', 'RT', 'H1', 'A7', 'L9', 'U6',
       'D8', 'HB', 'F2', 'N9', 'O3', 'P2', 'GW', 'HR', 'U1', 'J5', 'I3',
       'P6', 'G3', 'W6', 'Y1', 'T2', 'Z2', 'J1', 'J8', 'Z3', 'V6', 'Y3',
       'HS', 'Z7', 'L3', 'P7', 'H9', 'H6', 'K8', 'R7', 'CM', 'HH', 'Q8',
       'M2'], dtype=object)
```

Based on different types of building class, we compute the average sales price and count the total number for each category as shown in the data frame below. In our data set, there are two columns refer to building class. One is building class at present, the other is building class at time of sale. We decide to use building class at time of sale because we are focusing on the relationship of variables and sales price and building class at present may differ from building class at time of sale, so building class at present doesn't matter in our case.

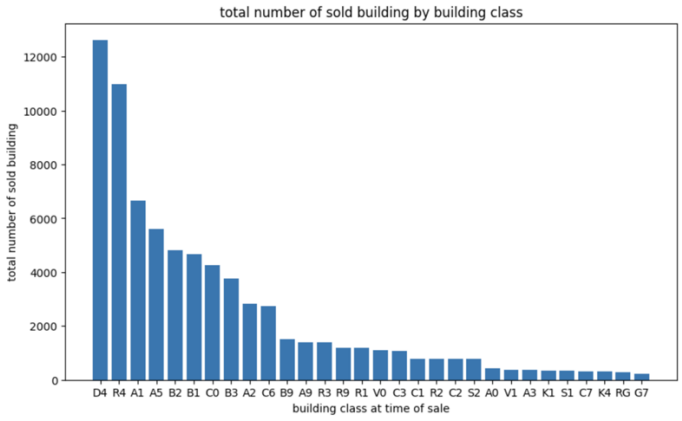
BUILDING CLASS AT TIME OF SALE	avg(SALE PRICE)	count(1)
Z9	3243628.5409836066	140
RG	305466.39366515836	281
C6	472013.59521943575	2743
A9	536916.4502369668	1395
H6	7.535E7	2
D7	2.1996417849056605E7	64
R6	973033.7923497268	187
D5	2.9869796470588237E7	21
W6	0.0	2
V3	16833.333333333332	8
F2	2618750.0	5
R8	3667276.8292682925	46
A6	219079.14285714287	90
RA	1.93206434E7	9
K8	225000.0	1
E9	4488266.05	74
M4	4035833.3333333335	6
V0	441913.91957104555	1116
I5	4122744.0	14
L8	6593105.181818182	22

only showing top 20 rows

We also created two plots for our findings. The first plot is average sales price of building class as shown in the picture below. From this picture, we can see that the price is extremely right skewed, so I took the top 20 for plotting. The highest average price is O4, which refers to the office only with or without comm with 20 stories or more. The second and the third highest is H1 and H2, which represent luxury hotel and full service hotel. Follows by O3, which is office with 7 to 19 stories. For now, we can conclude that the price of office building is much higher than other type, follows by luxury hotel.

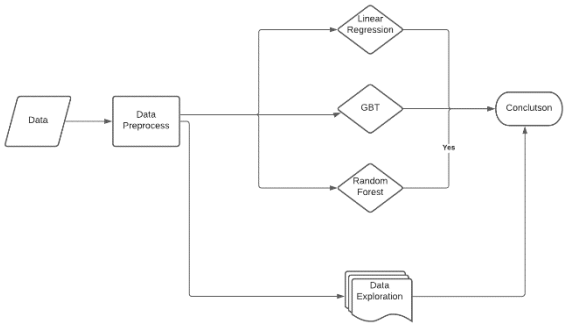


Another plot we create is the count of sold building by building class. This plot is right skewed as well, but not as severe as average sales price below, and thus I took top 30 for the plot. From the plot, we can see that the types of the top two sold buildings are R4 and D4. R4 is condo, or residential unit in elevator buildings. D4 is elevator cooperative buildings. They were sold more than 12,000 in the last year. Follows by A1 and A5, which are two stories and one family attached or semi-detached. The third group is B2 and B1 which are two family frame and two family brick.



Methodology

Our process follows the flow diagram.Data will be cleaned first. NAs will be removed and unreasonable data types will be regulated into the form qualified for the models inputs. The dataset will be separated into training and testing part. After that, three models, linear regression, random forest and GBT, will be applied to solve the regression problem. In the evaluation step, mean square error is utilized to grade the prediction accuracy of our models. With the help of the models and evaluation, we can draw our conclusion for this project.



Models

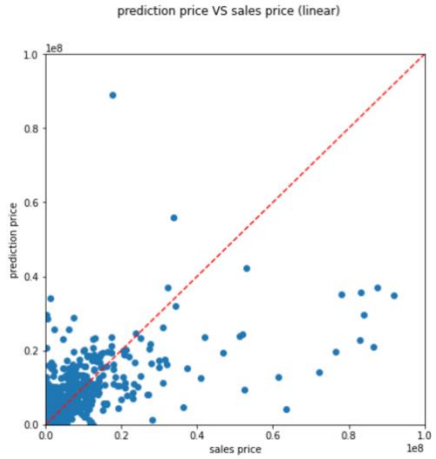
Linear regression

This model is designed to predict property sales price based on the property information. it contains the columns, borough, neighborhood, building class category, residential units, commercial units, land square feet, gross square feet, building age as the predictors. We use both borough and neighborhood as the location information of property to make the location more detailed. As borough, neighborhood, building class category are all categorical columns and don't have a specific order, we convert them into dummy variables. Considering

the fact that unit property price varies from one location to another, we also multiply borough columns by land square feet and gross square feet columns and get new features borough* gross square feet, borough* land square feet so that the coefficient of gross square feet in linear model can be different for different borough.

As we usually do in homework, we split the dataset into two parts, one part for training and the other for testing, and use MSE as the metric to evaluate the linear model.

The MSE score we get is 9.10489e+12, which means the mean estimation error on the price is more than one million dollars. The accuracy of the property price estimation is beyond our expectation. After trying to add more columns, the result doesn't change a lot. We then check the original data in the CSV file and find that the sales price of property of larger square feet is sometimes lower than that of property of the same information other than square feet. We think some factors have more influence on sales price than the elements listed in the dataset like decoration. If one apartment is within the walking distance of some universities like NYU, it tends to have higher sales price compared to another apartment in the same borough far from NYU. The following graph shows the relation of our price prediction and the real price.



The closer the blue dots are to the red dotted line, the more accurate the linear model is.

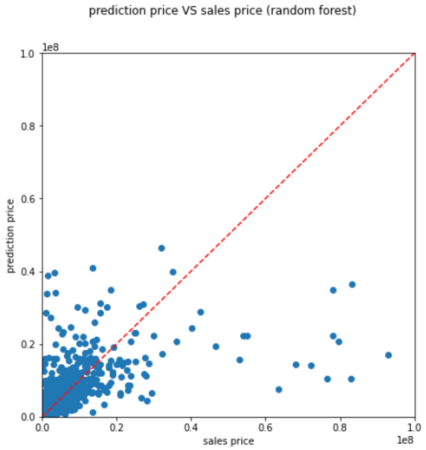
Next, we use inference to determine the most important predictor order. We create a new pipeline which encapsulates a standard scalar and a linear regression object. After fitting the pipe, we create a pandas data frame with 2 columns named coefficient and value. The coefficient column contains the coefficient names and the value column contains the regression model coefficient absolute values.

	coefficient	value
105	neighborhood_tribeca	8.393703e+07
196	neighborhood_midtown_cbd	3.092497e+07
270	CLASS_34	2.276959e+07
277	CLASS_10	2.011012e+07
34	neighborhood_midtown_west	1.786323e+07
95	neighborhood_downtown_fulton_ferry	1.723994e+07
26	neighborhood_soho	1.516637e+07
253	neighborhood_financial	1.515842e+07
278	CLASS_26	1.459667e+07
240	neighborhood_flatiron	1.237781e+07

We can see the top ten predictors are all from neighborhood and building class category which is not consistent with our common sense.

Random Forest

We use the same methods as linear model to transform categorical variables. To shorten the running time of the model, we only use borough as the location information of property.
The random forest model utilized a grid search to tune the regularization and elastic net parameters. The grid utilized num trees of 10, 30, 50 and max depth of 10 and 15. The trained model variations are evaluated by the mean square error (MSE) testing results after applying the model to the test dataset. The best model uses the num trees of 30 and max depth of 15. The best random forest model was able to achieve a MSE score of 6.10829e+12 which is a little better than the result of linear regression model. The following graph shows the relation of our price prediction and the real price.



Below is the most important predictor in random forest model.

	feature	importance
36	gross_square_feet	0.248924
33	RESIDENTIAL UNITS	0.131941
34	COMMERCIAL UNITS	0.104896
3	borough_Manhatten	0.101680
37	building_age	0.099092
35	land_square_feet	0.072694
29	CLASS_26	0.056407
11	CLASS_08	0.031235
25	CLASS_07	0.014930
28	CLASS_10	0.014499

From the table, the model's most important feature is the gross square feet of the property. The residential units of the property are the second feature people care about. This shows living area is people's most important concern. It does make sense. Besides, the commercial units and the building age can only explain some buyers' interest on properties. These two features are less preferred by the buyers.

The Gradient boosted trees Model

This model is designed to address the relationship between the property area and the price.

It will contain the total units, the land square feet, gross square feet, and the year built as the factors. It will focus more on the property conditions rather than the neighboring environment. This model will answer the question: how the property area affects the property price in New York.

This model is used to solve a regression problem. After the dataset split, a part of the dataset is used for training and the other can be used for testing the accuracy of the model. The testing result can show us the gap between the model estimation on prices and the real sales of the property.

Model 3 Data Transformations

GBT model. The outliers and NAs in the related columns are removed first. In addition, since the original data type in these columns are string, they are transformed into float type for the future model processing. This model uses only 4 features in total. The following features were not included:

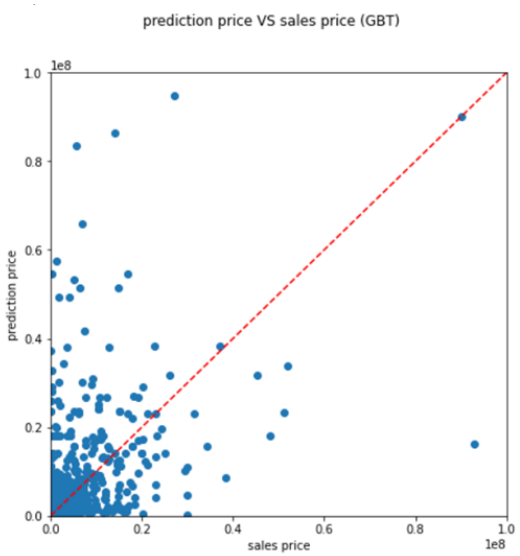
- Data about the property location: borough, address, apartment number, zip code, block
- Categorical data about the property feature: neighborhood building class category, building class at present, residential units, commercial units, lot, easement, tax class at present:
- Data about the sales of the property. tax class at time of sale, building class at time of sale, sale date:

Model 3 Evaluation

The GBT model utilized a grid search to tune the regularization and elastic net parameters. The grid utilized max iteration of 10, 20, 50 and max depth of 5 and 10. The trained model variations are evaluated by the mean square error (MSE) testing results after applying the model to the test dataset. The best model uses the max iteration of 10 and max depth of 10.

Model 3 Interpretation

The best GBT model was able to achieve a MSE score of 10^12.This shows that the accuracy of the property price estimation is within our expectation. The following graph shows the relation of our price prediction and the real price.



Model 3 Inference and Key Findings

The feature coefficients were extracted from the model and sorted based on absolute value to provide insight on feature importance. The ordered feature coefficients are provided below.

	feature	importance
0	TOTAL UNITS	0.334043
3	YEAR BUILT	0.332740
2	GROSS SQUARE FEET	0.199619
1	LAND SQUARE FEET	0.133598

From the table, the model's most important feature is the total units of the property. The built-year of the property is the second feature people care about. This shows buyers have some preference on the property age. Besides, the gross square feet and the land square feet can only explain some buyers' interest on properties. These two features are less preferred by the buyers. To answer our business questions, we can conclude that the total units and the building age are quite important factors for buyers.

Conclusion

Our main goal of predicting property sales price is to provide a tool for people who want to buy or sell property. Using this tool, they can get a reasonable price if they input the property information. The mse of linear model is 9.10489e+12. The mse of random forest model is 6.10829e+12. The mse of GBT model is 5.01303e+13. The random forest performs best.
The most important feature of the property is the gross square feet, residential units, commercial units and building age. These are the factors people most care about when

buying or selling property.
The overall trend of the property sales price is ascending. The property of Manhantton and Madison has the highest price.

CONCLUSION

This portfolio has demonstrated the successful implementation of these learning objectives and the major practice areas in data science. Data was collected and managed using web scraping and application programming interfaces in conjunction with database solutions to be analyzed using statistical methods and data mining techniques for tasks such as regression, classification, or clustering. Various data visualizations were paired with clustering techniques to identify patterns which directed the respective analyses; actionable recommendations were developed to reflect tangible business decisions

Communications skills were developed and displayed in the organization and delivery of insights, expressing them in terms which could be simply understood and acted upon. The ethical dimensions of data science practice were also reinforced in these applications by selecting only relevant data and considering user privacy when analyzing personally identifiable information.

Syracuse University's School of Information Studies provides students the opportunity to synthesize the collection, management, and analysis of data, as well as the delivery of actionable insights using various data science techniques.